The Ladder: A Reliable Leaderboard for Machine Learning Competitions

Moritz Hardt M@MRTZ.ORG

Avrim Blum AVRIM@CS.CMU.EDU
Carnegie Mellon University

Abstract

The organizer of a machine learning competition faces the problem of maintaining an accurate leaderboard that faithfully represents the quality of the best submission of each competing team. What makes this estimation problem particularly challenging is its sequential and adaptive nature. As participants are allowed to repeatedly evaluate their submissions on the leaderboard, they may begin to overfit to the holdout data that supports the leaderboard. Few theoretical results give actionable advice on how to design a reliable leaderboard. Existing approaches therefore often resort to poorly understood heuristics such as limiting the bit precision of answers and the rate of resubmission.

In this work, we introduce a notion of *leaderboard accuracy* tailored to the format of a competition. We introduce a natural algorithm called *the Ladder* and demonstrate that it simultaneously supports strong theoretical guarantees in a fully adaptive model of estimation, withstands practical adversarial attacks, and achieves high utility on real submission files from an actual competition hosted by Kaggle.

Notably, we are able to sidestep a powerful recent hardness result for adaptive risk estimation that rules out algorithms such as ours under a seemingly very similar notion of accuracy. On a practical note, we provide a completely parameter-free variant of our algorithm that can be deployed in a real competition with no tuning required whatsoever.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

1. Introduction

Machine learning competitions have become an extremely popular format for solving prediction and classification problems of all kinds. A number of companies such as Netflix have organized major competitions in the past and some start-ups like Kaggle specialize in hosting machine learning competitions. In a typical competition hundreds of participants will compete for prize money by repeatedly submitting classifiers to the host in an attempt to improve on their previously best score. The score reflects the performance of the classifier on some subset of the data, which are typically partitioned into two sets: a training set and a test set. The training set is publicly available with both the individual instances and their corresponding class labels. The test set is publicly available as well, but the class labels are withheld. Predicting these missing class labels is the goal of the participant and a valid submission is simply a list of labels—one for each point in the test set.

The central component of any competition is the leaderboard which ranks all teams in the competition by the score of their best submission. This leads to the fundamental problem of maintaining a leaderboard that accurately reflects the true strength of a classifier. What makes this problem so challenging is that participants may begin to incorporate the feedback from the leaderboard into the design of their classifier thus creating a dependence between the classifier and the data on which it is evaluated. In such cases, it is well known that the holdout set no longer gives an unbiased estimate of the classifier's true performance. To counteract this problem, existing solutions such as the one used by Kaggle further partition the test set into two parts. One part of the test set is used for computing scores on the public leaderboard. The other is used to rank all submissions after the competition ended. This final ranking is often referred to as the *private leaderboard*. While this solution increases the quality of the private leaderboard, it does not address the problem of maintaining accuracy on the public leaderboard. Indeed, numerous posts on the forums of Kaggle report on the problem of "overfitting to the holdout" meaning that

some scores on the public leaderboard are inflated compared to final scores. To mitigate this problem Kaggle primarily restricts the rate of re-submission and to some extent the numerical precision of the released scores.

Yet, in spite of its obvious importance, there is relatively little theory on how to design a leaderboard with rigorous quality guarantees. Basic questions remain difficult to assess, such as, can we a priori quantify how accurate existing leaderboard mechanisms are and can we design better methods?

While the theory of estimating the true loss of a classifier or set of classifiers from a finite sample is decades old, much of theory breaks down due to the *sequential* and *adaptive* nature of the estimation problem that arises when maintaining a leaderboard. First of all, there is no a priori understanding of which learning algorithms are going to be used, the complexity of the classifiers they are producing, and how many submissions there are going to be. Indeed, submissions are just a list of labels and do not even specify how these labels were obtained. Second, any submission might incorporate statistical information about the withheld class labels that was revealed by the score of previous submissions. In such cases, the public leaderboard may no longer provide an unbiased estimate of the true score. To make matters worse, very recent results suggest that maintaining accurate estimates on a sequence of many adaptively chosen classifiers may be computationally intractable (Hardt & Ullman, 2014; Steinke & Ullman, 2014).

1.1. Our Contributions

We introduce a notion of accuracy called *leaderboard accuracy* tailored to the format of a competition. Intuitively, high leaderboard accuracy entails that each score represented on the leaderboard is close to the true score of the corresponding classifier on the unknown distribution from which the data were drawn. Our primary theoretical contributions are the following.

1. We show that there is a simple and natural algorithm we call *Ladder* that achieves high leaderboard accuracy in a fully adaptive model of estimation in which we place no restrictions on the data analyst whatsoever. In fact, we don't even limit the number of submissions an analyst can make. Formally, our worst-case upper bound shows that if we normalize scores to be in $[0, 1]$ the maximum error of our algorithm on any estimate is never worse than $O((\log(kn)/n)^{1/3})$ where k is the number of submissions and n is the size of the data used to compute the leaderboard. In contrast, we observe that the error of the Kaggle mechanism (and similar solutions) scales with the number of submissions as $\sqrt{k}$ so that our algorithm features an exponential improvement in k.

2. We also prove an information-theoretic lower bound on the leaderboard accuracy demonstrating that no estimator can achieve error smaller than $\Omega((\log(k)/n)^{1/2})$.

Complementing our theoretical worst-case upper bound and lower bound, we make a number of practical contributions:

1. We provide a *parameter-free* variant of our algorithm that can be deployed in a real competition with no tuning required whatsoever.

2. To demonstrate the strength of our parameter-free algorithm we conduct two opposing experiments. The first is an adversarial—yet practical—attack on the leaderboard that aims to create as much of a bias as possible with a given number of submissions. We compare the performance of the Kaggle mechanism to that of the Ladder mechanism under this attack. We observe that the accuracy of the Kaggle mechanism diminishes rapidly with the number of submissions, while our algorithm encounters only a small bias in its estimates.

3. In a second experiment, we evaluate our algorithm on real submission files from a Kaggle competition. The data set presents a difficult benchmark as little overfitting occurred and the errors of the Kaggle leaderboard were generally within the expected statistical deviations given the properties of the data set. Even on this benchmark our algorithm produced a leaderboard that is very close to that computed by Kaggle. Through a sequence of significance tests we assess that the differences between the two leaderboards on this competition are not statistically significant.

In summary, our algorithm supports strong theoretical results while suggesting a simple and practical solution. Importantly, it is one and the same parameter-free algorithm that withstands our adversarial attack and simultaneously achieves high utility in a real Kaggle competition.

An important aspect of our algorithm is that it only releases a score to the participant if the score presents a statistically significant improvement over the previously best submission of the participant. Intuitively, this prevents the participant from exploiting or overfitting to minor fluctuations in the observed score values.

1.2. Related Work

There is a vast literature on preventing overfitting in the context of model assessment and selection. See, for example, Chapter 7 of (Hastie et al., 2001) for background. Two particularly popular practical approaches are various forms

of cross-validation and bootstrapping. It is important to note though that when scoring a submission for the leaderboard, neither of these techniques applies. One problem is that participants submit only a list of labels and not the corresponding learning algorithms. In particular, the organizer of the competition has no means of retraining the model on a different split of the data. Similarly, the natural bootstrap estimate of the expected loss of a classifier given a finite sample is simply the empirical average of the loss on the finite sample, which is what existing solutions release anyway. The other substantial obstacle is that even if these methods applied, their theoretical guarantees in the adaptive setting of estimation are largely not understood.

A highly relevant recent work (Dwork et al., 2015), that inspired us, studies a more general question: Given a sequence of *adaptively chosen* bounded functions $f_1, \ldots, f_k \colon X \to \{0,1\}$ over a domain X, estimate the expectations of these function $\mathbb{E}\, f_1, \ldots, \mathbb{E}\, f_k$ over an unknown distribution $\mathcal{D}$, given n samples from this distribution. If we think of each function as expressing the loss of one classifier submitted to the leaderboard, then such an algorithm could in principle be used in our setting. The main result of (Dwork et al., 2015) is an algorithm that achieves maximum error

$$O\left(\min\left\{\log^{\frac{3}{7}}(k)\log^{\frac{1}{7}}(|X|)/n^{\frac{2}{7}}, (\log|X|\log(k)/n)^{\frac{1}{4}}\right\}\right).$$

This bound readily implies a corresponding result for leaderboard accuracy albeit worse than the one we show. One issue is that this algorithm requires the entire test set to be withheld and not just the labels as is required in the Kaggle application. The bigger obstacle is that the algorithm is unfortunately not computationally efficient and this is inherent. In fact, no computationally efficient algorithm can give non-trivial error on $k > n^{2+o(1)}$ adaptively chosen functions as was shown recently (Hardt & Ullman, 2014; Steinke & Ullman, 2014) under a standard computational hardness assumption.

Matching this hardness result, there is a computationally efficient algorithm in (Dwork et al., 2015) that achieves an error bound of $O(k^{1/5}\log(k)^{3/5}/n^{2/5})$ which implies a bound on leaderboard accuracy that is worse than ours for all $k > n^{1/3}$. They also give an algorithm (called EffectiveRounds) with accuracy $O(\sqrt{r\log(k)/n})$ when the number of "rounds of adaptivity" is at most r. While we do not have a bound on r in our setting better than k[1], the proof technique relies on sample splitting and a similar argument could be used to prove our upper bound. However, our argument does not require sample splitting and this is very important for the practical applicability of the algorithm. Subsequently to our result, (Bassily et al., 2015) and (Nissim & Stemmer, 2015) improved upon the quantitative bounds of (Dwork et al., 2015).

We sidestep the hardness result by going to a more specialized notion of accuracy that is surprisingly still sufficient for the leaderboard application. However, it does not resolve the more general question raised in (Dwork et al., 2015). In particular, we do not always provide a loss estimate for each submitted classifier, but only for those that made a significant improvement over the previous best. This seemingly innocuous change is enough to circumvent the aforementioned hardness results.

Acknowledgments

We thank Ben Hamner at Kaggle Inc., for providing us with the submission files from the Photo Quality competition, as well as many helpful discussions. We are grateful to Aaron Roth for pointing out an argument similar to that appearing in the proof of Theorem 3.1 in a different context. We thank John Duchi for many stimulating discussions.

1.3. Preliminaries

Let X be a data domain and Y be a finite set of class labels, e.g., $X = \mathbb{R}^d$ and $Y = \{0,1\}$. Rather than speaking of the score of a classifier we will use the term *loss* with the understanding that smaller is better. A loss function is a mapping of the form $\ell \colon Y \times Y \to [0,1]$ and a classifier is a mapping $f \colon X \to Y$. A standard loss function is the $0/1$-loss defined as $\ell_{01}(y,y') = 1$ if $y \neq y'$ and 0 otherwise.

We assume that we are given a sample $S = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ drawn i.i.d. from an unknown distribution $\mathcal{D}$ over $X \times Y$. We define the *empirical loss* of a classifier f on the sample S as

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i)\,.$$

The *true loss* is defined as

$$R_{\mathcal{D}}(f) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\ell(f(x),y))\right]\,.$$

Throughout this paper we assume that S consists of n i.i.d. draws from $\mathcal{D}$ and ℓ is a loss function with bounded range.

ly identified using $B = O(\eta^{-1}\log(t/\eta))$ bits of information. This shows that the tree itself cannot grow larger than 2^B and we may take a union bound over all 2^B nodes.

2. Sequential and adaptive loss estimation

In this section we formally define the adaptive model of estimation that we work in and present our definition of *leaderboard accuracy*. Given a sequence of classifiers $f_1,\ldots,f_k$

[1] The parameter r corresponds to the depth of the adaptive tree we define in the proof of Theorem 3.1. While we bound the size of the tree, the depth could be as large as k.

and a finite sample S of size n, a fundamental estimation problem is to compute estimates $R_1, \ldots, R_k$ such that

$$\Pr\{\exists t\colon\ |R_t - R_{\mathcal{D}}(f_t)| > \varepsilon\} \leqslant \delta\,. \tag{1}$$

The standard way of estimating the true loss is via the empirical loss. If we assume that all functions $f_1, \ldots, f_k$ are fixed independently of the sample S, then Hoeffding's bound and the union bound imply

$$\Pr\{\exists t\colon\ |R_S(f_t) - R_{\mathcal{D}}(f_t)| > \varepsilon\} \leqslant 2k\exp(-2\varepsilon^2 n)\,. \tag{2}$$

In the *adaptive* setting, however, we assume that the classifier f_t may be chosen as a function of the previous estimates and the previously chosen classifiers. Formally, there exists a mapping $\mathcal{A}$ such that for all $t \in [k]$, $f_t = \mathcal{A}(f_1, R_1, \ldots, f_{t-1}, R_{t-1})$. We will assume for simplicity that $\mathcal{A}$ is a deterministic algorithm. The tuple $(f_1, R_1, \ldots, f_{t-1}, R_{t-1})$ is nevertheless a random variable due to the random sample used to compute the estimates.

Unfortunately, in the case where the choice of f_t depends on previous estimates, we may no longer apply Hoeffding's bound to control $R_S(f_t)$. In fact, recent work (Hardt & Ullman, 2014; Steinke & Ullman, 2014) shows that no computationally efficient estimator can achieve error $o(1)$ on more than $n^{2+o(1)}$ adaptively chosen functions (under a standard hardness assumption). Since we're primarily interested in a computationally efficient algorithm, these hardness results demonstrate that the goal of achieving the accuracy guarantee specified in inequality (1) is too stringent in the adaptive setting when k is large. We will therefore introduce a weaker notion of accuracy called *leaderboard accuracy* under which we can circumvent the hardness results and nevertheless achieve a guarantee strong enough for our application.

2.1. Leaderboard accuracy

The goal of an accurate leaderboard is to guarantee that at each step $t \leqslant k$, the leaderboard accurately reflects the *best* classifier among those classifiers $f_1, \ldots, f_k$ submitted so far. In other words, while we do not need an accurate estimate for each f_t, we wish to maintain that the t-th estimate R_t correctly reflects the minimum loss achieved by any classifier so far. This leads to the following definition.

Definition 2.1. Given an adaptively chosen sequence of classifiers $f_1, \ldots, f_k$, we define the *leaderboard error* of estimates $R_1, \ldots, R_k$ as

$$\mathrm{lberr}(R_1, \ldots, R_k) \stackrel{\mathrm{def}}{=} \max_{1\leqslant t\leqslant k} \left|\min_{1\leqslant i\leqslant t} R_{\mathcal{D}}(f_i) - R_t\right| \tag{3}$$

Given an algorithm that achieves high leaderboard accuracy there are two simple ways to extend it to provide a full leaderboard:

1. Use one instance of the algorithm for each team to maintain the best score achieved by each team.
2. Use one instance of the algorithm for each rank on the leaderboard. When a new submission comes in, evaluate it against each instance in descending order to determine its place on the leaderboard.

The first variant is straightforward to implement, but requires the assumption that competitors don't use several accounts (a practice that is typically against the terms of use of a competition). The second variant is more conservative and does not need this assumption.

3. The Ladder mechanism

We introduce an algorithm called the Ladder Mechanism that achieves small leaderboard accuracy. The algorithm is very simple. For each given function, it compares the empirical loss estimate of the function to the previously smallest loss. If the estimate is below the previous best by some margin, it releases the estimate and updates the best estimate. Importantly, if the estimate is not smaller by a margin, the algorithm releases the previous best loss (rather than the new estimate). A formal description follows in Figure 1.

Algorithm 1 Ladder mechanism

Input: Data set S, step size $\eta > 0$
Assign initial estimate $R_0 \leftarrow \infty$.
for round $t = 1, 2, \ldots$ **do**
  Receive function $f_t\colon X \to Y$
  if $R_S(f_t) < R_{t-1} - \eta$ **then**
    Assign $R_t \leftarrow [R_S(f_t)]_\eta$.
  else
    Assign $R_t \leftarrow R_{t-1}$.
  end if
end for

Theorem 3.1. *For any sequence of adaptively chosen classifiers $f_1, \ldots, f_k$, the Ladder Mechanism satisfies for all $t \leqslant k$ and $\varepsilon > 0$,*

$$\Pr\left\{\left|\min_{1\leqslant i\leqslant t} R_{\mathcal{D}}(f_i) - R_t\right| > \varepsilon + \eta\right\} \leqslant \exp\left(-2\varepsilon^2 n + (1/\eta + 2)\log(4t/\eta) + 1\right)\,.$$

In particular, for some $\eta = O(n^{-1/3}\log^{1/3}(kn))$, the Ladder Mechanism achieves with high probability,

$$\mathrm{lberr}(R_1, \ldots, R_k) \leqslant O\left(\frac{\log^{1/3}(kn)}{n^{1/3}}\right)\,.$$

Proof. Let $\mathcal{A}$ be the adaptive analyst generating the function sequence. Fix $t \leqslant k$. The algorithm $\mathcal{A}$ naturally defines a rooted tree $\mathcal{T}$ of depth t recursively defined as follows:

1. The root is labeled by $f_1 = \mathcal{A}(\emptyset)$.

2. Each node at depth $1 < i < t$ corresponds to one realization $(h_1, r_1, \ldots, h_{i-1}, r_{i-1})$ of the random variable $(f_1, R_1, \ldots, f_{i-1}, R_{i-1})$ and is labeled by $h_i = \mathcal{A}(h_1, r_1, \ldots, h_{i-1}, r_{i-1})$. Its children are defined by each possible value of the output R_i of Ladder Mechanism on the sequence $h_1, r_1, \ldots, r_{i-1}, h_i$.

Claim 3.2. *Let $B = (1/\eta + 2)\log(4t/\eta)$. Then, $|\mathcal{T}| \leqslant 2^B$.*

Proof. To prove the claim, we will uniquely encode each node in the tree using B bits of information. The claim then follows directly. The compression argument is as follows. We use $\lfloor \log(t) \rfloor \leqslant \log(2t)$ bits to specify the depth of the node in the tree. We then specify the index of each $1 \leqslant i \leqslant t$ for which $R_i \leqslant R_{i-1} - \eta$ together with the value R_i. Note that since $R_i \in [0,1]$ there can be at most $\lceil 1/\eta \rceil \leqslant (1/\eta)+1$ many such steps. Moreover, there are at most $\lceil 1/\eta \rceil$ many possible values for $R_i = [R_S(f_i)]_\eta$. Hence, specifying all such indices requires at most $(1/\eta+1)(\log(2/\eta)+\log(2t))$ bits. It is easy that this uniquely identifies each node in the graph, since for every index i not explicitly listed we know that $R_i = R_{i-1}$. The total number of bits we used is: $(1/\eta + 1)(\log(2/\eta) + \log(2t)) + \log(2t) \leqslant (1/\eta + 2)\log(4t/\eta) = B$. ∎

The theorem now follows by applying a union bound over all nodes in $\mathcal{T}$ and using Hoeffding's inequality for each fixed node. Let F be the set of all functions appearing in $\mathcal{T}$.

$$\begin{aligned}\Pr\{\exists f \in F\colon\ |R_{\mathcal{D}}(f) - R_S(f)| > \varepsilon\} &\leqslant 2|F|\exp(-2\varepsilon^2 n) \\ &\leqslant 2^{B+1}\exp(-2\varepsilon^2 n) \\ &\leqslant 2\exp(-2\varepsilon^2 n + B)\,.\end{aligned}$$

In particular,

$$\Pr\left\{\left|\min_{1\leqslant i\leqslant t} R_{\mathcal{D}}(f_i) - \min_{1\leqslant i\leqslant t} R_S(f_i)\right| > \varepsilon\right\} \leqslant 2e^{-2\varepsilon^2 n + B}\,.$$

Moreover, it is clear that conditioned on the event that

$$\left|\min_{1\leqslant i\leqslant t} R_{\mathcal{D}}(f_i) - \min_{1\leqslant i\leqslant t} R_S(f_i)\right| \leqslant \varepsilon,$$

at step i^* where the minimum of $R_{\mathcal{D}}(f_i)$ is attained, the Ladder Mechanism must output an estimate R_{i^*} which is within $\varepsilon + \eta$ of $R_{\mathcal{D}}(f_{i^*})$. This concludes the proof. ∎

3.1. A lower bound on leaderboard accuracy

We next show that $\Omega(\sqrt{\log(k)/n})$ is a lower bound on the best possible leaderboard accuracy that we might hope to achieve. This is true even if the functions are not adaptively chosen but fixed ahead of time.

Theorem 3.3. *There are classifiers $f_1, \ldots f_k$ and a bounded loss function for which we have the minimax lower bound*

$$\inf_R \sup_{\mathcal{D}} \mathbb{E}\left[\mathrm{lberr}(R(x_1, \ldots, x_n))\right] \geqslant \Omega\left(\sqrt{\frac{\log k}{n}}\right).$$

Here the infimum is taken over all estimators $R\colon X^n \to [0,1]^k$ that take n samples from a distribution $\mathcal{D}$ and produce k estimates $R_1, \ldots, R_k = \widehat{\theta}(x_1, \ldots, x_n)$. The expectation is taken over n samples from $\mathcal{D}$.

The proof follows by reduction from a high-dimensional mean estimation problem and an application of Fano's inequality. Please see the full version of this paper (arXiv:1502.04585) for details.

4. Parameter-free Ladder mechanism

When applying the Ladder Mechanism in practice it can be difficult to choose a fixed step size η ahead of time that will work throughout an entire competition. We therefore now give a completely parameter-free version of our algorithm that we will use in our experiments. The algorithm adaptively finds a suitable step size based on previous submissions to the algorithm. The idea is to perform a statistical significance test to judge whether the given submission improves upon the previous one. The test is such that as the best classifier gets increasingly accurate, the step size shrinks accordingly.

The empirical loss of a classifier is the average of n bounded numbers and follows a very accurate normal approximation for sufficiently large n so long as the loss is not biased too much towards 0. In our setting, the typical loss if bounded away form 0 so that the normal approximation is reasonable. In order to test whether the empirical loss of one classifier is significantly below the empirical loss of another classifier, it is appropriate to perform a one-sided *paired* t-test. A paired test has substantially more statistical power in settings where the loss vectors that are being compared are highly correlated as is common in a competition.

To recall the definition of the test, we denote the *sample standard deviation* of an n-dimensional vector vector u as

$$\mathrm{std}(u) = \sqrt{\tfrac{1}{n-1}\textstyle\sum_{i=1}^n (u_i - \mathrm{mean}(u))^2}\,, \qquad (4)$$

where $\mathrm{mean}(u)$ denotes the average of the entries in u. With this notation, the paired t-test statistic given two vectors u and v is defined as

$$t = \sqrt{n} \cdot \frac{\mathrm{mean}(u - v)}{\mathrm{std}(u - v)}\,. \qquad (5)$$

Keeping this definition in mind, Algorithm 2 is now very natural. On top of the loss estimate, it also maintains the

loss vector of the previously best classifier (starting with the trivial all zeros loss vector).

Algorithm 2 Parameter-free Ladder Mechanism

Input: Data $S = \{(x_1, y_1), \dots (x_n \dots, y_n)\}$
Assign initial estimate $R_0 \leftarrow \infty$
Assign initial loss vector $\ell_0 = (0)_{i=1}^n$.
for round $t = 1, 2 \dots$ **do**
  Receive function $f_t \colon X \to Y$.
  Compute loss vector $l_t \leftarrow (\ell(f_t(x_i), y_i))_{i=1}^n$
  if $R_S(f_t) < R_{t-1} - \mathrm{std}(l_t - l_{t-1})/\sqrt{n}$ **then**
    $R_t \leftarrow [R_S(f_t)]_{1/n}$.
  else
    Assign $R_t \leftarrow R_{t-1}$ and $l_t \leftarrow l_{t-1}$.
  end if
end for

Algorithm 2 releases the estimate of $R_S(f_t)$ up to an error of $1/n$ which is significantly below the typical step size of $\Omega(1/\sqrt{n})$. Looking back at our analysis, this is not a problem since such an estimate only reveals $\log(n)$ bits of information which is the same up to constant factors as an estimate that is accurate to within $1/\sqrt{n}$. The more critical quantity is the step size as it controls how often the algorithm releases a new estimate.

In the following sections we will show that the parameter-free Ladder mechanism achieves high accuracy both under a strong attack as well as on a real Kaggle competition.

4.1. Interpretation of the significance test

For sufficiently large n, the test statistic on the left hand side of (5) is well approximated by a Student's t-distribution with $n - 1$ degrees of freedom. The test performed in our algorithm at each step corresponds to refuting the null hypothesis roughly at the 0.15 significance level.

It is important to note, however, that our use of this significance test is primarily heuristic. This is because for $t > 1$, due to the adaptive choices of the analyst, the function f_t may in general not be independent of the sample S. In such a case, the Student approximation is no longer valid. Besides we apply the test many times, but do not control for multiple comparisons. Nevertheless, the significance test is an intuitive guide for deciding which improvements are statistically significant.

5. The boosting attack

In this section we describe a new canonical attack that an adversarial analyst might perform in order to boost their ranking on the public leaderboard. Besides being practical in some cases, the attack also serves as an analytical tool to assess the accuracy of concrete mechanisms.

For simplicity we describe the attack only for the $0/1$-loss although it generalizes to other reasonable functions such as the clipped logarithmic loss often used by Kaggle. We assume that the hidden solution is a vector $y \in \{0,1\}^n$. The analyst may submit a vector $u \in \{0,1\}^n$ and observe (up to small enough error) the loss

$$\ell_{01}(y, u) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell_{01}(u_i, y_i)$$

The attack proceeds as follows:

1. Pick $u_1, \dots, u_k \in \{0,1\}^n$ uniformly at random.
2. Observe loss estimates $l_1, \dots, l_k \in [0,1]$.
3. Let $I = \{i \colon l_i \leqslant 1/2\}$.
4. Output $u^* = \mathrm{maj}\left(\{u_i \colon i \in I\}\right)$, where the majority function is applied coordinate-wise.

The vector y corresponds to the target set of labels used for the public leaderboard which the analyst does not know. The vectors $u_1, \dots, u_k$ represent the labels given by a sequence of k classifiers.

The next theorem follows from a standard "boosting argument" using properties of the majority function and the fact that each u_i for $i \in I$ has a somewhat larger than expected correlation with y.

Theorem 5.1. *Assume that $|l_i - \ell_{01}(y, u_i)| \leqslant n^{-1/2}$ for all $i \in [k]$. Then, the boosting attack finds a vector $u^* \in \{0,1\}^n$ so that with probability $2/3$,*

$$\frac{1}{n} \sum_{i=1}^{n} \ell_{01}(u_i^*, y_i) \leqslant \frac{1}{2} - \Omega\left(\sqrt{\frac{k}{n}}\right).$$

The previous theorem in particular demonstrates that the Kaggle mechanism has poor leaderboard accuracy if it is invoked with rounding parameter $\alpha \leqslant 1/\sqrt{n}$. The currently used rounding parameter is 10^{-5} which satisfies this assumption for all $n \leqslant 10^{10}$. To make the discussion more precise, we briefly state a reference mechanism similar to the one Kaggle uses. We will refer to this algorithm as the "Kaggle mechanism" in our discussion below. As we did for the Ladder Mechanism we describe the algorithm as if the analyst was submitting classifiers $f \colon X \to Y$. In reality the analyst only submits a list of labels. It is easy to see that such a list of labels is sufficient to compute the empirical loss which is all the algorithm needs to do. The input set S in the description of our algorithm corresponds to the set of data points (and corresponding labels) that Kaggle uses for the public leaderboard.

Algorithm 3 Kaggle reference mechanism

Input: Data set S, rounding parameter $\alpha > 0$ (typically 0.00001)
for round $t \leftarrow 1, 2, \ldots :$ **do**
  Receive function $f_t \colon X \to Y$
  Assign $R_t \leftarrow [R_S(f_t)]_\alpha$.
end for

Corollary 5.2. *There is a sequence of adaptively chosen classifiers $f_1, \ldots, f_k$ such that if R_i denotes the minimum of the first i loss estimates returned by the Kaggle mechanism with accuracy $\alpha \leqslant 1/\sqrt{n}$ where n is the size of the data set, then with probability $2/3$ the estimates $R_1, \ldots, R_k$ have leaderboard error*

$$\mathrm{lberr}(R_1, \ldots, R_k) \geqslant \Omega\left(\sqrt{\frac{k}{n}}\right) .$$

5.1. Experiments with the boosting attack

Figure 1 compares the performance of the Ladder mechanism with that of the standard Kaggle mechanism under the boosting attack. We chose $N = 12000$ as the total number of labels of which $n = 4000$ labels are used for determining the public leaderboard under either mechanism. Other parameter settings lead to a similar picture, but these settings correspond roughly to the properties of the real data set that we will analyze later. The Kaggle mechanism gives answers that are accurate up to a rounding error of 10^{-5}. Note that $1/\sqrt{4000} \approx 0.0158$ so that the rounding error is well below the critical level of $1/\sqrt{n}$. The vector y in the description of our attack corresponds to the 4000 labels used for the public leaderboard. Since the answers given by Kaggle only depend on these labels, the remaining labels play no role in the attack. Importantly, the attack does not need to know the indices of the labels used for the public leaderboard within the entire vector of labels.

The 8000 coordinates not used for the leaderboard remain unbiased random bits throughout the attack as no information is revealed. In particular, the final submission u^* is completely random on those 8000 coordinates and only biased on the other 4000 coordinates used for the leaderboard. Therefore, once we evaluate the final submission u^* on the test set consisting of the remaining 8000 coordinates, the resulting loss is close to its expected value of $1/2$, i.e. the expected loss of a random $0/1$-vector. What we observe, however, is that the Kaggle mechanism gives a strongly biased estimate of the loss of u^*.

The blue line in Figure 1 displays the performance of the parameter-free version of the Ladder mechanism. Instead of selecting all the vectors with loss at most $1/2$ we modified the attack to be more effective against the Ladder Mechanism. Specifically, we selected all those vectors that successfully lowered the score compared to the previous best. As we have no information about the correlation of the remaining vectors, there is no benefit in including them in the boosting step. Even with this more effective attack, the Ladder mechanism gives a result that is correct to within the expected maximum deviation of the score on k random vectors. The intuitive reason is that every time a vector lowers the best score seen so far, the probability of a subsequent vector crossing the new threshold drops off by a constant factor. In particular there cannot be more than $O(\log(k))$ such steps thus creating a bias of at most $O(\sqrt{\log(k)/n})$ in the boosting step.

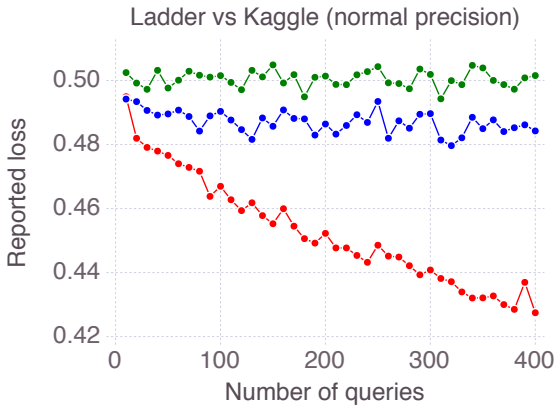

Figure 1. Performance of the parameter free Ladder Mechanism compared with the Kaggle Mechanism. Top **green** line: Independent test set. Middle **blue** line: Ladder. Bottom **red** line: Kaggle.

6. Experiments on real Kaggle data

To demonstrate the utility of the Ladder mechanism we turn to real submission data from Kaggle's "Photo Quality Prediction" challenge[2]. The holdout set contained 12000 samples of which Kaggle used 8400 for the private leaderboard and 3600 for the public leaderboard. Two hundred teams entered the competition producing a total of 1830 submissions. We were able to parse 1785 submissions successfully. The exact number of correctly parsed submissions by Kaggle differs slightly. Our code is available at https://github.com/mrtzh/Ladder.jl.

6.1. Using Ladder instead of Kaggle's mechanism

In our first experiment we use the parameter-free Ladder mechanism in place of the Kaggle mechanism across all 1785 submissions and recompute both the public and the private leaderboard. Our primary finding is that the resulting

[2] https://www.kaggle.com/c/PhotoQualityPrediction

rankings turn out to be extremely close to those computed by Kaggle. To illustrate this we consider the top 50 submissions as determined by the private leaderboard. For each of these submission we compare the "private score" (score on the 8400 private samples) with the "public score" (score on the 3600 remaining samples) computed by either the Ladder mechanism or the Kaggle mechanism. The results are shown in Figure 2.

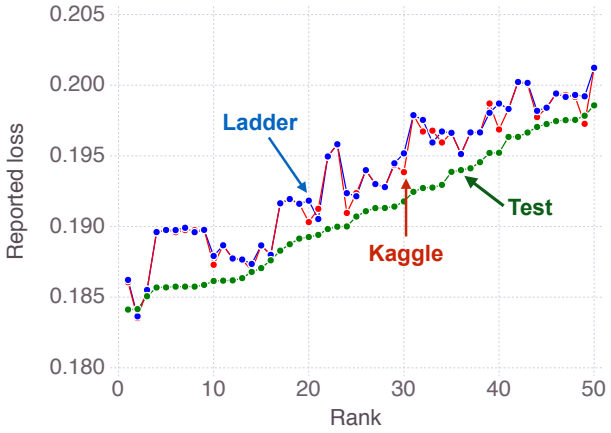

Figure 2. X-axis: Rank according to private test set. Y-axis: Public scores reported by Ladder mechanism and Kaggle mechanism.

As we can see in Figure 2, the scores computed by the Ladder mechanism are almost identical to those computed by Kaggle. We can also see a small amount of underfitting between the public and private scores; the losses on the public leaderboard generally tend to be slightly higher than on the private leaderboard. This appears to be due random fluctuations in the proportion of hard examples in the public holdout set.

To assess this possibility and gain further insight into the magnitude of statistical deviations of the scores, we randomly split the private holdout set into two equally sized parts and recompute the Kaggle leaderboards on each split. We repeat the process 20 times independently and look at the standard deviations of the scores across these 20 repetitions. Figure 3 shows the results demonstrating that the statistical deviations due to random splitting are large relative to the difference in mean scores. In particular the amount of underfitting observed on the original split is within one standard deviation of the mean scores which cluster close to the diagonal line. We also observed (not reflected in the figure) that the top 50 scores are highly correlated so that across different splits the points are either mostly above or mostly below the diagonal line. This must be due to the fact that the best submissions in this competition used related classifiers that fail to predict roughly the same label set.

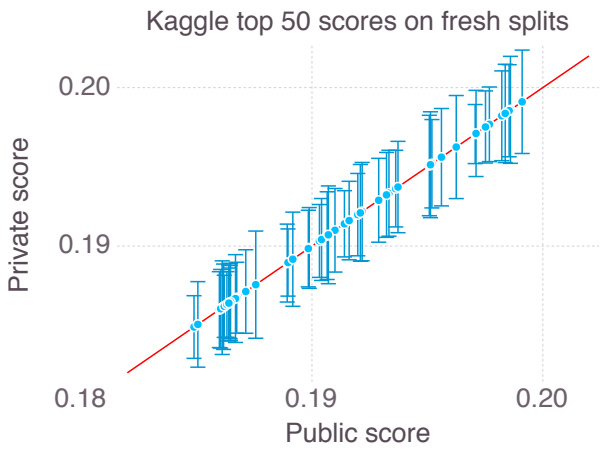

Figure 3. Behavior of private and public scores under fresh splits.

7. Conclusion

In a first step, we hope that the Ladder mechanism will be helpful in making machine learning competitions more reliable. But we believe that the Ladder mechanism could be useful well beyond the scope of machine learning competitions. For example, a data scientist might use the Ladder to keep track of her own progress in building a good model from a data set while avoiding overfitting to her holdout set. In such a situation the Ladder mechanism gives concrete guidance on how to safely reuse a holdout set. A similar application arises in the problem of searching for good hyperparameters of a learning algorithm. Whenever an adaptive search strategy is used, there is an increased danger of overfitting. It would be interesting to see how the Ladder mechanism can be used in this context. Finally, in the context of false discovery in the empirical sciences (Ioannidis, 2005; Gelman & Loken, 2014), one could imagine using the the Ladder mechanism as a way of keeping track of scientific progress on important public data sets.

Our algorithm can also be seen as an intuitive explanation for why overfitting to the holdout is sometimes not a major problem even in the adaptive setting. If indeed every analyst only uses the holdout set to test if their latest submission is well above the previous best, then they effectively simulate our algorithm.

A beautiful theoretical problem is to resolve the gap between our upper and lower bound. On the practical side, it would be interesting to use the Ladder mechanism in a real competition. One interesting question is if the Ladder mechanism actually encourages higher quality submissions by requiring a certain level of statistically significant improvement over previous submissions.

Acknowledgments

We thank Ben Hamner at Kaggle Inc., for providing us with the submission files from the Photo Quality competition, as well as many helpful discussions. We are grateful to Aaron Roth for pointing out an argument similar to that appearing in the proof of Theorem 3.1 in a different context. We thank John Duchi for many stimulating discussions.

References

Bassily, Raef, Smith, Adam, Steinke, Thomas, and Ullman, Jonathan. More general queries and less generalization error in adaptive data analysis. *CoRR*, abs/1503.04843, 2015.

Dwork, Cynthia, Feldman, Vitaly, Hardt, Moritz, Pitassi, Toniann, Reingold, Omer, and Roth, Aaron. Preserving statistical validity in adaptive data analysis. In *Proc. 47th Symposium on Theory of Computing (STOC)*. ACM, 2015.

Gelman, Andrew and Loken, Eric. The statistical crisis in science. *American Scientist*, 102(6):460, 2014.

Hardt, Moritz and Ullman, Jonathan. Preventing false discovery in interactive data analysis is hard. In *Proc. 55th Foundations of Computer Science (FOCS)*, pp. 454–463. IEEE, 2014.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *The Elements of Statistical Learning*. Springer, 2001.

Ioannidis, John P. A. Why Most Published Research Findings Are False. *PLoS Medicine*, 2(8):124, 2005. doi: 10.1371/journal.pmed.0020124.

Nissim, Kobbi and Stemmer, Uri. On the generalization properties of differential privacy. *CoRR*, abs/1504.05800, 2015.

Steinke, Thomas and Ullman, Jonathan. Interactive fingerprinting codes and the hardness of preventing false discovery. *CoRR*, abs/1410.1228, 2014.