Learning Linear and Kernel Predictors with the 0–1 Loss Function

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

Ohad Shamir
Microsoft Research
and The Hebrew University
ohadsh@cs.huji.ac.il

Karthik Sridharan
Toyota Technological Institute
karthik@tti-c.org

Abstract

Some of the most successful machine learning algorithms, such as Support Vector Machines, are based on learning linear and kernel predictors with respect to a convex loss function, such as the hinge loss. For classification purposes, a more natural loss function is the 0-1 loss. However, using it leads to a non-convex problem for which there is no known efficient algorithm. In this paper, we describe and analyze a new algorithm for learning linear or kernel predictors with respect to the 0-1 loss function. The algorithm is parameterized by L, which quantifies the effective width around the decision boundary in which the predictor may be uncertain. We show that without any distributional assumptions, and for any fixed L, the algorithm runs in polynomial time, and learns a classifier which is worse than the optimal such classifier by at most ϵ. We also prove a hardness result, showing that under a certain cryptographic assumption, no algorithm can learn such classifiers in time polynomial in L.

1 Introduction

One of the main workhorses of machine learning are linear predictors, used in algorithms such as Support Vector Machines, Perceptron, Adaboost, Linear Regression and more. A linear predictor is parameterized by a vector $\mathbf{w}$, and given an instance $\mathbf{x}$, predicts according to $\langle \mathbf{w}, \mathbf{x} \rangle$. A powerful extension of linear predictors are kernel predictors, where 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 $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ in the feature space.

For binary classification tasks, the common approach is to take the sign of $\langle \mathbf{w}, \mathbf{x} \rangle$ as the prediction. The natural way to quantify the performance of such a classifier is using the 0-1 loss function: for a given instance $\mathbf{x}$ and a true binary label $y \in \{0, 1\}$, we incur a loss of 1 if $\mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle) \neq y$, and 0 otherwise.

However, there is no efficient algorithm known for finding the linear predictor $\mathbf{w}$ which minimizes the number of classification errors on an arbitrary dataset. With general kernel predictors, this is aggravated by a statistical problem, namely that it is impossible to obtain generalization guarantees which relate good performance on the training data to a good performance on the distribution from which the training data was sampled.

In practice, a common solution to both problems has been to replace the non-convex 0-1 loss function by a convex surrogate, such as the hinge loss (defined as $\max\{0, 1 - y\langle \mathbf{w}, \mathbf{x} \rangle\}$). With respect to such loss functions, one can efficiently find the best predictor using convex optimization, as well as obtain statistical guarantees, even for general kernel predictors. However, this comes at the price of solving a somewhat different problem than what we are really after, namely minimizing classification mistakes. Worse, there is no simple way to relate these two problems for finite samples (although there do exist some recent results on the *asymptotic* relationship between the two [Zhang, 2004; Bartlett *et al.*, 2006]).

In this paper, we present and analyze a simple algorithm for learning linear or kernel predictors for binary classification, with respect to the 0-1 loss function. To obtain non-trivial guarantees, the learned predictors allow themselves a small region of uncertainty (or randomness) close to the decision boundary, whose effective width is parameterized by L. We show that for any fixed L, the algorithm learns a classifier which is worse than the optimal such classifier by at most ϵ, and runs in time polynomial in ϵ. Moreover, the algorithm has the interesting property of being provably competitive not only with respect to such classifiers, but with respect to a much larger set of predictors. In addition, while our guarantees are worst-case, the runtime of the algorithm can be much smaller, depending on the data distribution. We also prove a hardness result, showing that under a certain cryptographic assumption, no algorithm can learn such classifiers in time polynomial in L.

2 Preliminaries

Following the standard statistical learning framework, we assume that there is an unknown distribution $\mathcal{D}$ over the set of labeled examples, $\mathcal{X} \times \{0, 1\}$, and our primary goal is to find a classifier, $h : \mathcal{X} \to \{0, 1\}$, with low error in expectation

over $\mathcal{D}$:

$$\mathrm{err}_{\mathcal{D}}(h) \stackrel{\mathrm{def}}{=} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[|h(\mathbf{x}) - y|] \,. \quad (1)$$

The learning algorithm is allowed to sample a training set of labeled examples, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, where each example is sampled i.i.d. from $\mathcal{D}$, and it returns a classifier. Following the agnostic PAC learning framework [Kearns *et al.*, 1992], we say that an algorithm (ϵ, δ)-learns a concept class H of classifiers using m examples, if with probability at least $1 - \delta$ over a random choice of m examples, the algorithm returns a classifier $\hat{h}$ that satisfies

$$\mathrm{err}_{\mathcal{D}}(\hat{h}) \;\leq\; \inf_{h\in H} \mathrm{err}_{\mathcal{D}}(h) + \epsilon \,. \quad (2)$$

We note that $\hat{h}$ does not necessarily belong to H. Namely, we are concerned with *improper* learning, which is as useful as proper learning for the purpose of deriving good classifiers. A common learning paradigm is the Empirical Risk Minimization (ERM) rule, which returns a classifier that minimizes the average error over the training set,

$$\hat{h} = \mathop{\mathrm{argmin}}_{h\in H} \frac{1}{m} \sum_{i=1}^{m} |h(\mathbf{x}_i) - y_i| \,. \quad (3)$$

The class of classifiers discussed in the introduction, also known as *halfspaces*, is defined as follows. Let $\mathcal{X}$ be the set of all vectors in the unit ball. Let $\phi_{0-1} : \mathbb{R} \to \mathbb{R}$ be the function $\phi_{0-1}(a) = \mathbf{1}(a \geq 0) = \frac{1}{2}(\mathrm{sgn}(a) + 1)$. The class of halfspaces is the set of classifiers

$$H_{\phi_{0-1}} \stackrel{\mathrm{def}}{=} \{\mathbf{x} \mapsto \phi_{0-1}(\langle \mathbf{w}, \mathbf{x}\rangle) \,:\, \mathbf{w} \in \mathcal{X}\} \,.$$

If $\mathcal{X}$ lives in an n-dimensional Euclidean space, then standard learning-theoretic results allow us to obtain a guarantee of the form (2) using the ERM learning rule (Equation (3)). The size of the required training data scales linearly with n. However, with kernel predictors, which maps data to a high or even infinite-dimensional space, we must use a different class in order to obtain a guarantee of the form given in Equation (2).

One way to define a slightly different concept class is to approximate the non-continuous function, ϕ_{0-1}, 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

$$\phi_{\mathrm{sig}}(a) \stackrel{\mathrm{def}}{=} \frac{1}{1 + \exp(-4L\, a)} \,, \quad (4)$$

which is a L-Lipschitz function. An illustration of this transfer function is given in Figure 1. Analogously to the definition of $H_{\phi_{0-1}}$, for a general transfer function ϕ we define H_ϕ to be the set of predictors $\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x}\rangle)$. Since now the range of ϕ is not $\{0, 1\}$ but rather the entire interval $[0, 1]$, we interpret $\phi(\langle \mathbf{w}, \mathbf{x}\rangle)$ as the probability to output the label 1. The definition of $\mathrm{err}_{\mathcal{D}}(h)$ remains[1] as in Equation (1).

The advantage of using a L-Lipschitz transfer function is that one can obtain learning guarantees which depend only on (ϵ, δ) and L, rather than the dimensionality of the data. For example, based on techniques form [Bartlett and Mendelson, 2002], one can show the following:

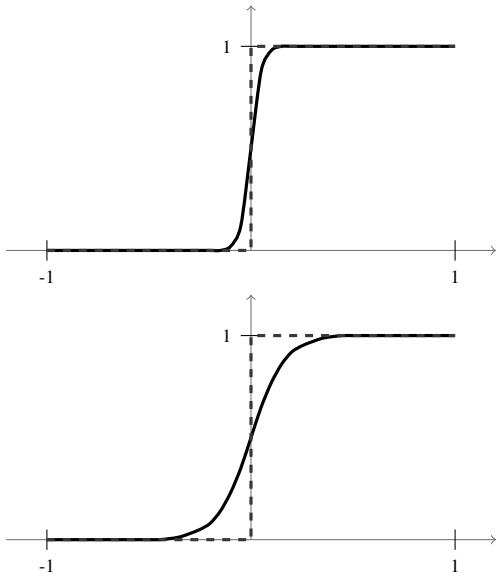

Figure 1: Illustrations of transfer functions for $L = 10$ (top) and $L = 3$ (bottom): the 0-1 transfer function (dashed blue line) and the sigmoid transfer function (black line).

Theorem 1 *Let $\epsilon, \delta \in (0, 1)$ and let ϕ be an L-Lipschitz transfer function. Let m be an integer satisfying*

$$m \;\geq\; \left(\frac{2L + 3\sqrt{2\ln(8/\delta)}}{\epsilon}\right)^2 \,.$$

Then, for any distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, the ERM algorithm (ϵ, δ)-learns the concept class H_ϕ using m examples.

The above theorem tells us that the sample complexity of learning H_ϕ is $\tilde{\Omega}(L^2/\epsilon^2)$, regardless of the dimensionality of $\mathcal{X}$. This allows us to learn with kernels, when the dimensionality of $\mathcal{X}$ can even be infinite.

From the computational complexity point of view, the result given in Theorem 1 is problematic, since the ERM algorithm should solve the non-convex optimization problem

$$\mathop{\mathrm{argmin}}_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \frac{1}{m} \sum_{i=1}^{m} |\phi(\langle \mathbf{w}, \mathbf{x}_i\rangle) - y_i| \,. \quad (5)$$

The main focus of this paper is the derivation and analysis of a simple learning algorithm that (ϵ, δ)-learns the class H_{sig} using time and sample complexity which is polynomial for any fixed L.

3 Main Results

In this section we present our main results. Recall that we would like to derive an algorithm which learns the class H_{sig}. However, the ERM optimization problem associated with H_{sig} is non-convex. The main idea behind our construction is to learn a larger hypothesis class, denoted H_B, which

[1] Note that in this case $\mathrm{err}_{\mathcal{D}}(h)$ can be interpreted as $\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}, b\sim\phi(\langle \mathbf{w},\mathbf{x}\rangle)}[y \neq b]$.

approximately contains H_{sig}, and for which the ERM optimization problem becomes convex. The price we need to pay is that from the statistical point of view, it is more difficult to learn the class H_B than the class H_{sig}, therefore the sample complexity increases.

The class H_B we use is a class of kernel predictors. The kernel function we use is defined as

$$K(\mathbf{x},\mathbf{x}') \stackrel{\text{def}}{=} \frac{1}{1-\nu\langle\mathbf{x},\mathbf{x}'\rangle}\,, \tag{6}$$

where $\nu \in (0,1)$ is a parameter. If our original learning problem pertained to linear predictors, then $\langle\mathbf{x},\mathbf{x}'\rangle$ is the standard inner product between vectors in Euclidean space. If our original learning problem pertained to kernel predictors, then $\langle\mathbf{x},\mathbf{x}'\rangle$ is the kernel inner product. For example, if we wish to learn kernel predictors using the Gaussian kernel, defined as $k(\mathbf{x},\mathbf{x}') = \exp(-\left\|\mathbf{x}-\mathbf{x}'\right\|^2/\sigma^2)$, then the kernel function we will use to learn with respect to the 0-1 loss is $1/(1-\nu\exp(-\left\|\mathbf{x}-\mathbf{x}'\right\|^2/\sigma^2))$.

To simplify the presentation we will set $\nu = 1/2$, although in practice other choices might be more effective. It is easy to verify that K is a valid kernel function (see for example [Cristianini and Shawe-Taylor, 2004]). Therefore, there exists some feature mapping $\psi : \mathcal{X} \to \mathbb{V}$, where $\mathbb{V}$ is an inner product space with $\langle\psi(\mathbf{x}),\psi(\mathbf{x}')\rangle = K(\mathbf{x},\mathbf{x}')$. The class H_B is defined to be:

$$H_B \stackrel{\text{def}}{=} \left\{\mathbf{x}\mapsto\langle\mathbf{v},\psi(\mathbf{x})\rangle \,:\, \mathbf{v}\in\mathbb{V},\ \|\mathbf{v}\|^2 \le B\right\}. \tag{7}$$

The algorithm we present and analyze is straightforward: first, it uses the ERM learning rule and finds the kernel predictor $\mathbf{v}\in\mathbb{V}$ which minimizes the average absolute value of its mis-predictions on the training data:

$$\mathbf{v} = \operatorname*{argmin}_{\mathbf{v}:\|\mathbf{v}\|^2\le B} \frac{1}{m}\sum_{i=1}^{m} |\langle\mathbf{v},\psi(\mathbf{x}_i)\rangle - y_i|, \tag{8}$$

Since the objective function is defined only via inner products with $\psi(\mathbf{x}_i)$, and the constraint on $\mathbf{v}$ is defined by the ℓ_2-norm, it follows by standard results that there is an optimal solution $\mathbf{v}^\star$ that can be written as $\mathbf{v}^\star = \sum_{i=1}^m \alpha_i\psi(\mathbf{x}_i)$. Therefore, instead of optimizing over $\mathbf{v}$, we can optimize over the set of weights $\alpha_1,\ldots,\alpha_m$ by solving the equivalent optimization problem

$$\min_{\alpha_1,\ldots,\alpha_m} \frac{1}{m}\sum_{i=1}^{m}\left|\sum_{j=1}^{m}\alpha_j K(\mathbf{x}_j,\mathbf{x}_i) - y_i\right| \tag{9}$$
$$\text{s.t.}\quad \sum_{i,j=1}^{m}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) \le B\,,$$

and defining the predictor $\mathbf{x}\mapsto\langle\mathbf{v},\mathbf{x}\rangle$ to be $\mathbf{x}\mapsto\sum_{i=1}^m K(\mathbf{x}_i,\mathbf{x})$. We note that this optimization problem is convex and can be solved efficiently (i.e. in time $\text{poly}(m)$) by any of several possible methods.

After finding this $\mathbf{v}$, the algorithm constructs and returns the following randomized binary classifier: Given an instance $\mathbf{x}$, the classifier predicts 1 with probability $[\langle\mathbf{v},\psi(\mathbf{x})\rangle]_{[0,1]}$ (where $[\cdot]_{[0,1]}$ clips values to the range $[0,1]$), and 0 otherwise.

The main result we prove in this section is the following:

Theorem 2 *Let $\epsilon,\delta\in(0,1)$ and let $L\ge 3$. Let $B = 2L^4 + \exp\left(7L\log\left(\frac{2L}{\epsilon}\right)+3\right)$ and let m be a sample size that satisfies $m \ge \frac{8B}{\epsilon^2}\left(2+9\sqrt{\ln(8/\delta)}\right)^2$. Then with probability at least $1-\delta$, the predictor $\hat{h}$ returned by the algorithm satisfies*

$$\text{err}_{\mathcal{D}}(\hat{h}) \le \min_{h\in H_{\text{sig}}}\text{err}_{\mathcal{D}}(h_{\text{sig}}) + \epsilon\,.$$

As discussed earlier, our algorithm runs in time polynomial in the sample size, and thus runs in time $\text{poly}(1/\epsilon)$ for any fixed L. We note that the bound on B is far from being the tightest possible in terms of constants and second-order terms. Also, the assumption of $L\ge 3$ is rather arbitrary, and is meant to simplify the presentation of the bound.

The algorithm and accompanying analysis has two important properties: first, the classifier returned by the algorithm is near-optimal not only with respect to the class of linear (or kernel) predictors $H_{\phi_{0-1}}$, but also with respect to the class H_B, which contains a much larger set of transfer functions (see Lemma 1 below). In particular, it is near-optimal with respect to the "best" transfer function in H_B, where by best we mean the one which attains the smallest error rate over the distribution $\mathcal{D}$. A second important property is that the runtime of our algorithm depends on the parameter B, for which the bound in Theorem 2 is only a worst-case bound over any possible distribution. Since in practice B is chosen via cross-validation, it is plausible that in many real-world scenarios the runtime of our algorithm will be much smaller.

To prove Theorem 2, we start with analyzing the time and sample complexity of learning H_B. A standard analysis (using tools from [Bartlett and Mendelson, 2002]) tells us that the sample complexity of learning H_B with the ERM rule is order of B/ϵ^2 examples:

Theorem 3 *Let $\epsilon,\delta\in(0,1)$, let $B\ge 1$, and let m be a sample size that satisfies*

$$m \ge \frac{2B}{\epsilon^2}\left(2+9\sqrt{\ln(8/\delta)}\right)^2.$$

Then, for any distribution $\mathcal{D}$, the ERM algorithm (ϵ,δ)-learns H_B.

Next, as discussed earlier, we note that the ERM problem with respect to H_B (namely, optimizing Equation (8) or equivalently Equation (9)) can be solved in time $\text{poly}(m)$.

It is left to understand why the class H_B approximately contains the class H_{sig}. Recall that for any transfer function, ϕ, we define the class H_ϕ to be all the predictors of the form $\mathbf{x}\mapsto\phi(\langle\mathbf{w},\mathbf{x}\rangle)$. The first step is to show that H_B contains the union of H_ϕ over all polynomial transfer functions that satisfy a certain boundedness condition on their coefficients.

Lemma 1 *Let P_B be the following set of polynomials (possibly with infinite degree)*

$$P_B \stackrel{\text{def}}{=} \left\{p(a) = \sum_{j=0}^{\infty}\beta_j a^j \,:\, \sum_{j=0}^{\infty}\beta_j^2 2^j \le B\right\}. \tag{10}$$

Then,

$$\bigcup_{p\in P_B} H_p \;\subset\; H_B\,.$$

Proof To simplify the proof, we will assume that $\mathcal{X}$ is simply the unit ball in $\mathbb{R}^n$ (the case of kernel predictors can be proven similarly), for an arbitrarily large but finite n. Consider the mapping $\psi : \mathcal{X} \to \mathbb{R}^{\mathbb{N}}$ defined as follows: for any $\mathbf{x} \in \mathcal{X}$, we let $\psi(\mathbf{x})$ be an infinite vector, indexed by $k_1 \ldots, k_j$ for all $(k_1, \ldots, k_j) \in \{1, \ldots, n\}^j$ and $j = 0 \ldots \infty$, where the entry at index $k_1 \ldots, k_j$ equals $2^{-j/2} x_{k_1} \cdot x_{k_2} \cdots x_{k_j}$. The inner-product between $\psi(\mathbf{x})$ and $\psi(\mathbf{x}')$ for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ can be calculated as follows,

$$\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = \sum_{j=0}^{\infty} \sum_{(k_1,\ldots,k_j) \in \{1,\ldots,n\}^j} 2^{-j} x_{k_1} x'_{k_1} \cdots x_{k_j} x'_{k_j}$$

$$= \sum_{j=0}^{\infty} 2^{-j} (\langle \mathbf{x}, \mathbf{x}' \rangle)^j \;=\; \frac{1}{1 - \frac{1}{2}\langle \mathbf{x}, \mathbf{x}' \rangle}.$$

This is exactly the kernel function defined in Equation (6) (recall that we set $\nu = 1/2$) and therefore ψ maps to the feature space defined by K. Consider any polynomial $p(a) = \sum_{j=0}^{\infty} \beta_j a^j$ in P_B, and any $\mathbf{w} \in \mathcal{X}$. Let $\mathbf{v}_{\mathbf{w}}$ be an element in $\mathbb{R}^{\mathbb{N}}$ explicitly defined as being equal to $\beta_j 2^{j/2} w_{k_1} \cdots w_{k_j}$ at index $k_1, \ldots, k_j$ (for all $k_1, \ldots, k_j \in \{1, \ldots, n\}^j, j = 0 \ldots \infty$). By definition of ψ and $\mathbf{v}_{\mathbf{w}}$, we have that

$$\langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle = \sum_{j=0}^{\infty} \sum_{k_1,\ldots,k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdot \cdots x_{k_j}$$

$$= \sum_{j=0}^{\infty} \beta_j (\langle \mathbf{w}, \mathbf{x} \rangle)^j \;=\; p(\langle \mathbf{w}, \mathbf{x} \rangle).$$

Similarly, it can be verified that

$$\|\mathbf{v}_{\mathbf{w}}\|^2 = \sum_{j=0}^{\infty} \beta_j^2 2^j \sum_{k_1} w_{k_1}^2 \sum_{k_2} w_{k_2}^2 \cdots \sum_{k_j} w_{k_j}^2$$

$$= \sum_{j=0}^{\infty} \beta_j^2 2^j \left(\|\mathbf{w}\|^2\right)^j \;\leq\; B.$$

Thus, the predictor $\mathbf{x} \mapsto \langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle$ belongs to H_B and is the same as the predictor $\mathbf{x} \mapsto p(\langle \mathbf{w}, \mathbf{x} \rangle)$. This proves that $H_p \subset H_B$ for all $p \in P_B$ as required. ■

Finally, the following lemma states that with a sufficiently large B, there exists a polynomial in P_B which approximately equals ϕ_{sig}. This implies that H_B approximately contains H_{sig}.

Lemma 2 *Let ϕ_{sig} be as defined in Equation (4), where for simplicity we assume $L \geq 3$. For any $\epsilon > 0$, let*

$$B = 2L^4 + \exp\left(7L \log\left(\tfrac{2L}{\epsilon}\right) + 3\right).$$

Then there exists $p \in P_B$ such that

$$\forall \mathbf{x}, \mathbf{w} \in \mathcal{X}, \;\; |p(\langle \mathbf{w}, \mathbf{x} \rangle) - \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)| \leq \epsilon \,.$$

The proof of the lemma is based on a Chebyshev approximation technique and is given in the full version of our paper [Shalev-Shwartz *et al.*, 2010].

Finally, Theorem 2 is obtained as follows: Combining Theorem 3 and Lemma 1 we get that with probability at least $1 - \delta$,

$$\text{err}_{\mathcal{D}}(\hat{h}) \;\leq\; \min_{h \in H_B} \text{err}_{\mathcal{D}}(h) + \epsilon/2 \leq \min_{p \in P_B} \min_{h \in H_p} \text{err}_{\mathcal{D}}(h) + \epsilon/2 \,. \tag{11}$$

From Lemma 2 we obtain that for any $\mathbf{w} \in \mathcal{X}$, if $h(\mathbf{x}) = \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ then there exists a polynomial $p_0 \in P_B$ such that if $h'(\mathbf{x}) = p_0(\langle \mathbf{w}, \mathbf{x} \rangle)$ then $\text{err}_{\mathcal{D}}(h') \leq \text{err}_{\mathcal{D}}(h) + \epsilon/2$. Since it holds for all $\mathbf{w}$, we get that

$$\min_{p \in P_B} \min_{h \in H_p} \text{err}_{\mathcal{D}}(h) \leq \min_{h \in H_{\text{sig}}} \text{err}_{\mathcal{D}}(h) + \epsilon/2 \,.$$

Combining this with Equation (11), and using the fact that clipping predictions to $[0, 1]$ can only decrease the error, Theorem 2 follows.

4 Hardness

In this section, we derive a hardness result for distribution-free learning of H_{sig} with respect to the 0-1 loss. The hardness result relies on the hardness of learning intersections of halfspaces in the standard PAC model[2], proven by [Klivans and Sherstov, 2006]. The hardness result is representation-independent —it makes no restrictions on the learning algorithm and in particular also holds for improper learning algorithms. The hardness result is based on the following cryptographic assumption:

Assumption 1 *There is no polynomial time solution to the $\tilde{O}(n^{1.5})$-unique-Shortest-Vector-Problem.*

In a nutshell, given a basis $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$, the $\tilde{O}(n^{1.5})$-unique-Shortest-Vector-Problem consists of finding the shortest nonzero vector in $\{a_1 \mathbf{v}_1 + \ldots + a_n \mathbf{v}_n : a_1, \ldots, a_n \in \mathcal{Z}\}$, even given the information that it is shorter by a factor of at least $\tilde{O}(n^{1.5})$ than any other non-parallel vector. This problem is believed to be hard - there are no known sub-exponential algorithms, and it is known to be NP-hard if $\tilde{O}(n^{1.5})$ is replaced by a small constant (see [Klivans and Sherstov, 2006] for more details).

Our hardness result is the following:

Theorem 4 *Let L be a Lipschitz constant and let H_{sig} be the class defined by the L-Lipschitz transfer function ϕ_{sig}. Then, based on Assumption 1, there is no algorithm that runs in time $\text{poly}(L, 1/\epsilon, 1/\delta)$ and (ϵ, δ)-learns the class H_{sig}.*

Proof

With assumption 1, [Klivans and Sherstov, 2006] proved the following:

Theorem 5 (Theorem 1.2 in [Klivans and Sherstov, 2006]) *Let $\mathcal{X} = \{\pm 1\}^n$, let $H = \{\mathbf{x} \mapsto \phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2) : \theta \in \mathbb{N}, \mathbf{w} \in \mathbb{N}^n, |\theta| + \|\mathbf{w}\|_1 \leq poly(n)\}$, and let $H_k = \{\mathbf{x} \mapsto (h_1(\mathbf{x}) \wedge \ldots \wedge h_k(\mathbf{x})) : \forall i, h_i \in H\}$. Then, based on Assumption 1, H_k is not efficiently learnable in the standard PAC model for any $k = n^{\rho}$ where $\rho > 0$ is a constant.*

[2] In the *standard* PAC model, we assume that some hypothesis in the class has $\text{err}_{\mathcal{D}}(h) = 0$, while in our setting, $\text{err}_{\mathcal{D}}(h)$ might be strictly greater than zero for all $h \in H$.

The above theorem implies the following.

Lemma 3 *Based on Assumption 1, there is no algorithm that runs in time* $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$ *and* (ϵ, δ)*-learns the class* H *defined in Theorem 5.*

Proof To prove the lemma we show that if there is a polynomial time algorithm that learns H in the *agnostic* model, then there exists a weak learning algorithm (with a polynomial edge) that learns H_k in the standard (non-agnostic) PAC model. In the standard PAC model, weak learning implies strong learning [Schapire, 1990], hence the existence of a weak learning algorithm that learns H_k will contradict Theorem 5.

Indeed, let $\mathcal{D}$ be any distribution such that there exists $h^\star \in H_k$ with $\mathrm{err}_\mathcal{D}(h^\star) = 0$. Let us rewrite $h^\star = h_1^\star \wedge \ldots \wedge h_k^\star$ where for all i, $h_i^\star \in H$. To show that there exists a weak learner, we first show that there exists some $h \in H$ with $\mathrm{err}_\mathcal{D}(h) \leq 1/2 - 1/2k^2$.

Since for each $\mathbf{x}$ if $h^\star(\mathbf{x}) = 0$ then there exists j s.t. $h_j^\star(\mathbf{x}) = 0$, we can use the union bound to get that

$$\begin{aligned} 1 &= \mathbb{P}[\exists j : h_j^\star(\mathbf{x}) = 0 | h^\star(\mathbf{x}) = 0] \\ &\leq k \max_j \mathbb{P}[h_j^\star(\mathbf{x}) = 0 | h^\star(\mathbf{x}) = 0] \,. \end{aligned}$$

So, for j that maximizes $\mathbb{P}[h_j^\star(\mathbf{x}) = 0 | h^\star(\mathbf{x}) = 0]$ we get that $\mathbb{P}[h_j^\star(\mathbf{x}) = 0 | h^\star(\mathbf{x}) = 0] \geq 1/k$. Therefore,

$$\begin{aligned} \mathrm{err}_\mathcal{D}(h_j^\star) &= \mathbb{P}[h^\star(\mathbf{x}) = 0]\ \mathbb{P}[h_j^\star(\mathbf{x}) = 1 | h^\star(\mathbf{x}) = 0] \\ &= \mathbb{P}[h^\star(\mathbf{x}) = 0]\,(1 - \mathbb{P}[h_j^\star(\mathbf{x}) = 0 | h^\star(\mathbf{x}) = 0]) \\ &\leq \mathbb{P}[h^\star(\mathbf{x}) = 0]\,(1 - 1/k) \,. \end{aligned}$$

Now, if $\mathbb{P}[h^\star(\mathbf{x}) = 0] \leq 1/2 + 1/k^2$ then the above gives

$$\mathrm{err}_\mathcal{D}(h_j^\star) \leq (1/2 + 1/k^2)(1 - 1/k) \leq 1/2 - 1/2k^2 \,,$$

where the inequality holds for any positive integer k. Otherwise, if $\mathbb{P}[h^\star(\mathbf{x}) = 0] > 1/2 + 1/k^2$, then the constant predictor $h(\mathbf{x}) = 0$ has $\mathrm{err}_\mathcal{D}(h) < 1/2 - 1/k^2$. In both cases we have shown that there exists a predictor in H with error of at most $1/2 - 1/2k^2$.

Finally, if we can agnostically learn H in time $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$, then we can find h' with $\mathrm{err}_\mathcal{D}(h') \leq \min_{h \in H} \mathrm{err}_\mathcal{D}(h) + \epsilon \leq 1/2 - 1/2k^2 + \epsilon$ in time $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$ (recall that $k = n^\rho$ for some $\rho > 0$). This means that we can have a weak learner that runs in polynomial time, and this concludes our proof. ■

We now turn to prove Theorem 4 itself. Let h be a hypothesis in the class H defined in Theorem 5 and take any $\mathbf{x} \in \{\pm 1\}^n$. Then, there exist an integer θ and a vector of integers $\mathbf{w}$ such that $h(\mathbf{x}) = \phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$. However, since $\langle \mathbf{w}, \mathbf{x} \rangle - \theta$ is also an integer, we see that

$$|\phi_{0,1}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2) - \phi_{\mathrm{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)| \leq \frac{1}{1 + \exp(2L)} \,.$$

This means that for any $\epsilon > 0$, if we pick $L = \frac{\log(2/\epsilon - 1)}{2}$ and define $h_{\mathrm{sig}}(\mathbf{x}) = \phi_{\mathrm{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta - 1/2)$, then $|h(\mathbf{x}) - h_{\mathrm{sig}}(\mathbf{x})| \leq \epsilon/2$. Furthermore, letting $\mathbf{x}' \in \mathbb{R}^{n+1}$ denote the concatenation of $\mathbf{x}$ with the constant 1 and letting $\mathbf{w}' \in \mathbb{R}^{n+1}$ denote the concatenation of $\mathbf{w}$ with the scalar $(-\theta - 1/2)$ we obtain that $h_{\mathrm{sig}}(\mathbf{x}) = \phi_{\mathrm{sig}}(\langle \mathbf{w}', \mathbf{x}' \rangle)$. Last, let us normalize $\tilde{\mathbf{w}} = \mathbf{w}'/\|\mathbf{w}'\|$, $\tilde{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}'\|$, and redefine L to be

$$L = \frac{\|\mathbf{w}'\| \|\mathbf{x}'\| \log(2/\epsilon - 1)}{2} \tag{12}$$

so that $h_{\mathrm{sig}}(\mathbf{x}) = \phi_{\mathrm{sig}}(\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle)$. Thus we see that if there exists an algorithm that runs in time $\mathrm{poly}(L, 1/\epsilon, 1/\delta)$ and $(\epsilon/2, \delta)$-learns the class H_{sig}, then since for all $h \in H$ exists $h_{\mathrm{sig}} \in H_{\mathrm{sig}}$ such that $|h_{\mathrm{sig}}(\mathbf{x}) - h(\mathbf{x})| \leq \epsilon/2$, there also exists an algorithm that (ϵ, δ)-learns the concept class H defined in Theorem 5 in time polynomial in $(L, 1/\epsilon, 1/\delta)$ (for L defined in Equation 12). But by definition of L in Equation 12 and the fact that $\|\mathbf{w}'\|$ and $\|\mathbf{x}'\|$ are of size $\mathrm{poly}(n)$, this means that there is an algorithm that runs in time polynomial in $(n, 1/\epsilon, 1/\delta)$ and (ϵ, δ)-learns the class H, which contradicts Lemma 3. ■

5 Related work

The problem of learning linear and kernel predictors has been extensively studied before, mainly in the framework of Support Vector Machines [Vapnik, 1998; Cristianini and Shawe-Taylor, 2004; Schölkopf and Smola, 2002]. In the special case where the data is separable (namely, there exists a predictor $\mathbf{w}$ such that $y\langle \mathbf{w}, \mathbf{x} \rangle \geq 0$ for any example $(\mathbf{x}, y)$), it is possible to learn a predictor in polynomial time, say by using linear programming. The learning problem becomes much more difficult when the data is not separable.

A paper closely related to ours is [Ben-David and Simon, 2000], whose techniques can be adapted to develop an exhaustive-search algorithm for our problem, whose runtime is $\mathrm{poly}\left(\exp\left(\frac{L^2}{\epsilon^2}\log(\frac{L}{\epsilon})\right)\right)$. In comparison, the runtime of our algorithm is exponentially smaller. Moreover, the algorithm of [Ben-David and Simon, 2000] performs an exhaustive search over all $(L/\epsilon)^2$ subsets of the m examples in the training set, and therefore its runtime is always order of m^{L^2/ϵ^2}. In contrast, our algorithm's runtime depends on a parameter B, which is bounded by $\exp(L)$ only under a worst-case assumption. Depending on the underlying distribution, B can be much smaller than the worst-case bound. In practice, we will cross-validate for B, and therefore the worst-case bound will often be pessimistic.

Another related line of work has been learning of linear predictors with respect to the 0-1 loss function, but making distributional assumptions on the data. For instance, [Kalai *et al.*, 2005] show that when the distribution over $\mathcal{X} \subseteq$ is uniform, and assuming it is a subset of $\mathbb{R}^n$, then it is possible to learn in time $\mathrm{poly}(n^{1/\epsilon^4})$. This was further generalized by [Blais *et al.*, 2008], who showed that similar bounds hold for product distributions. However, such distributional assumptions are not very realistic. Moreover, these works are characterized by explicit dependence on the dimension of $\mathcal{X}$, and therefore are not adequate for the kernel-based setting we consider in this paper, in which the dimensionality of $\mathcal{X}$ can even be infinite. Interestingly, in [Shalev-Shwartz *et al.*,

2009] we show that the very same algorithm we use in this paper recovers the same complexity bound of [Kalai *et al.*, 2005].

In terms of hardness results, there exist strong results on hardness of *proper* learning, where the learning algorithm must return a linear predictor (e.g. [Guruswami and Raghavendra, 2006; Ben-David and Simon, 2000] and references therein). We emphasize that we allow improper learning, which is just as useful for the purpose of learning good classifiers, and thus these hardness results do not apply.

6 Discussion

In this paper we described and analyzed a new technique for agnostically learning linear and kernel predictors with the 0-1 loss function. The bound we derive is polynomial for any fixed L, the Lipschitz coefficient of the transfer function. While we prove that (under a certain cryptographic assumption) no algorithm can have a polynomial dependence on L, the immediate open question is whether the dependence on L can be further improved.

A perhaps surprising property of our analysis is that we propose a single algorithm, returning a single classifier, which is simultaneously competitive against *all* transfer functions $p \in P_B$. In particular, it learns with respect to the "optimal" transfer function, where by optimal we mean the one which attains the smallest error rate, $\mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x} \rangle) - y|]$, over the distribution $\mathcal{D}$.

Our algorithm boils down to a simple ERM algorithm using a particular kernel function. In fact, it is possible to show that the standard Support Vector Machine algorithm, using the hinge-loss and our particular kernel, can also give similar guarantees. It is therefore interesting to study if there is something special about the kernel we propose or maybe other kernel functions (e.g. the Gaussian kernel) can give similar guarantees.

Acknowledgments

We would like to thank Adam Klivans for helping with the Hardness results. Shai Shalev-Shwartz is supported by the Israeli Science Foundation grant number 598-10.

References

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

[Bartlett *et al.*, 2006] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

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

[Blais *et al.*, 2008] E. Blais, R. O'Donnell, and K Wimmer. Polynomial regression under arbitrary product distributions. In *COLT*, 2008.

[Cristianini and Shawe-Taylor, 2004] N. Cristianini and J. Shawe-Taylor. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Guruswami and Raghavendra, 2006] V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of the 47th Foundations of Computer Science (FOCS)*, 2006.

[Kalai *et al.*, 2005] A. Kalai, A.R. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th Foundations of Computer Science (FOCS)*, 2005.

[Kearns *et al.*, 1992] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *COLT*, pages 341–352, July 1992. To appear, *Machine Learning*.

[Klivans and Sherstov, 2006] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *FOCS*, 2006.

[Schapire, 1990] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Schölkopf and Smola, 2002] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[Shalev-Shwartz *et al.*, 2009] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Agnostically learning halfspaces with margin errors. Technical report, Toyota Technological Institute, 2009.

[Shalev-Shwartz *et al.*, 2010] S. Shalev-Shwartz, O. Shamir, and K. Sridharan. Learning kernel-based halfspaces with the zero-one loss, 2010. Technical Report, available at arXiv:1005.3681.

[Vapnik, 1998] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[Zhang, 2004] T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.