Using More Data to Speed-up Training Time

Shai Shalev-Shwartz
The Hebrew University
shais@cs.huji.ac.il

Ohad Shamir
Microsoft Research New-England
ohad@microsoft.com

Eran Tromer
Tel-Aviv University
tromer@cs.tau.ac.il

Abstract

In many recent applications, data is plentiful. By now, we have a rather clear understanding of how more data can be used to improve the accuracy of learning algorithms. Recently, there has been a growing interest in understanding how more data can be leveraged to reduce the required training runtime. In this paper, we study the runtime of learning as a function of the number of available training examples, and underscore the main high-level techniques. We provide some initial positive results showing that the runtime can decrease exponentially while only requiring a polynomial growth of the number of examples, and spell-out several interesting open problems.

1 Introduction

Machine learning are now prevalent in a large range of scientific, engineering and every-day tasks, ranging from analysis of genomic data, through vehicle and aircraft control to locating information on the web and providing users with personalized recommendations. Meanwhile, our world has become increasingly "digitized" and the amount of data available for training is dramatically increasing. By now, we have a rather clear understanding of how more data can be used to improve the *accuracy* of learning algorithms. In this paper we study how more data can be beneficiary for constructing more *efficient* learning algorithms.

Roughly speaking, one way to show how more data can reduce the training runtime is as follows. Consider learning by finding a hypothesis in the hypothesis class that minimizes the training error. In many situations, this search problem is computationally hard. One can circumvent the hardness by replacing the original hypothesis class with a different (larger) hypothesis class, such that the search problem in the larger class is computationally easier (e.g., the search problem in the new hypothesis class reduces to a convex optimization problem). On the flip side, from the statistical point of view, the estimation error in the new hypothesis class might be larger than the estimation error in the original class, and thus, with a small number of examples, learning the larger class might lead to overfitting even though the same amount of examples suffices for the original hypothesis class. However, having more training examples keeps the overfitting in check. In particular, if the number of extra examples we need for learning the new class is only polynomially larger than the original number of examples, we end up with an efficient algorithm for the original problem. If, however, we don't have those extra examples, our only option is to learn the original hypothesis class, which may be computationally harder.

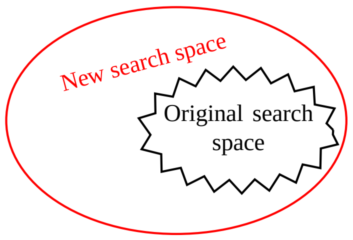


Figure 1: The Basic Approach

The goal of this paper is to present a formal model for studying the runtime of learning algorithms as a function of the available number of examples. After defining the formal model, we present a binary classification learning problem for which we can provably (based on standard cryptographic assumption) demonstrate an inverse dependence of the runtime on the number of examples. While there have been previous constructions which demonstrated a similar phenomenon, assuming the existence of a "perfect" hypothesis, we show this in the much more natural agnostic model of learning. A possible criticism is that our learning problem is still rather synthetic. We continue with presenting several learning problems, which arise in natural settings, that have more efficient algorithms by relying on the availability of more training data. Some of these examples are based on the intuition of Figure 1, but some are also based on other ideas and techniques. However, for all these problems, the analysis is based on upper bounds without having matching lower bounds. This raises several interesting open problems.

1.1 Related Work

[6] were the first to jointly study the computational and sample complexity, and to show that a tradeoff between runtime and sample size exists. In particular, they distinguish between the information theoretic sample complexity of a class and its computational sample complexity, the latter being the number of examples needed for learning the class in polynomial time. They presented a learning problem which is not efficiently learnable from a small training set, and is efficient learnable from a polynomially larger training set. [11] showed that for a concept class composed of 1-decision-lists over $\{0,1\}^n$, which can be learned inefficiently using $O(1)$ examples, no algorithm can learn it efficiently using $o(n)$ examples, and there is an efficient algorithm using $\Omega(n)$ examples. The construction was also extended to k decision-lists, $k \geq 1$. with larger gaps.

In contrast to [6, 11], which focused on learning under the realizable case (namely, that the labels are generated by some hypothesis in the class), we mostly focus on the more natural *agnostic* setting, where any distribution over the example domain is possible, and there may be no hypothesis h in our class that never errs. This is not just a formality - in both [6, 11], the construction crucially relies on the fact that the labels are provided by some hypothesis in the class. In terms of techniques, we rely on the cryptographic assumption that one-way permutations exist, which is the same assumption as in [11] and similar to the assumption in [6]. We note that cryptographic assumptions are common in proving lower bounds for efficient learnability, and in some sense they are even necessary [2]. However, our construction is very different. For example, in both [6, 11], revealing information on the identity of the "correct" hypothesis is split among many different examples. Therefore, efficient learning is possible after sufficiently many examples are collected, which then allows us to return the "correct" hypothesis. In our agnostic setting, there is no "correct" hypothesis, so this kind of approach cannot work. Instead, our efficient learning procedure computes and returns an improper predictor, which is not in the hypothesis class at all.

A potential weaknesses of our example, as well as the example given in [6], is that our hypothesis class does not consist of "natural" hypotheses. The class employed in [11] is more natural, but it is also a very carefully constructed subset of decision lists. The goal of the second part of the paper is to demonstrate gaps (though based on upper bounds) for natural learning problems.

Another contribution of our model is that it captures the exact tradeoff between sample and computational complexity rather then only distinguishing between polynomial and non-polynomial time, which may not be refined enough. Bottou and Bousquet [5] initiated a study on learning in the *data laden domain* – a scenario in which data is plentiful and computation time is the main bottleneck. This is the case in many real life applications nowadays. Shalev-Shwartz and Srebro [13] continued this line of research and showed how for the problem of training Support Vector Machines, a joint statistical-computational analysis reveals how the runtime of stochastic-gradient-descent can potentially *decrease* with the number of training examples. However, this is only demonstrated via upper bounds. More importantly, the advantage of having more examples only improves running time by constant factors. In this paper, we will be interested in larger factors of improvement, which scale with the problem size.

2 Formal Model Description

We consider the standard model of supervised statistical learning, in which each training example is an instance-target pair and the goal of the learner is to use past examples in order to predict the targets associated with future instances. For example, in spam classification problems, an instance is an email message and the target is either $+1$ ('spam') or -1 ('benign'). We denote the instance domain by $\mathcal{X}$ and the target domain by $\mathcal{Y}$. A prediction rule is a mapping $h : \mathcal{X} \to \mathcal{Y}$. The performance of a predictor h on an instance-target pair, $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, is measured by a loss function $\ell(h(\mathbf{x}), y)$. For example, a natural loss function for classification problems is the 0-1 loss, $\ell(h(\mathbf{x}), y) = 1$ if $y \neq h(\mathbf{x})$ and 0 otherwise.

A learning algorithm, A, receives a training set of m examples, $S_m = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$, which are assumed to be sampled i.i.d. from an unknown distribution $\mathcal{D}$ over the problem domain $\mathcal{Z} \subseteq \mathcal{X} \times \mathcal{Y}$. Using the training data, together with any prior knowledge or assumptions about the distribution $\mathcal{D}$, the learner forms a prediction rule. The predictor is a random variable and we denote it by $A(S_m)$. The goal of the learner is to find a prediction rule with low generalization error (a.k.a. risk), defined as the expected loss:

$$\mathrm{err}(h) \stackrel{\mathrm{def}}{=} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}), y)] \ .$$

The well known no-free-lunch theorem tells us that no algorithm can minimize the risk without making some prior assumptions on $\mathcal{D}$. Following the agnostic PAC framework, we require that the learner will find a predictor whose risk will be close to $\inf_{h\in\mathcal{H}} \mathrm{err}(h)$, where $\mathcal{H}$ is called a hypothesis class (which is known to the learner).

We use $\mathrm{err}(A(S_m))$ to denote the expected risk of the predictor returned by A, where expectation is with respect to the random choice of the training set. We denote by $\mathrm{time}(A, m)$ the upper bound on the expected runtime[1] of the algorithm A when running on any training set of m examples. The main mathematical object that we propose to study is the following:

$$T_{\mathcal{H},\epsilon}(m) = \min\{t : \exists\, A \text{ s.t. } \forall\, \mathcal{D},\ \mathrm{time}(A, m) \le t \wedge \mathrm{err}(A(m)) \le \inf_{h\in\mathcal{H}} \mathrm{err}(h) + \epsilon\} \ , \tag{1}$$

where when no t satisfies the above constraint we set $T_{\mathcal{H},\epsilon}(m) = \infty$. Thus, $T_{\mathcal{H},\epsilon}(m)$ measures the required runtime to learn the class $\mathcal{H}$ with an excess error of ϵ given a budget of m training examples. Studying this function can show us how more data can be used to decrease the required runtime of the learning algorithm. The minimum value of m for which $T_{\mathcal{H},\epsilon}(m) < \infty$ is the information-theoretic sample complexity. This corresponds to the case in which we ignore computation time. The other extreme case is the value of $T_{\mathcal{H},\epsilon}(\infty)$. This corresponds to the *data laden domain*, namely data is plentiful and computation time is the only bottleneck.

We continue with few additional definitions. In general, we make no assumptions on the distribution $\mathcal{D}$. However, we sometime refer to the realizable case, in which we assume that the distribution $\mathcal{D}$ satisfies $\min_{h\in\mathcal{H}} \mathrm{err}(h) = 0$. The empirical error on the training examples, called the training error, is denoted by $\mathrm{err}_S(h) \stackrel{\mathrm{def}}{=} \frac{1}{m}\sum_{i=1}^m \ell(h(\mathbf{x}_i), y_i)$. A common learning paradigm is Empirical Risk Minimization, denoted $\mathrm{ERM}_{\mathcal{H}}$, in which the learner can output any predictor in $\mathcal{H}$ that minimizes $\mathrm{err}_S(h)$. A learning algorithm is called *proper* if it always returns a hypothesis from $\mathcal{H}$. Throughout this paper we are concerned with *improper* learning, where the returned hypothesis can be any efficiently computed function h from instances $\mathbf{x}$ to labels y. Note that improper learning is just as useful as proper learning for the purpose of deriving accurate predictors.

2.1 A Warm-up Example

To illustrate how more data can reduce runtime, consider the problem of learning the class of 3-term disjunctive normal form (DNF) formulas in the realizable case. A 3-DNF is a Boolean mapping, $h : \{0,1\}^d \to \{0,1\}$, that can be written as $h(\mathbf{x}) = T_1(\mathbf{x}) \vee T_2(\mathbf{x}) \vee T_3(\mathbf{x})$, where for each i, $T_i(\mathbf{x})$ is a conjunction of an arbitrary number of literals, e.g. $T_i(\mathbf{x}) = x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_7$.

Since the number of 3-DNF formulas is at most 3^{3d}, it follows that the information theoretic sample complexity is $O(d/\epsilon)$. However, it was shown [10, 9] that unless RP=NP, the search problem of finding a 3-DNF formula which is (approximately) consistent with a given training set cannot be performed in $\mathrm{poly}(d)$ time. On the other hand, we will show below that if $m = \Theta(d^3/\epsilon)$ then $T_{\mathcal{H},\epsilon}(m) = \mathrm{poly}(d/\epsilon)$. Note that there is no contradiction between the last two sentences, since the former establishes hardness of *proper* learning while the latter claims feasibility of improper learning.

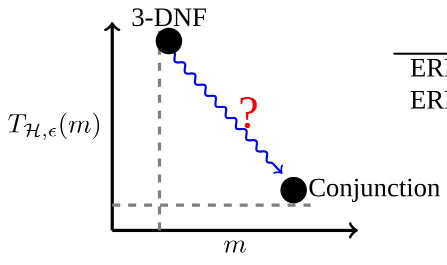
To show the positive result, observe that each 3-DNF formula can be rewritten as $\wedge_{u\in T_1, v\in T_2, w\in T_3}(u \vee v \vee w)$ for three sets of literals T_1, T_2, T_3. Define $\psi : \{0,1\}^d \to \{0,1\}^{2(2d)^3}$ such that for each triplet of literals u, v, w, there are two indices in $\psi(\mathbf{x})$, indicating if $u \vee v \vee w$ is true or false. Therefore, each 3-DNF can be represented as a single conjunction over $\psi(\mathbf{x})$. As a result, the class of 3-DNFs over $\mathbf{x}$ is a subset of the class of conjunctions over $\psi(\mathbf{x})$. The search problem of finding an ERM over the class of conjunctions is polynomially solvable (it can be

[1]To prevent trivialities, we also require that the runtime of applying $A(S_m)$ on any instance is at most $\mathrm{time}(A, m)$.

cast as a linear programming, or can be solved using a simple greedy algorithm). However, the information theoretic sample complexity of learning conjunctions over $2(2d)^3$ variables is $O(d^3/\epsilon)$. We conclude that if $m = \Theta(d^3/\epsilon)$ then $T_{\mathcal{H},\epsilon}(m) = \text{poly}(d/\epsilon)$.

It is important to emphasize that the analysis above is not satisfactory for two reasons. First, we do not know if it is not possible to improperly learn 3-DNFs in polynomial time using $O(d/\epsilon)$ examples. All we know is that the ERM approach is not efficient. Second, we do not know if the information theoretic sample complexity of learning conjunctions over $\psi(\mathbf{x})$ is $\Omega(d^3/\epsilon)$. Maybe the specific structure of the range of ψ yields a lower sample complexity.

But, if we do believe that the above analysis indeed reflects reality, we obtain two points on the curve $T_{\mathcal{H},\epsilon}(m)$. Still, we do not know how the rest of the curve looks like. This is illustrated below.



	Samples	Time
ERM over 3-DNF	d/ϵ	not $\text{poly}(d)$
ERM over Conjunctions	d^3/ϵ	$\text{poly}(d/\epsilon)$

3 Formal derivation of gaps

In this section, we formally show a learning problem which exhibits an inverse dependence of the runtime on the number of examples. As discussed in the Subsection 1.1, it is distinguished from previous work in being applicable to the natural agnostic setting, where we do not assume that a perfect hypothesis exist. Since this assumption was crucial in all previous works, the construction we use is rather different.

To present the result, we will need the concept of a *one-way permutation*. Intuitively, a one-way permutation over $\{0,1\}^n$ is a permutation which is computationally hard to invert. More formally, let $\mathcal{U}_n$ denote the uniform distribution over $\{0,1\}^n$, and let $\{0,1\}^*$ denote the set of all finite bit strings. Then we have the following definition:

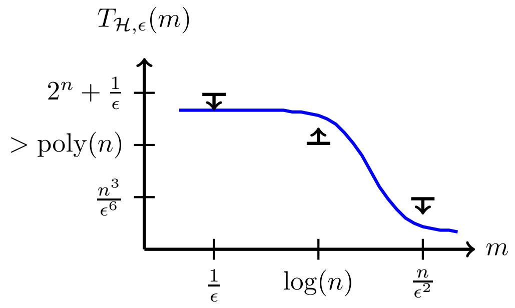
Definition 1. *A* one-way permutation $P : \{0,1\}^* \mapsto \{0,1\}^*$ *is a function which for any n, maps $\{0,1\}^n$ to itself; there exists an algorithm for computing $P(\mathbf{x})$, whose runtime is polynomial in the length of $\mathbf{x}$; and for any (possibly randomized) polynomial-time algorithm A and any polynomial $p(n)$ over n, $\Pr_{\mathbf{x}\sim\mathcal{U}_n}(A(P(\mathbf{x})) = \mathbf{x}) < \frac{1}{p(n)}$ for sufficiently large n.*

It is widely conjectured that such one-way permutations exist. One concrete candidate is the RSA permutation function, which treats $\mathbf{x} \in \{0,1\}^n$ as a number in $\{0,\ldots,2^n-1\}$, and returns $P(\mathbf{x}) = \mathbf{x}^3 \bmod N$, where N is a product of two "random" primes of length n such that $(p-1)(q-1)$ does not divide 3. However, since the existence of such a one-way permutation would imply $P \neq NP$, there is no formal proof that such functions exist (see [7] for this and related results).

Theorem 1. *There exists an agnostic binary classification learning problem over $\mathcal{X} = \{0,1\}^{2n}$ and $\mathcal{Y} = \{0,1\}$ with the following properties:*

- *It is inefficiently learnable with sample size $m = O(1/\epsilon)$, and running time $O(2^n + m)$.*
- *Assuming one-way permutations exist, there exist no polynomial-time algorithm based on a sample of size $O(\log(n))$.*
- *It is efficiently learnable with a sample of size $m = O(n/\epsilon^2)$. Specifically, the training time is $O(m)$, resulting in an improper predictor whose runtime is $O(m^3)$.*

The theorem implies that in the reasonable regime where $1/\epsilon \le \log(n) \le n/\epsilon^2$, we really get an inverse dependence of the runtime on the training size. The theorem is illustrated below:



To prove the theorem, we will define the following learning problem. Let $\mathcal{X} = \{0,1\}^{2n}$ and $\mathcal{Y} = \{0,1\}$. We will treat each $\mathbf{x} \in \mathcal{X}$ as a pair $(\mathbf{r}, \mathbf{s})$, where $\mathbf{r}$ refers to the first n bits in $\mathbf{x}$, and $\mathbf{s}$ to the last n bits. Let $\langle \mathbf{r}, \mathbf{r}' \rangle = \sum_{i=1}^{n} r_i r_i' \bmod 2$ to denote inner product over the field $GF(2)$. Let P be a one-way permutation. Then the example domain is the following subset of $\mathcal{X} \times \mathcal{Y}$:

$$\mathcal{Z} = \{((\mathbf{r},\mathbf{s}), b) \ : \ \mathbf{r}, \mathbf{s} \in \{0,1\}^n, \langle P^{-1}(\mathbf{s}), \mathbf{r} \rangle = b\}.$$

The loss function we use is simply the 0-1 loss, $\ell(h(\mathbf{x}), y) = \mathbf{1}_{h(\mathbf{x}) \neq y}$.

The hypothesis class $\mathcal{H}$ consists of randomized functions, parameterized by $\{0,1\}^n$, and defined as follows, where $\mathcal{U}_1$ is the uniform distribution on $\{0,1\}$:

$$\mathcal{H} = \left\{ h_{\mathbf{x}}(\mathbf{r},\mathbf{s}) = \begin{cases} \langle \mathbf{x}, \mathbf{r} \rangle & \mathbf{s} = P(\mathbf{x}) \\ b \sim \mathcal{U}_1 & \text{otherwise} \end{cases} \ : \ \mathbf{x} \in \{0,1\}^n \right\},$$

Learning with $O(\log(n))$ Samples is Hard

We consider the following "hard" set of distributions $\{\mathcal{D}_{\mathbf{x}}\}$, parameterized by $\mathbf{x} \in \{0,1\}^n$: each $\mathcal{D}_{\mathbf{x}}$ is a uniform distribution over all $((\mathbf{r}, P(\mathbf{x})), \langle \mathbf{x}, \mathbf{r} \rangle)$. Note that there are exactly 2^n such examples, one for each choice of $\mathbf{r} \in \{0,1\}^n$. Also, note that for any such distribution $\mathcal{D}_{\mathbf{x}}$, $\inf_{h \in \mathcal{H}} \mathrm{err}(h) = 0$, and this is achieved with the hypothesis $h_{\mathbf{x}}$.

First, we will prove that with a sample size $m = O(\log(n))$, any efficient learner fails on at least one of the distributions $\mathcal{D}_{\mathbf{x}}$. To see this, suppose on the contrary that we have an efficient distribution-free learner A, that works on all $\mathcal{D}_{\mathbf{x}}$, in the sense of seeing $m = O(\log(n))$ examples and then outputting some hypothesis h such that $h((\mathbf{r}, P(\mathbf{x}))) = \langle \mathbf{x}, \mathbf{r} \rangle$ with even some non-trivial probability (e.g. at least $1/2 + 1/\mathrm{poly}(n)$). We will soon show how we can use such a learner A, such that in probability at least $1/\mathrm{poly}(n)$, we get an efficient algorithm A', which given just $P(\mathbf{x})$ and r, outputs $\langle \mathbf{x}, \mathbf{r} \rangle$ with probability at least $1/2 + 1/\mathrm{poly}(n)$. However, by the Goldreich-Levin Theorem ([7], Theorem 2.5.2), such an algorithm can be used to efficiently invert P, violating the assumption that P is a one-way permutation.

Thus, we just need to show how given $P(\mathbf{x}), \mathbf{r}$, we can efficiently compute $\langle \mathbf{x}, \mathbf{r} \rangle$ with probability at least $1/\mathrm{poly}(n)$. The procedure works as follows: we pick $m = O(\log(n))$ vectors $\mathbf{r}_1, \ldots, \mathbf{r}_m$ uniformly at random from $\{0,1\}^n$, and pick uniformly at random bits $b_1, \ldots, b_m$. We then apply our learning algorithm A over the examples $\{((\mathbf{r}_i, P(\mathbf{x})), b_i)\}_{i=1}^m$, getting us some predictor h'. We then attempt to predict $\langle \mathbf{x}, \mathbf{r} \rangle$ by computing $h'((\mathbf{x}', P(\mathbf{x})))$.

To see why this procedure works, we note that with probability of $1/2^m = 1/\mathrm{poly}(n)$, we picked values for $b_1, \ldots, b_m$ such that $b_i = \langle \mathbf{x}, \mathbf{r}_i \rangle$ for all i. If this event happened, then the training set we get is distributed like m i.i.d. examples from $\mathcal{D}_{\mathbf{x}}$. By our assumption on A, and the fact that $\inf_h \mathrm{err}(h) = 0$, it follows that with probability at least $1/\mathrm{poly}(n)$, A will return a hypothesis which predicts correctly with probability at least $1/2 + 1/\mathrm{poly}(n)$, as required.

Inefficient Distribution-Free Learning Possible with $O(1/\epsilon)$ Samples

Ignoring computational constraints, we can use the following simple learning algorithm: given a training sample $\{(\mathbf{r}_i, \mathbf{s}_i), b_i\}_{i=1}^m$, find the most common value $\mathbf{s}'$ among $\mathbf{s}_1, \ldots, \mathbf{s}_m$, compute $\mathbf{x}' = P^{-1}(\mathbf{s}')$ (inefficiently, say by exhaustive search), and return the hypothesis $h_{\mathbf{x}'}$.

To see why this works, we will need the following lemma, which shows that if $h_{\mathbf{x}}$ has a low error rate, then $\mathbf{s} = P(\mathbf{x})$ is likely to appear frequently in the examples (the proof appears in Appendix A).

Lemma 1. *For any distribution $\mathcal{D}$ over examples, and any fixed $\mathbf{x} \in \{0,1\}^n$, it holds that $Pr_{\mathbf{s}}(\mathbf{s} = P(\mathbf{x})) = 1 - 2\text{err}(h_{\mathbf{x}})$.*

Suppose that $h_{\hat{\mathbf{x}}}$ is the hypothesis with a smallest generalization error in the hypothesis class. We now do a case analysis: if $\text{err}(h_{\hat{\mathbf{x}}}) > 1/2 - \epsilon$, then the predictor $h_{\mathbf{x}'}$ returned by the algorithm is almost as good. This because the probability in the lemma statement cannot be negative, so for *any* $\mathbf{x}'$ (and in particular the one used by the algorithm), we have $\text{err}(h_{\mathbf{x}'}) \leq 1/2$.

The other case we need to consider is that $\text{err}(h_{\hat{\mathbf{x}}}) \leq 1/2 - \epsilon$. By the lemma, $\hat{\mathbf{s}} = P(\hat{\mathbf{x}})$ is the value of $\mathbf{s}$ most likely to occur in the sample (since $h_{\hat{\mathbf{x}}}$ is the one with smallest generalization error), and its probability of being picked is at least $1 - 2 * (1/2 - \epsilon) = \epsilon$. This means that after $O(1/\epsilon)$ examples, then with overwhelming probability, the $\mathbf{s}'$ we pick is such that $\Pr_{\mathbf{s}}(\mathbf{s} = \hat{\mathbf{s}}) - \Pr_{\mathbf{s}}(\mathbf{s} = \mathbf{s}') \leq \epsilon/2$. But again by the lemma, this implies that $\text{err}(h_{\mathbf{x}'}) - \text{err}(h_{\hat{\mathbf{x}}})$ is at most $\epsilon/4$. So $h_{\mathbf{x}'}$ that our algorithm returns is an $\epsilon/4$-optimal classifier as required.

Efficient Distribution-Free Learning Possible with $O(n/\epsilon^2)$ Samples

We will need the following lemma, whose proof appears in Appendix A:

Lemma 2. *Let $\mathcal{D}'$ be some distribution over $\{0,1\}^n$, and suppose we sample m' vectors $\mathbf{r}_1, \ldots, \mathbf{r}_{m'}$ from that distribution. Then the probability that a freshly drawn vector $\mathbf{r}$ is not spanned by $\mathbf{r}_1, \ldots, \mathbf{r}_{m'}$ is at most n/m'.*

We use a similar algorithm to the one discussed earlier for inefficient learning. However, instead of finding the most common $\mathbf{s}'$, computing $\mathbf{x}' = P^{-1}(\mathbf{s}')$ and returning $h_{\mathbf{x}'}$, which cannot be done efficiently, we build a predictor which is at most ϵ worse than $h_{\mathbf{x}'}$, and doesn't require us to find $\mathbf{x}'$ explicitly.

To do so, let $\{((\mathbf{r}_{i_j}, \mathbf{s}_{i_j}), b_{i_j})\}_{j=1}^{m'}$ be the subset of examples for which $\mathbf{s}_{i_j} = \mathbf{s}'$. By definition of $\mathcal{Z}$, we know that for any such example, $\langle \mathbf{x}', \mathbf{r}_{i_j} \rangle = \langle P^{-1}(\mathbf{s}'), \mathbf{r}_{i_j} \rangle = b_{i_j}$. In other words, this gives us a set of values $\mathbf{r}_{i_1}, \ldots, \mathbf{r}_{i_{m'}}$, for which we know $\langle \mathbf{x}', \mathbf{r}_{i_1} \rangle, \ldots, \langle \mathbf{x}', \mathbf{r}_{i_{m'}} \rangle$. As a consequence, for any $\mathbf{r}$ in the linear subspace spanned by $\mathbf{r}_{i_1}, \ldots, \mathbf{r}_{i_{m'}}$, we can efficiently compute $\langle \mathbf{x}', \mathbf{r} \rangle$. Let B denote this subspace. Then our improper predictor works as follows, given some instance $(\mathbf{r}, \mathbf{s})$:

- If $\mathbf{s} = \mathbf{s}'$ and $\mathbf{r} \in B$, output $\langle \mathbf{x}', \mathbf{r} \rangle$ (note that this is the same output as $h_{\mathbf{x}'}$, by definition).
- If $\mathbf{s} \neq \mathbf{s}'$, output a random bit (note that this is the same output as $h_{\mathbf{x}'}$, by definition of $h_{\mathbf{x}'}$).
- If $\mathbf{s} = \mathbf{s}'$ and $\mathbf{r} \notin B$, output a bit uniformly at random.

Note that checking whether $\mathbf{r} \in B$ can always be done in at most $O(m'^3) \leq O(m^3)$ time, via Gaussian elimination.

Now, we claim that the probability of the third case happening is at most $\epsilon/2$. If this is indeed true, then our improper predictor is only $\epsilon/2$ worse (in terms of generalization error) from $h_{\mathbf{x}'}$, which based on the argument in the previous section, is already ϵ-close to optimal.

So let us consider the possibility that $\mathbf{s} = \mathbf{s}'$ and $r \notin B$. If $\Pr_{\mathbf{s}}(\mathbf{s} = \mathbf{s}') \leq \epsilon$, we are done, so let us suppose that $\Pr_{\mathbf{s}}(\mathbf{s} = \mathbf{s}') > \epsilon$. This means that m' is unlikely to be much smaller than ϵm. More precisely, by the multiplicative Chernoff bound, $\Pr(m' < \epsilon m/2) \leq \exp(-\epsilon m/8)$. Also, conditioned on some fixed $m' \geq \epsilon m/2$, Lemma 2 assures us that $\Pr(\mathbf{r} \notin B | \mathbf{s} = \mathbf{s}') \leq n/m' \leq 2n/\epsilon m$. Overall, we get the following (the probabilities are over the draw of the training set and an additional example $((\mathbf{r}, \mathbf{s}), b)$):

$$\begin{aligned}
&\Pr(\mathbf{s} = \mathbf{s}', \mathbf{r} \notin B) \;=\; \Pr(\mathbf{s} = \mathbf{s}', \mathbf{r} \notin B, m' < \epsilon m/2) + \Pr(\mathbf{s} = \mathbf{s}', \mathbf{r} \notin B, m' \geq \epsilon m/2) \\
&\leq \Pr(m' < \epsilon m/2) + \Pr(m' \geq \epsilon m/2, \mathbf{r} \notin B | \mathbf{s} = \mathbf{s}') \\
&\leq \exp(-\epsilon m/8) + \sum_{m'=\epsilon m/2}^{\infty} \Pr(m') \Pr(\mathbf{r} \notin B | \mathbf{s} = \mathbf{s}', m') \;\leq\; \exp(-\epsilon m/8) + \frac{2n}{\epsilon m}.
\end{aligned}$$

By taking $m = O(n/\epsilon^2)$ examples, we can ensure this to be at most order ϵ.

4 Gaps for natural learning problems

In this section we collect examples of natural learning problems in which we conjecture there is an inverse dependence of the training time on the sample size. Some of these examples already appeared explicitly in previous literature, but most are new, unpublished, or did not appear in such an explicit form. We base our inverse dependence conjecture on the current best known upper bounds. Of course, an immediate open question is to show matching lower bounds. However, our main goal here is to demonstrate general techniques of how to reduce the training runtime by requiring more examples.

4.1 Agnostically Learning Preferences

Consider the set $[d] = \{1, \ldots, d\}$, and let $\mathcal{X} = [d] \times [d]$ and $\mathcal{Y} = \{0, 1\}$. That is, each example is a pair (i, j) and the label indicates whether i is more preferable to j.

Consider the hypothesis class of all permutations over $[d]$ which can be written as $\mathcal{H} = \{h_{\mathbf{w}}(i,j) = \mathbf{1}[w_i > w_j] : \mathbf{w} \in \mathbb{R}^d\}$. The loss function is the 0-1 loss. Note that each hypothesis in $\mathcal{H}$ can be written as a Halfspace: $h_{\mathbf{w}}(i,j) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{e}^i - \mathbf{e}^j \rangle)$. Therefore, in the realizable case (namely, exists $h \in \mathcal{H}$ which perfectly predicts the labels of all the examples in the training set), solving the ERM problem can be performed in polynomial time. However, in the agnostic case, finding a Halfspace that minimizes the number of mistakes is in general NP hard. The sample complexity of agnostically learning a d-dimensional Halfspace is $\tilde{O}(d/\epsilon^2)$ and we therefore obtain that with a non-efficient algorithm, it is possible to learn using $\tilde{O}(d/\epsilon^2)$ examples.

On the other hand, in the following we show that with $m = \Theta(d^2/\epsilon^2)$ it is possible to learn preferences in time $O(m)$. The idea is to define the hypothesis class of all Boolean functions over $\mathcal{X}$, namely, $H_1 = \{H(i,j) = M_{i,j} : M \in \{0,1\}^{d^2}\}$. Clearly, $H \subset H_1$. In addition, $|H_1| = 2^{d^2}$ and therefore the sample complexity of learning H_1 using the ERM rule is $O(d^2/\epsilon^2)$. Last, it is easy to verify that solving the ERM problem can be easily done in time $O(m)$. So, overall, we obtain the following:

	Samples	Time
ERM over $\mathcal{H}$	d/ϵ^2	not $\mathrm{poly}(d)$
ERM over $\mathcal{H}_1$	d^2/ϵ^2	d^2/ϵ^2

4.2 Agnostic Learning of Kernel-based Halfspaces

We now consider the popular class of kernel-based linear predictors. In kernel predictors, the instances $\mathbf{x}$ are mapped to a high-dimensional feature space $\psi(\mathbf{x})$, and a linear predictor is learned in that space. Rather than working with $\psi(\mathbf{x})$ explicitly, one performs the learning implicitly using a kernel function $k(\mathbf{x}, \mathbf{x}')$ which efficiently computes inner products $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$.

Since the dimensionality of the feature space may be high or even infinite, the sample complexity of learning Halfspaces in the feature space can be too large. One way to circumvent this problem is to define a slightly different concept class by replacing the non-continuous sign function with a Lipschitz continuous function, $\phi : \mathbb{R} \to [0,1]$, which is often called a transfer function. For example, we can use a sigmoidal transfer function $\phi_{\mathrm{sig}}(a) = 1/(1 + \exp(-4L\,a))$, which is a L-Lipschitz function. The resulting hypothesis class is $\mathcal{H}_{\mathrm{sig}} = \{\mathbf{x} \mapsto \phi_{\mathrm{sig}}(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle) : \|\mathbf{w}\|_2 \le 1\}$, where we interpret the prediction $\phi_{\mathrm{sig}}(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle) \in [0,1]$ as the probability to predict a positive label. The expected 0-1 loss then amounts to $\ell(\mathbf{w}, (\mathbf{x}, y)) = |y - \phi_{\mathrm{sig}}(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle)|$.

Using standard Rademacher complexity analysis (e.g. [3]), it is easy to see that the information theoretic sample complexity of learning $\mathcal{H}$ is $O(L^2/\epsilon^2)$. However, from the computational complexity point of view, the ERM problem amounts to solving a non-convex optimization problem (with respect to $\mathbf{w}$). Adapting a technique due to [4] it is possible to show that an ϵ-accurate solution to the ERM problem cam be calculated in time $\exp\left(O\left(\frac{L^2}{\epsilon^2}\log(\frac{L}{\epsilon})\right)\right)$. The idea is to observe that the solution can be identified if someone reveals us a subset of $(L/\epsilon)^2$ non-noisy examples. Therefore we can perform an exhaustive search over all $(L/\epsilon)^2$ subsets of the m examples in the training set and identify the best solution.

In [12], a different algorithm has been proposed, that learns the class H_{sig} using time and sample complexity of at most $\exp\left(O\left(L\,\log(\frac{L}{\epsilon})\right)\right)$. That is, the runtime of this algorithm is exponentially smaller than the runtime required to solve the ERM problem using the technique described in [4], but the sample complexity is also exponentially larger. The main idea of the algorithm given in [12] is to define a new hypotheses class, $\mathcal{H}_1 = \{\mathbf{x} \mapsto \langle \mathbf{w}, \hat{\psi}(\psi(\mathbf{x}))\rangle : \|\mathbf{w}\|_2 \leq B\}$, where $B = O((L/\epsilon)^L)$ and $\hat{\psi}$ is a mapping function for which

$$\langle \hat{\psi}(\psi(\mathbf{x})), \hat{\psi}(\psi(\mathbf{x}'))\rangle \;=\; \frac{2}{2 - \langle \psi(\mathbf{x}), \psi(\mathbf{x}')\rangle} \;=\; \frac{2}{2 - k(\mathbf{x}, \mathbf{x}')} \;.$$

While it is not true that $\mathcal{H} \subset \mathcal{H}_1$, it is possible to show that $\mathcal{H}_1$ "almost" contains $\mathcal{H}$ in the sense that for each $h \in \mathcal{H}$ there exists $h_1 \in \mathcal{H}_1$ such that for all $\mathbf{x}$, $|h(\mathbf{x}) - h_1(\mathbf{x})| \leq \epsilon$. The advantage of $\mathcal{H}_1$ over $\mathcal{H}$ is that the functions in $\mathcal{H}_1$ are linear and hence the ERM problem with respect to $\mathcal{H}_1$ boils down to a convex optimization problem and thus can be solved in time $\mathrm{poly}(m)$, where m is the size of the training set. In summary, we obtain the following

	Samples	Time
ERM over $\mathcal{H}$	L^2/ϵ^2	$\mathrm{poly}\left(\exp\left(\frac{L^2}{\epsilon^2}\log(\frac{L}{\epsilon})\right)\right)$
ERM over $\mathcal{H}_1$	$\mathrm{poly}\left(\exp\left(L\,\log(\frac{L}{\epsilon})\right)\right)$	$\mathrm{poly}\left(\exp\left(L\,\log(\frac{L}{\epsilon})\right)\right)$

4.3 Additional Examples

In Appendix B we list additional examples of inverse dependence of runtime on sample size. These examples deal with other learning settings like online learning and unsupervised learning. These examples are interesting since they show other techniques to obtain faster algorithms using a larger sample. For example, we demonstrate how to use *exploration* for injecting structure into the problem, which leads better runtime. The price of the exploration is the need of a larger sample. For the unsupervised setting, we recall an existing example which shows polynomial gap for learning the support of a certain sparse vector.

5 Discussion

In this paper, we formalized and discussed the phenomena of an inverse dependence between the running time and the sample size. While this phenomena has also been discussed in some earlier works, it was under a restrictive realizability assumption, that a perfect hypothesis exists, and the techniques mostly involved finding this hypothesis. In contrast, we frame our discussion in the more modern approach of agnostic and improper learning.

In the first half of our paper, we provided a novel construction which shows such a tradeoff, based on a cryptographic assumption. While the construction indeed has an inverse dependence phenomenon, it is not based on a natural learning problem. In the second half of the paper, we provided more natural learning problems, which seem to have this phenomenon. Some of these problems were based on the intuition described in the introduction, but some were based on other techniques. However, the apparent inverse dependence in these problems is based on the assumption that the currently available upper bounds have matching lower bounds, which is not known to be true. Thus, we cannot formally prove that they indeed become computationally easier with the sample size.

Thus, a major open question is finding *natural* learning problems, whose required running time has *provable* inverse dependence with the sample size. We believe the examples we outlined hint at the existence of such problems, and provide clues as to the necessary techniques. Other problems are finding additional examples where this inverse dependence seems to hold, as well as finding additional techniques for making this inverse dependence happen. The ability to leverage large amounts of data to obtain more efficient algorithms would surely be a great asset to any machine learning application.

References

[1] A. Amini and M. Wainwright. High-dimensional analysis of semidefinite relaxations for sparse prinicpal components. *Annals of Statistics*, 37(5B):2877–2921, 2009.

[2] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *FOCS*, 2008.

[3] P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[4] S. Ben-David and H. Simon. Efficient learning of linear perceptrons. In *NIPS*, 2000.

[5] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2008.

[6] S. Decatur, O. Goldreich, and D. Ron. Computational sample complexity. *SIAM Journal on Computing*, 29, 1998.

[7] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[8] S. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In *International Conference on Machine Learning*, 2008.

[9] M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17:115–141, 1994.

[10] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, October 1988.

[11] R. Servedio. Computational sample complexity and attribute-efficient learning. *J. of Comput. Syst. Sci.*, 60(1):161–178, 2000.

[12] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Learning kernel-based halfspaces with the zero-one loss. In *COLT*, 2010.

[13] S. Shalev-Shwartz and N. Srebro. Svm optimization: Inverse dependence on training set size. In *ICML*, 2008.

A Technical Results

A.1 Proof of Lemma 1

Using the definition of $\mathcal{Z}$ and $h_{\mathbf{x}}$, we have

$$\begin{aligned}
1 - \mathrm{err}(h_{\mathbf{x}}) &= \Pr_{((\mathbf{r},\mathbf{s}),b)\sim\mathcal{D}}(b = h_{\mathbf{x}}(\mathbf{r},\mathbf{s})) \\
&= \Pr(\mathbf{s} = P(\mathbf{x}))\ \Pr(b = h_{\mathbf{x}}(\mathbf{r},\mathbf{s})|\mathbf{s} = P(\mathbf{x})) + \Pr(\mathbf{s} \neq P(\mathbf{x}))\ \Pr(b = h_{\mathbf{x}}(\mathbf{r},\mathbf{s})|\mathbf{s} \neq P(\mathbf{x})) \\
&= \Pr(\mathbf{s} = P(\mathbf{x})) * 1 + \Pr(\mathbf{s} \neq P(x)) * \frac{1}{2} = \frac{1}{2}(Pr(\mathbf{s} = P(\mathbf{x})) + 1).
\end{aligned}$$

Rearranging, we get the result.

A.2 Proof of Lemma 2

Let p_k denote the probability that after drawing $\mathbf{r}_1, \ldots, \mathbf{r}_k$, i.i.d., an independently drawn $\mathbf{r}_{k+1}$ is not spanned by $\mathbf{r}_1, \ldots, \mathbf{r}_k$. Also, let B_k be a Bernoulli random variable with parameter p_k. Whenever $B_k = 1$, the dimensionality of the subspace spanned by the vectors we drew so far increases by 1. Since we are in an n-dimensional space, we must have $B_1 + \ldots + B_{m'} \leq n$ with probability 1. In particular, we have

$$n \geq \mathbb{E}[B_1 + \ldots + B_{m'}] = p_1 + \ldots + p_{m'}.$$

Also, for any $k \leq m'$, by the assumption that the vectors are drawn i.i.d., we have

$$\begin{aligned}
p'_m &= \Pr(r_{m'+1} \notin \text{span}(r_1, \ldots, r_{m'})) \leq \Pr(r_{m'+1} \notin \text{span}(r_1, \ldots, r_k)) \\
&= \Pr(r_{k+1} \notin \text{span}(r_1, \ldots, r_k)) = p_k.
\end{aligned}$$

Combining the two inequalities, it follows that $m' p_{m'} \leq n$, so $p_{m'} \leq n/m'$ as required.

B Additional Examples

B.1 Online Multiclass Categorization with Bandit Feedback

This example is based on [8]. It deals with another variant of the multi-armed bandit problem. It shows how to use *exploration* for injecting structure into the problem, which leads to a decrease in the required runtime. The price of the exploration is a larger regret, which corresponds to the need of a larger number of online rounds for achieving the same target error.

The setting is as follows. At each online round, the learner first receives a vector $\mathbf{x}_t \in \mathbb{R}^d$ and need to predict one of k labels (corresponding to arms). Then, the environment picks the correct label y_t, without revealing it to the learner, and only tells the learner the binary feedback of if his prediction was correct or not.

We analyze the number of mistakes the learner will perform in T rounds, where we assume that there exists some matrix $W^\star \in \mathbb{R}^{k,d}$ such that at each round the correct label is $y_t = \arg\max_{y\in[k]}(W^\star\mathbf{x})_y$. We further assume that the score of the correct label is higher than the runner-up by at least γ and that the Frobenius norm of $W^\star$ is at most 1. We also assume that d is order of $1/\gamma^2$ (this is not restricting due to the possibility of performing random projections).

We now consider two algorithms. The first uses a multiclass version of the Halving algorithm (see [8]) which can be implemented with the bandit feedback and has a regret bound of $\tilde{O}(k^2 d)$. However, the runtime of this algorithm is 2^{kd}.

The second algorithm is the Banditron of [8]. The Banditron uses exploration for reducing the learning problem into the problem of learning multiclass classifier in the full information case, which can be performed efficiently using the Perceptron algorithm. In particular, in some of the rounds the Banditron guesses a random label, attempting to "fish" the relevant information. This exploration yields a higher regret bound of $O(\sqrt{kdT})$.

We can therefore draw the following table, which shows a tradeoff between running time and number of rounds required to obtain regret $\leq \epsilon$. Note that we will usually want ϵ to be much smaller than $1/k$.

	Rounds	Time
Inefficient alg.	k^2d/ϵ	$T2^{kd}$
Efficient alg.	kd/ϵ^2	Tkd

B.2 Sparse Principal Component Recovery

This example is taken from [1]. This time, it is in the context of *unsupervised* statistical learning.

The problem is as follows: we have an i.i.d. sample of vectors drawn from $\mathbb{R}^d$. The distribution is assumed to be Gaussian $\mathcal{N}(\mathbf{0}, \Sigma)$, with a "spiked" covariance structure. specifically, the covariance matrix Σ is assumed to be of the form $I_d + \mathbf{z}\mathbf{z}^\top$, where $\mathbf{z}$ is an unknown *sparse* vector, with only k non-zero elements of the form $\pm 1/\sqrt{k}$. Our goal in this setting is to detect the support of $\mathbf{z}$.

[1] provide two algorithms to deal with this problem. The first method is a simple diagonal thresholding scheme, which takes the empirical covariance matrix $\hat{\Sigma}$, and returns the k indices for which the diagonal entries of $\hat{\Sigma}$ are largest. It is proven that if $m \geq ck^2 \log(d-k)$ (for some constant c), then the probability of not perfectly identifying the support of $\mathbf{z}$ is at most $\exp(-O(k^2 \log(d-k)))$, which goes to 0 with k and d. Thus, we can view the sample complexity of this algorithm as $O(k^2 \log(d-k))$. In terms of running time, given a sample of size $m = O(k^2 \log(d-k))$, the method requires computing the diagonal of $\hat{\Sigma}$ and sorting it, for a total runtime of $O(k^2 d \log(d-k) + d\log(d)) = O(k^2 d \log(d))$.

The second algorithm is a more sophisticated semidefinite programming (SDP) scheme, which can be solved exactly in time $O(d^4 \log(d))$. Moreover, the sample complexity for perfect recovery is shown to be asymptotically $O(k \log(d-k))$. Summarizing, we have the following clear sample-time complexity tradeoff. Note that here, the gaps are only polynomial.

	Samples	Time
SDP	$k \log(d-k)$	$d^4 \log(d)$
Thresholding	$k^2 \log(d-k)$	$k^2 d \log(d)$