The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Climbing the Kaggle Leaderboard by Exploiting the Log-Loss Oracle

Jacob Whitehill
Worcester Polytechnic Institute
jrwhitehill@wpi.edu

Abstract

In the context of data-mining competitions (such as those organized by Kaggle), we show how access to an oracle that reports a contestant's log-loss score on the test set can be exploited to deduce the ground-truth of some of the test examples. By applying this technique iteratively to batches of m examples (for small m), all of the test labels can eventually be inferred. In this paper, (1) We demonstrate this attack on the first stage of a recent Kaggle competition (Intel & MobileODT Cancer Screening) and use it to achieve a log-loss of 0.00000 (and thus attain a rank of #4 out of 848 contestants), without ever downloading the training set or training a classifier to solve the actual task. (2) We prove an upper bound on the batch size m as a function of the floating-point resolution of the probability estimates that the contestant submits for the labels. (3) We derive, and demonstrate in simulation, a more flexible attack that can be used even when the oracle reports the accuracy on an unknown (but fixed) subset of the test set's labels. These results underline the importance of evaluating contestants based only on test data that the oracle does not examine.

Introduction

Data-mining competitions such as those as offered by Kaggle, DrivenData, KDD Cup, ImageNet LS-VRC (Russakovsky et al. 2015), and other organizations, have become a mainstay of machine learning. By establishing common rules of participation as well as training and testing datasets that are shared by all contestants, these competitions can help to advance the state-of-the-art of machine learning and big data analytics practice in a variety of application domains (Mangal and Kumar 2016; Kiss-Tóth and Takács 2014; De Cock et al. 2013). In order for the scientific results of these contests to have value, however, it is imperative that the methods by which candidates are evaluated be sound. The importance of fair evaluation is made more pressing by the availability of oracles, often provided by the organizers of the competitions themselves, that return the accuracy or loss value of the contestant's guesses with respect to the test labels. The purpose of such oracles is to help participants to pursue more promising algorithmic strategies and to improve the overall quality of contestants' submissions. But they also open up the possibility of systematic overfitting, either inadvertently or maliciously.

In this paper, we consider how an oracle that returns the *log-loss* of a contestant's guesses w.r.t. the ground-truth labels of a test set, can be exploited by an attacker to infer the test set's true labels. The log-loss is mathematically convenient because, unlike other metrics such as the AUC (Tyler and Chen 2000; Agarwal et al. 2005), which is calculated over *pairs* of examples, the log-loss can be computed for each example separately. Moreover, unlike the 0-1 loss that conveys only the number of correctly labeled examples, the log-loss measures how "close" the contestant's probability estimates are to ground-truth. The attack proposed in our paper can be effective despite limited floating-point resolution in the oracle's return values, and can be applied even if the oracle only computes the log-loss on an unknown (but fixed) subset of the test set. In a case study we performed for this paper, we applied the attack to achieve a perfect score on a recent Kaggle competition (Intel & MobileODT Cervical Cancer Screening), thereby attaining a rank on the first-stage competition leaderboard of #4 out of 848.

To be fair, Kaggle had structured their competition rules such that the first stage was mostly for informational purposes to let contestants know how their algorithmic approachs were faring compared to their competitors'. Moreover, leaderboard algorithms such as Ladder (Blum and Hardt 2015) could prevent many kinds of attacks, including the one we present in this paper. On the other hand, while many competition organizers employ sufficient safeguards, some currently do not: some competitions run by DrivenData[1], for example, evaluate contestants on test examples that are available to contestants during the entire competition. We are also unaware of any competition that currently uses robust leaderboard algorithms such as Ladder. Finally, even a temporary high ranking in a data-mining contest could conceivably hold ancillary value, e.g., by inducing a potential employer to take a look at a particular person's curriculum vitae. In any case, **the potential of exploiting a competition oracle underlines the importance of employing commonsense safeguards to preserve the integrity of contestant rankings**. In particular, the results in this paper suggest that evaluation of contestants' performance should

[1]https://www.drivendata.org/

be done strictly on test examples on which the oracle never reported accuracy.

Related work

Both intentional hacking (Whitehill 2016; Blum and Hardt 2015; Zheng 2017; Hardt 2017) and inadvertent overfitting (Dwork et al. 2015; Hardt and Ullman 2014) to test data in adaptive data analyses – including but not limited to data-mining competitions – have generated recent research interest in the privacy-preserving machine learning and computational complexity theory communities. In particular, Blum and Hardt (Blum and Hardt 2015) recently described a "boosting" attack with which a contestant can estimate the test labels such that, with probability $2/3$, the accuracy of the inferred labels w.r.t. the ground truth is better than chance. The essence of their attack is to query the oracle many times using randomly generated guess vectors, and then to compute the majority vote of each example's label over only those guess vectors that achieved a better loss value than would be expected by chance. In this sense, the attack they explore is *probabilistic* and attempts to infer all of the ground-truth labels at the same time. In contrast, our paper explores how a contestant can ascertain the ground-truth labels of the test set *definitively*, by inferring just a few examples at at time. In addition to illustrating the attack itself, they devised a "Ladder" algorithm that can be used to rank contestants' performance in a way that is robust to the boosting attack. In later work by Zheng (Zheng 2017), the Ladder mechanism was simplified to avoid redundant computation; moreover, a sample complexity bound was shown relating the number of examples in the test set to the desired precision of the leaderboard loss values.

For the particular accuracy metric of the Area Under the Receiving Operating Characteristics Curve (AUC), Whitehill (Whitehill 2016) demonstrated how an oracle that returns exact information on the AUC of a contestant's guesses can be used to infer a *few* ground-truth labels of the test set, provided that the AUC is high enough. Moreover they showed in a tiny simulation ($n = 16$ examples) that, even without a high starting AUC, the accuracy of the contestant's guesses can be improved by integrating over all possible ground-truth labelings that are consistent with the AUC value returned by the oracle. On the other hand, they also proved a weak form of lower bound on the number of possible binary ground-truth labelings for which the contestant's guesses achieve any fixed AUC c; this essentially imposes a severe practical limit on the size of the test sets in which this kind of attack is feasible.

Notation and Assumptions

We assume that the contestant's goal is to infer the ground-truth labels of the test set, which contains n examples, each of which belongs to one of c possible classes. We represent the ground-truth of the entire test set using a row-wise 1-hot matrix $\mathbf{Y}_n \in \{0,1\}^{n\times c}$, where $\mathbf{Y}_n \doteq [\mathbf{y}_1, \dots, \mathbf{y}_n]^\top$, each $\mathbf{y}_i = [y_{i1}, \dots, y_{ic}]^\top$, each $y_{ij} \in \{0,1\}$, and $\sum_j y_{ij} = 1$ for each i. Similarly, we represent the contestant's guesses using matrix $\widehat{\mathbf{Y}}_n \in \mathbb{R}^{n\times c}$, where $\widehat{\mathbf{Y}}_n \doteq [\widehat{\mathbf{y}}_1, \dots, \widehat{\mathbf{y}}_n]^\top$, each $\widehat{\mathbf{y}}_i = [\widehat{y}_{i1}, \dots, \widehat{y}_{ic}]$, each $\widehat{y}_{ij} \in (0,1)$, and $\sum_j \widehat{y}_{ij} = 1$ for each i. The loss function that we study in this paper is the *log-loss*, computed by function f, of the contestant's guesses with respect to the ground-truth:

$$\ell_n \doteq f(\mathbf{Y}_n, \widehat{\mathbf{Y}}_n) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij}\log \widehat{y}_{ij} \qquad (1)$$

Furthermore, we assume that the data-mining competition offers an oracle to which a contestant can submit her/his real-valued guesses for the test labels and obtain the log-loss ℓ_n of the guesses with respect to the ground-truth.

Example

To show how knowledge of the log-loss can reveal information about the ground-truth itself, consider a tiny test set in which there are just 2 examples and 3 classes, and suppose a contestant submits the following guesses to the oracle (where e is the base of the natural logarithm):

$$\widehat{\mathbf{Y}}_2 = \begin{bmatrix} e^{-2} & e^{-1} & 1-e^{-2}-e^{-1} \\ e^{-8} & e^{-4} & 1-e^{-8}-e^{-4} \end{bmatrix}$$

If the oracle reports that the log-loss of the contestant's guesses, with respect to the ground-truth, is 3, then the ground-truth labeling of these two examples *must* be

$$\mathbf{Y}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

since this is the only value of $\mathbf{Y}_2$ that satisfies

$$f(\mathbf{Y}_2, \widehat{\mathbf{Y}}_2) = -\frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{3} y_{ij}\log \widehat{y}_{ij} = 3$$

Problem formulation and proposed solution

Here we describe an attack (summarized in Algorithm 1) with which a contestant can iteratively (in batches) infer the ground-truth of the test set. The algorithm is structured so that the contestant incorporates the inferred ground-truth labels into her/his guesses during subsequent rounds, thereby enabling her/him to "climb" the leaderboard while waging the attack. For simplicity of notation (and without loss of generality), we assume that the examples in the test set are ordered such that (1) the labels of the first k examples have already been inferred; (2) the labels of the next m examples (which we call the *probed* examples) are to be inferred in the current round; and (3) the rest $(n-m-k)$ of the test labels will remain uninferred during the current round.

Since f is computed by summing over all n examples, its definition can be re-written as

$$\begin{aligned} \ell_n &= f(\mathbf{Y}_n, \widehat{\mathbf{Y}}_n) && (2) \\ &= -\frac{1}{n}\Bigg[\sum_{i=1}^{k}\sum_{j} y_{ij}\log\widehat{y}_{ij} + \sum_{i=k+1}^{k+m}\sum_{j} y_{ij}\log\widehat{y}_{ij} + \\ &\qquad \sum_{i=k+m+1}^{n}\sum_{j} y_{ij}\log\widehat{y}_{ij}\Bigg] && (3) \end{aligned}$$

If we assume that the first k examples have already been inferred correctly so that $\widehat{y}_{ij} = y_{ij}$ for each j and each $i \leq k$, then the first term in the RHS of Equation 2 is approximately 0.[2] Moreover, if we set $\widehat{y}_{ij}$ to $1/c$ for each j and each $i = k+m+1, \ldots, n$, then the third term in the RHS equals $-1 \times (n-m-k) \times \log c$ – regardless of the ground-truth labels of these examples. Based on these facts, we can deduce the log-loss ℓ_m due to *only* the m probed examples to be:

$$\begin{aligned} \ell_m &\doteq \frac{1}{m} \sum_{i=k+1}^{k+m} \sum_j y_{ij} \log \widehat{y}_{ij} \\ &= \frac{1}{m} \left((n-m-k) \log c - n \times \ell_n \right) \end{aligned} \quad (4)$$

Given knowledge of ℓ_m – which can be calculated from the value ℓ_n returned by the oracle – and given knowledge of $[\widehat{\mathbf{y}}_{k+1}, \ldots, \widehat{\mathbf{y}}_{k+m}]^\top$ – which the candidate her/himself controls – the ground-truth of the m probed examples can be inferred using exhaustive search over all possible c^m labelings (for small m). Over $\lceil n/m \rceil$ consecutive rounds, we can determine the ground-truth values of *all* of the test labels.

Algorithm 1 Infer the ground-truth of the test set using the oracle's response ℓ_n.

```
Input: A probe matrix $\mathbf{G}_m$, where $m$ is the number of probed examples in each round.
Input: An oracle that reports $f(\mathbf{Y}_n, \widehat{\mathbf{Y}}_n)$.
Output: The ground-truth labels $\mathbf{Y}_n$ for all $n$ examples.
  for round $r = 1, \ldots, \lceil n/m \rceil$ do
    1. Set $k \leftarrow (r-1) \times m$.
    2. Configure the probe matrix $\widehat{\mathbf{Y}}_n$:
    •  For $i = 1, \ldots, k$ (examples that have already been inferred), set $\widehat{\mathbf{y}}_i$ to the inferred $\mathbf{y}_i$.
    •  For $i = k+1, \ldots, k+m$ (the probed examples), set $\widehat{\mathbf{y}}_i$ to the corresponding row of $\mathbf{G}_m$.
    •  For $i = k+m+1, \ldots, n$ (examples that will remain uninferred), set $\widehat{y}_{ij}$ to $1/c$ for all $j$.
    3. Submit $\widehat{\mathbf{Y}}_n$ to oracle and obtain $\ell_n$.
    4. Compute $\ell_m$ (the loss on just the $m$ probed examples) according to Equation 4.
    5. Determine the ground-truth of the probed examples by finding $[\mathbf{y}_{k+1}, \ldots, \mathbf{y}_{k+m}]^\top$ that minimizes $\epsilon$.
  end for
```

Note that, due to the finite floating-point resolution of the oracle's return value, there is usually a small difference between the ℓ_m that is calculated based on the oracle's response and the true log-loss value of the m probed examples. We call this difference the *estimation error*:

$$\epsilon \doteq |\ell_m - f([\mathbf{y}_{k+1}, \ldots, \mathbf{y}_{k+m}]^\top, [\widehat{\mathbf{y}}_{k+1}, \ldots, \widehat{\mathbf{y}}_{k+m}]^\top)|$$

[2]In practice, competition oracles often enforce that each $\widehat{y}_{ij} \in [\gamma, 1-\gamma]$ where γ is a small number such as 1×10^{-15}; this results in a negligible cost of $k \log(1-(c-1)\gamma)$ for the second term. We discuss this later in the paper.

How to choose the guesses of the probed examples

The key to exploiting the oracle so as to infer the ground-truth correctly is to choose the guesses $\widehat{\mathbf{Y}}_n$ so that ℓ_m reveals the labels of the probed examples uniquely. To simplify notation slightly, we let $\mathbf{G}_m \doteq [\widehat{\mathbf{y}}_{k+1}, \ldots, \widehat{\mathbf{y}}_{k+m}]^\top$ – which we call the *probe matrix* – represent the contestant's guesses for the m probed examples. The probe matrix can stay the same across all submission rounds.

If the contestant's guesses were real numbers in the mathematical sense – i.e., with infinite decimal resolution – then $\mathbf{G}_m$ could be set to random values, constrained so that each row sums to 1. With probability 1, the log-loss ℓ_m would then be unique over all possible instantiations of $[\mathbf{y}_{k+1}, \ldots, \mathbf{y}_{k+m}]^\top$. However, in practice, both the guesses and the log-loss reported by the oracle have finite precision, and "collisions" – different values of the ground-truth that give rise to the same, or very similar, losses – could occur. Consider, for example, the probe matrix below:

$$\begin{bmatrix} 0.53595382 & 0.20743777 & 0.25660840 \\ 0.76336402 & 0.17982958 & 0.05680643 \\ 0.83539897 & 0.02825473 & 0.13634628 \\ 0.88845736 & 0.10858667 & 0.00295598 \end{bmatrix}$$

Here, no two elements are closer than 0.025 apart. Yet two possible values for the ground-truth labeling $\mathbf{Y}_4$ result in log-loss values (approximately 1.18479 and 1.18488, respectively) that are less than 10^{-4} apart; these candidate labelings are

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

If the oracle returned a log-loss of, say, 1.185, then it would be ambiguous which of the two values of $\mathbf{Y}_4$ was the correct one.

Collision avoidance

In order to avoid collisions, we need to choose $\mathbf{G}_m$ so that the minimum distance – over all possible pairs of different ground-truth labelings of the m examples – is large enough so that even a floating-point approximation of ℓ_m can uniquely identify the ground-truth. We can thus formulate a constrained optimization problem in which we express the *quality* Q of $\mathbf{G}_m$ as:

$$Q(\mathbf{G}_m) \doteq \min_{\mathbf{Y}_m \neq \mathbf{Y}'_m} |f(\mathbf{Y}_m, \mathbf{G}_m) - f(\mathbf{Y}'_m, \mathbf{G}_m)| \quad (5)$$

where $\mathbf{Y}_m, \mathbf{Y}'_m$ are *distinct* ground-truth labelings, and we wish to find

$$\mathbf{G}^*_m \doteq \arg\max_{\mathbf{G}_m} Q(\mathbf{G}_m)$$

subject to the constraints that each row of $\mathbf{G}_m$ be a probability distribution. Optimization algorithms do exist to solve constrained minimax and maximin problems (Madsen and Schjær-Jacobsen 1978; Brayton et al. 1979). However, in practice, the optimization problem above presents difficulties: computing the minimum loss difference over all unique pairs of ground-truths requires a minimax objective function with $c^m(c^m-1)/2$ components, which grows large very fast. Alternatively, the set of loss values (over all unique ground-truths) can be sorted, and then the minimum distance can be computed using just c^m-1 operations. But this latter implementation is not differentiable, and hence finite differencing must be employed, which is also slow. Instead, to optimize the minimax objective, we resorted to a heuristic that is designed to sample the guesses $\widehat{y}_{ij}$ so that the sum of the logarithms of a randomly chosen subset of the guesses – one for each example i – is far apart within the range of 32-bit floating-point numbers. In particular, we set $\widehat{y}_{ij}=a\times 10^b$ and sampled $a\sim\mathcal{U}([0,1])$ and $b\sim\mathcal{U}(\{-14,13,\ldots,-1,0\})$ for each i and each $j<c$. For $j=c$, we sampled $a\sim\mathcal{U}([0,1])$ and b was fixed to 0. Finally, we normalized each $\widehat{\mathbf{y}}_i$ so that the entries sum to 1. Based on this heuristic, we used Monte-Carlo sampling (with 10000 samples) to optimize the maximin expression above. In particular, for $m=6$, we obtained a matrix $\mathbf{G}_6$ for which $Q(\mathbf{G}_6)=0.00152$. This number is substantially greater than the largest estimation error we ever encountered during our attacks (see section below describing our demonstration on Kaggle) and thus enabled us to conduct our attack in batches of 6 probed excamples. However, for $m=7$, we were never able to find a $\mathbf{G}_7$ for which $Q(\mathbf{G}_7)>0.0001$.

Bound on number of probed examples m

The oracles in data-mining competitions such as Kaggle often impose lower and upper bounds on the submitted probabilities so that $\widehat{y}_{ij}\in[\gamma,1-\gamma]$ for each i,j, where γ is a small number such as 10^{-15}. This ensures that the log-loss is well defined (so that $\log 0$ is never evaluated) but also indirectly imposes an upper bound on how many examples m can be probed during each round. Below we show a upper bound on the quality of a probe matrix $\mathbf{Q}_m$ as a function of γ:

Proposition 1. *Let m be the number of probed examples and let c be the number of possible classes. Let $\gamma\in(0,1)$ represent the minimum value, imposed by the oracle, of any guess $\widehat{y}_{ij}$. Then the quality $Q(\mathbf{G}_m)$ of any probe matrix $\mathbf{G}_m$ is bounded above by $\frac{\log(1-(c-1)\gamma)-\log\gamma}{c^m-1}$.*

Proof. Since γ is the minimum value of any element in $\mathbf{G}_m$, then $1-(c-1)\gamma$ is the maximum value. Each of the m probed examples must therefore contribute at least $-\log(1-(c-1)\gamma)$ and at most $-\log\gamma$ to the log-loss. Averaged over all m examples, the log-loss must therefore be in the closed interval

$$\begin{aligned} I &\doteq \left[-\frac{1}{m}\sum_i \log(1-(c-1)\gamma),\quad -\frac{1}{m}\sum_i \log\gamma\right] \\ &= [-\log(1-(c-1)\gamma),-\log\gamma] \end{aligned}$$

Since there are c classes, then there are c^m possible ground-truth labelings and corresponding log-losses. The maximum value of $Q(\mathbf{G}_m)$ – i.e., the minimum distance, over all possible pairs of distinct ground-truth labelings, between corresponding log-loss values – is attained when the log-losses are distributed across I so that the c^m-1 "gaps" between consecutive pairs of log-loss values are equal in size. Therefore, the maximum value of $Q(\mathbf{G}_m)$ is at most

$$\delta=\frac{\log(1-(c-1)\gamma)-\log\gamma}{c^m-1}$$

□

Since δ decreases exponentially in m, and since δ must be kept larger than the maximum estimation error ϵ observed when executing Algorithm 1, then m must necessarily be kept small.In practice we were not able to find satisfactory $\mathbf{G}_m$ even for $m\geq 7$. Nonetheless, even with $m=6$, we were able to climb the leaderboard of a recent Kaggle competition successfully.

Experiment: Kaggle Competition

We tested Algorithm 1 on the Intel & MobileODT Cervical Cancer Screening competition hosted by Kaggle in May-June 2017. The objective of the competition was to develop an automatic classifier to analyze cervical scans of women who are at-risk for cervical cancer and to predict the most effective treatment based on the scan. Such a classifier could potentially save many lives, especially in rural parts of the world in which high-quality medical care is lacking. During the first stage of the contest, the competition website provided each contestant with training images (1821) and associated training labels (with $c=3$ categories), as well as testing images ($n=512$). The goal of the competition was to predict the test labels with high accuracy. To help competitors identify the most promising classification methods, Kaggle provided an oracle – which each contestant could query up to 5 times per day – that reported the log-loss on *all* 512 test examples *without* any added noise. After the first-stage submission deadline (June 14, 2017), the second stage of the competition began, using a larger test set and an oracle that reported the loss on only a fixed subset of the test samples.

Cheating during the first stage: Since the oracle during the first stage of the competition returned the log-loss on the *entire* test set, it provided an ideal environment in which to demonstrate Algorithm 1. We performed the attack in phases according to the following procedure: For the first 2 queries, we probed only a single test label (i.e., $m=1$) just to verify that our code was working correctly. For the next 30 queries, we probed $m=4$ labels (using the $\mathbf{G}_4$ shown in the appendix). For the remaining queries (after we had found $\mathbf{G}_6$ with large enough Q), we probed $m=6$ examples per query. The maximum estimation error ϵ, over all rounds, between the log-loss returned by the oracle and the loss calculated based on the inferred ground-truth, was less than 0.0061. This was less than half of $Q(\mathbf{G}_6)$ for the probe matrix we used and thus allowed us to infer the ground-truth unambiguously. During the competition we did not perform any supervised learning of cervical scan images (i.e., the intended purpose of the competition) whatsoever.

Results: The progression of our attack is shown in Figure 1. In short, with less than 100 oracle queries (well within the oracle query limit given the total duration of the competition), we were able to infer the ground-truth labels of all the test examples perfectly. We note that, during consecutive iterations of the attack, the attained log-loss need not always decrease – this is because the the reduction in loss due to inferring more examples can sometimes be dwarfed by an increase in loss due to which specific entries of $\mathbf{G}_m$ are selected by the corresponding ground-truth of the probed examples. Nevertheless, the proposed attack is able to recover perfectly all m probed labels during every round. The progression of (usually decreasing) log-loss values ℓ_n is plotted in Figure 1. By the last iteration, we had recovered the ground-truth values of all 512 examples correctly and thus attained a loss of 0.00000. Since we were tied with several other contestants who also achieved the same loss (whether by legitimate means or by cheating), and since Kaggle apparently uses the *submission time* as a secondary rank basis (and we were not the first party to achieve a perfect score), we were ranked in 4th place on the first-stage leaderboard (see Figure 2).

Second stage: During the second stage of the same Kaggle competition, the organizers published a larger test set that included the 512 test examples from the first-stage as a subset. Moreover, these same 512 examples were the basis of both the oracle results and the leaderboard rankings up until the conclusion of the second-stage competition. Hence, for a brief period of about two weeks, we were able to maintain the illusion of a top-ranked Kaggle competitor achieving a perfect score. To be clear: the *final*, *definitive* results of the competition (announced on June 21, 2017) – including who won the \$100,000 prize money – were based on the log-loss on the *entire* test set, not just the subset. Naturally, our ranking declined precipitously at this point (to 225th place out of 848 contenders) since our guesses on the remaining 75% examples were just $1/c$.

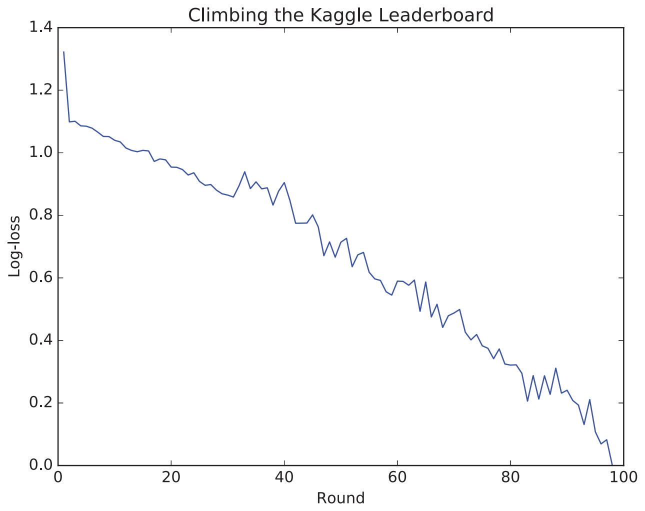


Figure 1: Climbing the Kaggle leaderboard, using the proposed log-loss oracle exploitation algorithm, of the Intel & MobileODT Cervical Cancer Screening 2017 competition (first stage). After 98 rounds of Algorithm 1, we had inferred all 512 test labels correctly and thereby achieved a log-loss of 0.00000.

Cheating when the Oracle Reports Accuracy on a Subset of Examples

It is more common in data-mining competitions for the oracle to report accuracy on only a *subset* of the test set. Here we describe how a contestant can still cheat, using similar methods as described above, when the oracle reports accuracy on a *fixed subset* of examples (i.e., the same subset for each oracle query). In this setting, the contestant submits a matrix $\widehat{\mathbf{Y}}_n$ with n rows, but the log-loss obtained from the oracle is based on only $s \leq n$ examples. Note that, in contrast to Algorithm 1, here we treat "already inferred" examples in the same way as the "uninferred" examples – we assign their guesses to be $\widehat{y}_{ij} = 1/c$ for all i, j (instead of setting them to their inferred values). The only drawback of this simplification is that the attacker cannot simultaneously infer the ground-truth *and* decrease her/his log-loss (in the manner illustrated by Figure 1) – rather, the contestant must wait until after she/he has inferred the ground-truth to "cash in" and jump to a higher leaderboard rank. We describe the new attack below:

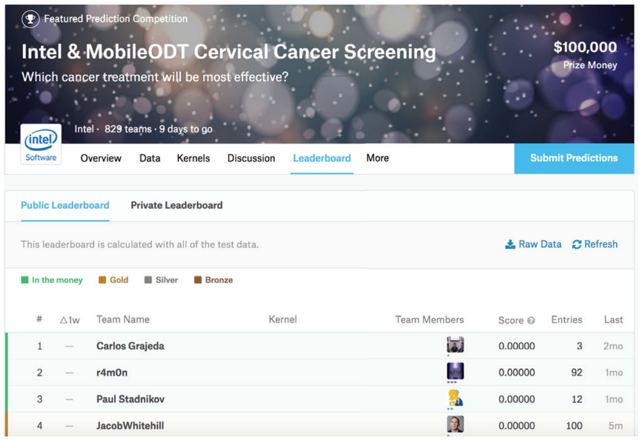
Determining the size of the subset s: The first step of the attack is to determine the value of s. To this end, it is useful to identify a *single* test example that is definitely in the s-element subset on which the oracle reports accuracy. Finding such an example can be achieved by setting the guesses $\widehat{y}_{ij}$ to $1/c$ for all but one example and setting the guesses to random values (but not equal to $1/c$) for a single "probe" example i. If the loss reported by the oracle is *not* equal to $-\log c$, then the probe example must be one of the s evaluated examples; otherwise, another example is chosen and the procedure is repeated. Assuming that the fraction s/n is not too small, then this procedure should only take a few oracle queries.

Given a single example at index i that is known to be among the s evaluated examples, along with the log-loss value ℓ_s summed over all s evaluated examples, the contestant can determine s. To see how, notice that the $s-1$ examples that are *not* example i contribute a log-loss of $\frac{s-1}{s}\log c$, and that example i contributes $-\frac{1}{s}y_{ij}\log\widehat{y}_{ij}$. Therefore, the contestant can iterate (jointly) over all n possible values for $s \in \{1, \ldots, n\}$ *and* all c possible values of $\mathbf{y}_i$ to find

$$\arg\min_s \left\{ \min_{\mathbf{y}_i} \left| \frac{1}{s}\left((s-1)\log c - \sum_j y_{ij}\log\widehat{y}_{ij} \right) - \ell_s \right| \right\} \tag{6}$$

The solution is the number of examples s on which the oracle reports the log-loss. We note that, in practice, due to finite resolution of the oracle's response and the contestant's guesses, the inferred value of s can sometimes be inaccurate. Nevertheless, even an imperfect estimate of s can often be used to infer the ground-truth of the s evaluated test examples with high accuracy (see simulation results below).

Inferring the labels of a batch of probe examples: Now that s has been inferred, the contestant can probe the labels of m examples at a time. To infer $\mathbf{Y}_m$, the contestant



Intel & MobileODT Cervical Cancer Screening

Which cancer treatment will be most effective?

$100,000 Prize Money

Intel · 829 teams · 9 days to go

This leaderboard is calculated with all of the test data.

🟩 In the money 🟧 Gold ⬜ Silver 🟫 Bronze

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	Carlos Grajeda			0.00000	3	2mo
2	—	r4m0n			0.00000	92	1mo
3	—	Paul Stadnikov			0.00000	12	1mo
4	—	JacobWhitehill			0.00000	100	5m

Figure 2: Our leaderboard rank on the *first* stage of the Intel & MobileODT Cervical Cancer Screening 2017 climbed to #4 after we inferred the labels of all the test examples using Algorithm 1.

must consider whether each probed example i is in the s-element subset on which the oracle reports the log-loss. To this end, we define $\mathbf{z} \in \{0,1\}^m$ so that z_i is 1 if example i is in the s-element subset and 0 otherwise. The L_1-norm $\|\mathbf{z}\|_1$ of this vector thus equals the number of probed examples that are also in the s-element subset, and we can compute the contribution of the probed examples to the log-loss as $-\frac{1}{s}\sum_i \sum_j z_i y_{ij} \log \widehat{y}_{ij}$. The remaining log-loss is accounted for by the $(s - \|\mathbf{z}\|_1)$ "uninferred" examples and amounts to $\frac{s-\|\mathbf{z}\|_1}{s} \log c$. Therefore, to determine $\mathbf{Y}_m$ and $\mathbf{z}$, the contestant must optimize

$$\arg\min_{\mathbf{Y}_m} \left\{ \min_{\mathbf{z}\in\{0,1\}^m} \left| \left(-\frac{1}{s} \sum_i \sum_j z_i y_{ij} \log \widehat{y}_{ij} + \frac{s - \|\mathbf{z}\|_1}{s} \log c \right) - \ell_s \right| \right\} \quad (7)$$

using brute-force search (which is easy since m is small). Naturally, row i of the inferred matrix $\mathbf{Y}_m$ is valid only if $z_i = 1$.

Choosing the guesses: Similar to the section above, we need to optimize the probe matrix so as to to minimize collisions. In this setting, however, we must be concerned not just with different possible ground-truth labelings $\mathbf{Y}_m$, but also with the indicator variables $\mathbf{z}$ – both of which "select" different elements $\widehat{y}_{ij}$ to add to the log-loss. We thus revise the quality function to maximize the minimum distance, over all distinct pairs $\mathbf{Y}_m \neq \mathbf{Y}'_m$ *and* over all distinct pairs $\mathbf{z} \neq \mathbf{z}'$, of the corresponding log-loss values (see appendix).

Simulation

Instead of applying the algorithm above to the Intel-MobileODT competition[3], we conducted a simulation. In particular, we simulated a test set containing $n = 2048$ examples (from $c = 3$ classes) where $s = 512$ evaluated examples were randomly sampled (but fixed over all oracle queries) across the entire test set. The simulated contestant first probed just single examples to find one of the s evaluated examples. Then, it inferred s by optimizing Equation 6. The contestant then proceeded to submit batches of $m = 4$ probed examples (for $\lceil n/m \rceil$ total rounds) and infer their labels based on the oracle's response by optimizing Equation 7 using exhaustive search.

To assess how the floating-point resolution p of the oracle's response impacts the accuracy of the labels inferred by the contestant, we varied $p \in \{1, 2, 3, 4, 5\}$, where p was the number of digits after the decimal point. ($p = 1$ means that the oracle's responses were rounded to the nearest 0.1; $p = 2$ to the nearest 0.01, etc.). For each p value, we conducted 100 simulations. As the probe matrix for all simulations, we used $\widetilde{G}_4$ (see appendix). At the end of each simulation, we computed the accuracy of the inferred labels versus ground-truth, and then averaged the accuracy rate over all 100 simulations.

Results: Results are shown (for each p) in Table 1, where columns 2-4 represent the median, mean, and standard deviation of the accuracy (% correct) of the inferred test labels, and columns 5-7 represent the same statistics for the accuracy (absolute difference) between the true s (the number of evaluated examples in the test set that is needed in Equation 7) and inferred $\hat{s}$. As expected, the accuracy of inferred labels increased with higher floating-point resolution p. In many simulations, the contestant's best inference of s was incorrect, and yet many (and often most) of the inferred test labels were still correct. In fact, the average correlation (over all values of p) between $|s - \hat{s}|$ and the accuracy of the inferred test labels w.r.t. ground-truth, was only $-.107$, suggesting that correct inference of the test labels is relatively robust to errors in inference of s.

Summary and Conclusions

We derived an iterative algorithm whereby a contestant can illicitly improve her/his leaderboard score in a data-mining competition by exploiting information provided by an oracle that reports the log-loss of the contestant's guesses w.r.t. the ground-truth labels of the test set. During each iteration of the attack, the contestant infers the ground-truth labels of batches of m probed examples of the test set. We proved that the maximum batch size m whose labels can be inferred in each round is fundamentally limited by the floating-point resolution of the contestant's guesses. Nevertheless, the attack is practical, and we demonstrated it on a recent Kaggle competition and thereby attained a leaderboard ranking of #4 out of 848 (for the first stage of the contest), without ever even downloading the training or testing data. For more general scenarios in which the oracle reports the log-loss on only a fixed subset of test examples, we derived a second algorithm and showed in simulation that it can infer the ground-truth of the test examples with high accuracy, despite inaccuracy in inferring which examples are part of the

[3]The oracle in the second stage of the Intel-MobileODT 2017 competition evaluated only a subset of the test examples, but it turned out that this subset was exactly the 512 images from the first-stage test set; hence, there was nothing new to infer.

Table 1: Results of simulations (for varying floating-point resolution p of the oracle's response) in which the contestant tries to infer the test labels based on a log-loss score that is calculated based only on a random, but fixed, subset of s test examples. Results for each p are averaged over 100 simulations.

	Label Accuracy (% correct)			s Accuracy ($\lvert s - \hat{s} \rvert$)		
p	Median	Mean	StdDev	Median	Mean	StdDev
1	39.46	38.77	3.66	1181.50	1170.27	144.84
2	48.89	47.76	5.54	1126.00	1060.82	333.74
3	59.72	59.40	4.81	1445.50	1129.56	662.61
4	80.41	78.71	13.73	3.00	204.55	444.78
5	100.00	93.55	11.92	0.00	169.85	434.34

evaluated subset.

In terms of practical implications, our findings further underline the importance of evaluating contestants in data-mining competitions based only on test examples that the loss/accuracy oracle never examined. Other mechanisms intended to make rankings more robust, such as imposing a limit on the number of oracle queries submitted per day, can easily be circumvented by registering multiple contestant accounts.

Future work: According to the rules for the Intel & MobileODT Cancer Screening competition, the winning contestant is obliged (before receiving the prize money) to submit her/his *training code* that she/he used to train the classifier that won the competition. Without any modification to either the code itself or the hyperparameter settings, the training code should be able to re-generate a classifier that can produce the same set of guesses as those used to win the competition. The point here is to ensure that *bona fide* machine learning research was conducted in the spirit intended by the competition organizers, rather than just competition hacking. Future research on data-mining competition hacking could examine how even this safeguard might be overcome. In particular, it would be interesting to explore how a neural network architecture and hyperparameter settings might be "reverse-engineered", based on both the training and testing sets, to yield a relatively simple neural network design (so as not to arouse suspicion) that performs well on both the training and testing sets, but not necessarily on independent, held-out data.

Appendix: Probe Matrices $\mathbf{G}_m$

For the experiment on the Kaggle competition, we used the matrices $\mathbf{G}_4$ and $\mathbf{G}_6$ shown below (note that e here means "times 10 to the power…"):

$$\mathbf{G}_4 = \begin{bmatrix} \text{3.17090802e-01} & \text{6.03843391e-01} & \text{7.90658068e-02} \\ \text{3.34653412e-01} & \text{6.64893789e-01} & \text{4.52799011e-04} \\ \text{4.44242183e-01} & \text{5.42742523e-01} & \text{1.30152938e-02} \\ \text{3.02254057e-01} & \text{1.41415552e-01} & \text{5.56330391e-01} \end{bmatrix}$$

and $Q(\mathbf{G}_4) = 0.019248$.

$$\mathbf{G}_6 = \begin{bmatrix} \text{3.72716316e-13} & \text{3.17270110e-06} & \text{9.99996841e-01} \\ \text{4.03777185e-11} & \text{2.98306441e-06} & \text{9.99997020e-01} \\ \text{1.51235222e-11} & \text{9.45069790e-02} & \text{9.05493021e-01} \\ \text{7.54659835e-10} & \text{6.77224932e-07} & \text{9.99999344e-01} \\ \text{1.84318694e-09} & \text{2.37398371e-01} & \text{7.62601614e-01} \\ \text{9.75336131e-12} & \text{1.44393380e-06} & \text{9.99998569e-01} \end{bmatrix}$$

and $Q(\mathbf{G}_6) = 0.001526$.

For the simulation of our second algorithm that can infer the ground-truth on a random (but fixed) subset of test examples, we defined a new quality function

$$\widetilde{Q}(\widetilde{\mathbf{G}}_m) \doteq \min_{\substack{\mathbf{z}_m, \\ \mathbf{Y}_m \neq \mathbf{Y}'_m}} |\widetilde{f}(\mathbf{z}_m, \mathbf{Y}_m, \mathbf{G}_m) - \widetilde{f}(\mathbf{z}_m, \mathbf{Y}'_m, \mathbf{G}_m)|$$

where

$$\widetilde{f}(\mathbf{z}_m, \mathbf{Y}_m, \widehat{\mathbf{Y}}_m) = -\frac{1}{m} \sum_i \sum_j z_i y_{ij} \log \widehat{y}_{ij}$$

In our simulated attack against the oracle, we used:

$$\widetilde{\mathbf{G}}_4 = \begin{bmatrix} \text{3.34296189e-02} & \text{6.06806998e-06} & \text{9.66564298e-01} \\ \text{6.80901580e-15} & \text{8.52564275e-02} & \text{9.14743602e-01} \\ \text{1.78242549e-01} & \text{2.03901175e-12} & \text{8.21757436e-01} \\ \text{1.22676250e-02} & \text{1.40922994e-03} & \text{9.86323118e-01} \end{bmatrix}$$

and $\widetilde{Q}(\widetilde{\mathbf{G}}_4) = 0.012750$.

References

Agarwal, S.; Graepel, T.; Herbrich, R.; Har-Peled, S.; and Roth, D. 2005. Generalization bounds for the area under the ROC curve. In *Journal of Machine Learning Research*, 393–425.

Blum, A., and Hardt, M. 2015. The ladder: A reliable leaderboard for machine learning competitions. *arXiv preprint arXiv:1502.04585*.

Brayton, R.; Director, S.; Hachtel, G.; and Vidigal, L. 1979. A new algorithm for statistical circuit design based on quasi-newton methods and function splitting. *IEEE Transactions on Circuits and Systems* 26(9):784–794.

De Cock, M.; Roy, S. B.; Savvana, S.; Mandava, V.; Dalessandro, B.; Perlich, C.; Cukierski, W.; and Hamner, B. 2013. The microsoft academic search challenges at kdd cup 2013. In *Big Data, 2013 IEEE International Conference on*, 1–4. IEEE.

Dwork, C.; Feldman, V.; Hardt, M.; Pitassi, T.; Reingold, O.; and Roth, A. L. 2015. Preserving statistical validity in adaptive data analysis. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, 117–126. ACM.

Hardt, M., and Ullman, J. 2014. Preventing false discovery in interactive data analysis is hard. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, 454–463. IEEE.

Hardt, M. 2017. Climbing a shaky ladder: Better adaptive risk estimation. *arXiv preprint arXiv:1706.02733*.

Kiss-Tóth, C., and Takács, G. 2014. A dynamic programming approach for 4d flight route optimization. In *Big Data (Big Data), 2014 IEEE International Conference on*, 24–28. IEEE.

Madsen, K., and Schjær-Jacobsen, H. 1978. Linearly constrained minimax optimization. *Mathematical Programming* 14(1):208–223.

Mangal, A., and Kumar, N. 2016. Using big data to enhance the bosch production line performance: A kaggle challenge. In *Big Data (Big Data), 2016 IEEE International Conference on*, 2029–2035. IEEE.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 1–42.

Tyler, C., and Chen, C.-C. 2000. Signal detection theory in the 2AFC paradigm: attention, channel uncertainty and probability summation. *Vision Research* 40(22):3121–3144.

Whitehill, J. 2016. Exploiting an oracle that reports AUC scores in machine learning contests. In *AAAI*, 1345–1351.

Zheng, W. 2017. Toward a better understanding of leaderboard. *arXiv preprint arXiv:1510.03349*.