Discriminative Learning can Succeed where Generative Learning Fails

Philip M. Long[1] and Rocco A. Servedio*[2]

[1] Google, Mountain View, CA, U.S.A.
plong@google.com
[2] Columbia University, New York, NY, U.S.A.
rocco@cs.columbia.edu

Abstract. Generative algorithms for learning classifiers use training data to separately estimate a probability model for each class. New items are then classified by comparing their probabilities under these models. In contrast, discriminative learning algorithms try to find classifiers that perform well on all the training data.
We show that there is a learning problem that can be solved by a discriminative learning algorithm, but not by any generative learning algorithm (given minimal cryptographic assumptions). This statement is formalized using a framework inspired by previous work of Goldberg [3].

1 Introduction

If objects and their classifications are generated randomly from a joint probability distribution, then the optimal way to predict the class y of an item x to is maximize $\Pr[y|x]$. Applying Bayes' rule, this is equivalent to maximizing $\Pr[x|y]\Pr[y]$. This motivates what has become known as the *generative* approach to learning a classifier, in which the training data is used to learn $\Pr[\cdot|y]$ and $\Pr[y]$ for the different classes y, and the results are used to approximate the behavior of the optimal predictor for the source (see [1, 5]).

In the *discriminative* approach, the learning algorithm simply tries to find a classifier that performs well on the training data [12, 5, 9, 6]. Discriminative algorithms can (and usually do) process examples from several classes together at once, e.g. maximum margin algorithms use both positive and negative examples together to find a large margin hypothesis separating the two classes.

The main result of this paper is a computational separation between generative and discriminative learning. We describe a learning problem and prove that it has the following property: a discriminative algorithm can solve the problem in polynomial time, but no generative learning algorithm can (assuming that cryptographic one-way functions exist).

Our analysis demonstrates the possible cost of largely processing the examples from different classes separately, as generative methods do. Goldberg [3]

* Supported in part by NSF award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.

was the first to study the effect of this limitation. His analyses concerned a modification of the PAC model in which

- the examples belonging to each class are analyzed separately,
- each analysis results in a scoring function for that class, and
- future class predictions are made by comparing the scores assigned by the different scoring functions.

He designed algorithms that provably solve a number of concrete learning problems despite the constraint of processing examples from different classes separately, and identified conditions that allow a discriminative PAC learner to be modified to work in the generative setting. The main open question formulated in [3] is whether there is a learning problem that can be solved by a discriminative algorithm but cannot be solved by a generative algorithm. We establish our main result in a framework closely related to the one proposed in [3]. The main difference between our formulation and Goldberg's is that we define a learning problem to be a collection of possible joint probability distributions over items and their classifications, whereas Goldberg defined a learning problem to be a concept class as in the PAC model.

Roughly, our proof works as follows. In the learning problem we consider the domain is divided into three parts, and a separate function provides 100% accuracy on each part. The third part is really hard: its function is cryptographically secure against any adversary (or learner) which does not "know" the "key" to the function. On the other hand, the first two parts are easy, and descriptions of their two functions can be combined to compute the key of the third function.

A discriminative algorithm can succeed by learning the first two parts and using the results to obtain the key to the third function. On the other hand, each of the first two parts is hard to learn from one kind of example: one part is hard to learn from positive examples only, and the other is hard to learn from negative examples only. Thus in the generative learning framework, the scoring function obtained using positive examples only contains no information about the subfunction which is hard to learn from positive examples, and thus in and of itself this positive scoring function contains no useful information about the key for the third function. An analogous statement is true for the negative scoring function. The tricky part of the analysis is to show that the overall predictor used by the generative algorithm – which incorporates information from both the positive and negative scoring functions – is similarly useless on the third part. Intuitively this is the case because the two scoring functions are combined in a very restricted way (simply by comparing the values that they output), and this makes it impossible for the final classifier to fully exploit the information contained in the two scoring functions.

Related work. Aside from Goldberg's paper, the most closely related work of which we are aware is due to Ng and Jordan [8]. They showed that Naive Bayes, a generative algorithm, can converge to the large-sample limit of its accuracy much more quickly than a corresponding discriminative method. For generative algorithms that work by performing maximum likelihood over restricted classes of models, they also showed, given minimal assumptions, that the large-sample

limit of their accuracy is no better than a corresponding discriminative method. Note that these results compare a particular generative algorithm with a particular discriminative algorithm. In contrast, the analysis in this paper exposes a fundamental limitation faced by any generative learning algorithm, due to the fact that it processes the two classes separately.

Section 2 contains preliminaries including a detailed description and motivation of the learning model. In Section 3 we give our construction of a learning problem that separates the two models and give a high-level idea of the proof. Sections 4 and 5 give the proof of the separation.

Due to space constraints some proofs are omitted; see [7] for these proofs.

2 Definitions and main result

Given a domain X, we say that a *source* is a probability distribution P over $X \times \{-1,1\}$, and a *learning problem* $\mathcal{P}$ is a set of sources. Throughout this paper the domain X will be $\{0,1\}^n \times \{1,2,3\}$.

2.1 Discriminative learning

The discriminative learning framework that we analyze is the *Probably Approximately Bayes* (PAB) [2] variant of the PAC [11] learning model. In the PAB model, in a learning problem $\mathcal{P}$ a learning algorithm is given a set of m labeled examples drawn from an unknown source $P \in \mathcal{P}$. The goal is to output a hypothesis function $h : X \to \{-1,1\}$ which with probability $1-\delta$ satisfies $\Pr_{(x,y)\in P}[h(x) \neq y] \leq \text{Bayes}(P) + \epsilon$, where $\text{Bayes}(P)$ is the least error rate that can be achieved on P, i.e. the minimum, over all functions h, of $\Pr_{(x,y)\in P}[h(x) \neq y]$. In a setting (such as ours) where the domain X is parameterized by n, an *efficient* learning algorithm for $\mathcal{P}$ is one that uses $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ many examples, runs in $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ time, and outputs a hypothesis that can be evaluated on any point in $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ time.

2.2 Generative learning

Goldberg [3] defined a restricted "generative" variant of PAC learning. Our analysis will concern a natural extension of his ideas to the PAB model.

Roughly speaking, in the generative model studied in this paper, the algorithm first uses only positive examples to construct a "positive scoring function" $h_+ : X \to \mathbf{R}$ that assigns a "positiveness" score to each example in the input domain. It then uses only negative examples to construct (using the same algorithm) a "negative scoring function" $h_- : X \to \mathbf{R}$ that assigns a "negativeness" score to each example. The classifier output by the algorithm is the following: given example x, output 1 or -1 according to whether or not $h_+(x) > h_-(x)$.

We now give a precise description of our learning framework. In our model

- A sample $S = (x_1, y_1)$,...,(x_m, y_m) is drawn from the unknown source P;
- An algorithm A is given a filtered version of S in which
  - examples (x_t, y_t) for which $y_t = 1$ are replaced with x_t, and
  - examples (x_t, y_t) for which $y_t = -1$ are replaced with $\diamond$

and A outputs $h_+ : X \to \mathbf{R}$.
- Next, the same algorithm A is given a filtered version of S in which
  - examples (x_t, y_t) for which $y_t = 1$ are replaced with $\diamond$, and
  - examples (x_t, y_t) for which $y_t = -1$ are replaced with x_t

  and A outputs $h_- : X \to \mathbf{R}$.
- Finally, let $h : X \to \{-1, 1\}$ be defined as $h(x) = \mathrm{sgn}(h_+(x) - h_-(x))$. If $h_+(x) = h_-(x)$ then we view $h(x)$ as outputing $\perp$ (undefined).

Algorithm A is said to be a *generative PAB learning algorithm* for $\mathcal{P}$ if for all $P \in \mathcal{P}$, for all $0 < \epsilon < \frac{1}{2}$, $0 < \delta < 1$, the hypothesis h obtained as above, with probability at least $1 - \delta$, satisfies $\Pr_{(x,y)\in P}[h(x) \neq y] \leq \mathrm{Bayes}(P) + \epsilon$. The notions of runtime and efficiency are the same as in the standard PAB framework. It is easy to see that any learning problem that can be efficiently PAB learned in the generative framework we have described can also be efficiently learned in the standard PAB framework.

2.3 Main result

With these definitions in place we can state our main result:

Theorem 1. *If one-way functions exist, there is a learning problem that is efficiently learnable in the PAB model, but not in the generative PAB model.*

2.4 Two unsupervised learners are not better than one

Using different algorithms for the positive and negative examples cannot help a generative learning algorithm much; this can be formalized using an idea due to Goldberg [3]. This leads to the following extension of Theorem 1 (see Section 6 of [7] for a proof of this extension):

Theorem 2. *Suppose the generative PAB learning model is relaxed so that separate algorithms can be applied to the positive and negative examples. Then it remains true that if one-way functions exist, then there is a learning problem that can be solved in polynomial time in the standard PAB model, but not in the generative PAB model.*

3 The construction and the main idea

Our construction uses pseudorandom functions; defined by Goldreich *et al.* in 1986 [4], these are central objects in modern cryptography.

Definition 1. *A* pseudorandom function family *(PRFF) is a collection of functions* $\{f_s : \{0,1\}^{|s|} \to \{1,-1\}\}_{s\in\{0,1\}^*}$ *with the following two properties:*

1. (efficient evaluation) *there is a deterministic algorithm which, given an* n*-bit seed* s *and an* n*-bit input* x*, runs in time poly*(n) *and outputs* $f_s(x)$;

2. (pseudorandomness) *for all constants $c > 0$, all probabilistic polynomial-time (p.p.t.) oracle algorithms A, and all sufficiently large n, we have that*

$$|\Pr_{F \in \mathcal{R}_n}[A^F(1^n) \text{ outputs } 1] - \Pr_{s \in \{0,1\}^n}[A^{f_s}(1^n) \text{ outputs } 1]| < 1/n^c.$$

Here "$\Pr_{F \in \mathcal{R}_n}$" indicates that F is a truly random function chosen uniformly from the set of all 2^{2^n} Boolean functions mapping $\{0,1\}^n$ to $\{-1,1\}$, and "$\Pr_{s \in \{0,1\}^n}$" indicates that s is chosen uniformly from $\{0,1\}^n$.

The notation "$A^g(1^n)$" indicates that A is run with black-box oracle access to g on a vacuous input of length n (so since A is a polynomial-time algorithm, it runs for at most poly(n) time steps). Intuitively, the pseudorandomess property ensures that in any probabilistic poly(n)-time computation which is executed with oracle access to a truly random function, a randomly chosen pseudorandom function may be used instead without affecting the outcome of the computation in a noticeable way. Well known results [4, 10] imply that pseudorandom function families exist if and only if any one-way function exists.

3.1 The construction

We first define a class C of Boolean functions in which each function is specified by a triple (r, s, b) where $r, s \in \{-1, 0, 1\}^n$ and $b \in \{-1, 1\}$. We will use the functions in C to define the set of sources which constitute our learning problem.

A function $c_{r,s,b} \in C$ takes two inputs: an n-bit string $x \in \{0,1\}^n$ and an index $i \in \{1, 2, 3\}$. We refer to examples of the form (x, i) as *type-i examples* for $i = 1, 2, 3$. The value of $c_{r,s,b}(x, i)$ is defined as follows:

$$c_{r,s,1}(x,i) = \begin{cases} AND_r(x) & \text{if } i = 1 \\ OR_s(x) & \text{if } i = 2 \\ f_{|r| \oplus |s|}(x) & \text{if } i = 3, \end{cases} \qquad c_{r,s,-1}(x,i) = \begin{cases} OR_r(x) & \text{if } i = 1 \\ AND_s(x) & \text{if } i = 2 \\ f_{|r| \oplus |s|}(x) & \text{if } i = 3. \end{cases}$$

Here AND_r is the conjunction of literals over $x_1, \ldots, x_n$ that is indexed by r; for instance if $n = 3$ and $r = (r_1, r_2, r_3) = (1, 0, -1)$ then $AND_r(x)$ is $x_1 \wedge \overline{x}_3$. OR_s is similarly the disjunction that is indexed by s. The notation "$|r|$" denotes the n-bit string $(|r_1|, \ldots, |r_n|) \in \{0,1\}^n$, and the bitwise XOR $y \oplus z$ of two n-bit strings $y, z \in \{0,1\}^n$ is the n-bit string $(y_1 \oplus z_1, \ldots, y_n \oplus z_n)$. The family $\{f_t\}_{t \in \{0,1\}^n}$ is a PRFF as described at the start of Section 3 above.

Now we describe the learning problem $\mathcal{P}$ that we use to prove our main result. Each source P in $\mathcal{P}$ is *realizable*, i.e. there is a function mapping X to $\{-1, 1\}$ with 100% accuracy (so the Bayes optimal error is 0). Specifically, for each $c_{r,s,b} \in C$, there is a source $P_{r,s,b}$ which is a distribution over labelled examples $((x, i), c_{r,s,b}(x, i))$. Thus to describe $P_{r,s,b}$ it suffices to describe the marginal distributions over the domain $X = \{0,1\}^n \times \{1, 2, 3\}$ of inputs to $c_{r,s,b}$; i.e. we need to describe the distribution over positive examples, and the distribution over negative examples. These marginal distributions are as follows: for each $i = 1, 2, 3$ the distribution allocates $1/3$ of the total probability mass to type-i

examples. For each $i = 1, 2, 3$, half of this $1/3$ mass is distributed uniformly over the positive type-i examples, and half over the negative type-i examples.

Note that the above description assumes that there are indeed both positive and negative examples of type i. If for some i all type-i examples have the same label, then the entire $1/3$ probability mass for type-i examples is uniformly distributed over all 2^n examples (x, i). Note that AND_r always has at least one positive example and OR_s always has at least one negative example, and consequently each source in $\mathcal{P}$ has at least $1/6$ probability weight on each label. Note also that it is possible that for a given $t \in \{0,1\}^n$, the member f_t of the pseudo-random function family used on the type-3 examples could be identically 1 or identically -1. However, the pseudorandomness of $\{f_t\}$ ensures that for any $c > 0$, for large enough n, at least a $1 - \frac{1}{n^c}$ fraction of functions in $\{f_t\}_{t\in\{0,1\}^n}$ have a fraction of positive (negative) examples which is bounded in $[\frac{1}{2} - \frac{1}{n^c}, \frac{1}{2} + \frac{1}{n^c}]$. (Otherwise, by drawing poly(n) many random examples and estimating the fraction of positive examples using this sample, a poly(n)-time algorithm would be able to distinguish a random function from $\{f_t\}_{t\in\{0,1\}^n}$ from a truly random function with nonnegligible advantage over random guessing.)

3.2 The idea

In this section we sketch the high-level idea of why discriminative algorithms can efficiently solve this learning problem while generative algorithms cannot.

Discriminative learners can succeed: Let $P_{r,s,b}$ be any element of $\mathcal{P}$. A simple argument which we sketch in Section 4 shows that a discriminative learner can use the labelled type-1 examples (type-2 respectively) to efficiently exactly identify r (s, respectively). It can guess and check the value of b, and thus can w.h.p. exactly identify the unknown source in poly(n) time.

Generative learners cannot succeed: We show that no generative algorithm can construct a hypothesis that w.h.p. has high accuracy on type-3 examples.

More precisely, we define a particular probability distribution $\mathcal{D}$ over the sources in $\mathcal{P}$ and show that for a source selected from this distribution, no poly(n)-time generative learning algorithm can w.h.p. output a hypothesis h whose accuracy on type-3 examples is bounded away from $1/2$. This means that the overall accuracy of such a learner cannot be substantially greater than $5/6$.

The distribution $\mathcal{D}$ is as follows: to draw a source $P_{r,s,b}$ from $\mathcal{D}$,

- Toss a fair coin and set b to ± 1 accordingly;
- Select r and s by drawing each one from the following distribution TARGET over $\{-1, 0, 1\}^n$: a string x drawn from TARGET has each x_i independently set to be -1, 0 or 1 with probabilities $1/4$, $1/2$ and $1/4$ respectively.

Note that under $\mathcal{D}$ the strings $|r|$ and $|s|$ are independently and uniformly distributed over $\{0,1\}^n$. This will be useful later since it means that even if one of the strings r, s is given, the seed $|r| \oplus |s|$ to the pseudorandom function $f_{|r|\oplus|s|}$ is uniformly distributed over $\{0,1\}^n$ as required by Definition 1.

Let $P_{r,s,b}$ be a source drawn from $\mathcal{D}$. Let us suppose for now that $b = 1$, and let us consider the execution of A when it is run using a sample in which only

positive examples drawn from $P_{r,s,1}$ (i.e. positive examples of the concept $c_{r,s,1}$) are uncovered. Recall that under the conditions of the generative model that we consider, the algorithm A does not "know" that it is being run using positive versus negative examples; it only receives a set of unlabelled examples.

(Throughout the following informal discussion we assume that both $r \neq 0^n$ and $s \neq 0^n$; note that the probability that either of these strings is 0^n is at most $2/2^n$. We further assume that $f_{|r| \oplus |s|}$ is not identically 1 or identically -1; recall from the discussion at the end of Section 3.1 that this fails to hold with probability $1/n^{\omega(1)}$. Under these assumptions a random example from $P_{r,s,1}$ has a 1/6 chance of being a positive/negative type-1/2/3 example.)

The examples that A receives will be distributed as follows:

- **Type-1 examples** $(x, 1)$**:** By our assumptions, 1/3 of the uncovered examples that A receives will be type-1 examples; these examples are uniformly distributed over all $x \in \{0,1\}^n$ that satisfy $AND_r(x)$. As we will see in Section 4, it is easy for A to completely identify r using these examples.
- **Type-2 examples** $(x, 2)$**:** By our assumptions, 1/3 of the uncovered examples A receives will be type-2 examples, each of which has x uniformly distributed over all strings that satisfy OR_s. As we will show in Section 5, for any $r \in \{-1, 0, 1\}^n$ the distribution of these type-2 examples (taken over the random choice of s from TARGET and the random draw of the examples from $P_{r,s,1}$) is statistically indistinguishable from the uniform distribution over $\{0,1\}^n$ to any algorithm (such as A) that receives only poly(n) many draws. Thus, as far as A can tell, the type-2 examples it receives are independently and uniformly drawn from $\{0,1\}^n$; intuitively we view this as meaning that A gets no useful information about s from the type-2 examples, so we informally view $|s|$ as uniform random and unknown to A.
- **Type-3 examples** $(x, 3)$**:** By our assumptions, 1/3 of the uncovered examples A receives will be type-3 examples. Intuitively, since $|s|$ is uniform random and unknown to A, even though r is known to A, the seed $t = |r| \oplus |s|$ to the pseudorandom function is uniform random and unknown to A. It follows from the definition of pseudorandomness that the function f_t is indistinguishable to algorithm A from a truly random function, so type-3 examples give no useful information to A; as far as A can tell, the type-3 examples it receives are simply uniform random strings drawn from $\{0,1\}^n$.

Thus we may informally view the hypothesis that A constructs, when run on positive examples drawn from $P_{r,s,1}$ where r and s were drawn from TARGET, as being determined only by the information "$(r, 1)$" (meaning that r is the string that governs the distribution of type-1 examples in the sample used for learning); the type-2 and type-3 examples that A receives are indistinguishable from uniform random strings. (The indistinguishability is statistical for the type-2 examples and computational for the type-3 examples; see Proposition 1 and Lemma 1 respectively of Section 5, where we make these arguments precise.) We thus write $h_{r,1}$ to denote the hypothesis that A constructs in this case.

An analogous argument shows that we may view the hypothesis that A constructs when run on negative examples drawn from $c_{r,s,1}$ as being determined

only by the information "$(s, 2)$" (meaning that s is the string that governs the distribution of type-2 examples in the sample); in this case the type-1 and type-3 examples in the sample are indistinguishable from truly random strings. We write $h_{s,2}$ to denote the hypothesis that A constructs in this case.

Now let us consider a setting in which the target source is $P_{-r,-s,-1}$ (where for $z \in \{-1, 0, 1\}^n$, the string $-z$ is simply $(-z_1, \ldots, -z_n)$) and r, s (or equivalently $-r, -s$) are independently drawn from TARGET. This time we will consider the execution of A when it is run using a sample in which only *negative* examples from $P_{-r,-s,-1}$ are uncovered, with the same assumptions on r, s and $f_{|r \oplus s|}$ as above. The examples that A receives will be distributed as follows:

- **Type-1 examples** $(x, 1)$**:** By definition of $P_{-r,-s,-1}$, 1/3 of the uncovered examples that A receives will be type-1 examples. These examples are uniformly distributed over all $x \in \{0, 1\}^n$ that do *not* satisfy $OR_{-r}(x)$, i.e. over all x that satisfy AND_r. Thus the negative type-1 examples in this case are distributed identically to the positive type-1 examples for $P_{r,s,1}$.
- **Type-2 examples** $(x, 2)$**:** 1/3 of the uncovered examples A receives will be type-2 examples, each of which has x uniformly distributed over all strings that do not satisfy AND_{-s}, i.e over all strings that satisfy OR_s. Thus the negative type-2 examples for $P_{-r,-s,-1}$ are distributed identically to the positive type-2 examples for $P_{r,s,1}$ (and as in that case, algorithm A gets no useful information about s from the type-2 examples, so we may view s as uniform random and unknown to A).
- **Type-3 examples** $(x, 3)$**:** The seed $|-r| \oplus |-s| \in \{0, 1\}^n$ is identical to the seed $t = |r| \oplus |s|$ that arose from $P_{r,s,1}$ above. As above, since s is uniform random and unknown to A, the function f_t is indistinguishable from a truly random function to A.

Thus we have arrived at the following crucial observation: *A cannot distinguish between when it is run on positive examples from $P_{r,s,1}$ versus negative examples from $P_{-r,-s,-1}$.* (The two distributions differ only in the type-3 examples, where in the negative $c_{-r,-s,-1}$ case they are uniform over $f_t(-1)$ and in the positive $c_{r,s,-1}$ case they are uniform over $f_t(1)$. By the pseudorandomness of f_t these distributions are indistinguishable from each other, since they are each indistinguishable from the uniform distribution over $\{0, 1\}^n$.) So we may informally view the hypothesis that A constructs as being $h_{r,1}$ in both cases.

Likewise, whether A is run on negative examples from $P_{r,s,1}$ or positive examples from $P_{-r,-s,-1}$, the resulting hypothesis is $h_{s,2}$ in both cases.

Now suppose that A is a successful generative learning algorithm in the PAB sense, i.e. the final hypothesis obtained from source $P_{r,s,1}$ (which we denote $h_{r,s,1}$, and which equals $\mathrm{sgn}(h_{r,1}(x, i) - h_{s,2}(x, i))$) has very high accuracy. Since the overall error rate of $h_{r,s,1}$ is at least 1/3 of its error rate on type-3 examples, this means that $h_{r,s,1}(x, 3)$ must be well-correlated with $f_{|r|\oplus|s|}(x)$. On the other hand, as argued above, the final hypothesis $h_{-r,-s,-1}$ obtained from source $P_{-r,-s,-1}$ is $\mathrm{sgn}(h_{s,2}(x, i) - h_{r,1}(x, i))$, and this must have high accuracy on type-3 examples from this source; so $h_{-r,-s,-1}(x, 3)$ is well-correlated with

$f_{|-r|\oplus|-s*|}(x)$. But this is impossible because $f_{|-r|\oplus|-s|}$ is identical to $f_{|r|\oplus|s|}$ whereas $h_{-r,-s,-1}(x,3)$ is easily seen to be the negation of $h_{r,s,1}(x,3)$.

This concludes our informal presentation of why learning problem $\mathcal{P}$ is hard for any generative learning algorithm. In Section 5 we give a precise cryptographic instantiation of the above intuitive argument to prove that generative algorithms cannot succeed.

4 Discriminative Algorithms can Succeed

Theorem 3. *There is a polynomial-time discriminative learning algorithm that can solve learning problem $\mathcal{P}$.*

Proof Sketch. We use Valiant's algorithm [11], which keeps all literals that are not eliminated as possibilities by the training data, to learn r and s. The probability that any incorrect literal is not eliminated by q examples is at most $2n(1/2)^q$. So r and s can be learned exactly; it is easy to "guess and check" b, and thus learn the target $c_{r,s,b}$ exactly. (See [7] for a full proof.) ■

5 Generative Algorithms must Fail

We prove the following theorem, which shows that no generative learning algorithm can succeed on learning problem $\mathcal{P}$.

Theorem 4. *Let A be any poly(n)-time algorithm that operates in the generative learning framework and has the following property: when run on examples from any source in $\mathcal{P}$, with probability at least $1-1/n$ A outputs a final hypothesis h whose error rate is at most ϵ. Then $\epsilon \geq \frac{1}{6} - o(1)$.*

Let us set up the framework. Let A be any poly(n) time generative algorithm. We can view A as a mapping from (filtered) samples to hypotheses. Given a sample S we write $A(S)$ to denote the hypothesis that A outputs on S, and we write $A(S)(x)$ to denote the real-valued output of this hypothesis on x.

5.1 Positive examples from $P_{r,s,1}$

Fix any $r \in \{-1,0,1\}^n$. Consider a source $P_{r,s,1}$ where s is drawn from TARGET. We first show that for any generative algorithm A that takes as input a sample of $m = \text{poly}(n)$ many examples from such a source with only the positive examples exposed, the type-2 examples in its sample are statistically indistinguishable from uniform random examples over $\{0,1\}^n$.

To make this precise, we need the following definitions.

Definition 2. *If P is a source, define P_+ to be the probability distribution over $X \cup \{\diamond\}$ obtained by (i) choosing (x,y) according to P, and (ii) emitting x if $y = 1$ and emitting $\diamond$ if $y = -1$. Define P_- analogously with the labels reversed.*

Definition 3. *Let $D^+_{r,1}$ be the distribution over sets S_+ of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which is defined as follows: to draw a set S_+ from $D^+_{r,1}$ (i) first draw s from* TARGET*, and (ii) then draw each of the m examples in S_+ independently from $(P_{r,s,1})_+$.*

Definition 4. *Let $\widetilde{D}^+_{r,1}$ be the distribution over sets $\widetilde{S}_+$ of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ defined as follows: to draw a set $\widetilde{S}_+$ from $\widetilde{D}^+_{r,1}$, (i) first draw s and S_+ as described above from $D^+_{r,1}$, and (ii) then replace each type-2 example $(x,2)$ in S_+ with a new example $(z,2)$ where each time z is an independent and uniform string in $\{0,1\}^n$.*

A fairly direct calculation establishes the following (see [7] for full proof):

Proposition 1. *For large enough n, for any $r \in \{-1,0,1\}^n$, the distributions $D^+_{r,1}$ and $\widetilde{D}^+_{r,1}$ have statistical distance at most $2^{-n/8}$.*

Thus the distributions $D^+_{r,1}$ and $\widetilde{D}^+_{r,1}$ are statistically indistinguishable. We now recall the notion of *computational* indistinguishability of two distributions:

Definition 5. *Let $p(n)$ be a fixed polynomial and let $\{X_n\}_{n\geq 1}$ and $\{Y_n\}_{n\geq 1}$ be two families where for each n, both X_n and Y_n are distributions over $\{0,1\}^{p(n)}$. $\{X_n\}_{n\geq 1}$ and $\{Y_n\}_{n\geq 1}$ are said to be* computationally indistinguishable *if for all constants $c > 0$, all p.p.t. algorithms A, and all sufficiently large n, we have*

$$|\Pr_{S_X \in X_n}[A(S_X) = 1] - \Pr_{S_Y \in Y_n}[A(S_Y) = 1]| < 1/n^c.$$

Intuitively, two distributions are computationally indistinguishable (henceforth abbreviated c.i.) if no probabilistic polynomial-time algorithm can distinguish whether a random draw comes from one of the distributions or the other with nonnegligible advantage over a random guess. We will use the following facts:

- Computational indistinguishability is transitive: if X_n and Y_n are c.i., and Y_n and Z_n are c.i., then X_n and Z_n are c.i..
- If X_n and Y_n are c.i., and Y_n and Z_n have statistical distance $\|Y_n - Z_n\|_1 = 1/n^{\omega(1)}$, then X_n and Z_n are c.i..

We now show that for any generative algorithm A that takes as input a sample of $m = \text{poly}(n)$ many positive examples from $P_{r,s,1}$ (where $0^n \neq r$ is any fixed string and s is drawn from TARGET), the type-2 and type-3 examples in its sample are computationally indistinguishable from uniform random examples over $\{0,1\}^n$. That is, positive examples for OR_s cannot be reliably distinguished from uniform draws from $\{0,1\}^n$ in polynomial time, and neither can uniform random elements of $f^{-1}_{|r|\oplus|s|}(1)$.

Definition 6. *Let $\widehat{D}^+_{r,1}$ be the distribution over sets $\widehat{S}_+$ of m examples from $\{0,1\}^n \times \{1,2,3\}$ which is defined as follows: to draw a set $\widehat{S}_+$ from $\widehat{D}^+_{r,1}$, (i) first draw s and S_+ as described above from $D^+_{r,1}$, and (ii) for $i = 2,3$ replace each type-i example (x,i) in S_+ with an example (z,i) where each time z is an independent and uniform random string from $\{0,1\}^n$.*

Lemma 1. *For any $0^n \neq r \in \{-1,0,1\}^n$, the two distributions $D^+_{r,1}$ and $\widehat{D}^+_{r,1}$ are computationally indistinguishable.*[3]

[3] The lemma also holds for $r = 0^n$, but this result suffices and has a simpler proof.

Proof. Suppose to the contrary that $D^+_{r,1}$ and $\widehat{D}^+_{r,1}$ are not computationally indistinguishable. Let Z be a p.p.t. algorithm which is such that

$$|\Pr_{S_+ \in D^+_{r,1}}[Z(S_+) = 1] - \Pr_{\widehat{S}_+ \in \widehat{D}^+_{r,1}}[Z(\widehat{S}_+) = 1]| > 1/n^c \quad (1)$$

for some $c > 0$ and infinitely many n. We show how such a Z can be used to obtain a distinguishing algorithm that "breaks" the PRFF, and thus obtain a contradiction.

Consider the following algorithm Z', which uses Z as a subroutine and accesses f as an oracle: Given black-box access to a function $f\colon \{0,1\}^n \to \{0,1\}$, construct an m-element sample S by performing the following m times:

- Toss a fair coin; if "heads," output $\diamond$. If "tails:"
  (*) Choose a uniform random index $i \in \{1,2,3\}$. If $i = 1$, output "$(x,1)$" where x is a uniformly chosen input that satisfies AND_r (i.e. $(x,1)$ is a random type-1 example). If $i = 2$, output "$(x,2)$" where x is a uniform random n-bit string. If $i = 3$, give random n-bit inputs to f until one is obtained (call it x) for which $f(x) = 1$; output "$(x,3)$." If more than n random n-bit inputs are tried without finding one which has $f(x) = 1$, abort the procedure (an arbitrary sample that is fixed ahead of time may be output in this case, say for example m copies of $(0^n, 1)$).

Now run Z on S and output whatever it outputs.

Recall that our plan is to show that Z, which can tell apart $D^+_{r,1}$ from $\widehat{D}^+_{r,1}$, can be used to tell a pseudo-random function from a truly random function. Roughly speaking, our first proposition says that $\widehat{D}^+_{r,1}$ is a faithful proxy for the result of applying a truly random function:

Proposition 2. *Suppose f is a truly random function. Let $D_{truerand}$ denote the distribution over samples S that results from performing (*) above $m = poly(n)$ times with f. Then the statistical distance between $D_{truerand}$ and $\widehat{D}^+_{r,1}$ is $\frac{1}{n^{\omega(1)}}$.*

Proof Sketch for Proposition 2 (see [7] for full proof): We first show that wlog we can assume that (*) above does not abort, that s (chosen in the definition of $\widehat{D}^+_{r,1}$) is not 0^n, and that $f_{|r|\oplus|s|}$ takes both positive and negative values.

Given these assumptions, the uniform random choice of an index $i \in \{1,2,3\}$ in the executions of (*) under $D_{truerand}$ faithfully simulates what is going on in $\widehat{D}^+_{r,1}$. It can be shown that we thus have that the distribution of type-1 and type-2 examples under the two distributions $D_{truerand}$ and $\widehat{D}^+_{r,1}$ are identical.

We now analyze the distribution of type-3 examples. Under $\widehat{D}^+_{r,1}$, each draw is with probability $1/3$ a type-3 example $(x,3)$ where x is uniform over $\{0,1\}^n$. Under $D_{truerand}$, each draw is with probability $1/3$ a type-3 example $(x,3)$ where x is drawn uniformly from $f^{-1}(1)$, where f is a truly random function (chosen once and for all before the m draws are made). Thus, for any $m' \leq m$ the probability of receiving exactly m' type-3 examples is the same under each of the two distributions. An easy Chernoff bound shows that with probability at

least $1-\frac{1}{2^n}$, the fraction of positive examples for a truly random f is $\frac{1}{2}\pm 1/2^{\Theta(n)}$. Thus, with high probability, a truly random f has $f^{-1}(1)$ uniformly distributed over an exponentially large set. This implies that a polynomial-size sample is exponentially unlikely to have any repetitions among the positive examples of f. Symmetry implies that, conditioned on a fixed values of the number m' of positive type-3 examples, and conditioned on the event that they are distinct, any set of m' examples are equally likely to be chosen. This is of course also the case if we draw m' examples uniformly from $\{0,1\}^n$. This establishes that the contribution to the statistical distance between $\widehat{D}^+_{r,1}$ and $D_{truerand}$ from type-3 examples is at most $1/2^{\Omega(n)}$, and establishes the proposition. (Proposition 2)■

Our next proposition shows that $D^+_{r,1}$ is a faithful proxy for the result of using a pseudo-random function.

Proposition 3. *Suppose f is a pseudorandom function, i.e. $f = f_t$ where t is drawn uniformly from $\{0,1\}^n$. Let $D_{pseudorand}$ denote the distribution over samples S in which the positive examples are obtained using (*) with this choice of f. Then the statistical distance between $D_{pseudorand}$ and $\widetilde{D}^+_{r,1}$ is at most $\frac{1}{n^{\omega(1)}}$.*

Proof Sketch for Proposition 3 (see [7] for full proof): As in the case of Proposition 2, we have that with probability $1 - 1/n^{\omega(1)}$ both (i) the string s chosen in the definition of $\widetilde{D}^+_{r,1}$ is not 0^n and (ii) the seed $|r| \oplus |s|$ is such that $f_{|r|\oplus|s|}$ assumes both $+$ and $-$ values, so we may assume that (i) and (ii) hold.

As in the earlier proof, since $r \neq 0^n$ this implies that each positive example from $\widetilde{D}^+_{r,1}$ has probability $1/3$ of being a type-1, type-2, or type-3 example, and the same is true for each example from $D_{pseudorand}$. Given this, the distribution of type-1 examples is easily seen to be the same under $\widetilde{D}^+_{r,1}$ and under $D_{pseudorand}$, and the same is true for the distribution of type-2 examples. The distribution of type-3 examples under $D_{pseudorand}$ is that each is chosen uniformly at random from $f_t^{-1}(1)$ where t is uniform random over $\{0,1\}^n$, whereas the distribution of type-3 examples under $\widetilde{D}^+_{r,1}$ is that each is chosen uniformly at random from $f_t^{-1}(1)$ where $t = |r| \oplus |s|$; this string is uniform random conditioned on the event that (i) and (ii) both hold. Since the probability that either (i) or (ii) fails to hold is $1/n^{\omega(1)}$, the proposition follows. (Proposition 3)■

Propositions 3 and 1 together yield that $D_{pseudorand}$ has statistical distance $1/n^{\omega(1)}$ from $D^+_{r,1}$. Combining this with Proposition 2 and Equation (1), we have that the p.p.t. algorithm Z' satisfies

$$\left| \Pr_{s\in\{0,1\}^n}[(Z')^{f_s}(1^n) \texttt{ outputs } 1] - \Pr_{F\in\mathcal{R}_n}[(Z')^{F}(1^n) \texttt{ outputs } 1]\right| > 1/n^{c'}.$$

for infinitely many n, where c' is any constant larger than c. But this violates the fact that $\{f_t\}$ is a PRFF. This proves Lemma 1. ■

5.2 Negative examples from $P_{-r,-s,-1}$

We now give results for negative examples from $P_{-r,-s,-1}$ that are dual to the results we gave for positive examples from $P_{r,s,1}$ in the last section. This will let

us show (Corollary 1 below) that positive examples drawn from $P_{r,s,1}$ are computationally indistinguishable from negative examples drawn from $P_{-r,-s,-1}$.

Fix any $r \in \{-1,0,1\}^n$. We now consider a source $P_{-r,-s,-1}$ where s (or equivalently $-s$) is drawn from TARGET. In analogy with Definitions 3 and 4, let $D^-_{-r,-1}$ be the distribution over sets S_- of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which is defined as follows: to draw a set S_- from $D^-_{-r,-1}$, (i) first draw s from TARGET, and (ii) then draw each of the m examples in S_- independently from $(P_{-r,-s,1})_-$. Let $\widehat{D}^-_{-r,-1}$ be the distribution over sets $\widehat{S}_-$ of m examples from $\{0,1\}^n \times \{1,2,3\}$ which is defined as follows: to draw a set $\widehat{S}_-$ from $\widehat{D}^-_{r,1}$, (i) first draw s and S_- as described above from $D^-_{-r,-1}$, and (ii) for $i = 2,3$ replace each type-i example (x,i) in S_- with a fresh uniform example (z,i).

Dual arguments to those in Section 5.1 give the following Lemma 1 analogue:

Lemma 2. *For any $0^n \neq r \in \{-1,0,1\}^n$, the distributions $D^-_{-r,-1}$ and $\widehat{D}^-_{-r,-1}$ are computationally indistinguishable.*

The following proposition relates $D^+_{r,1}$ and $D^-_{-r,-1}$ (see [7] for proof):

Proposition 4. *For any $0 \neq r \in \{-1,0,1\}^n$, the distributions $\widehat{D}^+_{r,1}$ and $\widehat{D}^-_{-r,-1}$ have statistical distance at most $1/n^{\omega(1)}$.*

Lemma 1, Lemma 2 and Proposition 4 together give:

Corollary 1. *For any $0 \neq r \in \{-1,0,1\}^n$, the distributions $D^+_{r,1}$ and $D^-_{-r,-1}$ are computationally indistinguishable.*

5.3 Negative examples from $P_{r,s,1}$ & positive examples from $P_{-r,-s,-1}$

Dual arguments to those of Sections 5.1 and 5.2 can be used to show that negative examples from $P_{r,s,1}$ and positive examples from $P_{-r,-s,-1}$ are c.i.. More precisely, fix any $s \in \{-1,0,1\}$. Similar to Definitions 3 and 4, let $D^-_{s,1}$ be the distribution over sets S_- of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which is defined as follows: to draw a set S_- from $D^-_{s,1}$, (i) first draw r from TARGET, and (ii) then draw each of the m examples in S_- independently from $(P_{r,s,1})_-$. Let $\widehat{D}^-_{s,1}$ be the distribution over sets $\widehat{S}_-$ of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which is defined as follows: to draw $\widehat{S}_-$ from $\widehat{D}^-_{s,1}$, (i) first draw r and S_- as described above from $D^-_{s,1}$, and (ii) for $i = 1,3$ replace each type-i example (x,i) in S_- with a new uniform example (z,i). Dual arguments to Section 5.1 give:

Lemma 3. *For any $0^n \neq s \in \{-1,0,1\}^n$, the two distributions $D^-_{s,1}$ and $\widehat{D}^-_{s,1}$ are computationally indistinguishable.*

Fix any $s \in \{-1,0,1\}$. Let $D^+_{-s,-1}$ be the distribution over sets S_+ of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which is defined as follows: to draw a set S_+ from $D^+_{-s,-1}$, (i) first draw r from TARGET, and (ii) then draw each of the m examples in S_+ independently from $(P_{-r,-s,-1})_+$. Let $\widehat{D}^+_{-s,-1}$ be the distribution over sets $\widehat{S}_+$ of m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$ which

is defined as follows: to draw a set $\widehat{S}_+$ from $\widehat{D}^+_{-s,-1}$, (i) first draw r and S_+ as described above from $D^+_{-s,-1}$, and (ii) for $i = 1, 3$ replace each type-i example (x, i) in S_+ with a fresh uniform example (z, i). As before, we have the following:

Lemma 4. *For any $0^n \neq s \in \{-1, 0, 1\}^n$, the two distributions $D^+_{-s,-1}$ and $\widehat{D}^+_{-s,-1}$ are computationally indistinguishable.*

Proposition 5. *For any $0^n \neq s \in \{-1, 0, 1\}^n$, the distributions $\widehat{D}^-_{s,1}$ and $\widehat{D}^+_{-s,-1}$ have statistical distance at most $1/n^{\omega(1)}$.*

Corollary 2. *For any $0^n \neq s \in \{-1, 0, 1\}^n$, the distributions $D^-_{s,1}$ and $D^+_{-s,-1}$ are computationally indistinguishable.*

5.4 Proof of Theorem 4

As in the theorem statement, let A be any poly(n)-time purported generative learning algorithm which, when run on examples from any source $P \in \mathcal{P}$, outputs a final hypothesis h whose error rate on P is at most ϵ with probability at least $1 - \delta$ where $\delta = 1/n$. We will show that $\epsilon \geq \frac{1}{6} - o(1)$.

Algorithm B will make use of oracle access to distributions D_Y and D_Z over m examples from $(\{0,1\}^n \times \{1,2,3\}) \cup \{\diamond\}$, and will output a bit. Here it is:

- Draw r, s independently from TARGET. Let $t = |r| \oplus |s|$.
- Draw S_Y from D_Y and S_Z from D_Z.
- Apply A to S_Y to get h_Y, and to S_Z to get h_Z, with parameters $\epsilon = \delta = 1/n$.
- Pick a uniform $x \in \{0,1\}^n$ and output the value $f_t(x) \cdot \mathrm{sgn}(h_Y(x, 3) - h_Z(x, 3))$.

h_Y and h_Z will be functions for scoring elements for positivity or negativity. By applying B with different sources, each function will adopt each role. This will let us conclude that the final accuracy on type-3 examples must be low.

1. Suppose first that D_Y is $((P_{r,s,1})_+)^m$ and D_Z is $((P_{r,s,1})_-)^m$. Then with probability at least $1 - \delta - 3\epsilon - 1/n$, the output of B must be 1. (To see this, recall that by assumption, for any $r, s \in \{-1, 0, 1\}^n$ the final hypothesis A produces should be ϵ-accurate with probability $1 - \delta$. Also, as noted in Section 3.1, for r, s drawn from TARGET with probability at least (say) $1 - 1/n^2$ we have that both $0^n \neq r, s$ and $f_{|r| \oplus |s|}$ has a fraction of positive examples which is bounded by $[\frac{1}{2} - \frac{1}{n^2}, \frac{1}{2} + \frac{1}{n^2}]$, and consequently each type-3 example $(x, 3)$ has total probability weight in $[\frac{1}{3}(\frac{1}{2} - \frac{1}{n^2}), \frac{1}{3}(\frac{1}{2} + \frac{1}{n^2})]$. Consequently if A's final hypothesis has overall error rate at most ϵ under $P_{r,s,1}$ over all of $\{0,1\}^n \times \{1,2,3\}$, then its error rate on uniformly chosen type-3 examples must certainly be at most $3\epsilon + 1/n$.) Let p_1 denote the probability that B outputs 1 in this case.
2. Now, suppose that D_Y is the distribution $((P_{-r,-s,-1})_-)^m$ and, as in case 1 above, D_Z is $((P_{r,s,1})_-)^m$. Let p_2 denote the probability that B outputs 1 in this case. By Corollary 1, we know that for every fixed $0^n \neq r \in \{-1, 0, 1\}^n$, the distributions $D^+_{r,1}$ (where s is drawn from TARGET) and $D^-_{-r,-1}$ (where

s is again drawn from TARGET) are computationally indistinguishable. It follows that the distributions $((P_{-r,-s,-1})_-)^m$ (where both r and s are drawn from TARGET) and $((P_{r,s,1})_+)^m$ (where both r and s are drawn from TARGET) are computationally indistinguishable. This gives us that $|p_1 - p_2| \leq 1/n^{\omega(1)}$, for otherwise B would be a polynomial-time algorithm that violates the computational indistinguishability of these distributions.

3. Now suppose that, as in Case 2, D_Y is the distribution $((P_{-r,-s,-1})_-)^m$, and that D_Z is $((P_{-r,-s,-1})_+)^m$. Let p_3 denote the probability that B outputs 1 in this case. As argued in case (2) above, Corollary 2 gives us that $((P_{r,s,1})_-)^m$ and $((P_{-r,-s,-1})_+)^m$ are computationally indistinguishable, where in both cases r, s are drawn from TARGET. This gives us that $|p_2 - p_3| < 1/n^{\omega(1)}$.

Putting together the pieces, we have that $p_3 \geq p_1 - \frac{1}{n^{\omega(1)}} \geq 1 - \delta - 3\epsilon - \frac{1}{n} - \frac{1}{n^{\omega(1)}} = 1 - o(1) - 3\epsilon$ (since $\delta = 1/n$). But under the assumption that A is a successful generative algorithm for $\mathcal{P}$, it must be the case in case (3) that $p_3 \leq \delta + 3\epsilon + o(1) = 3\epsilon + o(1)$. This is because in case (3) the hypothesis h_Y is the *negative* example hypothesis and h_Z is the *positive* example hypothesis, so the generative algorithm's final hypothesis on type-3 examples (which, as argued in case (1) above, has error at most $3\epsilon + 1/n$-accurate on such examples with probability at least $1 - \delta - o(1)$) is $\mathrm{sgn}(h_Z(x,3) - h_Y(x,3))$. We thus have $3\epsilon + o(1) > p_3 > 1 - o(1) - 3\epsilon$ which gives $\epsilon \geq \frac{1}{6} - o(1)$. (Theorem 4)■

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Wiley, 2000.

[2] P. Fischer, S. Pölt, and H. U. Simon. Probably almost Bayes decisions. In *Proceedings of the Fourth Annual COLT*, pages 88–94, 1991.

[3] P. Goldberg. When Can Two Unsupervised Learners Achieve PAC Separation? In *Proceedings of the 14th Annual COLT*, pages 303–319, 2001.

[4] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[5] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*, pages 487–493. Morgan Kaufmann, 1998.

[6] T. Jebara. *Machine learning: discriminative and generative*. Kluwer, 2003.

[7] P. Long and R. Servedio. Discriminative Learning can Succeed where Generative Learning Fails (full version). Available at http://www.cs.columbia.edu/~rocco/papers/discgen.html.

[8] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*, 2001.

[9] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum. Classification with hybrid generative/discriminative models. *NIPS*, 2004.

[10] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[11] L. G. Valiant. A theory of the learnable. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 436–445. ACM Press, 1984.

[12] V. Vapnik. *Estimations of dependences based on statistical data*. Springer, 1982.