How Does Knowledge of the AUC Constrain the Set of Possible Ground-Truth Labelings?

Jacob Whitehill
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA, USA
jrwhitehill@wpi.edu

Abstract

Recent work on privacy-preserving machine learning has considered how datamining competitions such as Kaggle could potentially be "hacked", either intentionally or inadvertently, by using information from an oracle that reports a classifier's accuracy on the test set (Blum and Hardt 2015; Hardt and Ullman 2014; Zheng 2015; Whitehill 2016). For binary classification tasks in particular, one of the most common accuracy metrics is the Area Under the ROC Curve (AUC), and in this paper we explore the mathematical structure of how the AUC is computed from an n-vector of real-valued "guesses" with respect to the ground-truth labels. Under the assumption of perfect knowledge of the test set AUC $c = p/q$, we show how knowing c constrains the set $\mathcal{W}$ of possible ground-truth labelings, and we derive an algorithm both to compute the exact number of such labelings and to enumerate efficiently over them. We also provide empirical evidence that, surprisingly, the number of compatible labelings can actually *decrease* as n grows, until a test set-dependent threshold is reached. Finally, we show how $\mathcal{W}$ can be efficiently whittled down, through pairs of oracle queries, to infer all the ground-truth test labels with complete certainty.

Introduction and Related Work

Datamining contests such as Kaggle and KDDCup can accelerate progress in many application domains by providing standardized datasets and a fair basis of comparing multiple algorithmic approaches. However, their utility will diminish if the integrity of leaderboard rankings is called into question due to either intentional or accidental overfitting to the test data. Recent research on privacy-preserving machine learning (Blum and Hardt 2015; Zheng 2015) has shown how information on the accuracy of a contestant's guesses, returned to the contestant by an oracle, can divulge information about the test data's true labels. Such oracles are often provided by the organizers of the competition themselves. For example, in the 2017 Intel & MobileODT Cervical Cancer Screening competition (Kaggle 2017), every contestant can submit her/his guesses up to 5 times per day, and for each submission the oracle returns the log-loss of the guesses with respect to the ground-truth values of the entire 512-element test set. The contestant can use the accuracy information to improve (hopefully) the classifier design and then re-submit.

AUC: For binary classification problems, one of the most commonly used accuracy metrics is the Area Under the Receiver Operating Characteristics Curve (AUC). In contrast to other accuracy metrics such as log-loss and 0/1 loss, which can be computed as the sum of example-wise losses over each example in the test set, the AUC statistic is computed over all possible *pairs* of test examples, such that each pair contains one example from each class. In a recent paper (Whitehill 2016), we showed that an oracle that provides contestants with information on the AUC of their guesses can inadvertently divulge information on the ground-truth labels of the test examples. The AUC can help the contestant to deduce the labels of particular examples, as well as the number of negative and positive examples; we provide examples below.

Labels of specific examples: Suppose that a tiny test set contains just 4 examples; a contestant's real-valued guesses for these labels is $\hat{\mathbf{y}} = (0.2, 0.5, 0.9, 0.1)$; and an oracle informs the contestant that her/his guesses have achieved an AUC of exactly $0.75 = 3/4$. How does perfect knowledge of the AUC (i.e., no noise added) constrain the set of possible binary ground-truth vectors for the test set? In this example, it turns out that there is *exactly one* possible ground-truth vector – namely $\mathbf{y} = (1, 0, 1, 0)$ – for which the AUC of the contestant's guesses is exactly 0.75. Hence, based on a single oracle query, the contestant has managed to deduce the test labels with complete certainty.

Number of negative and positive examples: Suppose that the oracle reports an AUC of $c = p/q$, where p/q is a reduced fraction, i.e., p and q are relatively prime. If n_0 and n_1 are the numbers of negative and positive examples in the test set (respectively), then q must divide the total number $(n_0 n_1)$ of pairs containing one example from each class. Knowledge of q thus constrains the set $\{(n_0, n_1) \mid n_0 + n_1 = n\}$ of possibilities for n_0 and n_1. For instance, if $n = 1000$ and $c = p/q = 450/479 \approx 0.939$, then (n_0, n_1) must be in the set $\{(42, 958), (479, 521), (521, 479), (958, 42)\}$ – i.e., only 4 choices instead of 999.

These simple examples raise more general questions: For a test set with n examples and a fixed AUC of $c = p/q$ (where $p, q \in \mathbb{Z}$), how many compatible binary ground-truth vectors are there? Does this number grow monotonically in n, or might there exist some "pathological" combinations of the

number of test examples n, number of positively labeled examples n_1, and the contestant's AUC c, such that this number is small? If the number is small, can the solution candidates be enumerated efficiently? This paper explores these questions in some detail.

Related work: In recent years there has been growing interest in the statistical validity of scientific results that are obtained from *adaptive* data analyses, in which the results of one experiment inform the design of the next (Dwork et al. 2015; Hardt and Ullman 2014). For the particular application of datamining contests – in which contestants can submit their guesses to an oracle, receive information on their accuracy, revise their guesses, and resubmit – a potential danger is that the rankings and accuracy statistics of different contestants may be unreliable. Therefore, the design of algorithms to generate contest leaderboards that are robust to "hacking", whether intentional as part of an attack or inadvertently due to adaptive overfitting, has begun to generate significant research interest (Blum and Hardt 2015; Zheng 2015). Blum and Hardt 2015 proposed an algorithm ("Ladder") that can reliably estimate the accuracy of a contestant's classifier on the true test data distribution, even when the classifier has been adaptively optimized based on the output of an oracle on the empirical test distribution.

While the availability of an oracle in datamining contests presents potential problems, it is also useful for helping contestants to focus their efforts on more promising algorithmic approaches. Our research is thus related to privacy-preserving machine learning and differential privacy (e.g., (Dwork 2011; Chaudhuri and Monteleoni 2009; Blum, Ligett, and Roth 2013)), which are concerned with how to provide useful aggregate statistics without disclosing private information about particular examples in the dataset. The AUC statistic, in particular, has been investigated in the context of privacy: (Stoddard, Chen, and Machanavajjhala 2014) proposed an algorithm for computing "private ROC" curves and associated AUC statistics. (Matthews and Harel 2013) showed how an attacker who already knows most of the test labels can estimate the remaining labels if he/she gains access to an empirical ROC curve, i.e., a set of classifier thresholds and corresponding true positive and false positive rates.

In previous work (Whitehill 2016), we showed a weak form of lower bound on the number of possible binary ground-truth vectors $\mathbf{y} \in \{0,1\}^n$ for which the contestant's guesses $\hat{\mathbf{y}}$ achieve any fixed AUC c. Specifically, for every AUC value $c = p/q \in (0,1)$, there exists an infinite sequence of dataset sizes ($n = 4q, 8q, 12q, \ldots$) such that the number of satisfying ground-truth vectors $\mathbf{y} \in \{0,1\}^n$ grows exponentially in n. However, this result does not preclude the possibility that there might be certain pathological cases – combinations of p, q, n_0, and n_1 – for which the number of satisfying ground-truth vectors is actually much smaller. Conceivably, there might be values of n that lie between integer multiples of $4q$ for which the number of satisfying solutions is small. Moreover, the lower bound in (Whitehill 2016) applies only to datasets that contain at least $4q$ examples and says nothing about smaller (but possibly still substantial) datasets.

Main contributions: The novel contributions of our paper are the following: (1) We show how a pair of oracle queries can retrieve the ground-truth label of any test example k. (2) We derive an algorithm to compute the exact number of n-dimensional binary ground-truth vectors for which a contestant's real-valued vector of guesses achieves a fixed AUC, along with an algorithm to efficiently generate all such vectors. (3) Based on (1) and (2) above, we describe a novel attack to infer all the test set labels in typically far fewer than $2n$ oracle calls. (4) We provide empirical evidence that the number of satisfying binary ground-truth vectors can actually *decrease* with increasing n, until a test set-dependent threshold is reached.

Notation and Assumptions

Let $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$ be the ground-truth binary labels of n test examples, and let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n) \in \mathbb{R}^n$ be the contestant's real-valued guesses. Let $\mathcal{L}_1(\mathbf{y}) = \{i : y_i = 1\}$ and $\mathcal{L}_0(\mathbf{y}) = \{i : y_i = 0\}$ represent the index sets of the examples that are labeled 1 and 0, respectively. Similarly define $n_1(\mathbf{y}) = |\mathcal{L}_1(\mathbf{y})|$ and $n_0(\mathbf{y}) = |\mathcal{L}_0(\mathbf{y})|$ to be the number of examples labeled 1 and 0 in $\mathbf{y}$, respectively. For brevity, we sometimes write simply n_1, n_0, $\mathcal{L}_0$, or $\mathcal{L}_1$ if the argument to these functions is clear from the context.

We assume that the contestant's guesses $\hat{y}_1, \ldots, \hat{y}_n$ are all *distinct* (i.e., $\hat{y}_i = \hat{y}_j \iff i = j$). In machine learning applications where classifiers analyze high-dimensional, real-valued feature vectors, this is common.

Importantly, but without loss of generality, we assume that the test examples are ordered according to $\hat{y}_1, \ldots, \hat{y}_n$, i.e., $\hat{y}_i > \hat{y}_j \iff i > j$. This significantly simplifies the notation.

Finally, we assume that the oracle provides the contestant with *perfect* knowledge of the AUC $c = p/q$, where p/q is a reduced fraction (i.e., the greatest common factor of p and q 1) on the *entire* test set, and that the contestant knows both p and q.

AUC Accuracy Metric

The AUC has two mathematically equivalent definitions (Tyler and Chen 2000; Agarwal et al. 2005): (1) the AUC is the Area under the Receiver Operating Characteristics (ROC) curve, which plots the true positive rate against the false positive rate of a classifier on some test set. The ROC thus characterizes the performance of the classifier over all possible thresholds on its real-valued output, and the AUC is the integral of the ROC over all possible false positive rates in the interval $[0,1]$. (2) The AUC represents the fraction of *pairs* of test examples – one labeled 1 and one labeled 0 – in which the classifier can correctly identify the positively labeled example based on the classifier output. Specifically, since we assume that all of the contestant's guesses are distinct, then the AUC can be computed as:

$$\text{AUC}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n_0 n_1} \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1} \mathbb{I}[\hat{y}_i < \hat{y}_j] \qquad (1)$$

Equivalently, we can define the AUC in terms of the number of *misclassified pairs* h:

$$\text{AUC}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{h(\mathbf{y}, \hat{\mathbf{y}})}{n_0 n_1}$$

where

$$h(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1} \mathbb{I}[\hat{y}_i > \hat{y}_j]$$

As is evident in Eq. 1, all that matters to the AUC is the *relative ordering* of the $\hat{y}_i$, not their exact values. Also, if all examples belong to the same class and either $n_1 = 0$ or $n_0 = 0$, then the AUC is undefined. Finally, the AUC is a *rational* number because it can be written as the fraction of two integers p and q, where q must divide $n_0 n_1$.

Iterative Attack to Deduce Each Example

Here we show that the binary label y_k of each example k can be deduced through a pair of oracle queries. In particular, when deducing example k, we set $\hat{y}_k$ to either 1 or 0; set the guesses for the remaining $n-1$ examples all to distinct values (so that there are no ties) that are greater than 0 and less than 1; and then submit the vector of guesses to the oracle. If the AUC achieved by $\hat{\mathbf{y}}$ is higher when $\hat{y}_k = 1$ than when $\hat{y}_k = 0$, then we know that $y_k = 1$. Thus, in total $2n$ oracle queries are required to infer the labels of the entire test set definitively.

Proposition 1. *Let* $\mathbf{y}$ *be a vector of* n *binary labels, and let* $\hat{\mathbf{y}}$ *be the corresponding vector of guesses. Let* c_1 *and* c_0 *be the AUC resulting when we set* $\hat{y}_k = 1$ *and* $\hat{y}_k = 0$*, respectively, and the values of all the other guesses to any distinct values in the open interval* $(0, 1)$*. Then* $c_1 > c_0 \iff y_k = 1$.

Proof. Suppose $y_k = 1$. Based on Equation 1, we can compute the number of correctly classified pairs first *without* example k, then *with* example k, and then add them together:

$$\begin{aligned} n_0 n_1 c_1 &= \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1} \mathbb{I}[\hat{y}_i < \hat{y}_j] \\ &= \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1 \setminus \{k\}} \mathbb{I}[\hat{y}_i < \hat{y}_j] + \sum_{i \in \mathcal{L}_0} \mathbb{I}[\hat{y}_i < \hat{y}_k] \\ &= \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1 \setminus \{k\}} \mathbb{I}[\hat{y}_i < \hat{y}_j] \quad + n_0 \end{aligned}$$

since $\hat{y}_k = 1$ which is greater than all other guesses. By similar reasoning, we have:

$$n_0 n_1 c_0 = \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1 \setminus \{k\}} \mathbb{I}[\hat{y}_i < \hat{y}_j] + 0$$

since $\hat{y}_k = 0$ which is less than all other guesses. Therefore, $c_1 > c_0$. If $y_k = 0$, then an analogous argument proves that $c_0 > c_1$. □

This attack enables the inference of all n test examples using $2n$ queries. In the next section, we show how a *single* oracle query can sometimes narrow down the set of ground-truth possibilities more quickly.

Computing the Exact Number of Binary Labelings for which AUC=c

We are interested in determining the number of unique binary vectors $\mathbf{y} \in \{0,1\}^n$ such that the contestant's guesses $\hat{\mathbf{y}} \in \mathbb{R}^n$ achieve a fixed AUC of c. The bulk of the effort is to derive a recursive formula for the number of unique binary vectors with a *fixed* number n_1 of 1s that give the desired AUC value.

Intuition: Given a real-valued vector $\hat{\mathbf{y}}$ representing the contestant's guesses and a corresponding binary vector $\mathbf{y}$ representing the ground-truth test labels, the number $h(\mathbf{y}, \hat{\mathbf{y}})$ of misclassified pairs of examples (such that each pair contains one example from each class) can be increased by 1 by "left-swapping" any occurrence of 1 in $\mathbf{y}$ (at index j') with a 0 that occurs immediately to the left of it (i.e., at index $j'-1$) – see Figure 1. To generate a vector $\mathbf{y}$ such that $h(\mathbf{y}, \hat{\mathbf{y}}) = d$ for any desired $d \in \{0, \ldots, q\}$, we start with a vector $\mathbf{r}$ in "right-most configuration" – i.e., where all the 0s occur to the left of all the 1s – because (as we will show) $h(\mathbf{r}, \hat{\mathbf{y}}) = 0$. We then apply a sequence of multiple left-swaps to each of the 1s in $\mathbf{r}$, and count the number of ways of doing so such that the total number is d. Because we want to determine the number of *unique* vectors $\mathbf{y}$ such that $h(\mathbf{y}, \hat{\mathbf{y}}) = d$, we restrict the numbers $s_1, \ldots, s_{n_1}$ of left-swaps applied to the n_1 different 1s in $\mathbf{r}$ (where the first 1 is left-swapped s_1 times, the second 1 is left-swapped s_2 times, etc.) so that $s_i \geq s_j$ for all $i < j$. This results in a proof that the number of possible ground-truth binary labelings, for any given value of n_1 and for which a given vector of guesses misclassifies d pairs of examples, is equal to the number of points in a n_1-dimensional discrete simplex $\{(s_1, \ldots, s_{n_1}) \in \mathbb{Z}^{n_1} \mid \sum_i s_i = d, s_i \geq 0 \ \forall i\}$ that has been truncated by the additional constraint that $n_0 \geq s_1 \geq \ldots \geq s_{n_1}$.

To get started: Since we assume (without loss of generality) that the contestant's guesses are ordered such that $\hat{y}_i < \hat{y}_j \iff i < j$, then we can simplify the definition of h to be:

$$h(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i \in \mathcal{L}_0} \sum_{j \in \mathcal{L}_1} \mathbb{I}[i > j] \tag{2}$$

Now we define "left-swap" and "right-most configuration" more precisely:

Definition 1. *For any* $\mathbf{y} \in \{0,1\}^n$ *and* $i \in \{2, \ldots, n\}$ *where* $y_i = 1$ *and* $y_{i-1} = 0$*, define the (partial) function* $\sigma : \{0,1\}^n \times \mathbb{Z}_+ \to \{0,1\}^n$ *such that* $\sigma(\mathbf{y}, i) = (y_1, \ldots, y_i, y_{i-1}, y_{i+1}, \ldots, y_n)$*. Function* σ *is said to perform a* ***left-swap on*** $\mathbf{y}$ ***from index*** i.

Definition 2. *For any* $\mathbf{y} \in \{0,1\}^n$*,* $i \in \{2, \ldots, n\}$*, and* $k < i$ *where* $y_i = 1$ *and* $y_{i-1} = \ldots = y_{i-k} = 0$*, define the (partial) function* $\rho : \{0,1\}^n \times \mathbb{Z}_+ \times \mathbb{Z}_+ \to \{0,1\}^n$*, where* $\rho(\mathbf{y}, i, k) = \sigma(\mathbf{y}, i)$ *for* $k = 1$*, and*

$$\rho(\mathbf{y}, i, k) = \underbrace{\sigma(\ldots(\sigma(\sigma}_{k}(\mathbf{y}, i), i-1), \ldots), i-(k-1))$$

for $k > 1$*. Function* ρ *is said to perform* k ***consecutive left-swaps on*** $\mathbf{y}$ ***from index*** i.

Example 1. *Let* $\mathbf{y} = (y_1, y_2, y_3, y_4, y_5) \in \{0,1\}^5$*. Then* $\sigma(\mathbf{y}, 4) = (y_1, y_2, y_4, y_3, y_5)$ *and* $\rho(\mathbf{y}, 4, 3) = (y_4, y_1, y_2, y_3, y_5)$.

Definition 3. *Let* $\mathbf{y}$ *be a vector of* n *binary labels such that* n_1 *entries are* 1*. Let* $\mathcal{L}_1(\mathbf{y}) = \{p_1, \ldots, p_{n_1}\}$*, ordered such that* $p_i < p_j \iff i < j$*, be the indices of the* 1*s in the vector. Then we say* $\mathbf{y}$ *is in a* ***right-most configuration*** *iff* $p_i = n - n_1 + i$ *for every* $i \in \{1, \ldots, n_1\}$.

Proposition 2. *Let $\mathbf{r} = (r_1, \ldots, r_n)$ be a binary vector of length n in right-most configuration such that n_1 entries are 1. Let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$ be a vector of n real-valued guesses, ordered such that $\hat{y}_i < \hat{y}_j \iff i < j$. Then $h(\mathbf{r}, \hat{\mathbf{y}}) = 0$.*

Proof. Since $\mathbf{r}$ is in right-most configuration, it is clear that the right-hand side of

$$h(\mathbf{r}, \hat{\mathbf{y}}) = \sum_{i \in \mathcal{L}_0(\mathbf{r})} \sum_{j \in \mathcal{L}_1(\mathbf{r})} \mathbb{I}[i > j]$$

sums to 0. □

Proposition 3. *Let $\mathbf{y}$ be a vector of n binary labels, and let $\mathbf{z} = \sigma(\mathbf{y}, j')$ be another vector of binary labels that is produced by a single left-swap of $\mathbf{y}$ at index $j' \in \{2, \ldots, n\}$, where $y_{j'} = 1$ and $y_{i'} = 0$, and $i' = j' - 1$. Let $\hat{\mathbf{y}}$ be a vector of real-valued guesses. Then the number of pairs misclassified by $\hat{\mathbf{y}}$ w.r.t. $\mathbf{z}$ is one more than the number of pairs misclassified by $\hat{\mathbf{y}}$ w.r.t. $\mathbf{y}$ – i.e., $h(\mathbf{z}, \hat{\mathbf{y}}) = h(\mathbf{y}, \hat{\mathbf{y}}) + 1$.*

Proof. To shorten the notation, let $\mathcal{L}_0 = \mathcal{L}_0(\mathbf{y}), \mathcal{L}_1 = \mathcal{L}_1(\mathbf{y})$, and let $\tilde{\mathcal{L}}_0 = \mathcal{L}_0(\mathbf{z}), \tilde{\mathcal{L}}_1 = \mathcal{L}_1(\mathbf{z})$. We can split the summation in Equation 2 into four sets of pairs (see Figure 1): (a) those involving neither i' nor j'; (b) those involving j' but not i'; (c) those involving i' but not j'; and (d) the single pair involving both i' and j'. By grouping the pairs this way, we obtain:

$$\begin{aligned} &h(\mathbf{z}, \hat{\mathbf{y}}) \\ &= \left(\sum_{\substack{i \in \\ \tilde{\mathcal{L}}_0 \setminus \{j'\}}} \sum_{\substack{j \in \\ \tilde{\mathcal{L}}_1 \setminus \{i'\}}} \mathbb{I}[i > j] \right) + \left(\sum_{\substack{i \in \\ \tilde{\mathcal{L}}_0 \setminus \{j'\}}} \mathbb{I}[i > i'] \right) \\ &\quad + \left(\sum_{\substack{j \in \\ \tilde{\mathcal{L}}_1 \setminus \{i'\}}} \mathbb{I}[j' > j] \right) + \mathbb{I}[j' > i'] \end{aligned}$$

Notice that $\tilde{\mathcal{L}}_0 = (\mathcal{L}_0(\mathbf{y}) \setminus \{i'\}) \cup \{j'\}$, and hence $\mathcal{L}_0(\mathbf{y}) \setminus \{i'\} = \tilde{\mathcal{L}}_0(\mathbf{z}) \setminus \{j'\}$. Similarly, $\tilde{\mathcal{L}}_1 \setminus \{i'\} = \mathcal{L}_1 \setminus \{j'\}$. Then we have:

$$\begin{aligned} h(\mathbf{z}, \hat{\mathbf{y}}) \quad = \quad & \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[i > j] + \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \mathbb{I}[i > i'] + \\ & \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[j' > j] \quad + \mathbb{I}[j' > i'] \end{aligned}$$

Since $i' + 1 = j'$, then there cannot exist any index $i \in \mathcal{L}_0 \setminus \{i'\}$ whose value is "between" i' and j'; in other words, $i > i' \iff i > j'$ for every $i \in \mathcal{L}_0 \setminus \{i'\}$. Similarly, $j' > j \iff i' > j$ for every $j \in \mathcal{L}_1 \setminus \{j'\}$. Hence:

$$\begin{aligned} h(\mathbf{z}, \hat{\mathbf{y}}) \quad = \quad & \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[i > j] + \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \mathbb{I}[i > j'] + \\ & \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[i' > j] \quad + \mathbb{I}[j' > i'] \end{aligned}$$

Finally, since $\mathbb{I}[j' > i'] = 1$ and $\mathbb{I}[i' < j'] = 0$, then:

$$\begin{aligned} h(\mathbf{z}, \hat{\mathbf{y}}) \quad = \quad & \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[i > j] + \sum_{\substack{i \in \\ \mathcal{L}_0 \setminus \{i'\}}} \mathbb{I}[i > j'] + \\ & \sum_{\substack{j \in \\ \mathcal{L}_1 \setminus \{j'\}}} \mathbb{I}[i' > j] \quad + \mathbb{I}[i' > j'] + 1 \\ = \quad & h(\mathbf{y}, \hat{\mathbf{y}}) + 1 \end{aligned}$$

□

Proposition 4. *Suppose a dataset contains n examples, of which n_1 are labeled 1 and $n_0 = n - n_1$ are labeled 0. Let $\mathcal{Y}_{n_1} = \{\mathbf{y} \in \{0,1\}^n : \sum_i y_i = n_1\}$, and let $\mathcal{S}_{n_1} = \{(s_1, \ldots, s_{n_1}) \in \mathbb{Z}^{n_1} : n_0 \geq s_1 \geq \ldots \geq s_{n_1} \geq 0\}$. Then $\mathcal{Y}_{n_1}$ and $\mathcal{S}_{n_1}$ are in 1-to-1 correspondence.*

Proof. Every binary vector $\mathbf{y} \in \mathcal{Y}_{n_1}$ of length n, of which n_1 entries are 1, can be described by a unique vector of integers in the set $\mathcal{P}_{n_1} = \{(p_1, \ldots, p_{n_1}) \in \mathbb{Z}_+ : 1 \leq p_1 < \ldots < p_{n_1} \leq n\}$ specifying the indices of the 1s in $\mathbf{y}$ in increasing order. In particular, $\mathcal{Y}_{n_1}$ and $\mathcal{P}_{n_1}$ are in 1-to-1 correspondence with a bijection $f_p : \mathcal{Y}_{n_1} \to \mathcal{P}_{n_1}$. Hence, if we can show a bijection $f_s : \mathcal{P}_{n_1} \to \mathcal{S}_{n_1}$, then we can compose f_s with f_p to yield a new function $f : \mathcal{Y}_{n_1} \to \mathcal{S}_{n_1}$; since the composition of two bijections is bijective, then f will be bijective.

We can construct such an f_s as follows:

$$f_s(p_1, \ldots, p_{n_1}) = (s_1, \ldots, s_{n_1}) \quad \text{where } s_i = n - n_1 + i - p_i$$

We must first show that $(s_1, \ldots, s_{n_1}) = f_s(p_1, \ldots, p_{n_1}) \in \mathcal{S}_{n_1}$ for every $(p_1, \ldots, p_{n_1}) \in \mathcal{P}_{n_1}$; in particular, we must show that $n_0 \geq s_1 \geq \ldots \geq s_{n_1} \geq 0$. Since every p_i is an integer, we have that $p_j - p_i \geq j - i$ for every $j > i$. Hence:

$$n - n_1 + p_j - p_i \geq n - n_1 + j - i$$

We then add i to and subtract p_j from both sides to obtain:

$$\begin{aligned} n - n_1 + i - p_i \quad &\geq \quad n - n_1 + j - p_j \\ s_i \quad &\geq \quad s_j \quad \forall j > i \end{aligned}$$

The two boundary cases are s_{n_1}:

$$s_{n_1} = n - n_1 + n_1 - p_{n_1} = n - p_{n_1} \geq 0$$

and s_1:

$$s_1 = n - n_1 + 1 - p_1 = n_0 + 1 - p_1 \leq n_0$$

f_s **is 1-to-1**: Suppose $(s_1, \ldots, s_{n_1}) = f_s(p_1, \ldots, p_{n_1})$ and $(s'_1, \ldots, s'_{n_1}) = f_s(p'_1, \ldots, p'_{n_1})$, and suppose $s_i = s'_i$ for each i. Then:

$$\begin{aligned} n - n_1 + i - p_i \quad &= \quad n - n_1 + i - p'_i \qquad (3) \\ &\iff \qquad (4) \\ p_i \quad &= \quad p'_i \qquad (5) \end{aligned}$$

for each i.

f_s **is onto**: For every $(s_1, \ldots, s_{n_1}) \in \mathcal{S}_{n_1}$, we can find $(p_1, \ldots, p_{n_1})$ such that $f_s(p_1, \ldots, p_{n_1}) = (s_1, \ldots, s_{n_1})$ by

Guess $\hat{\mathbf{y}}$	1	2	3	4	i'=5	j'=6	7	8	9	10	
Label **y**	0	0	1	0	0	1	0	1	1	1	$h(\mathbf{y}, \hat{\mathbf{y}})$=4
Label **z**	0	0	1	0	1	0	0	1	1	1	$h(\mathbf{z}, \hat{\mathbf{y}})$=5

Figure 1: Illustration of how performing a left-swap on binary vector $\mathbf{y}$ at index j' yields a new vector $\mathbf{z}$ such that the number of misclassified pairs $h(\mathbf{z}, \hat{\mathbf{y}})$ is one more than $h(\mathbf{y}, \hat{\mathbf{y}})$. Specifically, $\hat{\mathbf{y}}$ misclassifies pairs $(3, 4)$, $(3, 5)$, $(3, 7)$, and $(6, 7)$ w.r.t. to $\mathbf{y}$, since for each such pair (i, j), $\hat{y}_i < \hat{y}_j$ but $y_i > y_j$. In contrast, $\hat{\mathbf{y}}$ misclassifies $(3, 4)$, $(3, 6)$, $(3, 7)$, $(5, 6)$, and $(5, 7)$ w.r.t. to $\mathbf{z}$.

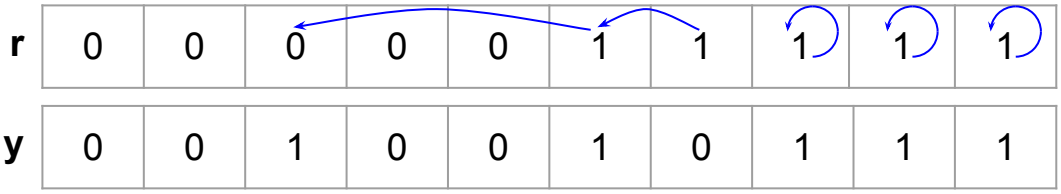

Figure 2: Illustration of how any binary vector $\mathbf{y}$ with n_1 1s can be produced by repeatedly left-swapping the 1's in a right-most binary vector $\mathbf{r}$. In the example above, the indices of the 1s in $\mathbf{y}$ are $p_1 = 3, p_2 = 6, p_3 = 8, p_4 = 9, p_5 = 10$. Left-swaps of the 1s in $\mathbf{r}$ to produce $\mathbf{y}$ are indicated with blue arrows, with $s_1 = 3, s_2 = 1$, and $s_3 = s_4 = s_5 = 0$.

setting $p_i = n - n_1 + i - s_i$ for each i. It only remains to be shown that $1 \leq p_1 < \ldots < p_{n_1} \leq n$: For any $j > i$,

$$p_j - p_i = j - i + s_i - s_j$$

Since $s_i \geq s_j$ (by definition of $\mathcal{S}_{n_1}$), we have:

$$\begin{aligned} p_j - p_i &\geq j - i \\ &> 0 \end{aligned}$$

and hence $p_j > p_i$. Moreover,

$$\begin{aligned} p_1 &= n - n_1 + 1 - s_1 \\ &\geq n - n_1 + 1 - n_0 \\ &\geq 1 \end{aligned}$$

and

$$\begin{aligned} p_{n_1} &= n - n_1 + n_1 - s_{n_1} \\ &\leq n \end{aligned}$$

□

Theorem 1. *Suppose a dataset contains n examples, of which n_1 are labeled 1 and n_0 are labeled 0. Let $\mathcal{Y}_{n_1}^{(d)} = \{\mathbf{y} \in \{0,1\}^n : \sum_i y_i = n_1 \wedge h(\mathbf{y}, \hat{\mathbf{y}}) = d\}$, and let $\mathcal{S}_{n_1}^{(d)} = \{(s_1, \ldots, s_{n_1}) \in \mathbb{Z}^{n_1} : n_0 \geq s_1 \geq \ldots \geq s_{n_1} \geq 0 \wedge \sum_i s_i = d\}$. Then $\mathcal{Y}_{n_1}^{(d)}$ and $\mathcal{S}_{n_1}^{(d)}$ are in 1-to-1 correspondence.*

Proof. Since $\mathcal{Y}_{n_1}^{(d)} \subset \mathcal{Y}_{n_1}$ and $\mathcal{S}_{n_1}^{(d)} \subset \mathcal{S}_{n_1}$, and since $\mathcal{Y}_{n_1}$ and $\mathcal{S}_{n_1}$ are in 1-to-1 correspondence with bijection f (from Proposition 4) then we must show only that the image of $\mathcal{Y}_{n_1}^{(d)}$ through f is $\mathcal{S}_{n_1}^{(d)}$. Suppose that $(s_1, \ldots, s_{n_1}) = f(\mathbf{y})$ for some $\mathbf{y} \in \mathcal{Y}_{n_1}^{(d)}$. Then we can write $\mathbf{y}$ as

$$\mathbf{y} = \underbrace{\rho(\rho(\ldots(\rho}_{n_1}(\mathbf{r}, n-n_1+1, s_1), n-n_1+2, s_2)\ldots), n, s_{n_1})$$

In other words, $\mathbf{y} \in \mathcal{Y}_{n_1}^{(d)}$ can be obtained from $\mathbf{r}$ (a binary vector of length n, such that n_1 elements are labeled 1, in right-most configuration) by performing a sequence of consecutive left-swaps on the 1s in $\mathbf{r}$. To see this, observe that the first 1 in $\mathbf{r}$ is always immediately preceded by n_0 0s; hence, we can perform $s_1 \leq n_0$ consecutive left-swaps on $\mathbf{r}$ from index $n - n_1 + 1$. (See Figure 2 for an illustration.) Moreover, after performing these consecutive left-swaps, then the second 1 in $\mathbf{r}$ will be immediately preceded by s_1 0s; hence, we can perform $s_2 \leq s_1$ consecutive left-swaps on $\mathbf{r}$ from index $n - n_1 + 2$. After performing these consecutive left-swaps, then the third 1 in $\mathbf{r}$ will be immediately preceded by s_2 0s; and so on. After performing the consecutive left-swaps for each of the 1s in $\mathbf{r}$, then the position of the ith 1 in the resulting vector is $n - n_1 + i - s_i = p_i$ for each i, as desired.

Next, recall that, by Proposition 2, $h(\mathbf{r}, \hat{\mathbf{y}}) = 0$. Moreover, by Proposition 3, each left-swap increases the value of h by 1; hence, applying ρ to perform s_i consecutive left-swaps increases the value of h by s_i. Summing over all 1s results in a total of $\sum_{i=1}^{n_1} s_i$ misclassified pairs, i.e., $h(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_1} s_i$. But since $\mathbf{y} \in \mathcal{Y}_{n_1}^{(d)}$, we already know that $h(\mathbf{y}, \hat{\mathbf{y}}) = d$. Therefore, $\sum_i s_i = d$, and hence $(s_1, \ldots, s_{n_1}) \in \mathcal{Y}_{n_1}^{(d)}$. □

Summing over all possible n_1

Based on Theorem 1, we can compute the number, $v(n_0, n_1, d) \doteq |\mathcal{S}_{n_1}^{(d)}|$, of binary vectors of length $n =$

$n_0 + n_1$, such that n_1 of the entries are labeled 1 and for which $h(\mathbf{y}, \hat{\mathbf{y}}) = d$. Recall that the AUC can be computed by dividing the number d of misclassified pairs by the total number of example-pairs $n_0 n_1$. Hence, to compute the *total* number, $w(n, c)$, of binary vectors of length n for which $\text{AUC}(\mathbf{y}, \hat{\mathbf{y}}) = c$, we must first determine the set $\mathcal{N}_1$ of possible values for n_1, and then sum $v(n_0, n_1, d)$ over every value in $\mathcal{N}_1$ and the corresponding value d.

Suppose that the oracle reports an AUC of $c = p/q$, where p/q is a reduced fraction, Since c represents the fraction of all pairs of examples – one from each class – that are classified by the contestant's guesses correctly, then q must divide the total number $(n_0 n_1)$ of pairs in the test set. Hence:

$$\mathcal{N}_1 = \{n_1 : (0 < n_1 < n) \ \wedge \ (q \mid (n - n_1) n_1)\}$$

Since it is possible that $q < n_0 n_1$, we must scale $(q - p)$ by $n_0 n_1 / q$ to determine the actual number of misclassified pairs d. In particular, we define

$$d(n_1) = (q - p) n_0 n_1 / q = (q - p)(n - n_1) n_1 / q$$

Based on $\mathcal{N}_1$ and $d(n_1)$, we can finally compute:

$$\begin{aligned} w(n, c) &= \left| \bigcup_{n_1 \in \mathcal{N}_1} \mathcal{S}_{n_1}^{(d(n_1))} \right| \\ &= \sum_{n_1 \in \mathcal{N}_1} v(n - n_1, n_1, d(n_1)) \end{aligned}$$

since the $\mathcal{S}_{n_1}^{(d(n_1))}$ are disjoint.

Recursion Relation

We can derive a recursion relation for $v(n_0, n_1, d)$ as follows: Given any binary vector $\mathbf{r}$ of length n, with n_1 1s, in right-most configuration, we can apply $k \in \{0, 1, \ldots, \min(d, n_0)\}$ left-swaps on $\mathbf{r}$ from index $n - n_1 + 1$ (i.e., from the left-most 1) to yield $\mathbf{y} = \rho(\mathbf{r}, n - n_1 + 1, k)$. Then the vector $(y_{n-n_1-k+2}, y_{n-n_1-k+3}, y_{n-n_1-k+4}, \ldots, y_n)$ (i.e., the last $n_1 - 1 + k$ elements of $\mathbf{y}$) consists of k 0s followed by $(n_1 - 1)$ 1s; in other words, it is in right-most configuration. Thus, by iterating over all possible k and computing for each choice how many *more* left-swaps are necessary to reach a total of d, we can define v recursively:

$$v(n_0, n_1, d) = \sum_{k=0}^{\min(d, n_0)} v(k, n_1 - 1, d - k)$$

with initial conditions:

$$\begin{aligned} v(n_0, n_1, 0) &= 1 \quad \forall n_0 \geq 0, n_1 \geq 0 \\ v(0, n_1, d) &= 0 \quad \forall n_1 \geq 0, d > 0 \\ v(n_0, 0, d) &= 0 \quad \forall n_0 \geq 0, d > 0 \end{aligned}$$

Dynamic programming using a three-dimensional memoization table can be used to compute v in time $O(n_0 n_1 d)$.

The recursive algorithm above can also be used *constructively* (though with large space costs) to compute the set $\mathcal{W}$ of all binary vectors $\mathbf{y}$ of length n, of which n_1 are 1, such that $h(\mathbf{y}, \hat{\mathbf{y}}) = d$ for any d. In order to apply this construction, the test examples must first be sorted in increasing value of the contestant's guesses; the constructive algorithm is then applied to generate $\mathcal{W}$; and then the components of each of the possible binary vectors must be reordered to recover the original order of the test examples. An example of $\mathcal{W}$ for $n_0 = 3, n_1 = 8, c = p/q = 17/24$ is shown in Table 1.

	Test Labels y										
a	1	1	0	0	1	0	1	1	1	1	1
b	1	0	1	1	0	0	1	1	1	1	1
c	1	0	1	0	1	1	0	1	1	1	1
d	0	1	1	1	0	1	0	1	1	1	1
e	1	0	0	1	1	1	1	0	1	1	1
f	0	1	1	0	1	1	1	0	1	1	1
g	0	1	0	1	1	1	1	1	0	1	1
h	0	0	1	1	1	1	1	1	1	0	1

Table 1: Set $\mathcal{W}$ of possible ground-truth labelings $\mathbf{y}$ for $n_0 = 3, n_1 = 8, c = 17/24$. Columns correspond to different test examples (indexed by their rank order in $\hat{\mathbf{y}}$), and rows correspond to different possible values of $\mathbf{y}$. Each $\mathbf{y}$ is generated by performing $d = 7$ left-shifts on the vector $(0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1)$.

Growth of $w(n, c)$ in n for fixed c

In prior work (Whitehill 2016) we showed that, for every fixed rational $c = p/q \in (0, 1)$, the number of possible binary ground-truth vectors for which the contestant's guesses achieve AUC of exactly c, grows exponentially in n. However, their result applies only to datasets that are at least $n = 4q$ in size. What can happen for smaller n?

Using the recursive formula from the section above, we found empirical evidence that $w(n, c)$ may actually be (initially) monotonically *decreasing* in n, until n reaches a threshold (specific to q) at which it begins to increase again. As an example with $p = 1387$ and $q = 1440$ (and hence $d = 1440 - 1387 = 53$), we can compute the number of possible binary labelings that are compatible with an AUC of exactly $c = p/q = 1387/1440$ (which is approximately 96.3%) as a function of n:

n	$\mathcal{N}_1$	$w(n, c)$
76	$\{36, 40\}$	657488
77	$\{32, 45\}$	654344
78	$\{30, 48\}$	650822
84	$\{24, 60\}$	622952
92	$\{20, 72\}$	572728
98	$\{18, 80\}$	529382
106	$\{16, 90\}$	468686

Here, $w(n, c)$ decreases steadily until $n = 106$. We conjecture that $w(n, c)$ is monotonically non-increasing in n for $n \leq \min\{n_0 + n_1 : n_0 n_1 = 2q\}$, for every fixed c. This conjecture is consistent with some simulations we conducted and is based on the following intuition: By definition, the denominator (q) of the AUC $c = p/q$ must divide $n_0 n_1$. The number of possible ground-truth labelings for fixed n and

c depends, in part, on the set of tuples (n_0, n_1) such that $n_0 + n_1 = n$ and $q \mid n_0 n_1$. (See section "Summing over all possible n_1" above.) When n grows to be at least $2q$, then many more possibilities for (n_0, n_1) may become feasible because if (n_0, n_1) is feasible, then so is $(2n_0, n_1)$, and so is $(n_0, 2n_1)$.

While the number of satisfying solutions in this example for $n = 106$ is still in the hundreds of thousands, it is easily small enough to allow each possibility to be considered individually, as we explore below. We note that test sets on the order of hundreds of examples are not uncommon; the Intel & MobileODT Cervical Cancer Screening (Kaggle 2017) and ASSISTments datamining competitions (Heffernan et al. 2017) are two examples.

A More Efficient Exploit

In the sections above, we showed (1) a simple mechanism whereby a pair of oracle calls can reveal the label of any example k; and (2) an algorithm for enumerating all possible labelings compatible with the AUC c obtained by a vector of guesses. Below we describe how these two mechanisms can be combined to infer the identity of *all* test labels with fewer than $2n$ oracle queries:

1. Submit a vector of guesses $\hat{\mathbf{y}}$ and obtain the AUC c.
2. Determine the set $\mathcal{W}$ of all possible ground-truth vectors compatible with c and $\hat{\mathbf{y}}$.
3. Set $\mathcal{W}_1 = \mathcal{W}$.
4. For $\tau = 1, 2, \ldots$:
   (a) If $|\mathcal{W}_\tau| = 1$, then return $\mathbf{y} \in \mathcal{W}_\tau$.
   (b) Choose the example k (where examples are numbered according to their *rank order* in $\hat{\mathbf{y}}$) whose label y_k would provide the greatest reduction in the number of compatible members of $\mathcal{W}_\tau$. This is equivalent to finding $k = \arg\min_{k'} |0.5 - |\{\mathbf{y} \in W_\tau : y_{k'} = 1\}| / |\mathcal{W}_\tau||$.
   (c) Submit a pair of oracle queries to infer the ground-truth label of example k; call it l.
   (d) Compute $\mathcal{W}_{\tau+1} = \{\mathbf{y} \in \mathcal{W}_\tau : y_k = l\}$.

The loop at Step 4 is equivalent to traversing a binary tree from the root downwards. While constructing a minimum-height binary tree is NP-complete (Laurent and Rivest 1976), heuristics such as maximizing the reduction in uncertainty (Quinlan 2014) are often used, which we also use here.

Example 1: For the example ($n_0 = 3, n_1 = 8, c = 17/24$) shown in Table 1, we could choose $k = 1$ during the first iteration since there are 4 examples for which $y_1 = 1$ and 4 examples for which $y_1 = 0$. (Note that we could also choose $k = 2$.) Suppose that, using a pair of oracle queries, we deduce that $y_1 = 0$. Then for the second iteration we would choose $k = 2$ and infer y_2 using another pair of oracle queries, and so on. The binary decision tree is shown in Figure 3. Since the height of the tree is 4, then only $2 \times 4 + 1 = 9$ total oracle queries are required (including the initial query before the loop). This is fewer than the $2 \times 11 = 22$ queries using the naive approach presented earlier in the paper.

Example 2: Suppose we wished to infer the labels of a dataset consisting of $n = 106$ examples, and our guesses attained an AUC of $c = p/q = 1387/1440$ (as in the section above). Then the number of satisfying solutions is 468686. By following the algorithm listed above, we can iteratively query the oracle to most quickly reduce the number of possible solutions down to 1. In this example, it turns out that the maximum depth of any node in the decision tree is 35; hence, at most $35 \times 2 + 1 = 71$ queries (including the initial one) need to be made to infer all 106 ground-truth labels unambiguously. This is significantly less than $2 \times 106 = 212$ using a naive strategy.

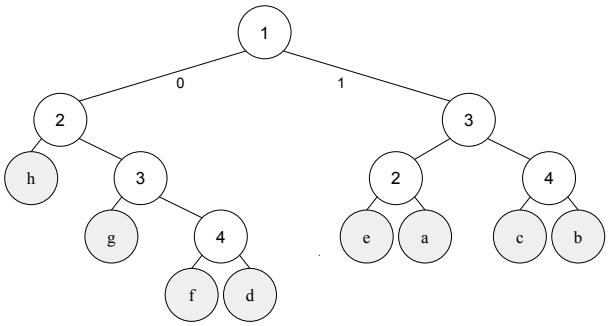

Figure 3: Binary decision tree to deduce which element of $\mathcal{W}$ is the actual ground-truth vector $\mathbf{y}$. Each internal node signifies the index (according to the rank order of $\hat{\mathbf{y}}$) of the example whose label should be inferred (using a pair of oracle queries). Each left edge corresponds to a label of 0. Leaf nodes indicate which ground-truth vector in Table 1 is correct.

Discussion

Here we discuss some points raised by the anonymous reviewers regarding the practicality of the proposed attack.

Is there experimental evidence that labels can be inferred in this way in a real competition? In order to perform the attack described in our paper in a real datamining contest, we would need to find an ongoing competition that fulfills several criteria:

(a) It uses AUC as the evaluation criterion.
(b) The size of the test set is modest, i.e., $O(100)$ examples.
(c) The oracle returns the AUC on the entire test set, not just a random subset.
(d) Contestants can submit multiple oracle queries during the competition.

Many recent competitions fulfill several of these criteria. For instance, the public leaderboard of the Intel-MobileODT 2017 competition fulfilled criteria (b), (c), and (d), but was evaluated using log-loss instead of AUC. The KDD Cup 2015 fulfilled criteria (a), (c), and (d), but contained thousands of test examples. There actually was one recent competition that fulfilled all four criteria – the ASSISTments Data Mining Competition 2017, with just 172 test examples. It may only be a matter of time before a new competition arises that meets all the criteria. A goal of our paper is to emphasize that doing so would be a bad idea.

To be clear: We do not claim that our attack is feasible (computationally, or within the allowed number of oracle queries) for *all* test sets, even when all the criteria above are fulfilled. In the vast majority of cases, the competition would not allow enough oracle queries, and/or the computational time and storage costs would become intractable. However, the purpose of our paper is to show that there *exist* test sets on which attacks could successfully be waged.

Does the proposed method apply if the AUC is reported as a floating-point number instead of an exact fraction p/q? Sometimes. Many competition oracles report AUC results with 5 decimal digits. If the value of p/q can be represented exactly using less than or equal to 5 digits, then no information is lost. Even if it cannot be, there may still be some useful information about (p, n_0, n_1) that can be extracted. Consider, for example, a competition of the size used in the ASSISTments Datamining Competition, with $n = 172$ examples. If the actual values of n_0 and n_1 were 100 and 72, respectively, and if a contestant's classifier achieved an AUC of (say) $6500/(100 * 72)$ which were truncated by the oracle at 5 decimal figures (i.e., 0.90278), then the contestant could greatly reduce the set of (n_0, n_1, p) that are compatible with this result to just 18 possibilities. Moreover, through subsequent oracle calls with slightly different AUCs (which could be performed by adding noise to the guesses), then the possible values can be whittled away further by intersecting the subsets of feasible (n_0, n_1). Of course, having more than just one possibility for (n_0, n_1, p) will weaken the attack, and the attacker might need to make far more oracle queries (to handle each possible tuple) to deduce the test set labels with certainty. Speculatively, we could imagine that one might generalize our proposed attack in this way, but this would require further research.

What would happen if the oracle added noise to c? Adding noise to the AUC returned by the oracle would likely thwart the proposed attack because it assumes perfect knowledge of p and q. However, in practice, datamining competition organizers are reluctant to add such noise to the oracle responses. One possible reason is that it might bother contestants that they could receive a score that is (sightly) lower than what they deserved.

Conclusions

We have investigated the mathematical structure of how the Area Under the Receiver Operating Characteristics Curve (AUC) accuracy metric is computed from the binary vector of ground-truth labels and a real-valued vector of guesses. In particular, we derived an efficient recursive algorithm with which to count the exact number of binary vectors for which the AUC of a fixed vector of guesses is some value c. We also derived a constructive algorithm with which to enumerate all such binary vectors. Finally, we showed a novel exploitation mechanism whereby the set $\mathcal{W}$ of possible ground-truth vectors can be iteratively whittled down using a binary decision tree to a unique value $\mathbf{y}$. As an important limitation noted above, our algorithm assumes *perfect* knowledge of the AUC, i.e., no noise added.

Future work: It would be interesting to explore more sophisticated adaptive mechanisms of choosing guess vectors $\hat{\mathbf{y}}$; based on the $\mathcal{W}$ resulting from the first vector of guesses, there might be a way of choosing the next guess vector $\hat{\mathbf{y}}$ that results in a more efficient deduction of the ground-truth values than the method we presented.

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*.

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

Blum, A.; Ligett, K.; and Roth, A. 2013. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)* 60(2):12.

Chaudhuri, K., and Monteleoni, C. 2009. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*.

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*.

Dwork, C. 2011. Differential privacy. In van Tilborg, H., and Jajodia, S., eds., *Encyclopedia of Cryptography and Security*. Springer US. 338–340.

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.

Heffernan, N.; Baker, R.; Woolf, B.; and Patikorn, T. 2017. ASSISTments data mining competition 2017. https://sites.google.com/view/assistmentsdatamining/data-mining-competition-2017.

Kaggle. 2017. Intel & mobileodt cervical cancer screening datamining competition. https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening.

Laurent, H., and Rivest, R. L. 1976. Constructing optimal binary decision trees is np-complete. *Information processing letters* 5(1):15–17.

Matthews, G. J., and Harel, O. 2013. An examination of data confidentiality and disclosure issues related to publication of empirical roc curves. *Academic radiology* 20(7):889–896.

Quinlan, J. R. 2014. *C4. 5: programs for machine learning*. Elsevier.

Stoddard, B.; Chen, Y.; and Machanavajjhala, A. 2014. Differentially private algorithms for empirical machine learning. *CoRR* abs/1411.5428.

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. 2015. Toward a better understanding of leaderboard. *arXiv preprint arXiv:1510.03349*.