

---

# Can We Learn to Gamble Efficiently?

---

Jacob Abernethy

UC Berkeley

jake@eecs.berkeley.edu

## 1 Introduction

Betting is an important problem faced by millions of sports fans each day. Presented with an upcoming matchup between team A and team B, and given the opportunity to place a 50/50 wager on either, where should a gambler put her money? This decision is not, of course, made in isolation: both teams will have played a number of decided matches with other teams throughout the season. Furthermore, a reasonable assumption to make is that the relation “A tends to beat B” is transitive. Under transitivity, the best prediction strategy is clearly to sort the teams by their abilities and predict according to this *ranking*.

The obvious difficulty is that the best ranking of the teams is not known in advance. But there’s a more subtle issue: even with knowledge of all match outcomes in advance, i.e. a list of items of the form (team  $X <$  team  $Y$ ), it’s NP-hard to determine the best ranking of the teams when the outcomes are noisy. This is exactly the infamous Minimum Feedback Arc Set problem.

The question we pose is as follows: can we design a non-trivial online prediction strategy in this setting which achieves *vanishing regret* relative to the best ranking in hindsight, even when the latter is computationally infeasible?

It is tempting to believe this is impossible, as competing with the best ranking would appear tantamount to finding the best ranking. However, this assertion is false: the algorithm *need not* learn a ranking explicitly, it must only output a prediction ( $X < Y$ ) or ( $Y < X$ ) when presented with a pair  $(X, Y)$ , and these predictions do not necessarily have to satisfy transitivity. Indeed, consider the following simple algorithm: treat each team pair  $(X, Y)$  as an independent learning problem (ignoring all other matchups). In the long run, this will achieve vanishing regret with respect to the best ranking. So why is this not desirable? The trivial approach unfortunately admits a bad regret bound: the algorithm must see  $O(n)$  matches per team before it can start to make decent predictions. On the other hand, there is an information-theoretic approach that requires only  $O(\log n)$  observations per team—the downside, unfortunately, is that this requires exponential computation. We would like to achieve this rate with an efficient method.

## 2 Problem Setup

We have a set of  $n$  teams with indices  $i = 1, 2, \dots, n$ . A learner is presented with a sequence of pairs  $(i_t, j_t)$  for  $t = 1, \dots, T$  and must predict  $\hat{y}_t \in [-1, 1]$ , where  $\hat{y}_t = 1$  implies that the learner believes that team  $i_t$  will beat team  $j_t$ , and vice versa when  $\hat{y}_t = -1$ . After making her prediction, the learner observes the outcome  $y_t \in \{-1, 1\}$  and suffers loss  $\ell(\hat{y}_t, y_t) := (1 - \hat{y}_t y_t)/2$ .

An online prediction algorithm  $A$  is a function that outputs predictions  $\hat{y}_t$  given input of the data  $(i_1, j_1, y_1), \dots, (i_{t-1}, j_{t-1}, y_{t-1})$  and the current matchup  $(i_t, j_t)$ . We can compare such a prediction algorithm to any *offline comparator* class  $\mathcal{F}$ , which is any collection of “skew-symmetric” mappings  $\phi : [n] \times [n] \rightarrow \{-1, 1\}$ , namely those that satisfy  $\phi(i, j) = -\phi(j, i)$  for all  $i, j$  (required since if  $i$  beats  $j$  then  $j$  doesn’t beat  $i$ ). We’ll consider two such classes, the class  $\mathcal{F}_{\text{all}}$  of *all* such mappings, and the class  $\mathcal{F}_{\text{perm}}$  of *permutations*, i.e. those mappings  $\phi$  which satisfy the transitive property,  $\phi(i, j) = 1$  and  $\phi(j, k) = 1$  implies  $\phi(i, k) = 1$ . The *regret* of any algorithm  $A$  with respect to any  $\mathcal{F}$  is defined as

$$\text{Regret}_{\mathcal{F}}(A) := \sum_{t=1}^T \ell(\hat{y}_t, y_t) - \min_{\phi \in \mathcal{F}} \sum_{t=1}^T \ell(\phi(i_t, j_t), y_t)$$

**Lemma 1** For any algorithm  $A$ ,  $\text{Regret}_{\mathcal{F}_{\text{perm}}}(A) \leq \text{Regret}_{\mathcal{F}_{\text{all}}}(A)$ .

**Proof:**[sketch] This follows trivially from the fact that  $\mathcal{F}_{\text{perm}} \subset \mathcal{F}_{\text{all}}$  ■

**Lemma 2** There exists an inefficient algorithm  $A$  such that  $\text{Regret}_{\mathcal{F}_{\text{perm}}}(A) = O(\sqrt{Tn \log n})$ .

**Proof:**[sketch] If we treat each permutation  $\phi \in \mathcal{F}_{\text{perm}}$  as an “expert” and run a standard experts algorithm, then well-known results (see, e.g., Cesa-Bianchi and Lugosi [1]) imply that the regret will scale as  $O(\sqrt{T \log(\text{no. of experts})})$ . The result follows since there are  $n!$  permutations, and  $\log n! = \Theta(n \log n)$ . ■

**Lemma 3** There exists an efficient algorithm  $A$  such that  $\text{Regret}_{\mathcal{F}_{\text{all}}}(A) = O(\sqrt{Tn^2})$ .

**Proof:**[sketch] Let  $i$  and  $j$  be arbitrary and assume without loss of generality that  $i < j$ . If we treat the problem “does  $i$  beat  $j$ ?” as an independent learning problem, then we can imagine that we have two experts: “ $i$  wins” and “ $j$  wins”. So if the matchup  $(i, j)$  occurs  $T_{i,j}$  times throughout the season, then on only these particular events we are guaranteed to achieve  $O(\sqrt{T_{i,j}})$  regret using a standard experts algorithm. If we do this for every pair  $i, j$  independently, then we can achieve

$$\text{Regret}_{\mathcal{F}_{\text{all}}} = O\left(\sum_{(i,j):i < j} \sqrt{T_{i,j}}\right).$$

Using the fact that, for any  $z_1, \dots, z_m \geq 0$ ,

$$\sum_{j=1}^m \sqrt{z_j} \leq \sqrt{m} \sqrt{\sum_{j=1}^m z_j}$$

and that  $\sum_{(i,j):i < j} T_{i,j} = T$ , immediately gives the desired bound. ■

This final Lemma suggests that sub-linear regret is not difficult for learning rankings. Unfortunately, this bound scales quite poorly in  $n$ . This is precisely because this algorithm, when predicting the outcome of a pair  $(X, Y)$ , does not leverage the information from any other matchups. For example, when the biggest winner plays the biggest loser for the first time, this algorithm would guess the outcome is a tossup.

**Open Problem:** Is it possible to find an efficient algorithm that closes the gap between the  $O(\sqrt{Tn^2})$  bound and the information-theoretic  $O(\sqrt{Tn \log n})$  rate? Is it even possible to do any better than  $O(\sqrt{Tn^2})$  efficiently?

## Prior Work

While not stated as we do here, essentially the same open question was posed by Kleinberg et al [2] in 2008. Their work also provides a very useful background on learning to rank.

## Acknowledgments

I would like to thanks to Adam Kalai for originally suggesting this open question, and to Hamid Nazer Zadeh and Eli Ben-Sasson for joining our unsuccessful attempt at solving it. Adam gets additional credit for regularly finding counterexamples to nearly all of our proposed solutions. Great work, Adam, really.

## References

- [1] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge Univ Pr, 2006.
- [2] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *ACM COLT*, 2008.