Error-Correcting Tournaments

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

[1] IBM Research, beygel@us.ibm.com
[2] Yahoo! Research, jl@yahoo-inc.com
[3] UC Berkeley, pradeepr@stat.berkeley.edu

Abstract. We present a family of adaptive pairwise tournaments that are provably robust against large error fractions when used to determine the largest element in a set. The tournaments use nk pairwise comparisons but have only $O(k + \log n)$ depth, where n is the number of players and k is the robustness parameter (for reasonable values of n and k). These tournaments also give a reduction from multiclass to binary classification in machine learning, yielding the best known analysis for the problem.

1 Introduction

Suppose we want to select the best among a set of competing players, each with an unknown skill level between 0 and 1. The simplest approach is to run a single-elimination tournament, where players pair up, play, and then winners pair up repeatedly until only one winner remains. This approach is common in practice, but it does not suit our paranoia—we want a system robust against random outcomes, biased refs, and bribing bettors. We investigate mechanisms and costs for allaying our paranoia.

Our goal is to design a robust tournament algorithm using pairwise comparisons between players to choose the best player. We measure robustness against an adversary who is charged according to the difference between the skills of the players she missorts, instead of the same amount for every missort. We do not insist that the algorithm returns the best player reliably, but no adversary should be able to make the algorithm return a player significantly worse than the best player, without paying a high cost.

The model considered here is directly motivated by a problem in machine learning, described in Section 4. We begin by reviewing previous directions of analysis that are related to our goal, and explain why they do not properly capture it.

Previous Approaches A common approach to analyzing tournaments is to use probabilistic assumptions about the outcomes of different pairings. For example, Feige et al. [8] assume that each outcome has a fixed probability of being erroneous, independently of other outcomes. Adler et al. [1] consider several probabilistic models (noise models), but all of them are based on the assumption that the best player beats any other player with probability greater than

half in any individual game, and all outcomes are independent. Herbrich and Graepel [11] model the performance of each player as a normally distributed random variable centered around the skill level; the outcome of each pairing is determined by the outcomes of the corresponding random variables. All these assumptions do not fit our understanding of what a bribing bookie or some other adversary is capable of.

In comparator-based selection networks [6, 14], the adversary can only fail to sort an unsorted pair (passing it on without any processing), but she cannot missort an already sorted pair. The adversary we are concerned with here could defeat such networks by simply missorting on the last comparison. Thus no tournament structure where one final comparison determines the winner is viable.

A simple way to avoid this problem is to use repeated comparisons, as in the best four-of-seven playoff in the World Series baseball championship. Ravikumar, Ganesan, and Lakshmanan [13] present a strategy which reliably finds the largest element in an n-element set with at most $(e+1)n-1$ comparisons, where e (known to the algorithm) is the number of times the adversary can lie. This algorithm does not perform well in our setting because we are concerned with the per-round *rate* of errors rather than the total number of errors, which strongly encourages small depth tournaments. In our model, the adversary is given a budget on allowed missorts *per round*. This is a constraint on budget rather than spending—the budget for all rounds can be used in one round, if the adversary desires, unlike in the prefix-bounded model, where the number of incorrect responses at no point can exceed some constant fraction of the number of queries [5, 3].

We also strengthen the adversary by allowing her to choose skill levels of players and charging her according to the difference in the skills for every missort (rather than charging the same amount per missort, as assumed by previous approaches).

The Problem More formally, a set of n players enter the tournament. The player corresponding to index $i \in \{1, \ldots, n\}$ is referred to as player i. Every player i has some skill level $s_i \in [0, 1]$. The games are played in successive rounds, and each player can participate in at most one game per round.

The winner of the tournament is determined through interaction between an *algorithm* B and an *adversary* A. In each round, B pairs players based on the outcomes of games in previous rounds. The outcomes are determined by the adversary A who also determines the skill levels (unknown to the algorithm).

The realized tournament $T = T(A, B)$ is a function of both B and A. The *loss* of B on T is given by $\ell_B(T) = \max_i s_i - s_w$, where w is the winner of T. The *cost* that adversary A pays on game (i, j) is $r(i, j) = s_l - \min\{s_i, s_j\}$, where $l \in \{i, j\}$ is the loser of the pair as determined by A. So the cost is 0 if A is truthful; otherwise, A pays the absolute difference between s_i and s_j. The *total cost* of A on T is defined as $\ell_A(T) = \sum_{(i,j)} r(i, j)$, where (i, j) ranges over the multiset of all pairs played in T. We judge the algorithm's success according to the following ratio.

Definition 1. *The* regret ratio *of a tournament algorithm B is the worst-case ratio over all adversaries A, of the loss of the resulting tournament $T = T(A, B)$ to the average per-round loss of A:*

$$\rho(B) = \max_A \frac{\ell_B(T)}{\ell_A(T)/d(T)},$$

where $d(T)$ is the depth (the number of rounds) of T.

We can think of the adversary as being penalized according to the average (over the rounds) of the loss from the missorts she induces. Thus the adversary has some per round budget, but there is no restriction on how this budget is used. One important property of our setting is that the tournament algorithm is oblivious to the loss budget of the (generally) unknown adversary. The performance of the tournaments presented here degrades gracefully as the adversary increases in power.

Simple Strategies that Do Not Work This section describes simples strategies that do not lead to a good regret ratio.

Pair players arbitrarily and eliminate any player after k losses The simplest example that shows that this strategy does not give a good regret ratio is when there are $n = 3$ players and $k = 2$. Suppose that the players have skills $s_1 = s_2 = 1, s_3 = 0$, and player 1 is paired with player 2 three times, resulting in the outcomes "1 wins, 2 wins, 1 wins," eliminating player 2. When player 3 is paired with player 1, and player 3 wins, player 1 is eliminated, resulting in player 3 winning with only one miscomparison, rather than two. Notice that this argument applies to a Swiss tournament, where players always face opponents with the same (or almost the same) number of wins.

Bracketing Approach Single, double, and triple-elimination tournaments are common in sports. The players are organized into *brackets*. Bracket i is a single-elimination tournament on all players except the winners of brackets $1, ..., i-1$. After the bracket winners are determined, the player winning bracket k plays the winner of bracket $k-1$ repeatedly until one player has suffered k total losses (they start with $k-1$ and $k-2$ losses respectively). The winner moves on to repeatedly pair against the winner of bracket $k-2$, and the process continues until only one player remains. This method does not scale well to large k, because the post-bracket phase takes $\sum_{i=1}^{k} i - 1 = O(k^2)$ rounds implying an adversary with a per-round spending budget has a power growing with k. Even for $n = 8$ and $k = 3$, our constructions have smaller maximum depth than the bracketed triple-elimination algorithm.

Make all games redundant We could construct a balanced binary tree over the set of players and repeat the pairing at each node $2k-1$ times. The resulting structure would have depth $(2k-1)\log_2 n$ and a regret ratio of at least $\log_2 n$, no better than the non-redundant single-elimination tournament.

Summary of Results We present a family of tournament algorithms parametrized by an integer $k \geq 1$ which controls the tradeoff between maximizing robustness (k large) and minimizing depth (k small). Setting $k = 1$, gives the familiar single-elimination tournament, with depth $\log_2 n$ and regret ratio of $\log_2 n$. Some other critical values of k are the following: If $k = 4 \ln n$, the construction gives regret ratio of 5.5 with depth $22 \ln n$. Setting $k = cn$, gives regret ratio of $4 + 1/c$ with depth $(4c + 1)n$.

One special case of interest is when one player has skill 1 while all other players have skill 0. In this special case, an adversary with a budget of $k - 1$ or fewer errors results in no loss ($\max_i s_i - s_w = 0$), while one with k or more errors can force maximal loss ($\max_i s_i - s_p = 1$).

We also analyze lower bounds on the regret ratio. A lower bound of 2 formalizes the fact that an adversary with the power to corrupt half of all outcomes can force a bad outcome. Since we allow skill levels, this lower bound applies even when the adversary's budget is small. When the depth of the tournament is independent (or nearly independent) of the game outcomes, we strengthen this lower bound to 3.

The tournament problem described here is also the problem of robustly reducing multiclass classification to binary classification in machine learning. Section 4 describes the problem and shows that the approach here gives the tightest known analysis for the problem.

2 Single Elimination

The $k = 1$ case corresponds to a single-elimination tournament where a player is knocked out after a single loss. The process continues until the sole remaining player is declared a winner. The proof for the $k = 1$ case is a useful lemma for the $k > 1$ case.

Different mechanisms for pairing players give different tournament structures. The multi-elimination case (Section 3) uses single-elimination structures other than maximally balanced binary trees, so we explicitly analyze all single-elimination tournaments here.

Lemma 1. (Single Elimination) *For all single-elimination algorithms B and adversaries A, $\ell_B(T) \leq \ell_A(T)$, where $T = T(A, B)$ is the resulting tournament.*

Proof: We use induction, starting at the leaves of T. By the induction hypothesis, the cost paid by A in any subtree of T bounds the loss of the subtree's output. Consider a subtree Q whose root corresponds to a game between players l and q, where player q wins according to A. Letting $\Gamma(Q)$ denote the set of leaves in Q, the loss of B on Q is given by $\ell_B(Q) = \max_{p \in \Gamma(Q)} s_p - s_q$. The hypothesis is trivially satisfied for trees with a single node. Inductively, assume that $\ell_A(L) \geq \ell_B(L)$ and $\ell_A(R) \geq \ell_B(R)$, for the left subtree L of Q providing l, and the right subtree R providing q.

There are two possibilities. Either the best label comes from the leaves of L or the leaves of R. The second possibility is easy since we have $\ell_B(Q) =$

$\max_{p \in \Gamma(R)} s_p - s_q = \ell_B(R) \leq \ell_A(R) \leq \ell_A(Q)$, as desired. For the first possibility, we have

$$\begin{aligned}\ell_B(Q) &= \max_{p \in \Gamma(L)} s_p - s_q = \max_{p \in \Gamma(L)} s_p - s_l + s_l - s_q \\ &= \ell_B(L) + s_l - s_q \leq r(l,q) + \ell_A(L) \leq \ell_A(Q),\end{aligned}$$

completing the proof. The inductive hypothesis for the root is $\ell_B(T) \leq \ell_A(T)$. ∎

The lemma above together with the definition of the regret ratio (and the fact that the single-elimination tournament algorithm produces a maximally balanced binary tree structure) imply the following corollary for $k = 1$.

Corollary 1. (Single Elimination) *For any $n > 1$, the regret ratio of the single-elimination tournament algorithm B on n players is $\rho(B) \leq \lceil \log_2 n \rceil$.*

3 Multiple Elimination: Error-Correcting Tournaments

Algorithm Description The *k-elimination algorithm* operates in two phases. (Throughout the paper, k is a positive natural number.) The first phase consists of k single-elimination tournaments over the n players. Since players can participate in at most one match per round, only one of these single elimination tournaments has a simple binary tree structure as in Figure 1.

In the conventional bracketed k-elimination tournament, the $(i+1)$-th tournament plays the losers of the i-th tournament. (The losers of round d of the i-th tournament play in round $(d+1)$ of the $(i+1)$-th tournament.) Thus, tournament i plays all players but the winners of brackets $1, \ldots, i-1$. Here, we make a crucial distinction: Both finalists of the i-th tournament play in the $(i+1)$-th tournament in the next round. The upshot of this is that each tournament plays *all* the players. As such, it functions more as a "parallelized" version of running k independent single elimination tournaments. The purpose of this first phase is to cull k "winners" from the n players, to feed into the second phase.

The second phase is a final elimination phase, where we select the winner from the k 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. Leaf nodes have charge 1. For a pair (l, q), where l is the winner of subtree L and q is the winner of subtree R, the charges c_R and c_L determine the number of times that player l or q, respectively, must beat the other player in order to win the subtree Q. Note that if $c_R = c_L$, as when the final phase structure is a balanced binary tree, this can be interpreted as a best c_R-of-$(2c_R - 1)$ outcome, as is familiar from many sports. An example for $k = 8$ is given in the rightmost subfigure of Figure 1.

Multiple Elimination Analysis We start with an analog of Lemma 1 for k-elimination tournaments.

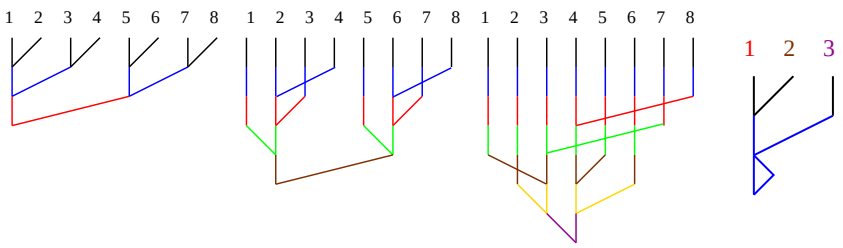

Fig. 1. The first three subfigures give an example of the first phase of a 3-elimination tournament on $n = 8$ players. Note that no player participates in more than one match per round, and that the precise pairing is dependent on who wins. The rightmost subfigure gives an example of the second phase for $k = 3$ (n is irrelevant) when the winner of the 3rd single elimination tournament is the overall winner. The structure is a single-elimination tournament with redundancy. In order to advance, a player must defeat its opponent as many times as there are leaves in the opponent's subtree.

Theorem 1. (k-elimination) *For all k-elimination algorithms B and adversaries A, $\ell_B(T) \leq \ell_A(T)/k$, where $T = T(A, B)$ is the resulting tournament.*

Proof: Define the *regret* of any player q as $\ell(q) = \max_{i\in\{1,\ldots,n\}} s_i - s_q$. The proof is by induction on the tree structure F of the final phase. The invariant for a subtree Q of F with winner q is

$$c_Q \ell(q) \leq \ell_A(Q) + \sum_{w_i \in \Gamma(Q)} \ell_A(T_i),$$

where w_i is the winner of the first phase single-elimination tournament T_i. When $Q = F$, we get the theorem statement. (The tournament T in the statement consists of tournaments $T_1, \ldots, T_k$, and F.)

When Q is a leaf w_i of F, we have $c_Q \ell(q) = \ell_B(T_i) \leq \ell_A(T_i)$, where the inequality is from Lemma 1.

Assume inductively that the hypothesis holds for the right subtree R and the left subtree L of Q with respective winners q and l:

$$c_R \ell(q) \leq \ell_A(R) + \sum_{w_i \in \Gamma(R)} \ell_A(T_i), \quad \text{and} \quad c_L \ell(l) \leq \ell_A(L) + \sum_{w_i \in \Gamma(L)} \ell_A(T_i).$$

In order for player q to win, it must have c_L wins over player l, which implies

$$\begin{aligned} \ell_A(Q) + \sum_{w_i \in \Gamma(Q)} \ell_A(T_i) &\geq c_L r(q,l) + \ell_A(R) + \ell_A(L) + \sum_{w_i \in \Gamma(R)} \ell_A(T_i) + \sum_{w_i \in \Gamma(L)} \ell_A(T_i) \\ &\geq c_L r(q,l) + c_R \ell(q) + c_L \ell(l) \geq c_Q \ell(q), \end{aligned}$$

completing the induction. Here the first inequality uses the fact that the adversary must pay at least $c_L r(q, l)$ to make q win. The second inequality follows by the inductive hypothesis. The third inequality comes from $\ell(l) + r(q, l) \geq \ell(q)$. ∎

To bound the regret ratio, we need to bound the depth of the resulting tournaments. We first deal with the simple case when $n = 2$.

Case 1. For $n = 2$ and $k \geq 1$, the regret ratio of a k-elimination tournament algorithm B is $\rho(B) \leq 2$.

Proof: If $n = 2$, there are k matches in the first phase and at most k in the second, so the total number of rounds is at most $2k$. Combining with Theorem 1 and the definition of $\rho(B)$, gives the theorem. ∎

In the general case, we have the following theorem. Part 1 shows that a regret ratio of 4 is achievable for very large k. Part 2 gives a better bound for all conceivable cases of practical interest.

Theorem 2. (Main Theorem)

Part 1. *For all $k \geq 2$ and $n > 2$, the regret ratio of a k-elimination tournament B on n players is $\rho(B) \leq 3 + \frac{\lceil k \rceil_2}{k} + \frac{n}{k}$, where $\lceil k \rceil_2$ is the smallest power of two greater than or equal to k. If k is a power of 2, the bound is $\rho(B) \leq 4 + n/k$.*

Part 2. *For all $n \leq 2^{62}$ and $k \leq 4 \log_2 n$,*

$$\rho(B) = 4 + \frac{2 \ln n}{k} + 2\sqrt{\frac{\ln n}{k}}.$$

Proof: The proof is a conjunction of Theorem 1 and the depth bound (Theorem 3) below. ∎

The depth bound follows from the following three lemmas.

Lemma 2. (First Phase Depth bound, Part 2) *For $n \leq 2^{62}$ and $k \leq 4 \log_2 n$, the first phase has depth at most $2(k-1) + \ln n + \sqrt{\ln n}\sqrt{\ln n + 4(k-1)}$.*

Proof: A precise depth bound for the general case is complicated by discrete effects. A similar problem, finding the number of rounds for a circuit in the half-lie case of the noisy max is known to be hard [6]. We construct the depth bound by analyzing a continuous relaxation of the problem. The relaxation allows players to be broken into fractions. Relative to this version, the actual problem has two important discretizations:

1. When a single-elimination tournament has only one player, that player can go and play in a different single-elimination tournament. This can have the effect of *decreasing* the depth.
2. When a single-elimination tournament has an odd number of players, the odd player does not play in the tournament. Thus the number of players does not quite halve, potentially *increasing* the depth.

In the continuous version, tournament i on round d has $\binom{d}{i-1} n/2^d$ players, where the first tournament corresponds to $i = 1$. Consequently, the number of players remaining in any of the tournaments is $\frac{n}{2^d} \sum_{i=1}^{k} \binom{d}{i-1}$. 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 $k-1$ or fewer heads in d coin flips is bounded by $e^{-2d\left(\frac{1}{2}-\frac{k-1}{d}\right)^2}$, and the probability that this occurs in n attempts is bounded by n times that. Setting this value to 1, we get $\ln n = 2d\left(\frac{1}{2}-\frac{k-1}{d}\right)^2$. Solving the equation for d, gives $d = 2(k-1)+\ln n+\sqrt{4(k-1)\ln n+(\ln n)^2}$. This last formula was verified computationally for $n < 2^{62}$ and $k < 4\log_2 n$ by discretizing n into factors of 2 and running a simple program to keep track of the number of players in each tournament at each level. For $n \in \{2^{l-1}+1, 2^l\}$, we used a pessimistic value of $n = 2^{l-1}+1$ in the above formula to compute the bound, and compared it to the output of the program for $n = 2^l$. ∎

Lemma 3. (First Phase Depth Bound, Part 1) *For all adversaries A, the depth of the k-th tournament is at most* $2k+n$.

Proof: For any $n > 2$ and any number of rounds, the sequence of k numbers giving the current number of players in each of the k single-elimination tournaments has the form

$$0^*(2+)^*(1+)0^*,$$

where $(x+)$ is a shortcut for x or more players, and x^* means that x occurs zero or more times.

The proof is by induction. We start with the sequence $n0^{k-1}$, when all players play in the first tournament.

Inductively, assume that the sequence is $0^*(2+)^*(1+)0^*$. For the left-most (2+), there are two cases: the number of players in the corresponding tournament is either 2 or (3+). In the first case, the entry becomes a 0; otherwise, it remains (2+). All other (2+) entries remain (2+), because at least one player is left, and one player propagates from the left. For the rightmost non-zero entry, the number of players is either 1 or (2+). No play happens in the first case, and the entry becomes (2+) due to a propagation from the left. In the second case, it remains (2+), but one or more players lose to other players and move to the next tournament forming a new (1+).

Starting from a sequence of the form $0^*(2+)^*(1+)0^*$, at least one new tournament is entered every two rounds, implying that the round of the first entry into tournament k is at most $2k$. After the first round of entry, at least one player loses every round, until no players remain in the tournament. Consequently, any tournament is active for at most n rounds, implying the bound. ∎

It is useful to have the notation $\lceil k \rceil_2$ for the smallest power of 2 larger than or equal to k. Similarly, let $\lfloor k \rfloor_2$ round down to the largest power of 2 smaller than or equal to k. If k is not a power of 2, we have $2\lfloor k \rfloor_2 = \lceil k \rceil_2$.

Lemma 4. (Second Phase Depth Bound) *In any realized k-elimination tournament, the second phase has depth at most* $k + \lceil k \rceil_2 - \lceil \log_2 k \rceil - 2$ *rounds for* $k > 1$.

Proof: When two players are compared at level $i \geq 1$, they play at most $2^i - 1$ times before one beats the other 2^{i-1} times and wins the level. Thus when k is a power of 2, the depth of the final phase is bounded by

$$\sum_{i=1}^{\log_2 k} (2^i - 1) = -\log_2 k + \sum_{i=1}^{\log_2 k} 2^i = -\log_2 k + 2k - 2,$$

If k is not a power of 2, the depth is maximized when one child of the root is a perfect binary tree with $\lfloor k \rfloor_2$ leaves. The other child has size $k - \lfloor k \rfloor_2$, implying that the root playoff adds depth at most $k - 1$. The final depth is thus $k - 1 + 2\lfloor k \rfloor_2 - \lceil \log_2 k \rceil - 2 = k + \lceil k \rceil_2 - \lceil \log_2 k \rceil - 2$ when k is not a power of 2, and $2k - \log_2 k - 2$ otherwise. In both cases, the theorem holds. ∎

Putting everything together gives the theorem.

Theorem 3. (Depth Bound) *For any realized k-elimination tournament, the number of rounds is at most*

$$\begin{cases} 3k + \lceil k \rceil_2 + 2\ln n + 2\sqrt{k \ln n} & n \leq 2^{62},\ k \leq 4\log_2 n \\ 3k + \lceil k \rceil_2 + n & n > 2,\ k \geq 2 \end{cases}$$

Proof: The proof uses Lemmas 2, 3, and 4. For the $n \leq 2^{62}, k \leq 4\log_2 n$ case, we bound $\sqrt{\ln n + 4(k-1)} \leq \sqrt{\ln n} + 2\sqrt{k}$ and eliminate subtractions in Lemma 4. For the general case, we simply add the depths of the first and second phases from Lemmas 3 and 4. ∎

4 Application to Learning

The tournament construction described here is useful for solving the problem of reducing multiclass classification to binary classification in machine learning.

In multiclass classification there are n labels, and the goal is to choose the most likely label, given features based on examples of previously classified instances. Binary, or two-class classification, is a well-studied special case with many efficient algorithms. A common approach to solving multiclass problems is to create meta-algorithms, which reduce the multiclass problem to a set of binary classification problems. Solutions to these binary problems are then used to infer multiclass labels. A question of interest in such settings is: "How do misclassifications in the binary problems affect the stitched multiclass solution?" The following correspondence maps this setting to the tournament problem:

1. Labels correspond to players.
2. The probability that the label is correct, conditioned on the observed features, defines the skill of the corresponding player.
3. The adversary corresponds to the learned binary classifiers.
4. The adversary controls all binary classifications, and her cost is the difference between her error rate and the smallest achievable error rate on the same binary problem.

5. The difference between the probability of the output label and the probability of the most likely label is the loss of the tournament (i.e., multiclass learning algorithm).

Several adversary design choices are motivated by this machine learning application. For example, it is critical in the machine learning application that the algorithm be oblivious to the adversary's budget (since some examples are much harder than others), that errors are adversarial rather than probabilistic (because predictions are correlated by the shared features), and that the adversary has a budget growing with the depth (because more comparisons imply more opportunities for mistakes to be made).

A desirable property of such reductions is their consistency: if the binary classification problems are solved optimally, the reduction should yield an optimal multiclass classifier. Known consistent methods are inadequate because they have $n-1$ rounds for every label [10], use more information than a single bit per pairing [2].

Perhaps the best known reduction result uses error-correcting output codes (ECOC) to predict the most likely label [7, 12]. There are two substantial drawbacks of the ECOC approach, which the tournament approach addresses.

1. Tournament reductions have a loss which is linear in the adversary's per round budget. The best ECOC approach has a loss which scales as the square root of the adversary's budget [12].
2. Operations performed by ECOC correspond to dividing the players into groups and playing one team against another. The tournament approach always compares only pairs of classes. It is easy to construct examples of multiclass learning problems where pairs of classes are separable by a hyperplane, but subsets of the classes are not.

Further details completely covering the single elimination tournament case are presented in a technical report (not intended as a separate publication) [4].

5 Lower Bound

The first lower bound says that for any algorithm B, there exists an adversary A with the average per-round error r such that A can make B incur loss $2r$ even if B knows the error budget. Thus an adversary who corrupts half of all outcomes can force a maximally bad outcome.

Theorem 4. *For any tournament algorithm B on $n > 2$ players there exists an adversary A such that for any per round budget $r \leq 1/d(T)$ the regret ratio is $\rho(B) \geq 2$.*

Proof: The adversary A picks any two players 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 $s_i = s_j = 1$ while setting all other skills to 0. The adversary pays nothing while the algorithm suffers loss 1, yielding $\rho(B) = \infty$.

Assume without loss of generality that i wins. The depth of the circuit is either c or at least $c + 1$, because each player can appear at most once in any round. If the depth is c, then since $n > 2$, some node is not involved in any pairing, and A can set the skill of that node to 1 while setting all other skills to 0, resulting in $\rho(B) = \infty$.

Otherwise, A sets $s_j = 2r$ while all other skills are 0. The total cost of A is at most $r(c + 1)$, while B pays $2r$. Multiplying by the depth bound $c + 1$, gives the 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 an error rate of roughly a third, such that A can make B incur the maximal loss, even if B knows the error rate.

Lemma 5. *For any tournament algorithm B with the number of rounds independent of the pairing outcomes, $\rho(B) \geq 3 - \frac{2}{n}$.*

Proof: Let B take q rounds to determine the winner, for any set of pairing outcomes. We design an adversary A with error budget $e = \frac{qn}{3n-2}$, such that A can make B incur the maximal loss of 1, even if B knows e.

The adversary's pairing outcome strategy is to answer consistently with $s_n = 1$, $s_1 = \cdots = s_{n-1} = 0$ for the first $\frac{2(n-1)}{n}e$ rounds, breaking ties arbitrarily. The total number of queries that B can ask during this stage is at most $(n-1)e$ since each player can play at most once in every round, and each pairing occupies two players. Thus the total number of losses at this point is at most $(n-1)e$, and there must exist a player i other than player n with at most e losses. In the remaining $q - \frac{2(n-1)}{n}e = e$ rounds, A answers consistently with $s_i = 1$ and all other skills being 0.

If B selects s_n as the winner, A can set $s_i = 1$ and $s_n = 0$, implying the e losses of player i in the first stage were erroneous giving a regret ratio of $\rho(B) \geq q/e = (3n - 2)/n$.

If B selects s_i instead, A can set $s_n = 1$ and $s_i = 0$. Since the number of queries between players i and n in the second stage is at most e A pays for at most e errors implying a regret ration of $\rho(B) \geq q/e = (3n - 2)/n$.

If B claims any other player to be the winner, it suffers the maximal loss of 1 on either of the two skill sets implying a regret ratio of ∞. ∎

6 Conclusion

We introduced the family of Error Correcting Tournaments and proved that they have a constant regret ratio against an adversary more powerful than any previously analyzed for tournaments. These tournaments are practical for games since they are optimized for small depth.

The results also have an immediate application to reducing multiclass classification to binary classification in machine learning. For more details about the basic constructions and experiments, see [4]. (The report has not been published and is not intended as a separate publication.)

References

1. M. Adler, P. Gemmell, M. Harchol-Balter, R. Karp, C. Kenyon. Selection in the Presence of Noise: The Design of Playoff Systems, *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 564–573, 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. A. Beygelzimer, J. Langford, and P. Ravikumar. Multiclass Classification with Filter Trees, available at `http:\\hunch.net\~jl/reductions/mc\_to\_b/invertedTree.pdf`.
5. R. Borgstrom, S. Rao Kosaraju. Comparison-base search in the presence of errors, STOC 1993.
6. P. Denejko, K. Diks, A. Pelc, M. Piotr'ow. Reliable Minimum Finding Comparator Networks, Fundamenta Informaticae, 42: 235–249, 2000.
7. T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, 2: 263–286, 1995.
8. U. Feige, D. Peleg, P. Raghavan and E. Upfal. Computing with unreliable information, *Symposium on Theory of Computing*, 128–137, 1990.
9. V. Guruswami and A. Sahai. Multiclass Learning, Boosting, and Error Correcting Codes, COLT 1999.
10. T. Hastie and R. Tibshirani. Classification by Pairwise Coupling, NIPS 1997.
11. R. Herbrich, T. Minka, and T. Graepel. TrueSkill(TM): A Bayesian Skill Rating System, NIPS 2007.
12. J. Langford and A. Beygelzimer. Sensitive Error Correcting Output Codes, COLT 2005.
13. 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.
14. A. C. Yao and F. F. Yao. On fault-tolerant networks for sorting. *SIAM Journal of Computing*, 14(1): 120–128, 1985.