# Error-Correcting Tournaments

Alina Beygelzimer[1], John Langford[2], and Pradeep Ravikumar[3]

[1] IBM Thomas J. Watson Research Center, Hawthorne, NY 10532, USA
beygel@us.ibm.com
[2] Yahoo! Research, New York, NY 10018, USA
jl@yahoo-inc.com
[3] University of California, Berkeley, CA 94720, USA
pradeepr@stat.berkeley.edu

**Abstract.** We present a family of pairwise tournaments reducing $k$-class classification to binary classification. These reductions are provably robust against a constant fraction of binary errors, and match the best possible computation and regret up to a constant.

## 1 Introduction

We consider the classical problem of multiclass classification, where given an instance $x \in X$, the goal is to predict the most likely label $y \in \{1, \ldots, k\}$, according to some unknown probability distribution.

A common general approach to multiclass learning is to reduce a multiclass problem to a set of binary classification problems [2, 6, 10, 11, 14]. This black-box approach is composable with any binary learning algorithm (and thus bias), including online algorithms, Bayesian algorithms, and even humans.

A key technique for analyzing reductions is *regret analysis*, which bounds the "regret" of the resulting multiclass classifier in terms of the average classification "regret" on the binary problems. Here *regret* is the difference between the incurred loss and the smallest achievable loss on the problem, i.e., excess loss due to suboptimal prediction.

The most commonly applied reduction is one-against-all, which creates a binary classification problem for each of the $k$ classes: The classifier for class $i$ is trained to predict whether the label is $i$ or not; predictions are done by evaluating each binary classifier and randomizing over those which predict "yes," or randomly if all answers are "no". This simple reduction is *inconsistent*, in the sense that given optimal (zero-regret) binary classifiers, the reduction may not yield an optimal multiclass classifier in the presence of noise. Optimizing squared loss of the binary predictions instead of the $0/1$ loss makes the approach consistent, but the resulting multiclass regret may be as high as $\sqrt{2kr}$, where $r$ is the average squared loss regret on the induced problems, which is upper bounded by the average binary classification regret via the Probing reduction [15].

The probabilistic error correcting output code approach (PECOC) [14] reduces $k$-class classification to learning $O(k)$ regressors on the interval $[0, 1]$, creating $O(k)$ binary examples per multiclass example at both training and test time, with a test time computation of $O(k^2)$. The resulting multiclass regret is bounded by $4\sqrt{r}$, where $r$ is the average squared loss regret of the regressors (which is upper bounded by the average binary classification regret via the Probing reduction [15]). Thus PECOC removes the dependence on the number of classes $k$. When only a constant number of labels have non-zero probability given $x$, the complexity can be reduced to $O(\log k)$ examples per multiclass example and $O(k \log k)$ computation per example [13].

This leads to several questions:

1. Is there a consistent reduction from multiclass to binary classification that does not have a square root dependence [17]? For example, an average binary regret of just $0.01$ may imply a PECOC multiclass regret of $0.4$.
2. Is there a consistent reduction that requires just $O(\log k)$ computation, matching the information theoretic lower bound? The well known tree reduction (see [9]) distinguishes between the labels using a balanced binary tree, where each non-leaf nodes predicts "Is the correct multiclass label to the left or right?". As shown in Section 2, this method is inconsistent.
3. Can the above be achieved with a reduction that only performs pairwise comparisons between classes? One fear associated with the PECOC approach is that it creates binary problems of the form "What is the probability that the label is in a given random subset of labels?," which may be hard to solve. Although this fear is addressed by regret analysis (as the latter operates only on *excess* loss), and is overstated in some cases [8, 13], it is still of some concern, especially with larger values of $k$.

The error-correcting tournament family presented here answers all of these questions in the affirmative. It provides an exponentially faster in $k$ method for multiclass prediction with the resulting multiclass regret bounded by $5.5r$, where $r$ is the average binary regret and every binary classifier logically compares two distinct class labels.

The result is based on a basic observation that if a non-leaf node fails to predict its binary label, which may be unavoidable due to noise in the distribution, nodes between this node and the root should have no preference for class label prediction. Utilizing this observation, we construct a reduction, called the Filter Tree, with the property that it uses $O(\log k)$ binary examples and $O(\log k)$ computation at training and test time with a multiclass regret bounded by $\log k$ times the average binary regret.

The decision process of a Filter Tree, viewed bottom up, can be viewed as a single-elimination tournament on a set of $k$ players. Using $c$ independent single-elimination tournaments is of no use as it does not affect the *average* regret of an adversary controlling the binary classifiers. Somewhat surprisingly, it is possible to have $c = \log k$ complete single-elimination tournaments between $k$ players in $O(\log k)$ rounds with no player playing twice in the same round [5]. All error-correcting tournaments first pair labels in consecutive interfering single-elimination tournaments, followed by a final carefully weighted single-elimination tournament that decides among the $\log_2 k$ winners of the first phase. As for the Filter Tree, test time evaluation can start at the root and proceed to a multiclass label with $O(\log k)$ computation.

This construction is also useful for the problem of robust search, yielding the first algorithm which allows the adversary to err a constant fraction of the time in the "full lie" setting [16] where a comparator can missort any comparison. Previous work either applied to the "half lie" case where a comparator can fail to sort but can not actively missort [5, 18] or to a "full lie" setting where an adversary has a fixed known bound on the number of lies [16] or a fixed budget on the fraction of errors so far [4, 3]. Indeed, it might even appear impossible to have an algorithm robust to a constant fraction of full lie errors since an error can always be reserved for the last comparison. By repeating the last comparison $O(\log k)$ times we can defeat this strategy.

The result here is also useful for the actual problem of tournament construction in games with real players. Our analysis does not assume that errors are *i.i.d.* [7], or have known noise distributions [1] or known outcome distributions given player skills [12]. Consequently, the tournaments we construct are robust against severe bias such as a biased referee or some forms of bribery and collusion. Furthermore, the tournaments we construct are shallow, requiring fewer rounds than $m$-elimination bracket tournaments, which do not satisfy the guarantee provided here. In an *$m$-elimination bracket tournament*, bracket $i$ is a single-elimination tournament on all players except the winners of brackets $1, \ldots, i-1$. After the bracket winners are determined, the player winning the last bracket $m$ plays the winner of bracket $m-1$ repeatedly until one player has suffered $m$ losses (they start with $m-1$ and $m-2$ losses respectively). The winner moves on to pair against the winner of bracket $m-2$, and the process continues until only one player remains. This method does not scale well to large $m$, as the final elimination phase takes $\sum_{i=1}^{m} i-1 = O(m^2)$ rounds. Even for $k = 8$ and $m = 3$, our constructions have smaller maximum depth than bracketed 3-elimination.

*Paper overview* Section 2 shows that the simple divide-and-conquer tree approach is inconsistent, motivating the Filter Tree algorithm described in section 3 (which applies to more general cost sensitive multiclass problems). Section 3.1 proves that the algorithm has the best possible computational dependence, and gives two upper bounds on the regret of the returned (cost-sensitive) multiclass classifier.

Section 4 presents the error-correcting tournament family parametrized by an integer $m \geq 1$, which controls the tradeoff between maximizing robustness ($m$ large) and minimizing depth ($m$ small). Setting $m = 1$ gives the Filter Tree, while $m = 4 \ln k$ gives a (multiclass to binary) regret ratio of $5.5$ with $O(\log k)$ depth. Setting $m = ck$ gives regret ratio of $3 + O(1/c)$ with depth $O(k)$. The results here provide a nearly free generalization of earlier work [5] in the robust search setting, to a more powerful adversary that can missort as well as fail to sort. Section 5 gives an algorithm independent lower bound of 2 on the regret ratio for large $k$. When the number of calls to a binary classifier is independent (or nearly independent) of the label predicted, we strengthen this lower bound to 3 for large $k$.

## 2 Inconsistency of Divide and Conquer Trees

One standard approach for reducing multiclass learning to binary learning is to split the set of labels in half, then learn a binary classifier to distinguish between the subsets, and repeat recursively until each subset contains one label. Multiclass predictions are made by following a chain of classifications from the root down to the leaves.

The following theorem shows that there exist multiclass problems such that even if we have an optimal classifier for the induced binary problem at each node, the tree reduction does not yield an optimal multiclass predictor.

**Notation:** Let $D$ be the underlying distribution over $X \times Y$, where $X$ is some observable feature space and $Y = \{1, \ldots, k\}$ is the label space. The *error rate* of a classifier

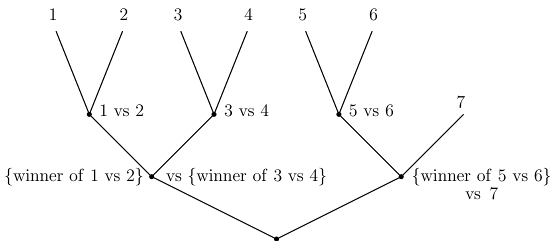

**Fig. 1.** Filter Tree. Each node predicts whether the left or the right input label is more likely, conditioned on a given $x \in X$. The root node predicts the best label for $x$.

$f : X \rightarrow Y$ on $D$ is given by $\mathrm{err}(f, D) = \mathbf{Pr}_{(x,y)\sim D}[f(x) \neq y]$. The *regret* of $f$ on $D$ is defined as $\mathrm{reg}(f, D) = \mathrm{err}(f, D) - \min_{f^*} \mathrm{err}(f^*, D)$.

The tree reduction transforms $D$ into a distribution $D_T$ over binary labeled examples by drawing a multiclass example $(x, y)$ from $D$, drawing a random non-leaf node $i$, and outputting instance $\langle x, i\rangle$ with label 1 if $y$ is in the left subtree of node $i$, and 0 otherwise. A binary classifier $f$ for this problem induces a multiclass classifier $T(f)$, via a chain of binary predictions starting from the root.

**Theorem 1.** *For all $k \geq 3$, for all binary trees over the labels, there exists a multiclass distribution $D$ such that* $\mathrm{reg}(T(f^*), D) > 0$ *for any* $f^* = \arg\min_f \mathrm{err}(f, D_T)$.

**Proof:** Find a node with one subset corresponding to two labels and the other subset corresponding to a single label. (If the tree is perfectly balanced, simply let $D$ assign probability 0 to one of the labels.) Since we can freely rename labels without changing the underlying problem, let the first two labels be 1 and 2, and the third label be 3. Choose $D$ with the property that $D(y = 1 \mid x) = D(y = 2 \mid x) = 1/4 + 1/100$, while $D(y = 3 \mid x) = 1/2 - 2/100$. Under this distribution, the fraction of examples for which label 1 or 2 is correct is $1/2 + 2/100$, so any minimum error rate binary predictor must choose either label 1 or label 2. Each of these choices has an error rate of $3/4 - 1/100$. The optimal multiclass predictor chooses label 3 and suffers an error rate of $1/2 + 2/100$, implying that the regret of the tree classifier based on an optimal binary classifier is $1/4 - 3/100 > 0$. ∎

## 3 The Filter Tree Algorithm

The Filter Tree algorithm is illustrated by Figure 1. It is equivalent to a single-elimination tournament on the set of labels structured as a binary tree $T$ over the labels. In the first round, the labels are paired according to the lowest level of the tree, and a classifier is trained for each pair to predict which of the two labels is more likely. (The labels that don't have a pair in a given round, win that round for free.) The winning labels from the first round are in turn paired in the second round, and a classifier is trained to predict whether the winner of one pair is more likely than the winner of the other. The process of training classifiers to predict the best of a pair of winners from the previous round is repeated until the root classifier is trained.

**Algorithm 1** `Filter-Train` (multiclass training set $S$, binary learner `Learn`)

```
for each non-leaf node $n$ in order from leaves to root do
    Set $S_n = \emptyset$
    for each $(x, y) \in S$ such that $y \in \Gamma(T_n)$ and all nodes $u$ on the path $n \rightsquigarrow y$ predict $y_u$ given $x$ do
            add $(x, y_n)$ to $S_n$
    end
    Let $c_n = \texttt{Learn}(S_n)$
end
return $c = \{c_n\}$
```

**Algorithm 2** `C-Filter-Train` (cost-sensitive training set $S$, importance-weighted binary learner `Learn`)

```
for each non-leaf node $n$ in the order from leaves to root do
    Set $S_n = \emptyset$
    for each example $(x, c_1, ..., c_k) \in S$ do
        Let $a$ and $b$ be the two classes input to $n$
        $S_n \leftarrow S_n \cup \{(x, \arg\min\{c_a, c_b\}, |c_a - c_b|)\}$
    end
    Let $c_n = \texttt{Learn}(S_n)$
end
return $c = \{c_n\}$
```

The setting above is akin to Boosting: At each round $t$, a booster creates an input distribution $D_t$ and calls an oracle learning algorithm to obtain a classifier with some error $\epsilon_t$ on $D_t$. The distribution $D_t$ depends on the classifiers returned by the oracle in previous rounds. The accuracy of the final classifier is analyzed in terms of $\epsilon_t$'s.

Let $T_n$ be the subtree of $T$ rooted at node $n$. The set of leaves of a tree $T$ is denoted by $\Gamma(T)$. Let $y_n$ be the bit specifying whether the multiclass label $y$ is in the left subtree of $n$ or not.

The key trick in the training stage (Algorithm 1) is to form the right training set at each interior node. A training example for node $n$ is formed conditioned on the predictions of classifiers in the round before it. Thus the learned classifiers from the first level of the tree are used to "filter" the distribution over examples reaching the second level of the tree.

Given $x$ and classifiers at each node, every edge in $T$ is identified with a unique label. The optimal decision at any non-leaf node is to choose the input edge (label) that is more likely according to the true conditional probability. This can be done by using the outputs of classifiers in the round before it as a filter during the training process: For each observation, we set the label to 0 if the left parent's output matches the multiclass label, 1 if the right parent's output matches, and reject the example otherwise.

The testing algorithm, `Filter-Test`, is very simple. Given a test example $x \in X$, we output the label $y$ such that every classifier on the path from $y$ to the root prefers $y$.

Algorithm 2 extends this idea to the cost-sensitive multiclass case where each choice has a different associated cost. Formally, a *cost-sensitive* $k$-class classification problem is defined by a distribution $D$ over $X \times [0, 1]^k$. The expected cost of a classifier $f : X \rightarrow \{1, ..., k\}$ of $D$ is $\ell(f, D) = E_{(x,c)\sim D}\left[c_{f(x)}\right]$. Here $c \in [0, 1]^k$ gives the cost of each of the $k$ choices for $x$. As in the multiclass case (which is a special case), the regret of $f$ on $D$ is defined as $\operatorname{reg}_{\mathrm{c}}(f, D) = \ell(f, D) - \min_{f^*} \ell(f^*, D)$.

The algorithm relies upon an importance weighted binary learning algorithm, which takes examples of the form $(x, y, w)$, where $x$ is a feature vector used for prediction, $y$ is a binary label, and $w \in [0, \infty)$ is the importance any classifier pays if it doesn't predict $y$ on $x$.

### 3.1 Filter Tree Analysis

Before doing the regret analysis, we note the computational characteristics of the Filter Tree. Since the algorithm is a reduction, we count the computational complexity in the reduction itself, assuming that the oracle calls take unit time.

1. Algorithm 1 requires $O(\log k)$ computation per multiclass example, by searching for the correct leaf in $O(\log k)$ time, then filtering back toward the root. This matches the information theoretic lower bound since simply reading one of $k$ labels requires $\log_2 k$ bits.
2. Algorithm 2 requires $O(k)$ computation per cost sensitive example, because there are $k - 1$ nodes, each requiring constant computation per example. Since any method must read the $k$ costs, this bound is tight.
3. The testing algorithm is the same for both multiclass and cost-sensitive variants, requiring $O(\log k)$ computation per example to descend a binary tree. Any method must write out labels of length $\log_2 k$ bits.

First, we define several concepts necessary to understand the analysis. Algorithm 2 transforms cost-sensitive multiclass examples into importance-weighted binary examples. This process implicitly transforms a distribution $D$ over cost sensitive multiclass examples into a distribution $D_{\mathtt{FT}}$ over importance-weighted binary examples.

There are many induced problems, one for each call to the oracle `Learn`. To simplify the analysis, we use a standard transformation allowing us to consider only a single induced problem: We add the node index $n$ as an additional feature into each importance weighted binary example, and then train based upon the union of all the training sets. The learning algorithm produces a single binary classifier $c(x, n)$ for which we can redefine $c_n(x)$ as $c(x, n)$. The induced distribution $D_{\mathtt{FT}}$ can be defined by the following process: (1) draw a cost-sensitive example $(x, c)$ from $D$, (2) pick a random node $n$, (3) create an importance-weighted sample according to the algorithm, except using $\langle x, n \rangle$ instead of $x$.

The theorem is quantified over all classifiers, and thus it holds for the classifier returned by the algorithm. In practice, one can either call the oracle multiple times to learn a separate classifier for each node (as we do in our experiments), or use iterative techniques for dealing with the fact that the classifiers are dependent on other classifiers closer to the leaves.

When reducing to importance-weighted classification, the theorem statement depends on importance weights. To remove the importances, we compose the reduction with the Costing reduction [19], which alters the underlying distribution using rejection sampling on the importance weights. This composition transforms $D_{\mathtt{FT}}$ into a distribution $D'$ over binary examples.

We use the folk theorem from [19] saying that for all binary classifiers $f$ and all importance weighted binary distributions $P$, the importance weighted binary regret of $f$ on $P$ is upper bounded by $E_{(x,y,w)\sim P}[w]$ times the binary regret of $f$ on the induced binary distribution.

The core theorem relates the regret of a binary classifier $f$ to the regret of the induced cost sensitive classifier $\texttt{Filter-Test}(f)$.

**Theorem 2.** *For all binary classifiers $f$ and all cost sensitive multiclass distributions $D$,*

$$\mathrm{reg}_{\mathrm{c}}(\texttt{Filter-Test}(f), D) \leq \mathrm{reg}(f, D')E_{(x,c)\sim D}\sum_{n\in T} w(n,x,c),$$

*where $w(n,x,c)$ is the importance weight in Algorithm 2 (the difference in cost between the two labels that node $n$ chooses between on $x$), and $D'$ is the induced distribution as defined above.*

Before proving the theorem, we state the corollary for multiclass classification.

**Corollary 1.** *For all binary classifiers $f$ and all multiclass distributions $D$ on $k$ labels, for all Filter Trees of depth $d$,* $\mathrm{reg}(\texttt{Filter-Test}(f), D) \leq d\cdot \mathrm{reg}(f, D_{\texttt{FT}})$.

(Since all importance weights are either 0 or 1, we don't need to apply Costing.) The proof of the corollary given the theorem is simple since for any $(x,y)$, the induced $(x,c)$ has at most one node per level with induced importance weight 1; all other importance weights are 0. Therefore, $\sum_n w(n,x,c) \leq d$.

Theorem 3 provides an alternative bound for cost-sensitive classification. It is the first known bound giving a worst-case dependence of less than $k$.

**Theorem 3.** *For all binary classifiers $f$ and all cost-sensitive $k$-class distributions $D$,* $\mathrm{reg}_{\mathrm{c}}(\texttt{Filter-Test}(f), D) \leq k\,\mathrm{reg}(f, D')/2$, *where $D'$ is as defined above.*

The remainder of this section proves Theorems 2 and 3.

**Proof:** (Theorem 2) It is sufficient to prove the claim for any $x \in X$ because that implies that the result holds for all expectations over $x$.

Conditioned on the value of $x$, each label $y$ has a distribution over costs $c_y$ with an expected value $E_{c\sim D|x}[c_y]$. The zero regret cost sensitive classifier predicts according to $\arg\min_y E_{c\sim D|x}[c_y]$. Suppose that $\texttt{Filter-Test}(f)$ predicts $y'$ on $x$, inducing cost sensitive regret $\mathrm{reg}_{\mathrm{c}}(y', D|x) = E_{c\sim D|x}[c_{y'}] - \min_y E_{\boldsymbol{c}\sim D|x}[c_y]$.

First, we show that the sum over the binary problems of the importance weighted regret is at least $\mathrm{reg}_{\mathrm{c}}(y', D|x)$, using induction starting at the leaves. The induction hypothesis is that the sum of the regrets of importance-weighted binary classifiers in any subtree bounds the regret of the subtree output.

For node $n$, each importance weighted binary decision between class $a$ and class $b$ has an importance weighted regret which is either 0 or $r_n = |E_{\boldsymbol{c}\sim D|x}[c_a - c_b]| = |E_{\boldsymbol{c}\sim D|x}[c_a] - E_{\boldsymbol{c}\sim D|x}[c_b]|$, depending on whether the prediction is correct or not.

Assume without loss of generality that the predictor outputs class $b$. The regret of the subtree $T_n$ rooted at $n$ is given by $r_{T_n} = E_{\boldsymbol{c}\sim D|x}[c_b] - \min_{y\in\Gamma(T_n)} E_{\boldsymbol{c}\sim D|x}[c_y]$.

As a base case, the inductive hypothesis is trivially satisfied for trees with one label. Inductively, assume that $\sum_{n'\in L} r_{n'} \geq r_L$ and $\sum_{n'\in R} r_{n'} \geq r_R$ for the left subtree $L$ of $n$ (providing $a$) and the right subtree $R$ (providing $b$).

There are two possibilities. Either the minimizer comes from the leaves of $L$ or the leaves of $R$. The second possibility is easy since we have

$$r_{T_n} = E_{\boldsymbol{c}\sim D|x}[c_b] - \min_{y\in\Gamma(R)} E_{\boldsymbol{c}\sim D|x}[c_y] = r_R \le \sum_{n'\in R} r_{n'} \le \sum_{n'\in T_n} r_{n'},$$

which proves the induction.

For the first possibility, we have

$$\begin{aligned} r_{T_n} &= E_{\boldsymbol{c}\sim D|x}[c_b] - \min_{y\in\Gamma(L)} E_{\boldsymbol{c}\sim D|x}[c_y] \\ &= E_{\boldsymbol{c}\sim D|x}[c_b] - E_{\boldsymbol{c}\sim D|x}[c_a] + E_{\boldsymbol{c}\sim D|x}[c_a] - \min_{y\in\Gamma(L)} E_{\boldsymbol{c}\sim D|x}[c_y] \\ &= E_{\boldsymbol{c}\sim D|x}[c_b] - E_{\boldsymbol{c}\sim D|x}[c_a] + r_L \le r_n + \sum_{n'\in L} r_{n'} \le \sum_{n'\in T_n} r_{n'}, \end{aligned}$$

which completes the induction. The inductive hypothesis for the root is that $\mathrm{reg}_{\mathrm{c}}(y', D|x) \le \sum_{n\in T} r_n$, implying $\mathrm{reg}_{\mathrm{c}}(y', D|x) \le \sum_{n\in T} r_n = (k-1)\cdot r_i(f, D_{\texttt{FT}})$, where $r_i$ is the importance weighted binary regret on the induced problem.

Using the folk theorem from [19], we have $r_i(f, D_{\texttt{FT}}) = \mathrm{reg}(f, D')E_{(x,y,w)\sim D_{\texttt{FT}}}[w]$. The expected importance is $\frac{1}{k-1}E_{(x,c)\sim D}\sum_{n\in T} w(n,x,c)$. Plugging this in, we get the theorem. ■

The proof of Theorem 3 makes use of the following inequality. Consider a Filter Tree $T$ evaluated on a cost-sensitive multiclass instance with cost vector $c \in [0,1]^k$. Let $S_T$ be the sum of importances over all nodes in $T$, and $I_T$ be the sum of importances over the nodes where the class with the larger cost was selected for the next round. Let $c_T$ denote the cost of the winner chosen by $T$.

**Lemma 1.** *For any Filter Tree $T$ on $k$ labels, $S_T + c_T \le I_T + \frac{k}{2}$.*

**Proof:** The inequality follows by induction, the result being clear when $k = 2$. Assume that the claim holds for the two subtrees, $L$ and $R$, providing their respective inputs $l$ and $r$ to the root of $T$, and $T$ outputs $r$ without loss of generality. Using the inductive hypotheses for $L$ and $R$, we get $S_T + c_T = S_L + S_R + |c_r - c_l| + c_r \le I_L + I_R + \frac{k}{2} - c_l + |c_r - c_l|$. If $c_r \ge c_l$, we have $I_T = I_L + I_R + (c_r - c_l)$, and $S_T + c_T \le I_T + \frac{k}{2} - c_l \le I_T + \frac{k}{2}$, as desired. If $c_r < c_l$, we have $I_T = I_L + I_R$ and $S_T + c_T \le I_T + \frac{k}{2} - c_r \le I_T + \frac{k}{2}$, completing the proof. ■

**Proof:** (Theorem 3) We will fix $(x, \boldsymbol{c}) \in X \times [0,1]^k$ and take the expectation over the draw of $(x, \boldsymbol{c})$ from $D$ as the last step.

Consider a Filter Tree $T$ evaluated on $(x, \boldsymbol{c})$ using a given binary classifier $b$. As before, let $S_T$ be the sum of importances over all nodes in $T$, and $I_T$ be the sum of importances over the nodes where $b$ made a mistake. Recall that the regret of $T$ on $(x, \boldsymbol{c})$, denoted in the proof by $\mathrm{reg}_T$, is the difference between the cost of the tree's output and the smallest cost $c^*$. The importance-weighted binary regret of $b$ on $(x, \boldsymbol{c})$ is simply $I_T/S_T$. Since the expected importance is upper bounded by 1, $I_T/S_T$ also bounds the binary regret of $b$.

The inequality we need to prove is $\mathrm{reg}_T S_T \le \frac{k}{2} I_T$. The proof is by induction on $k$, the result being trivial if $k = 2$. Assume that the assertion holds for the two subtrees, $L$ and $R$, providing their respective inputs $l$ and $r$ to the root of $T$. (The number of classes in $L$ and $R$ can be taken to be even, by splitting the odd class into two classes with the same cost as the split class, which has no effect on the quantities in the theorem statement.)

Let the best cost $c^*$ be in the left subtree $L$. Suppose first that $T$ chooses $r$ and $c_r > c_l$. Let $w = c_r - c_l$. We have $\text{reg}_L = c_l - c^*$ and $\text{reg}_T = c_r - c^* = \text{reg}_L + w$. The left hand side of the inequality is thus $\text{reg}_T\, S_T = (\text{reg}_L + w)(S_R + S_L + w) = w(\text{reg}_L + S_R + S_L + w) + \text{reg}_L(S_L + S_R) \leq w\left(\text{reg}_L + I_R + I_L - c_r - c_l + w + \frac{k}{2}\right) + \text{reg}_L\left(I_R + I_L - c_l - c_r + \frac{k}{2}\right) \leq \frac{k}{2}w + I_R(w + \text{reg}_L) + I_L(w + \text{reg}_L) + \text{reg}_L\left(\frac{k}{2} - c_r - c_l\right) \leq \frac{k}{2}w + I_R(w + \text{reg}_L) + I_L\left(w + \text{reg}_L + \frac{k}{2} - c_r - c_l\right) \leq \frac{k}{2}w + I_R(w + \text{reg}_L) + \frac{k}{2}I_L \leq \frac{k}{2}(w + I_R + I_L) = \frac{k}{2}I_T$. The first inequality follows from lemma 1. The second and fourth follow from $w(\text{reg}_L - c_l - c_r + w) \leq 0$. The third follows from $\text{reg}_L \leq I_L$. The fifth follows from $\text{reg}_T \leq \frac{k}{2}$ for $k \geq 2$.

The proofs for the remaining three cases ($c_T = c_l < c_r$, $c_T = c_l > c_r$, and $c_l > c_r = c_T$) use the same machinery as the proof above.

*Case 2:* $T$ outputs $l$, and $c_l < c_r$. In this case $\text{reg}_T = \text{reg}_L = c_l - c^*$. The left hand side can be rewritten as $\text{reg}_T\, S_T = \text{reg}_L(S_R + S_L + c_r - c_l) = \text{reg}_L\, S_L + \text{reg}_L(S_R + c_r - c_l) \leq \text{reg}_L\left(I_L + I_R - 2c_l + \frac{k}{2}\right) \leq I_R + \text{reg}_L\left(I_L - 2c_l + \frac{k}{2}\right) \leq I_R + I_L\left(\text{reg}_L - 2c_l + \frac{k}{2}\right) \leq I_R + \frac{k}{2}I_L \leq \frac{k}{2}I_T$. The first inequality from the lemma, the second from $\text{reg}_L \leq 1$, the third from $\text{reg}_L \leq I_L$, the fourth from $-c_L - c^* < 0$, and the fifth because $I_T = I_L + I_R$.

*Case 3:* $T$ outputs $l$, and $c_l > c_r$. We have $\text{reg}_T = \text{reg}_L = c_l - c^*$. The left hand side can be written as

$$\begin{aligned}\text{reg}_T S_T = \text{reg}_L(S_R + S_L + c_l - c_r) &\leq \frac{|L|}{2}I_L + \text{reg}_L\left(I_R + \frac{k - |L|}{2} - c_r + c_l - c_r\right)\\ &\leq \frac{k}{2}I_L + I_R + (c_l - 2c_r) \leq \frac{k}{2}(I_L + I_R + (c_l - c_r)) = \frac{k}{2}I_T,\end{aligned}$$

The first inequality follows from the inductive hypothesis and the lemma, the second from $\text{reg}_L < 1$ and $\text{reg}_L < I_L$, and the third from $c_r > 0$ and $k/2 > 1$.

*Case 4:* $T$ outputs $r$, and $c_l > c_r$. Let $w = c_l - c_r$. We have $\text{reg}_T = c_r - c^* = \text{reg}_L - w$. The left hand side can be written as

$$\begin{aligned}\text{reg}_T S_T &= (\text{reg}_L - w)(S_R + S_L + w)\\ &= \text{reg}_L S_L - wS_L + (\text{reg}_L - w)(S_R + w)\\ &\leq \frac{|L|}{2}I_L - w\left(I_L + \frac{|L|}{2} - c_l\right) + (\text{reg}_L - w)\left(I_R + c_l - 2c_r + \frac{k - |L|}{2}\right)\\ &\leq \frac{|L|}{2}I_L - w\left(I_L + \frac{|L|}{2} - c_l\right) + (I_L - w)\frac{k - |L|}{2} + (\text{reg}_L - w)\left(I_R + c_l - 2c_r\right)\\ &\leq \frac{k}{2}(I_L + I_R) - w\frac{k}{2} - w(I_L - c_l) + (\text{reg}_L - w)(c_l - 2c_r).\end{aligned}$$

The first inequality follows from the inductive hypothesis and the lemma, the second from $\text{reg}_L \leq I_L$, and the third from $\text{reg}_L \leq \frac{k}{2}$.

The last three terms are upper bounded by $-w - w\text{reg}_L + wc_l + \text{reg}_L c_l - 2c_r \text{reg}_L - wc_l + 2wc_r \le -w - \text{reg}_L(c_r + c_l) + \text{reg}_L c_l + 2wc_r \le -w - (c_l - c^*)c_r + wc_r + (c_l - c_r)c_r \le 0,$ and thus can be ignored, yielding $\text{reg}_T S_T \le \frac{k}{2}(I_L + I_R) = \frac{k}{2} I_T$, which completes the proof. Taking the expectation over $(x, \boldsymbol{c})$ completes the proof. ■

### 3.2 Lower Bound

The following simple example shows that the theorem is essentially tight in the worst case.

Let $k$ be a power of two, and let every label have cost 0 if it is is even, and 1 otherwise. The tree structure is a complete binary tree of depth $\log k$ with the nodes being paired in the order of their labels. Suppose that all pairwise classifications are correct, except for class $k$ wins all its $\log k$ games leading to cost-sensitive multiclass regret 1. If $T$ is the resulting filter tree, we have $\text{reg}_T = 1$, $S_T = \frac{k}{2} + \log k - 1$, and $I_T = \log k$, leading to the reget ratio of $\frac{\text{reg}_T S_T}{I_T} \le \frac{k/2 + \log k - 1}{\log k} = \Omega(\frac{k}{2 \log k})$, almost matching the theorem's bound of $\frac{k}{2}$ the regret ratio.

## 4 Error-Correcting Tournaments

In this section we first state and then analyze error correcting tournaments. As this section builds on the previous section, understanding the previous should be considered prerequisite for reading this section. For simplicity, we work with only the multiclass case. An extension for cost-sensitive multiclass problems is possible using the importance weighting techniques of the previous section.

### 4.1 Algorithm Description

An error-correcting tournament is one of a family of $m$-elimination tournaments where $m$ is a natural number. An *$m$-elimination tournament* operates in two phases. The first phase consists of $m$ single-elimination tournaments over the $k$ labels where a label is paired against another label at most once per round. Consequently, only one of these single elimination tournaments has a simple binary tree structure—see for example Figure 2 for an $m = 3$ elimination tournament on $k = 8$ labels. There is substantial freedom in exactly how the pairings of the first phase are done—our bounds are dependent on the depth of any mechanism which pairs labels in $m$ distinct single elimination tournaments. One such explicit mechanism is stated in [5]. Note that once an $(x, y)$ example has lost $m$ times, it is eliminated and no longer influences training at the nodes closer to the root.

The second phase is a final elimination phase, where we select the winner from the $m$ winners of the first phase. It consists of a redundant single-elimination tournament, where the degree of redundancy increases as the root is approached. To quantify the redundancy, let every subtree $Q$ have a *charge $c_Q$* equal to the number of leaves under the subtree. First phase winners at the leaves of final elimination tournament have charge 1. For any non-leaf node comparing subtree $R$ to subtree $L$, the importance weight of a binary example is set to $\max\{c_R, c_L\}$. For reference, in tournament applications, an importance weight can be expressed by playing games repeatedly where the winner of $R$ must beat the winner of $L$ $c_L$ times to advance, and vice versa.

One complication arises: what happens when the two labels compared are the same? In this case, the importance weight is set to $0$, indicating there is no preference in the pairing amongst the two choices.

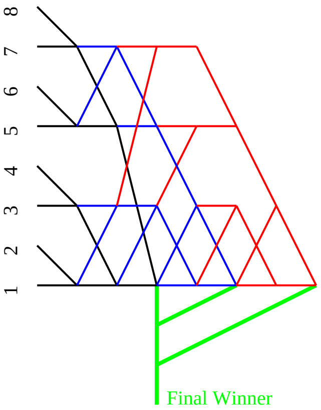

**Fig. 2.** An example of a 3-elimination tournament on $k = 8$ players. There are $m = 3$ distinct single elimination tournaments in first phase—one in black, one in blue, and one in red. After that, a final elimination phase occurs over the three winners of the first phase. The final elimination tournament has an extra weighting on the nodes, detailed in the text.

### 4.2 Error Correcting Tournament Analysis

A key concept throughout this section is the *importance depth*, defined as the worst-case length (number of games) of the overall tournament, where importance-weighted matches in the final elimination phase are played as repeated games. In Theorem 6 we prove a bound on the importance depth.

The computational bound per example is essentially just the importance depth.

**Theorem 4.** (Structural Depth Bound) *For any $m$-elimination tournament, the training and test computation is $O(m + \ln k)$ per example.*

**Proof:** The proof is by simplification of the importance depth bound (theorem 6), which bounds the sum of importance weights at all nodes in the circuit.

To see that the importance depth controls the computation, first note that the importance depth bounds the circuit depth since all importance weights are at least 1. At training time, any one example is used at most once per circuit level starting at the leaves. At testing time, an unlabeled example can have its label determined by traversing the structure from root to leaf. ∎

### 4.3 Regret analysis

Our regret theorem is the analogue of corollary 1 for error-correcting tournaments. Using the one classifier trick detailed there, the reduction transforms a multiclass distribution $D$ into an induced distribution $\mathrm{ECT}(D)$ over binary labeled examples. Let $f_{\mathrm{ECT}}$ denote the multiclass predictor induced by a binary classifier $f$.

It is useful to have the notation $\lceil m \rceil_2$ for the smallest power of 2 larger than or equal to $m$.

**Theorem 5.** (Main Theorem) *For all distributions $D$ over $k$-class examples, all binary classifiers $f$, all $m$-elimination tournaments* ECT, *the ratio of* $\mathrm{reg}(f_{\mathrm{ECT}}, D)$ *to* $\mathrm{reg}(f, \mathrm{ECT}(D))$ *is upper bounded by*

$$\begin{cases} 2 + \frac{\lceil m \rceil_2}{m} + \frac{k}{2m} & \textit{for all } m \geq 2 \textit{ and } k > 2 \\ 4 + \frac{2 \ln k}{m} + 2\sqrt{\frac{\ln k}{m}} & \textit{for all } k \leq 2^{62} \textit{ and } m \leq 4 \log_2 k \end{cases}$$

The first case shows that a regret ratio of 3 is achievable for very large $m$. The second case is the best bound for cases of common interest. For $m = 4 \ln k$ it gives a ratio of 5.5.

**Proof:** The proof holds for each input $x$, and hence in expectation over $x$. For a fixed $x$, we can define the regret of any label $y$ as $r_y = \max_{y' \in \{1,\cdots,k\}} D(y' \mid x) - D(y \mid x)$.

A node $n$ comparing two labels $y$ and $y'$ has regret $r_n$, which is $|D(y' \mid x) - D(y \mid x)|$ if the most probable label is not predicted, and 0 otherwise. The regret of a tree $T$ is defined as $r_T = \sum_{n \in T} r_n$.

The first part of the proof is by induction on the tree structure $F$ of the final phase. The invariant for a subtree $Q$ of $F$ won by label $q$ is $c_Q r_q \leq r_Q + \sum_{w \in \Gamma(Q)} r_{T_w}$, where $w$ is the winner of the first phase single-elimination tournament $T_w$.

When $Q$ is a leaf $w$ of $F$, we have $c_Q r_q = r_q \leq r_{T_i}$, where the inequality is from Corollary 1 noting that $d$ times the average binary regret is the sum of binary regrets.

Assume inductively that the hypothesis holds at node $n$ for the right subtree $R$ and the left subtree $L$ of $Q$ with respective winners $q$ and $l$: $c_R r_q \leq r_R + \sum_{w \in \Gamma(R)} r_{T_w}$ and $c_L r_l \leq r_L + \sum_{w \in \Gamma(L)} r_{T_w}$. Now, a chain of inequalities holds, completing the induction: $r_Q + \sum_{w \in \Gamma(Q)} r_{T_w} \geq c_L r_n + r_R + r_L + \sum_{w \in \Gamma(R)} r_{T_w} + \sum_{w \in \Gamma(L)} r_{T_w} \geq c_L r_n + c_R r_q + c_L r_l \geq c_Q r_q$. Here the first inequality uses the fact that the adversary must pay at least $c_L r_n$ to make $q$ win. The second inequality follows by the inductive hypothesis. The third inequality comes from $r_l + r_n \geq r_q$. To finish the proof, $m\,\mathrm{reg}(f_{\mathrm{ECT}}, D \mid x) = c_F r_f \leq r_F + \sum_{w \in \Gamma(F)} r_{T_w} \leq d\,\mathrm{reg}(f, \mathrm{ECT}(D \mid x))$, where $d$ is the maximum importance depth and the last quantity follows from the folk theorem in [19]. Applying the importance depth theorem 6 and algebra complete the proof. ∎

The depth bound follows from the following three lemmas.

**Lemma 2.** (First Phase Depth bound) *The importance depth of the first phase tournament is bounded by the minimum of*

$$\begin{cases} \lceil \log_2 k \rceil + m \lceil \log_2(\lceil \log_2 k \rceil + 1) \rceil \\ 1.5 \lceil \log_2 k \rceil + 3m + 1 \\ \lceil \frac{k}{2} \rceil + 2m \\ \textit{For } k \leq 2^{62} \textit{ and } m \leq 4 \log_2 k,\ 2(m-1) + \ln k + \sqrt{\ln k}\sqrt{\ln k + 4(m-1)}. \end{cases}$$

**Proof:** The depth of the first phase is bounded by the classical problem of robust minimum finding with low depth. The first three cases hold because any such construction upper bounds the depth of an error correcting tournament, and one such construction has these bounds [5].

For the fourth case, we construct the depth bound by analyzing a continuous relaxation of the problem. The relaxation allows the number of labels remaining in each single elimination tournament of the first phase to be broken into fractions. Relative to this version, the actual problem has two important discretizations:

1. When a single-elimination tournament has only a single label remaining, it enters the next single elimination tournament. This can have the effect of *decreasing* the depth compared to the continuous relaxation.
2. When a single-elimination tournament has an odd number of labels remaining, the odd label does not play that round. Thus the number of players does not quite halve, potentially *increasing* the depth compared to the continuous relaxation.

In the continuous version, tournament $i$ on round $d$ has $\frac{\binom{d}{i-1}k}{2^d}$ labels, where the first tournament corresponds to $i = 1$. Consequently, the number of labels remaining in any of the tournaments is $\frac{k}{2^d}\sum_{i=1}^{m}\binom{d}{i-1}$. We can get an estimate of the depth by finding the value of $d$ such that this number is 1.

This value of $d$ can be found using the Chernoff bound. The probability that a coin with bias $1/2$ has $m-1$ or fewer heads in $d$ coin flips is bounded by $m^{-2d\left(\frac{1}{2}-\frac{m-1}{d}\right)^2}$, and the probability that this occurs in $k$ attempts is bounded by $k$ times that. Setting this value to 1, we get $\ln k = 2d\left(\frac{1}{2}-\frac{m-1}{d}\right)^2$. Solving the equation for $d$, gives $d = 2(m-1) + \ln k + \sqrt{4(m-1)\ln k + (\ln k)^2}$. This last formula was verified computationally for $k < 2^{62}$ and $m < 4\log_2 k$ by discretizing $k$ into factors of 2 and running a simple program to keep track of the number of labels in each tournament at each level. For $k \in \{2^{l-1}+1, 2^l\}$, we used a pessimistic value of $k = 2^{l-1}+1$ in the above formula to compute the bound, and compared it to the output of the program for $k = 2^l$. ■

**Lemma 3.** (Second Phase Depth Bound) *In any $m$-elimination tournament, the second phase has importance depth at most $\lceil m \rceil_2 - 1$ rounds for $m > 1$.*

**Proof:** When two labels are compared in round $i \geq 1$, the importance weight of their comparison is at most $2^{i-1}$. Thus we have $\sum_{i=1}^{\lceil \log_2 m \rceil - 1} 2^{i-1} + \lfloor m \rfloor_2 = \lceil m \rceil_2 - 1$. ■

Putting everything together gives the importance depth theorem.

**Theorem 6.** (Importance Depth Bound) *For all $m$-elimination tournaments, the importance depth is upper bounded by*

$$\begin{cases} \lceil \log_2 k \rceil + m\lceil \log_2(\lceil \log_2 k \rceil + 1) \rceil + \lceil m \rceil_2 \\ 1.5\lceil \log_2 k \rceil + 3m + \lceil m \rceil_2 \\ \left\lceil \frac{k}{2} \right\rceil + 2m + \lceil m \rceil_2 \\ \textit{For } k \leq 2^{62} \textit{ and } m \leq 4\log_2 k\textit{, } 2m + \lceil m \rceil_2 + 2\ln k + 2\sqrt{m \ln k}. \end{cases}$$

**Proof:** We simply add the depths of the first and second phases from Lemmas 2 and 3. For the last case, we bound $\sqrt{\ln k + 4(m-1)} \leq \sqrt{\ln k} + 2\sqrt{m}$ and eliminate subtractions in Lemma 3. ■

# 5 Lower Bound

All of our lower bounds hold for a somewhat more powerful adversary which is more natural in a game playing tournament setting. In particular, we disallow reductions which use importance weighting on examples, or equivalently, all importance weights are set to 1. Note that we can modify our upper bound to obey this constraint by transforming final elimination comparisons with importance weight $i$ into $2i - 1$ repeated comparisons and use the majority vote. This modified construction has an importance depth which is at most $m$ larger implying the ratio of the adversary and the reduction's regret increases by at most 1.

The first lower bound says that for any reduction algorithm $B$, there exists an adversary $A$ with the average per-round regret $r$ such that $A$ can make $B$ incur regret $2r$ even if $B$ knows $r$ in advance. Thus an adversary who corrupts half of all outcomes can force a maximally bad outcome. In the bounds below, $f_B$ denotes the multiclass classifier induced by a reduction $B$ using a binary classifier $f$.

**Theorem 7.** *For any deterministic reduction $B$ from $k > 2$ classification to binary classification, there exists a choice of $D$ and $f$ such that* $\operatorname{reg}(f_B, D) \geq 2 \operatorname{reg}(f, B(D))$.

**Proof:** The adversary $A$ picks any two labels $i$ and $j$. All comparisons involving $i$ but not $j$, are decided in favor of $i$. Similarly for $j$. The outcome of comparing $i$ and $j$ is determined by the parity of the number of comparisons between $i$ and $j$ in some fixed serialization of the algorithm. If the parity is odd, $i$ wins; otherwise, $j$ wins. The outcomes of all other comparisons are picked arbitrarily.

Suppose that the algorithm halts after some number of queries $c$ between $i$ and $j$. If neither $i$ nor $j$ wins, the adversary can simply assign probability $1/2$ to $i$ and $j$. The adversary pays nothing while the algorithm suffers loss 1, yielding a regret ratio of $\infty$.

Assume without loss of generality that $i$ wins. The depth of the circuit is either $c$ or at least $c + 1$, because each label can appear at most once in any round. If the depth is $c$, then since $k > 2$, some label is not involved in any query, and the adversary can set the probability of that label to 1 resulting in $\rho(B) = \infty$.

Otherwise, $A$ can set the probability of label $j$ to be 1 while all others have probability 0. The total regret of $A$ is at most $\lfloor \frac{c+1}{2} \rfloor$, while the regret of the winning label is 1. Multiplying by the depth bound $c + 1$, gives a regret ratio of at least 2. ■

Note that the number of rounds in the above bound can depend on $A$. Next, we show that for any algorithm $B$ taking the same number of rounds for any adversary, there exists an adversary $A$ with a regret of roughly one third, such that $A$ can make $B$ incur the maximal loss, even if $B$ knows the power of the adversary.

**Lemma 4.** *For any deterministic reduction $B$ to binary classification with number of rounds independent of the query outcomes, there exists a choice of $D$ and $f$ such that* $\operatorname{reg}(f_B, D) \geq (3 - \frac{2}{k}) \operatorname{reg}(f, B(D))$.

**Proof:** Let $B$ take $q$ rounds to determine the winner, for any set of query outcomes. We will design an adversary $A$ with incurs regret $r = \frac{qk}{3k-2}$, such that $A$ can make $B$ incur the maximal loss of 1, even if $B$ knows $r$.

The adversary's query answering strategy is to answer consistently with label 1 winning for the first $\frac{2(k-1)}{k} r$ rounds, breaking ties arbitrarily. The total number of queries that $B$ can ask during this stage is at most $(k-1)r$ since each label can play at most once in every round, and each query occupies two labels. Thus the total amount of regret at this point is at most $(k-1)r$, and there must exist a label $i$ other than label $k$ with at most $r$ losses. In the remaining $q - \frac{2(k-1)}{n}r = r$ rounds, $A$ answers consistently with label $i$ and all other skills being 0.

Now if $B$ selects label 1, $A$ can set $D(i \mid x) = 1$ with $r/q$ average regret from the first stage. If $B$ selects label $i$ instead, $A$ can choose that $D(1 \mid x) = 1$. Since the number of queries between labels $i$ and $k$ in the second stage is at most $r$, the adversary can incurs average regret at most $r/q$. If $B$ chooses any other label to be the winner, the regret ratio is unbounded. ▮

## References

1. M. Adler, P. Gemmell, M. Harchol-Balter, R. Karp, and C. Kenyon. Selection in the presence of noise: The design of playoff systems, SODA 1994.
2. E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers, *Journal of Machine Learning Research*, 1: 113–141, 2000.
3. J. Aslam and A. Dhagat. Searching in the presence of linearly bounded errors, STOC 1991.
4. R. Borgstrom, S. Rao Kosaraju. Comparison-base search in the presence of errors, STOC 1993.
5. P. Denejko, K. Diks, A. Pelc, and M. Piotr'ow. Reliable minimum finding comparator networks, *Fundamenta Informaticae*, 42: 235–249, 2000.
6. T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, 2: 263–286, 1995.
7. U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information, *Symposium on Theory of Computing*, 128–137, 1990.
8. D. Foster and D. Hsu, `http://hunch.net/?p=468`.
9. J. Fox. *Applied regression analysis, linear models, and related methods*, Sage Publications, 1997.
10. V. Guruswami and A. Sahai. Multiclass learning, Boosting, and Error Correcting Codes, COLT 1999.
11. T. Hastie and R. Tibshirani. Classification by pairwise coupling, NIPS 1997.
12. R. Herbrich, T. Minka, and T. Graepel. TrueSkill(TM): A Bayesian skill rating system, NIPS 2007.
13. D. Hsu, J. Langford, S. Kakade, and T. Zhang, Multi-label prediction via compressed sensing, arXiv:0902.1284v1, 2009.
14. J. Langford and A. Beygelzimer. Sensitive Error Correcting Output Codes, COLT 2005.
15. J. Langford and B. Zadrozny, Estimating class membership probabilities using classifier learners, AISTAT 2005.
16. B. Ravikumar, K. Ganesan, and K. B. Lakshmanan. On selecting the largest element in spite of erroneous information, Lecture Notes in Computer Science, 247: 88–99, 1987.
17. B. Williamson, personal communication.
18. A. C. Yao and F. F. Yao. On fault-tolerant networks for sorting. *SIAM Journal of Computing*, 14(1): 120–128, 1985.
19. B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting, ICDM 2003.