Technical Report
TTIC-TR-2009-4

June 2009

Agnostically Learning Halfspaces with Margin Errors

Shai Shalev-Shwartz
Toyota Technological Institute, Chicago, USA
`shai@tti-c.org`

Ohad Shamir
The Hebrew University, Jerusalem, Israel
`ohadsh@cs.huji.ac.il`

Karthik Sridharan
Toyota Technological Institute, Chicago, USA
`karthik@tti-c.org`

ABSTRACT

We describe and analyze a new algorithm for agnostically learning halfspaces with respect to the *margin* error rate. Roughly speaking, this corresponds to the worst-case error rate after each point is perturbed by a noise vector of length at most μ. Margin based analysis is widely used in learning theory and is considered the most successful theoretical explanation for the statistical properties of several learning algorithms, such as Support Vector Machines and AdaBoost. The proposed algorithm can learn n-dimensional halfspaces in time $\mathrm{poly}(n \exp(\frac{1}{\mu} \log(\frac{1}{\mu\epsilon})))$, for *any* distribution, where μ is the margin parameter and the error rate of the learned classifier is at most ϵ plus the margin error rate of the optimal halfspace. This improves over the bound $\mathrm{poly}(n \exp((\frac{1}{\mu})^2 \log(\frac{1}{\mu\epsilon})))$, derived by Ben-David and Simon [7]. If the distribution over instances is uniform on the unit ball, we recover the $\mathrm{poly}(n^{1/\epsilon^4})$ complexity bound of Kalai et al [17]. Furthermore, the dependence on the dimension n in our complexity bound can be replaced by the time required to calculate inner-products. This enables us to efficiently learn halfspaces in possibly infinite dimensional Hilbert spaces by using the so-called kernel trick.

1 Introduction

Some of the most important machine learning tools are based on learning halfspaces, also known as linear classifiers. Examples include the Perceptron [23], Support Vector Machines [27], and AdaBoost [15]. A linear classifier $h_{\mathbf{w}}$, parameterized by a vector $\mathbf{w}$, is a boolean function defined as $h(\mathbf{x}) = \mathbf{1}(\langle \mathbf{w}, \mathbf{x} \rangle > 0)$, where $\langle \mathbf{w}, \mathbf{x} \rangle$ is the inner product between $\mathbf{w}$ and the vector instance $\mathbf{x}$, and $\mathbf{1}(\text{predicate})$ is 1 if the predicate holds and 0 otherwise. While the expressive power of linear classifiers seems to be rather restricted – for example, it is impossible to express XOR functions using halfspaces – one can use the so-called kernel trick to implicitly map the instances into a higher dimension space and then learn a linear classifier in that space. Kernel-based linear classifiers can handle not only non-linear decision boundaries but can also be utilized for efficiently classifying non-vectorial instances such as trees and strings (see for example [11]). The kernel trick has had tremendous impact on machine learning theory and algorithms over the past decade.

In the agnostic PAC learning framework of [18], applied to binary classification, there is an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$ (where $\mathcal{X}$ is some domain), and the error rate of a classifier $h : \mathcal{X} \to \{0, 1\}$ is defined as:

$$\mathrm{err}(h) \stackrel{\mathrm{def}}{=} \mathbb{E}[|h(\mathbf{x}) - y|] \ , \tag{1}$$

where expectation is with respect to $\mathcal{D}$. Since both $h(\mathbf{x})$ and y are in $\{0, 1\}$ we can rewrite $\mathrm{err}(h)$ as $\Pr[h(\mathbf{x}) \neq y]$. The current definition will allow us later on to deal with more general classifiers such as randomized classifiers. The learning algorithm is allowed to sample pairs $(\mathbf{x}, y)$ from $\mathcal{D}$, and its goal is to produce a near-optimal classifier $\hat{h}$ with respect to a fixed concept class H of classifiers:

$$\mathrm{err}(\hat{h}) \ \leq \ \min_{h \in H} \mathrm{err}(h) + \epsilon \ . \tag{2}$$

When learning (origin-centered) halfspaces, the concept class equals $\{\mathbf{x} \mapsto \mathbf{1}(\langle \mathbf{w}, \mathbf{x} \rangle > 0)\}_{\mathbf{w}:\|\mathbf{w}\|=1}$. 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. For agnostically learning halfspaces when $\mathcal{X} = \mathbb{R}^n$, the best current result is the algorithm of [17], with complexity $\mathrm{poly}(n)$ for any constant $\epsilon > 0$. However, this algorithm crucially assumes a restricted set of marginal distribution on $\mathcal{X}$: either uniform on the unit ball, uniform on $\{-1, +1\}^n$, or log-concave. This was further generalized by [8], who showed that similar bounds hold for product distributions. Beside distributional assumptions, these works are also characterized by explicit dependence on the dimension of $\mathcal{X}$.

It is well known that the VC dimension of halfspaces in an n-dimensional space equals n. This implies that the number of training examples, as well as the complexity of learning halfspaces, scales at least linearly with the dimension n [27]. In fact, without imposing more assumptions, at least n examples are necessary for learning an n-dimensional halfspaces with a guarantee of the form given in Equation (2). Since kernel-based learning algorithms allow $\mathcal{X}$ to be an infinite dimensional inner product space, the performance requirement of the learned classifier must be changed. The most common approach is to require that the learned classifier will be competitive with the *margin* error rate of the optimal halfspace. Formally, the μ-margin error rate of a halfspace of the form $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{1}(\langle \mathbf{w}, \mathbf{x} \rangle > 0)$ is defined as:

$$\mathrm{err}_{\mu}(\mathbf{w}) \;=\; \Pr[h_{\mathbf{w}}(\mathbf{x}) \neq y \vee |\langle \mathbf{w}, \mathbf{x} \rangle| \leq \mu] \ . \tag{3}$$

Intuitively, $\mathrm{err}_{\mu}(\mathbf{w})$ is the error rate of $h_{\mathbf{w}}$ had we μ-shifted each point in the worst possible way. Based on the definition of $\mathrm{err}_{\mu}(\mathbf{w})$ we can restate the goal of the learner as finding a classifier h that satisfies:

$$\mathrm{err}(h) \;\leq\; \min_{\mathbf{w}:\|\mathbf{w}\|=1} \mathrm{err}_{\mu}(\mathbf{w}) + \epsilon \ . \tag{4}$$

Bounds of the above form are called margin-based bounds and are widely used in the statistical analysis of Support Vector Machines and AdaBoost. It was shown [4, 22] that $m = \Theta(\log(1/\delta)/(\mu\,\epsilon)^2)$ examples are sufficient (and necessary) to learn a classifier for which Equation (4) holds with probability of at least $1-\delta$. Note that m does not depend on the dimension. This fact allows us to learn even in infinite dimensional spaces.

From a computational perspective, if $\min_{\mathbf{w}} \mathrm{err}_{\mu}(\mathbf{w}) = 0$, then it is possible (e.g. using the kernel Perceptron of [14]) to learn a classifier that satisfies Equation (4) in time $\mathrm{poly}(\kappa/(\mu\epsilon))$, where κ is the time required to implement a single inner product. The reason we distinguish between κ and the dimension is that when using kernels, the dimension is very large while κ can still be a small number (for more details, see appendix A). The learning problem becomes much more difficult when $\min_{\mathbf{w}} \mathrm{err}_{\mu}(\mathbf{w}) > 0$. A computational complexity analysis under margin assumptions was first carried out in [7] (see also the hierarchical worst-case analysis recently proposed in [6]). The technique used in [7] is the observation that in the noise-free case, an optimal halfpsace can be expressed as a linear sum of at most $1/\mu^2$ examples. Therefore, one can perform an exhaustive search over all sub-sequences of $1/\mu^2$ examples, and choose the optimal halfspace. Combining this with the margin-based sample complexity bound described previously, the total runtime complexity of the approach is $\mathrm{poly}(\kappa \exp((\frac{1}{\mu})^2 \log(\frac{1}{\mu\epsilon})))$. Also, we note that practical algorithms such as support vector machines [27] often replace the 0-1 error function with a convex surrogate, and then apply convex optimization tools. However, there are no guarantees on how well the surrogate function approximates the 0-1 error function (there do exist some recent results on the *asymptotic* relationship between these error functions in some cases (cf. [5]), but these do not apply to the finite-sample setting we are studying). In terms of negative results, we note that there exist strong hardness of approximation results for *proper* learning *without* margin (see for example [16, 13] and the references therein). There are also hardness results for proper learning with sufficiently small margins [7]. 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.

We propose a new algorithm for learning linear classifiers, with or without kernels, for arbitrary distributions. The main techniques for the derivation and analysis are:

- We approximate the 0-1 function $\mathbf{1}(a > 0)$ with polynomials of possibly infinite degree. The idea of approximating this function with a polynomial was first proposed by [17]. However, while [17] try to approximate the function with a low-degree polynomial, we require instead that the coefficients of the polynomials are bounded in a sense that will be formally defined in Section 1.1. The principle that "the size of the parameters is more important than their number" was one of the main advantages in the analysis of the statistical properties of several learning algorithms (e.g. [3]). Here, we use this principle for obtaining an efficient algorithm.

- After applying the polynomial approximation, the learning problem boils down to a convex optimization problem in an infinite-dimensional space. Building on Wahba's representer theorem [28], we show that the problem can be solved efficiently using the kernel trick.

- To further improve the complexity of our algorithm, we show how an approximate solution can be obtained after a single pass over the examples. The idea is to apply an online learning algorithm together with a technique called "online-to-batch" conversion.

- We show that our algorithm produces an approximate solution with respect to *all* polynomial approximations of the 0-1 function that satisfy a certain boundedness condition on their coefficients. The bound holds for the same algorithm, simultaneously for all polynomials. Intuitively, the algorithm learns both the halfspace, and (implicitly) the optimal approximation polynomial for the data at the same time.

- To derive explicit quantitative bounds, we describe and analyze specific polynomial approximations, focusing on the technique of Chebyshev polynomial approximation on top of a sigmoid function. The coefficients of this polynomial approximation can be bounded explicitly using tools from complex analysis. In particular, we rely on the fact that the sigmoid is a meromorphic function over the complex field. With these polynomials, we prove that our algorithm can learn a classifier which satisfies Equation (4) in time $\text{poly}(\kappa\ \exp(\frac{1}{\mu}\log(\frac{1}{\mu\epsilon}))$. This improves over the $\text{poly}(\kappa\ \exp((\frac{1}{\mu})^2\log(\frac{1}{\mu\epsilon})))$ bound, which is implied by the work of [7].

- The latter bound holds for *any* distribution and is dimension free up to computing inner products, which with kernels can be computed efficiently even in infinite-dimensional spaces. However, the generality of our algorithm allows us to use it seamlessly in the plain vanilla setting of learning halfspaces in $\mathbb{R}^n$ without kernels. In this case, we can recover the $\text{poly}(n^{1/\epsilon^4})$ complexity bound of [17], with the same assumption of uniform distribution over the unit ball. Interestingly, the very same algorithm works both under the margin-based analysis or the uniform distribution assumption.

1.1 Main Results

Let $\mathcal{X}$ be a compact subset of a reproducing kernel Hilbert space (or RKHS), which w.l.o.g. will be taken to be the unit ball around the origin. For simplicity, one can think of $\mathcal{X}$ as simply the unit ball in $\mathbb{R}^n$. For kernel-based classifiers, however, we will need the more general notion of an RKHS, which is a certain complete inner product space (see appendix A for more details). Let κ be the time required to perform an inner product $\langle \mathbf{x}, \mathbf{x}' \rangle$ between pairs of elements in $\mathcal{X}$. In general, when $\mathcal{X}$ is an n-dimensional space, we have $\kappa = n$. However, when using kernels, κ can be a small constant independent from the dimension of $\mathcal{X}$.

To derive and analyze our results, we accommodate classifiers with real-valued output, as well as randomized classifiers. Randomized classifiers can be defined as mappings of the form $h : \mathcal{X} \to [0, 1]$, where the predicted label for an instance $\mathbf{x}$ is chosen to be 1 with probability $h(\mathbf{x})$, and 0 otherwise. In that case, the definition of $\text{err}(h)$ given in Equation (1) can be rewritten as $\mathrm{err}(h) = \Pr_{(\mathbf{x},y)\sim\mathcal{D},\hat{y}\sim h(\mathbf{x})}[\hat{y} \neq y]$. We will also use the definition of $\mathrm{err}(\cdot)$ given in Equation (1) for functions with arbitrary real-valued output, which will be discussed later on.

The concept classes we deal with below will be of the form

$$H_\phi = \{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x}\rangle)\}_{\mathbf{w}:\|\mathbf{w}\|\leq 1},$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is denoted as the *transfer function.* For $\phi_{0-1}(a) \stackrel{\text{def}}{=} \mathbf{1}(a > 0)$, $H_{\phi_{0-1}}$ is simply the class of halfspaces[1]. However, we will also investigate smoother transfer functions which approximate the 0-1 transfer function. The resulting concept classes, which can be thought of as "fuzzy" halfspaces, will allow us to derive learning guarantees for margin errors, but might also be of independent interest as a natural generalization of the concept class of halfspaces.

Our main theorem is the following:

Theorem 1 *Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X} \times \{0,1\}$. For $B > 0$, let*

$$P_B = \left\{ p(a) = \sum_{j=0}^{\infty} \beta_j\, a^j \;:\; \sum_{j=0}^{\infty} \beta_j^2\, 2^j \leq B \right\}$$

be the set of all polynomials (possibly of infinite degree) with an ℓ_2-bound on the growth of their coefficients. For any $\epsilon > 0, \delta > 0$, let

$$m = \Omega\left(\frac{B \log(1/\delta)}{\epsilon^2}\right) .$$

Then, the algorithm described in Section 2 agnostically learns $H_B \stackrel{\text{def}}{=} \bigcup_{p\in P_B} H_p$ in time $O(\kappa m^2)$. Namely, with probability at least $1-\delta$, the classifier $\tilde{f} : \mathcal{X} \to [0,1]$ returned by the algorithm satisfies

$$\mathrm{err}(\tilde{f}) \leq \min_{h\in H_B} \mathrm{err}(h) + \epsilon.$$

[1]For simplicity, we are concerned only with origin-centered halfspaces, namely we do not allow an additional bias term. It is easy to extend our results to learning general halfspaces using the following standard trick: for any given example, add a constant coordinate of 1 to each $\mathbf{x}$. Any general halfspace in $\mathcal{X}$, defined as $\mathrm{sgn}\langle \mathbf{w}, \mathbf{x}\rangle + b$ now corresponds to the origin-centered halfspace $\mathrm{sgn}\langle [\mathbf{w}, b], [\mathbf{x}, 1]\rangle$. The only change is that the norm bounds on $\mathbf{x}$ and $\mathbf{w}$ are now somewhat different, but this can be dealt with by making some trivial changes to the analysis.

We note that the 2-factor in the definition of P_B is rather arbitrary, and can be replaced with ν^{-j} for an arbitrary $\nu \in (0,1)$. The price we pay is an additional $1/(1-\nu)$ factor in the bound for m in the theorem. For clarity of representation, we chose the value of $\nu = 0.5$.

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 $\mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x}\rangle) - y|]$. Naturally, the optimal transfer function depends on the distribution $\mathcal{D}$, which is unknown to us. In the rest of this section, we derive more explicit complexity results for learning concept classes defined by ϕ_{0-1} or smooth approximations of it. This allows us to derive complexity results for learning with respect to margin error (Equation (4)). It is important to note that while for the sake of the analysis we must specify explicitly some transfer functions, the algorithm itself is not affected by this specification.

Our first explicit bound involves the 0-1 transfer function. While the 0-1 transfer function cannot be expressed as a polynomial in P_B for any finite B, it can be approximated by a polynomial in P_B. In particular, the following lemma shows that by imposing a (rather strong) uniform distribution assumption on the marginal distribution over $\mathcal{X}$, one can approximate the 0-1 transfer function by a polynomial. This technique was first proposed by [17] for the very same purpose. In fact, we use exactly the same Hermite polynomials construction as in [17]. However, while [17] shows that the 0-1 transfer function can be approximated by a low degree polynomial, we are concerned with polynomials having bounded coefficients. By showing that the approximating polynomial has bounded coefficients, we are able to re-derive the results in [17] for a uniform distribution with a different algorithm.

Lemma 1 *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{0,1\}$, where $\mathcal{X}$ is the unit ball in $\mathbb{R}^n$ and the marginal distribution of $\mathcal{D}$ on $\mathcal{X}$ is uniform. For any $\epsilon \in (0,1)$, if $B = \mathrm{poly}(n^{1/\epsilon^4})$, then there exists $p \in P_B$ such that*

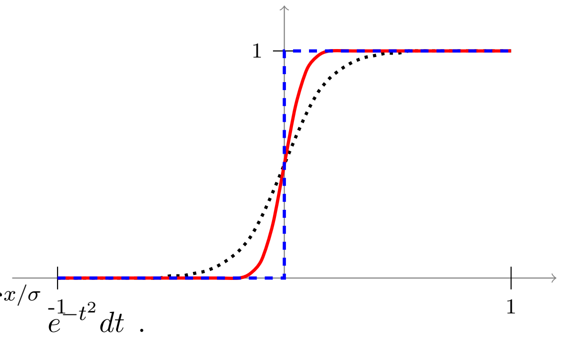
$$\mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x}\rangle) - y|] \;\leq\; \mathbb{E}[|\phi_{0-1}(\langle \mathbf{w}, \mathbf{x}\rangle) - y|] + \epsilon \;.$$

As a direct corollary, using Theorem 1, we obtain the following:

Corollary 1 *Assume that the conditions of Theorem 1 and Lemma 1 hold. For any $\epsilon > 0, \delta > 0$, let $m = \mathrm{poly}\left(n^{1/\epsilon^4} \log\left(\frac{1}{\delta}\right)\right)$. Then the algorithm described in Section 2 agnostically learns $H_{\phi_{0-1}}$ in time $O(\kappa m^2)$.*

As mentioned before, a similar bound has been obtained by [17] and the above merely shows that our algorithm is not weaker than the one described in [17]. Moreover, this bound makes restrictive distributional assumptions. We note that [17, 8] further relaxed the distributional assumptions, but our focus in this paper is different and therefore we made no attempt to recover all of their results.

To derive distribution-free results, we will use bounds which are not with respect to the 0-1 transfer function, but rather with respect to smoother transfer functions. A simple preliminary example is the Gaussian cumulative distribution function with variance $\sigma^2/2$, defined as



$$\phi_{\mathrm{erf}}(a) \overset{\mathrm{def}}{=} \frac{1}{2}+\frac{1}{2}\mathrm{erf}\left(\frac{x}{\sigma}\right) = \frac{1}{2}+\frac{1}{\sqrt{\pi}}\int_0^{x/\sigma} e^{-t^2}\,dt\ . \tag{5}$$

Figure 1: Illustrations of the transfer functions used in the paper: the 0-1 transfer function (dashed blue line); the erf transfer function (red line, for $\sigma = 0.1$); and the sigmoid transfer function (dotted black line, for $\sigma = 0.1$).

The variance parameter σ controls the steepness of the transfer function. As σ decreases, the erf transfer function resembles more and more the 0-1 transfer function. An illustration of ϕ_{erf} is given in Figure 1. This function equals its Taylor series expansion at any point, which is an infinite-degree polynomial with bounded coefficients, hence can be analyzed using Theorem 1. However, the derived complexity results, in terms of ϵ and the margin parameter μ, are not superior to those in [7], and are hence relegated to appendix B.

To get better results, we will use a different transfer function and approximating polynomial. The transfer function will be the sigmoid function, defined as

$$\phi_{\mathrm{sig}}(x) = \frac{1}{1+e^{-x/\sigma}}. \tag{6}$$

As in the ϕ_{erf} case, σ controls the steepness of the curve, which resembles the 0-1 transfer function more and more as $\sigma \to 0$ (see Figure 1). The following lemma shows how the sigmoid transfer function can be approximated by a polynomial. The proof is based on a Chebyshev approximation technique.

Lemma 2 *Let ϕ_{sig} be as defined in Equation (6), where for simplicity we assume $\sigma \le 1/4$. For any $\epsilon > 0$, let*

$$B = 10 + \frac{1}{12\sigma^4} + \frac{3}{2}\exp\left(4\log_{1+\pi\sigma}\left(\frac{2\sigma + 1/\pi}{\pi\sigma\epsilon}\right)\right).$$

Then there exists $p \in P_B$ such that

$$\mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x}\rangle) - y|] \le \mathbb{E}[|\phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x}\rangle) - y|] + \epsilon.$$

By Theorem 1, it follows that:

Corollary 2 *Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X} \times \{0,1\}$. Let $\sigma > 0$ be a variance parameter and let ϕ_{sig} be as defined in Equation (6). Denote B as in Lemma 2, and let*

$$m = \Omega\left(\frac{B\log(1/\delta)}{\epsilon^2}\right) .$$

Then the algorithm described in Section 2 agnostically learns $H_{\phi_{\text{sig}}}$ in time $O(\kappa m^2)$.

Finally, we use this corollary to derive a margin error guarantee:

Theorem 2 *Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X} \times \{0,1\}$, and let μ be a margin parameter. Let $\epsilon > 0, \delta > 0$ such that $\mu/\log(1/\epsilon) \le 1/4$, and set*

$$B = 10 + \tfrac{\log^4(1/\epsilon)}{12\mu^4} + \frac{3}{2}\exp\left(\left(\tfrac{2.1\log(1/\epsilon)}{\mu}\right)\left(\log\left(\tfrac{1}{\mu}\right) + \log\left(\tfrac{1}{\pi\epsilon}\right) + \log\log\left(\tfrac{1}{\epsilon}\right)\right)\right) ,$$

$$m = \Omega\left(\frac{B\log(1/\delta)}{\epsilon^2}\right) .$$

Then, the algorithm described in Section 2 runs in time $O(\kappa m^2)$, and returns a classifier $\tilde{f} : \mathcal{X} \to [0,1]$ such that with probability at least $1-\delta$, $\text{err}(\tilde{f}) \le \min_{\mathbf{w}:\|\mathbf{w}\|\le 1} \text{err}_\mu(\mathbf{w}) + \epsilon$.

2 The Algorithm

Our algorithm is based on implicitly solving a convex optimization problem in an infinite dimensional space. We first present the explicit infinite-dimensional optimization problem, and then show it can be solved efficiently, with a pseudo-code provided.

To simplify the presentation, we derive the algorithm without the use of kernels, therefore $\mathcal{X}$ is simply the unit ball in $\mathbb{R}^n$. In Appendix A we show how the results can be extended to kernel-based classifiers. 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

$$2^{-j/2} x_{k_1} \cdot x_{k_2} \cdots x_{k_j}.$$

Note that although the vector is infinite, inner products between such vectors can actually be computed efficiently, since for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$,

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

Let $K(\mathbf{x}, \mathbf{x}') = 2/(2 - \langle \mathbf{x}, \mathbf{x}' \rangle)$ denote the function which performs this inner product. The fact that we can efficiently perform inner products in infinite-dimensional spaces is known as the *kernel trick* in machine learning, and the function $K(\cdot, \cdot)$ is known as the *kernel function* (for more details, see again appendix A).

Given a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, our algorithm solves the following convex optimization problem:

$$\min_{\mathbf{v}:\|\mathbf{v}\|^2 \le B} \frac{1}{m} \sum_{i=1}^{m} |\langle \mathbf{v}, \psi(\mathbf{x}_i) \rangle - y_i| \ , \tag{7}$$

and the predictor is defined to be

$$\tilde{f}(\mathbf{x}) \ = \ \min\{1, \max\{0, \langle \tilde{\mathbf{v}}, \psi(\mathbf{x}) \rangle\}\} \ , \tag{8}$$

where $\tilde{\mathbf{v}}$ is a solution of Equation (7). Of course, this cannot be performed explicitly, since these objects are infinite-dimensional. Below, we explain how we can efficiently implement this algorithm.

Since the objective function and the predictor are 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 the Representer theorem [28] that there is an optimal $\tilde{\mathbf{v}}$ in the span of $\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)$ (see appendix A). Therefore, instead of learning a vector $\mathbf{v}$, we can learn a set of weights $\alpha_1, \ldots, \alpha_m$ based on the training set. Once these are founds, we can rewrite the definition of the predictor in Equation (8) as

$$\begin{aligned}\tilde{f}(\mathbf{x}) &= \min\{1, \max\{0, \langle \sum_{i=1}^{m} \alpha_i \psi(\mathbf{x}_i), \psi(\mathbf{x})\rangle\}\} \\ &= \min\{1, \max\{0, \sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x})\}\} \ .\end{aligned}$$

The r.h.s. can of course be computed efficiently.

As to the learning phase (namely, solving the optimization problem in Equation (7)), it can be rewritten in terms of the variables $\alpha_1, \ldots, \alpha_m$. The resulting optimization problem is convex with respect to the α's, and therefore can be solved by standard convex optimization tools such as the ellipsoid method or interior point methods. To further improve the complexity bound and to simplify the derivation of the generalization bound, we instead choose to use a simple stochastic gradient descent technique. This technique, well known in machine learning theory, can be seen as a variant of the perceptron algorithm. At time t, we update our current classifier $\mathbf{v}$, with respect to the gradient contributed by example $(\mathbf{x}_t, y_t)$, scaled by a learning rate $\eta = \sqrt{B/T}$:

$$\mathbf{v} = \mathbf{v} - \eta \,\mathrm{sgn}(\langle \mathbf{v}, \psi(\mathbf{x}_t)\rangle - y_t)\psi(\mathbf{x}_i),$$

where $\mathrm{sgn}(\cdot)$ is the sign function. If $\|\mathbf{v}\|^2 > B$, we shrink the classifier back to the feasible region: $\mathbf{v} = \mathbf{v}\sqrt{B/\|\mathbf{v}\|}$. To obtain the final classifier, we average over the ensemble of m classifiers obtained at every time step. This allows us to obtain a generalization guarantee with respect to the unknown underlying distribution. Since all the operations are only in terms of weights and inner products between elements in the mapped training data, we can use the kernel function $K(\cdot, \cdot)$ to implicitly perform these infinite-dimensional inner products. An optimized pseudo-code is provided in Figure 1.

It is easily verified that the algorithm runs in time $O(\kappa\, m^2)$. Its generalization guarantee, stated in the theorem below, is standard in the statistical machine learning literature (see [29, 10] for details).

Theorem 3 *Algorithm 1 presented below runs in time $O(\kappa\, m^2)$, and results in a classifier $\tilde{f} : \mathcal{X} \to [0, 1]$ such that with probability at least $1 - \delta$,*

$$\mathrm{err}(\tilde{f}) \leq \min_{\mathbf{v}:\|\mathbf{v}\|^2 \leq B} \mathbb{E}[|\langle \mathbf{v}, \psi(\mathbf{x})\rangle - y|] + O\left(\sqrt{\frac{B \log(1/\delta)}{m}}\right).$$

Algorithm 1 Approximated Half-Space Learning Algorithm

```
INPUT: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ ; $B$ ; $K(\mathbf{x}, \mathbf{x}') = 2/(2 - \langle \mathbf{x}, \mathbf{x}' \rangle)$
INITIALIZE: $\eta = \sqrt{m/B}$, $a = 0$, $\alpha_j = 0, \bar{\alpha}_j = 0$ for all $j = 1, \ldots, m$
For $i = 1, 2, \ldots, m$
    $b = \sum_{j=1}^{i-1} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)$          // Prediction of current classifier on $\mathbf{x}_i$
    $\alpha_i = -\eta \, \mathrm{sgn}(b - y_i)$
    $a = a + 2b - K(\mathbf{x}_i, \mathbf{x}_i) \, \alpha_i^2$          // Update classifier norm
    If $a > B$
        For $j = 1, \ldots, i$
            $\alpha_j = \alpha_j \sqrt{B/a}$
        $a = B$
    For $j = 1, \ldots, i$
        $\bar{\alpha}_j = \bar{\alpha}_j + \alpha_j / m$          // Track average of all classifiers so far
OUTPUT: predictor $\tilde{f}(\mathbf{x}) = \min\{1, \max\{0, \sum_{i=1}^{m} \bar{\alpha}_i K(\mathbf{x}_i, \mathbf{x})\}\}$
```

3 Proofs

Due to lack of space, some of the proofs are deferred to Appendix C.

3.1 Proof of Theorem 1

For simplicity, we will assume that $\mathcal{X}$ is a finite-dimensional space, with dimension n. The proof extends with minimal changes to infinite-dimensional spaces, which are relevant when we learn with kernels (see appendix A for more details). We equip the range of $\psi(\cdot)$ with the standard ℓ_2 norm. The key component in the proof is to show that the mapping $\psi(\cdot)$ does not blow up norms too much.

Lemma 3 *For all $\mathbf{x} \in \mathcal{X}$ it holds that $\|\psi(\mathbf{x})\|^2 \leq 2$. Also, for any $\mathbf{w}$ such that $\|\mathbf{w}\|^2 \leq 1$, and any polynomial $p(a) = \sum_{j=0}^{\infty} \beta_j a^j$ in P_B, if we 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 $k_1, \ldots, k_j \in \{1, \ldots, n\}^j, j = 0 \ldots \infty$) then $\|\mathbf{v}_{\mathbf{w}}\|^2 \leq B$.*

Proof The fact that for all $\mathbf{x} \in \mathcal{X}$ we have $\|\mathbf{x}\|^2 \leq 1$ yields:

$$\begin{aligned}\|\psi(\mathbf{x})\|^2 &= \sum_{j=0}^{\infty} \sum_{(k_1,\ldots,k_j)\in\{1,\ldots,n\}^j} 2^{-j} x_{k_1}^2 \cdots x_{k_j}^2 = \sum_{j=0}^{\infty} 2^{-j} (\langle \mathbf{x}, \mathbf{x}\rangle)^j \\ &\leq \sum_{j=0}^{\infty} 2^{-j} = 2 .\end{aligned}$$

To show the second part of the lemma, we have

$$\begin{aligned}\|\mathbf{v}_{\mathbf{w}}\|^2 &= \sum_{j=0}^{\infty} \sum_{k_1,\ldots,k_j} \beta_j^2 2^j w_{k_1}^2 \cdots w_{k_j}^2 = \sum_{j=0}^{\infty} \beta_j^2 2^j \sum_{k_1} w_{k_1}^2 \sum_{k_2} w_{k_2}^2 \cdots \sum_{k_j} w_{k_j}^2 \\ &= \sum_{j=0}^{\infty} \beta_j^2 2^j \left(\|\mathbf{w}\|^2\right)^j \leq B.\end{aligned}$$

■

Let $p(a) = \sum_{j=0}^{\infty} \beta_j a^j$ be an arbitrary polynomial in P_B. Let $\mathbf{w}^* = \arg\min_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \mathbb{E}[|p(\langle \mathbf{w}, \mathbf{x}\rangle) - y|]$ be the optimal linear classifier with respect to $p(\cdot)$ and let $\mathbf{v}_{\mathbf{w}^*}$ be its corresponding element in $\mathbb{R}^{\mathbb{N}}$ (defined as in Lemma 3). Let $\tilde{f}$ be the output classifier of the algorithm. Combining Lemma 3 with Theorem 3, we have that with probability at least $1-\delta$,

$$\operatorname{err}(\tilde{f}) \leq \mathbb{E}[|\langle \mathbf{v}_{\mathbf{w}^*}, \psi(\mathbf{x})\rangle - y|] + O\left(\sqrt{\frac{B\log(1/\delta)}{m}}\right) . \tag{9}$$

However, by definition of ψ and $\mathbf{v}_{\mathbf{w}^*}$, we have that

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

Plugging this into Equation (9), and using the optimality of $\mathbf{w}^*$, the theorem follows.

3.2 Proof of Theorem 3

Let $\alpha_1^{(t)}, \ldots, \alpha_m^{(t)}$ be the value of the $\alpha' s$ at the beginning of round t of the algorithm, and let $\mathbf{v}_t = \sum_i \alpha_i^{(t)} \psi(\mathbf{x}_i)$. The algorithm can be rewritten as

$\mathbf{v}_1 = \mathbf{0}$ and for $t > 1$,

$$g_t(\mathbf{v}) = |\langle \mathbf{v}, \psi(\mathbf{x}_t)\rangle - y_t| \ , \ \hat{\mathbf{v}}_{t+1} = \mathbf{v}_t - \eta \nabla g_t(\mathbf{v}_t) \ \text{ and } \ \mathbf{v}_{t+1} = \min_{\mathbf{v}:\|\mathbf{v}\|^2 \le B} \|\mathbf{v} - \hat{\mathbf{v}}_{t+1}\| \ ,$$

where $\nabla g_t(\mathbf{v}_t)$ is a (sub)-gradient of the function g_t at $\mathbf{v}_t$. This algorithm corresponds to the online convex optimization algorithm of [29, Section 2.2], and combining [29, Theorem 2] with the fact that $\|\nabla g_t(\mathbf{v}_t)\|^2 \le \|\psi(\mathbf{x}_t)\|^2 \le 2$ (see Lemma 3), it is easy to see that for any $\mathbf{v}^\star$ with $\|\mathbf{v}^\star\| \le \sqrt{B}$ we have:

$$\frac{1}{m}\sum_{t=1}^{m} g_t(\mathbf{v}_t) \le \frac{1}{m}\sum_{t=1}^{m} g_t(\mathbf{v}^\star) + O(\sqrt{B/m}) \ .$$

Next, using the 'averaging' online-to-batch technique (see details in [10, Corollary 2], where the basic idea is to combine Azuma concentration inequality together with Jensen's inequality) we get that with probability of at least $1-\delta$ over the choice of the training examples we have

$$\mathbb{E}[|\langle \bar{\mathbf{v}}, \psi(\mathbf{x})\rangle - y|] \ \le \ \mathbb{E}[|\langle \mathbf{v}^\star, \psi(\mathbf{x})\rangle - y|] + O(\sqrt{B\,\log(1/\delta)/m}) \ ,$$

where $\bar{\mathbf{v}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{v}_i$. Finally, our proof is concluded by noting that for any $y \in \{0,1\}$ we have

$$|\min\{1, \max\{0, \langle \bar{\mathbf{v}}, \psi(\mathbf{x})\rangle\}\} - y| \le |\langle \bar{\mathbf{v}}, \psi(\mathbf{x})\rangle - y| \ .$$

3.3 Proof sketch of Lemma 2 and Theorem 2

For a full proof, see appendix C. We approximate the sigmoid transfer function ϕ_{sig} using the technique of *Chebyshev polynomial approximation.* Chebyshev polynomials $\{T_n(\cdot)\}_{n=0}^{\infty}$ form an orthogonal basis for the space of continuous functions on $[-1,+1]$. In particular, ϕ_{sig} equals an infinite expansion $\sum_{n=0}^{\infty} \alpha_n T_n(\cdot)$, where $\alpha_n = \int_{-1}^{1} \phi_{\text{sig}}(x) T_n(x)/\sqrt{1-x^2}dx$. We truncate this into a finite polynomial $\sum_{n=0}^{N} \alpha_n T_n(\cdot)$ (where N is picked so as to optimize the bound), and estimate its coefficients to get the bound on B in Lemma 2. The major technical difficulty is estimating α_n for all n, since their defining integrals do not have a closed-form expression as a function of n and σ. For that, we turn to tools from complex analysis, utilizing the fact that ϕ_{sig} is a meromorphic function over the complex field (roughly speaking, it is sufficiently 'well behaved'). Theorem 2 is derived by determining and plugging in the largest σ so that ϕ_{sig} is ϵ-close to ϕ_{0-1} for any instance with margin larger than μ.

4 Discussion

We described and analyzed a new technique for agnostically learning halfspaces. Our algorithm recovers the result of [17] under uniform distribution assumption and improves the result of [7] for margin-based analysis. We also showed that our algorithm can agnostically learn concept classes of the form $\{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)\}$ for several smooth transfer functions, which can be thought of as "fuzzy" hyperplanes.

The immediate open question is whether the margin-based complexity bound can be further improved. To the best of our knowledge, all previous hardness results are for proper learning, where the returned classifier must be a halfspace. The focus of this paper is on improper learning, which is just as useful for the purpose of learning good classifiers. The achievable complexity bound for improper learning of halfspaces is therefore an important open problem. Another possible direction is to consider other types of margin-based analysis or transfer functions. For example, in the statistical learning literature, there are several definitions of "noise" conditions, some of them are related to margin, which lead to faster decrease of the error rate as a function of the number of examples (see for example [9, 26, 25]). Studying the computational complexity of learning under these conditions is left to future work. Finally, it would be interesting to further understand whether our results have implications to learning Boolean functions, for example by using Boolean kernels [19].

References

[1] C. Thomas-Agnan A. Berlinet. *Reproducing Kernel Hilbert Spaces in Probability and Statistics.* Springer, 2003.

[2] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions.* Dover, 1965.

[3] P. L. Bartlett. For valid generalization, the size of the weights is more important than the size of the network. In *Advances in Neural Information Processing Systems 9*, 1997.

[4] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Ams-*

terdam, The Netherlands, July 2001, Proceedings, volume 2111, pages 224–240. Springer, Berlin, 2001.

[5] 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.

[6] S. Ben-David. Alternative measures of computational complexity. In *TAMC*, 2006.

[7] S. Ben-David and H. Simon. Efficient learning of linear perceptrons. In *Advances in Neural Information Processing Systems 14*, 2000.

[8] E. Blais, R. ODonnell, and K Wimmer. Polynomial regression under arbitrary product distributions. In *COLT*, 2008.

[9] O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.

[10] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.

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

[12] D. Elliot. The evaluation and estimation of the coefficients in the Chebyshev series expansion of a function. *Mathematics of Computation*, 18(86):274–284, April 1964.

[13] V. Feldman, P. Gopalan, S. Khot, and A.K. Ponnuswami. New results for learning noisy parities and halfspaces. In *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[14] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[15] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

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

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

[18] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 341–352, July 1992. To appear, *Machine Learning*.

[19] Roni Khardon and Rocco A. Servedio. Maximum margin algorithms with boolean kernels. *J. Mach. Learn. Res.*, 6:1405–1429, 2005. ISSN 1533-7928.

[20] N. N. Lebedev. *Special Functions and Their Applications.* Dover, 1972.

[21] J.C. Mason. *Chebyshev Polynomials.* CRC Press, 2003.

[22] D. A. McAllester. Simplified PAC-Bayesian margin bounds. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, pages 203–215, 2003.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

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

[25] I. Steinwart and C. Scovel. Fast rates for support vector machines using gaussian kernels. *ANNALS OF STATISTICS*, 35:575, 2007.

[26] A. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

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

[28] G. Wahba. *Spline Models for Observational Data.* SIAM, 1990.

[29] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

A Learning with Kernels

Kernelized linear classifiers is a powerful extension of linear classifiers on $\mathbb{R}^n$, and an integral part of modern machine learning. Our algorithm can be easily combined with kernels, and the analysis of our algorithm (even for linear classifiers on $\mathbb{R}^n$) use ideas derived from kernel theory. In this appendix, we shortly explain the concept of kernelized linear classifiers; describe the main results needed for the analysis; and show how to combine our algorithm with kernels. For additional details and results, we refer the reader to [11, 24].

With standard linear classifiers, the prediction on an instance $\mathbf{x} \in \mathbb{R}^n$ using a classifier $h_{\mathbf{w}}$ parameterized by $\mathbf{w} \in \mathbb{R}^n$ is based on the inner product $\langle \mathbf{x}, \mathbf{w} \rangle$. *Kernel* linear classifiers predict by first mapping the instance $\mathbf{x}$ into an element $\psi(\mathbf{x})$ in some Hilbert space (namely, a complete inner-product space), and then predicting by computing the inner product $\langle \mathbf{w}, \psi(\mathbf{x}) \rangle$, where $\mathbf{w}$ is now an element in the Hilbert space. The power of these methods stem from the fact that these Hilbert spaces can be very high or even infinite dimensional. So while the classifier is always linear on the mapped instances in the Hilbert space, it can have a highly non-linear behavior on the original space from which instances were sampled. This phenomenon allows us to learn data which is not linearly separable. Moreover, the domain of $\psi(\cdot)$ need not even be $\mathbb{R}^n$, and it is possible to learn linear classifiers over non-vectorial objects such as strings and trees. This has had tremendous impact on machine learning theory and algorithms over the past decade.

On a first look, the obvious computational disadvantage of this approach is that an explicit evaluation of $\langle \mathbf{w}, \psi(\mathbf{x}) \rangle$ might be hard or even impossible to perform. Luckily, for a special class of Hilbert spaces called *reproducing kernel Hilbert spaces* (or RKHS for short), for any two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, the inner product of their mapping $\langle \psi(\mathbf{x}_1), \psi(\mathbf{x}_2) \rangle$ is equal to $k(\mathbf{x}_1, \mathbf{x}_2)$, where the *reproducing kernel* $k(\cdot, \cdot)$ is usually an explicit and easily computable function. In the trivial case of linear kernels, where $\psi(\cdot)$ is simply the identity transformation, we have that $k(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{x}_1, \mathbf{x}_2 \rangle$. However, many other functions are possible. For example, there exist an infinite dimensional RKHS whose kernel (called an RBF kernel in the literature) is defined as $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/\sigma^2)$ for an arbitrary $\sigma^2 > 0$.

Moreover, it is possible to prove that for many learning-related optimization problems (which include our formulation as a special case), there is an optimal classifier $\mathbf{w}^*$ in the RKHS which lies in the span of the mapped data points $\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)$. This classic result is known as Wahba's *representer theorem* in the literature [28], and basically holds whenever shifting $\mathbf{w}$ in a direction orthogonal to all data points cannot improve the objective function. Thus, when we attempt to find the optimal classifier, it is enough to determine the coefficients $\alpha_1, \ldots, \alpha_m$ in the expansion of $\mathbf{w}^* = \alpha_1 \psi(\mathbf{x}_1) + \ldots + \alpha_m \psi(\mathbf{x}_m)$. Once these coefficients are determined, prediction is easy: for any new instance $\mathbf{x}$, we predict according to

$$\langle \mathbf{w}, \psi(\mathbf{x}) \rangle = \sum_{i=1}^{m} \alpha_i \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}) \rangle = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}).$$

Thus, we can implicitly perform the inner product in a possibly infinite-dimensional Hilbert space (the term on the left), by doing a simple calculation involving a kernel function (the term on the right).

A useful feature of reproducing kernels is that they are closed to composition, multiplication and addition (including infinite summation under appropriate conditions). As a result, the function

$$K(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=0}^{\infty} (\langle \mathbf{x}_1, \mathbf{x}_2 \rangle)^j = \frac{2}{2 - \langle \mathbf{x}_1, \mathbf{x}_2 \rangle}$$

we have defined for our algorithm is in fact a reproducing kernel for some RKHS, since it is an infinite positive polynomial of the linear kernel $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$.

Moreover, the fact that kernels can be composed allows us to seamlessly combine our algorithm with kernels. All we need to change in the pseudocode described in Section 2 is to redefine $K(\cdot, \cdot)$ as $2/(2 - k(x, x'))$, where $k(\cdot, \cdot)$ is any reproducing kernel we wish. As a result, we can agnostically learn halfspaces in the (possibly) infinite-dimensional RKHS associated with $k(\cdot, \cdot)$, and thus learn *non-linear* classifiers which are near-optimal, with respect to the set of non-linear classifiers induced by halfspaces in the RKHS.

In terms of the analysis, for the proof of Theorem 1, we assumed that each instance $\mathbf{x}$ can be written as a vector $(x_1, \ldots, x_n)$. However, our analysis does not depend on the dimension and we do not need to assume that n is finite. Therefore, the analysis holds for infinite-dimensional Hilbert spaces as well. The only technicality is that in order to represent $\mathbf{x}$ as a (possibly infinite) vector, we need to show that our RKHS has a countable basis. This can be shown to hold with the mild requirement that the kernel $k(\cdot, \cdot)$ is continuous and bounded in the support of the data distribution (see [1]).

B The $\phi_{\mathrm{erf}}(\cdot)$ function

As discussed in the body of the paper, $\phi_{\mathrm{erf}}(\cdot)$ is a smooth transfer function which is relatively straightforward to analyze, although the resulting complexity bounds do not improve on the current results in the literature. In this appendix, we formally carry out the relevant analysis.

First, the following lemma tells us that $\phi_{\mathrm{erf}}(\cdot)$ is in fact an infinite degree polynomial in P_B, but with coefficients which decay sufficiently rapidly to give a meaningful bound on B.

Lemma 4 *Let $\phi_{\mathrm{erf}}(\cdot)$ be as defined in Equation (5), and*

$$B = \frac{1}{4} + \frac{4}{\pi\sigma^2}\left(1 + \frac{\sqrt{2}e}{\sigma^2}e^{4/\sigma^2}\right).$$

Then $\phi_{\mathrm{erf}}(\cdot) \in P_B$.

Proof By a standard fact, $\phi_{\mathrm{erf}}(\cdot)$ is equal to its infinite Taylor series expansion at any point, and this series equals

$$\phi_{\mathrm{erf}}(x) = \frac{1}{2} + \frac{1}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{\sigma^{2n+1}n!(2n+1)}.$$

Luckily, this is an (infinite degree) polynomial, and it is only left to calculate for which values of B does it belong to P_B. Plugging in the coefficients in the bound on B, we get that

$$\begin{aligned}
B &\leq \frac{1}{4} + \frac{1}{\pi}\sum_{n=0}^{\infty}\frac{2^{(2n+1)}}{\sigma^{2(2n+1)}(n!)^2(2n+1)^2} \leq \frac{1}{4} + \frac{1}{\pi}\sum_{n=0}^{\infty}\frac{(\sigma^2/2)^{-(2n+1)}}{(n!)^2} \\
&= \frac{1}{4} + \frac{1}{\pi\sigma^2/2}\left(1 + \sum_{n=1}^{\infty}\frac{(\sigma^2/2)^{-2n}}{(n!)^2}\right) \leq \frac{1}{4} + \frac{1}{\pi\sigma^2/2}\left(1 + \sum_{n=1}^{\infty}\frac{(\sigma^2/2)^{-2n}}{(n/e)^{2n}}\right) \\
&= \frac{1}{4} + \frac{1}{\pi\sigma^2/2}\left(1 + \sum_{n=1}^{\infty}\left(\frac{2e}{\sigma^2 n}\right)^{2n}\right).
\end{aligned}$$

Thinking of $(2e/\sigma^2 n)^{2n}$ as a continuous function of n, a simple derivative exercise shows that it is maximized for $n = 2/\sigma^2$, with value e^{4/σ^2}. Therefore, we can upper bound the series in the expression above as follows:

$$\begin{aligned}
\sum_{n=1}^{\infty}\left(\frac{2e}{\sigma^2 n}\right)^{2n} &= \sum_{n=1}^{\lfloor\sqrt{2}2e/\sigma^2\rfloor}\left(\frac{2e}{\sigma^2 n}\right)^{2n} + \sum_{n=\lceil\sqrt{2}2e/\sigma^2\rceil}^{\infty}\left(\frac{2e}{\sigma^2 n}\right)^{2n} \\
&\leq \frac{\sqrt{2}2e}{\sigma^2}e^{4/\sigma^2} + \sum_{n=\lceil\sqrt{2}2e/\sigma^2\rceil}^{\infty}\left(\frac{1}{2}\right)^n \leq \frac{\sqrt{2}2e}{\sigma^2}e^{4/\sigma^2} + 1.
\end{aligned}$$

From which we get

$$B \leq \frac{1}{4} + \frac{4}{\pi\sigma^2}\left(1 + \frac{\sqrt{2}e}{\sigma^2}e^{4/\sigma^2}\right).$$

■

By Theorem 1, it follows that:

Corollary 3 *Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X} \times \{0,1\}$. Let $\sigma > 0$ be a variance parameter and let ϕ_{erf} be as defined in Equation (5). Denote B as in Lemma 4, and let*

$$m = \Omega\left(\frac{B\log(1/\delta)}{\epsilon^2}\right) \ .$$

Then the algorithm described in Section 2 agnostically learns $H_{\phi_{\mathrm{erf}}}$ in time $O(\kappa m^2)$.

The next lemma connects the error with respect to $\phi_{\mathrm{erf}}(\cdot)$ and the margin error as defined in Equation (3).

Lemma 5 *For any $\mu > 0$ and $\epsilon > 0$, let $\sigma = \mu/\sqrt{\log(1/2\epsilon)}$ and let ϕ_{erf} be the corresponding erf transfer function. Then,*

$$\mathbb{E}[|\phi_{\mathrm{erf}}(\langle \mathbf{w}, \mathbf{x}\rangle) - y|] \ \leq \ \mathrm{err}_\mu(\mathbf{w}) + \epsilon \ .$$

Proof To satisfy the inequality in the lemma, it suffices to choose a value of σ so that for any $a \in [-1,1]$ such that $|a| \geq \mu$, it holds that $|\phi_{\mathrm{erf}}(a) - \phi_{0-1}(a)| \leq \epsilon$. This happens if σ^2 fulfills the inequality

$$1 - 2\epsilon \leq \mathrm{erf}\left(\frac{\mu}{\sigma}\right).$$

Using a standard inequality for the error function (see [2]), this will follow if we require that

$$\epsilon \geq \frac{e^{(-\mu/\sigma)^2}}{\sqrt{\pi}\left(\frac{\mu}{\sigma} + \sqrt{\left(\frac{\mu^2}{\sigma^2}\right)^2 + \frac{4}{\pi}}\right)}.$$

A stronger requirement is that $\epsilon \geq e^{-(\mu/\sigma)^2}/2$, which leads to

$$\sigma \leq \frac{\mu}{\sqrt{\log(1/2\epsilon)}}.$$

■

Combining the above lemma with Corollary 3 we obtain:

Theorem 4 *Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X} \times \{0,1\}$, and let μ be a margin parameter. Let $\epsilon > 0, \delta > 0$, and set*

$$B = \frac{1}{4} + \frac{4\log(1/2\epsilon)}{\pi\mu^2}\left(1 + \frac{\sqrt{2}e\log(1/2\epsilon)}{\mu^2}\exp\left(\frac{4\log(1/2\epsilon)}{\mu^2}\right)\right) ,$$
$$m = \Omega\left(\frac{B\log(1/\delta)}{\epsilon^2}\right)$$

Then, the algorithm described in Section 2 returns a hypothesis h such that with probability at least $1-\delta$, $\mathrm{err}(h) \leq \min_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \mathrm{err}_\mu(\mathbf{w}) + \epsilon$.

C Additional Proofs

C.1 Proof of Lemma 1

Our proof technique is closely related to the one in [17]. In particular, we use the same kind of approximating polynomials (based on Hermite polynomials). The main difference is that while in [17] the degree of the approximating polynomial was the dominating factor, for our algorithm the dominating factor is the size of the coefficients in the polynomial. We note that we have made no special attempt to optimize the proof or the choice of polynomials to our algorithm, and it is likely that the result below can be substantially improved. To maintain uniformity with the rest of the paper, we will assume that the half-space with which we compete passes through the origin, although the analysis below can be easily extended when we relax this assumption.

For the proof, we will need two auxiliary lemmas. The first one provides a polynomial approximation to ϕ_{0-1}, which is an L_2 approximation to ϕ_{0-1} under a Gaussian-like weighting, using *Hermite Polynomials*. The second lemma shows how to transform this L_2 approximating polynomial into a new L_1 approximating polynomial.

Lemma 6 *For any $d > 0$, there is a degree-d univariate polynomial $p_d(x) = \sum_{j=0}^{d} \beta_j x^j$ such that*

$$\int_{-\infty}^{\infty} (p_d(x) - \mathrm{sgn}(x))^2 \frac{\exp(-x^2)}{\sqrt{\pi}} dx = O\left(\frac{1}{\sqrt{d}}\right) . \qquad (10)$$

Moreover, it holds that $|\beta_j| \leq O(2^{(j+d)/2})$.

Proof Our proof closely follows that of theorem 6 in [17]. In that theorem, a certain polynomial is constructed, and it is proven there that it satisfies Equation (10). Thus, to prove the lemma it is enough to show the bound on the coefficients of that polynomial. The polynomial is defined there as

$$p_d(x) = \sum_{i=0}^{d} c_i \bar{H}_i(x),$$

where $\bar{H}_i(x) = H_i(x)/\sqrt{2^i i!}$, $H_i(x)$ is the i-th Hermite polynomial, and

$$c_i = \int_{-\infty}^{\infty} \operatorname{sgn}(x) \bar{H}_i(x) \frac{\exp(-x^2)}{\sqrt{\pi}} dx \ .$$

In the proof of theorem 6 in [17], it is shown that $|c_i| \leq C i^{-3/4}$, where $C > 0$ is an absolute constant. Letting β_j be the coefficient of x^j in $p_d(x)$, and $h_{n,j}$ be the coefficient of x^j in $H_n(x)$, we have

$$|\beta_j| = \left| \sum_{n=j}^{d} c_n \frac{h_{n,j}}{\sqrt{2^n n!}} \right| \leq C \sum_{n=j}^{d} \frac{|h_{n,j}|}{\sqrt{2^n n!}}. \tag{11}$$

Now, using a standard formula for $h_{n,j}$ (cf. [20]),

$$|h_{n,j}| = 2^j \frac{n!}{j! \left(\frac{n-j}{2}\right)!}$$

whenever $n = j \mod 2$, otherwise $h_{n,j} = 0$. Therefore, we have that for any n, j,

$$\frac{|h_{n,j}|}{\sqrt{2^n n!}} \leq 2^{j-n/2} \sqrt{\frac{n!}{(j!)^2 \left(\left(\frac{n-j}{2}\right)!\right)^2}}. \tag{12}$$

Now, we claim that $(((n-j)/2)!)^2 \geq (n-j)! 2^{j-n}$. This follows from $(((n-j)/2)!)^2$ being equal to

$$\prod_{i=0}^{\frac{n-j}{2}-1} \left(\frac{n-j-2i}{2}\right)\left(\frac{n-j-2i}{2}\right) \geq \prod_{i=0}^{\frac{n-j}{2}-1} \left(\frac{n-j-2i}{2}\right)\left(\frac{n-j-2i-1}{2}\right)$$
$$= 2^{j-n}(n-j)!.$$

Plugging this into Equation (12), we get that $|h_{n,j}|/\sqrt{2^n n!}$ is at most

$$2^{j/2}\sqrt{\frac{n!}{(j!)^2(n-j)!}} \leq 2^{j/2}\sqrt{\frac{n!}{j!(n-j)!}} = 2^{j/2}\sqrt{\binom{n}{j}} \leq 2^{j/2}2^{n/2}.$$

Plugging this in turn into Equation (11) and simplifying, the second part of the lemma follows. ■

Lemma 7 *For any positive integer d, define the polynomial $Q'_d(x) = p_d\left(\sqrt{\frac{n-3}{2}}x\right)$, where $p_d(\cdot)$ is defined in Lemma 6. Let $\mathcal{U}$ denote the uniform distribution on S^{n-1}. Then for any $\mathbf{w} \in S^{n-1}$,*

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{U}}[\left(Q'_d(\mathbf{w}\cdot\mathbf{x}) - \operatorname{sgn}(\mathbf{w}\cdot\mathbf{x})\right)^2] \leq O(1/\sqrt{d}).$$

As a result, if we define $Q_d(x) = Q'_d/2 + 1/2$, we get

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{U}}[(Q_d(\mathbf{w}\cdot\mathbf{x}) - \phi_{0-1}(\mathbf{w}\cdot\mathbf{x}))^2] \leq O(1/\sqrt{d}).$$

The first part of this lemma is identical (up to notation) to theorem 6 in [17], and we refer the reader to it for the proof. The second part is an immediate corollary.

With these lemmas at hand, we are now ready to prove the main result. Using the polynomial $Q_d(\cdot)$ from Lemma 7, we know it belongs to P_B for

$$B = \sum_{j=0}^{d} 2^j \left(\left(\sqrt{\frac{n}{2}}\right)^j \beta_j\right)^2 \leq O\left(\sum_{j=0}^{d} n^j 2^j 2^d\right) = O\left((4n)^d\right). \tag{13}$$

Now, recall by Theorem 1 that if we run our algorithm with these parameters, then the returned hypothesis $\tilde{f}$ satisfies the following with probability at least $1-\delta$:

$$\operatorname{err}(\tilde{f}) \leq \mathbb{E}[|Q_D(\langle\mathbf{w}^*,\mathbf{x}\rangle) - y|] + O\left(\sqrt{\frac{B\log(1/\delta)}{m}}\right). \tag{14}$$

Using Lemma 7, we have that

$$\left|\mathbb{E}[|Q(\langle\mathbf{w}^*,\mathbf{x}\rangle) - y|] - \mathbb{E}[|\phi_{0-1}(\langle\mathbf{w}^*,\mathbf{x}\rangle) - y|]\right| \leq \mathbb{E}[|Q(\langle\mathbf{w}^*,\mathbf{x}\rangle) - \phi_{0-1}(\langle\mathbf{w}^*,\mathbf{x}\rangle)|]$$
$$\leq \sqrt{\mathbb{E}[(Q(\langle\mathbf{w}^*,\mathbf{x}\rangle) - \phi_{0-1}(\langle\mathbf{w}^*,\mathbf{x}\rangle))^2]} \leq O(d^{-1/4}).$$

Plugging this back into Equation (14), and choosing $d = \Theta(1/\epsilon^4)$, the result follows.

C.2 Proof of Lemma 2 and Theorem 2

In order to approximate ϕ_{sig} with a polynomial, we will use the technique of *Chebyshev approximation* (cf. [21]). The idea is similar to the Hermite approximation discussed earlier, but here a different family of orthogonal polynomials is used. One can write any continuous function on $[-1,+1]$ as a *Chebyshev expansion* $\sum_{n=0}^{\infty} \alpha_n T_n(\cdot)$, where each $T_n(\cdot)$ is a particular n-th degree polynomial denoted as the n-th Chebyshev polynomial (of the first kind). These polynomials are defined as $T_0(x) = 1, T_1(x) = x$, and then recursively via $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$. For any n, $T_n(\cdot)$ is bounded in $[-1,+1]$. The coefficients in the Chebyshev expansion of ϕ_{sig} are equal to

$$\alpha_n = \frac{1+\mathbf{1}(n>0)}{\pi} \int_{x=-1}^{1} \frac{\phi_{\text{sig}}(x) T_n(x)}{\sqrt{1-x^2}} dx. \tag{15}$$

Truncating the series after some threshold $n = N$ provides an N-th degree polynomial which approximates the original function.

In order to obtain a bound on B, we need to understand the behavior of the coefficients in the Chebyshev approximation. These are determined in turn by the behavior of α_n as well as the coefficients of each Chebyshev polynomial $T_n(\cdot)$. The following two lemmas provide the necessary bounds.

Lemma 8 *For any $n > 1$, $|\alpha_n|$ in the Chebyshev expansion of ϕ_{sig} on $[-1,+1]$ is upper bounded as follows:*

$$|\alpha_n| \le \frac{2\sigma + 1/\pi}{(1+\pi\sigma)^n}.$$

Also, we have $|\alpha_0| \le 1$, $|\alpha_1| \le 2$.

Proof The coefficients α_n, $n = 1, \ldots$ in the Chebyshev series are given explicitly by

$$\alpha_n = \frac{2}{\pi} \int_{x=-1}^{1} \frac{\phi_{\text{sig}}(x) T_n(x)}{\sqrt{1-x^2}} dx. \tag{16}$$

For α_0, the same equality holds with $2/\pi$ replaced by $1/\pi$, so α_0 equals

$$\frac{1}{\pi} \int_{x=-1}^{1} \frac{\phi_{\text{sig}}(x)}{\sqrt{1-x^2}} dx,$$

which by definition of $\phi_{\text{sig}}(x)$, is at most $(1/\pi) \int_{x=-1}^{1} \left(\sqrt{1-x^2}\right)^{-1} dx = 1$. As for α_1, it equals

$$\frac{2}{\pi} \int_{x=-1}^{1} \frac{\phi_{\text{sig}}(x) x}{\sqrt{1-x^2}} dx,$$

whose absolute value is at most $(2/\pi)\int_{x=-1}^{1}\left(\sqrt{1-x^2}\right)^{-1}dx = 2$.

To evaluate the integral in Equation (16) for general n and σ, we will need to use some tools from complex analysis. The calculation follows [12], to which we refer the reader for justification of the steps and further details[2].

On the complex plane, the integral in Equation (16) can be viewed as a line integral over $[-1,+1]$. Using properties of Chebyshev polynomials, this integral can be converted into a more general complex-valued integral over an arbitrary closed curve C on the complex plane which satisfies certain regularity conditions:

$$\alpha_n = \frac{1}{\pi i}\int_C \frac{\phi_{\text{sig}}(z)dz}{\sqrt{z^2-1}(z\pm\sqrt{z^2-1})^n}dz, \tag{17}$$

where the sign in $\pm$ is chosen so that $|z\pm\sqrt{z^2-1}|>1$. In particular, for any parameter $\rho>1$, the set of points z satisfying $|z\pm\sqrt{z^2-1}|=\rho$ form an ellipse, which grows larger with ρ and with foci at $z=\pm1$ and which grows larger with ρ. Since we are free to choose C, we choose it as this ellipse while letting $\rho\to\infty$.

To understand what happens when $\rho\to\infty$, we need to characterize the singularities of $\phi_{\text{sig}}(z)$, namely the points z where $\phi_{\text{sig}}(z)$ is not well defined. Recalling that $\phi_{\text{sig}}(z)=(1+e^{-z/\sigma})^{-1}$, we see that the problematic points are $i(\pi+2\pi k)\sigma$ for any $k=\pm1,\pm2,\ldots$, where the denominator in $\phi_{\text{sig}}(z)$ equals zero. Note that this forms a discrete set of isolated points - in other words, ϕ_{sig} is a *meromorphic function.* The fact that ϕ_{sig} is 'well behaved' in this sense allows us to perform the analysis below.

The behavior of the function at its singularities is defined via the *residue* of the function at each singularity c, which equals $\lim_{z\to c}(z-c)\phi_{\text{sig}}(z)$ assuming the limit exists (in that case, the singularity is called a *simple pole*, otherwise a higher order limit might be needed). In our case, the residue for the singularity at $i\pi\sigma$ equals

$$\lim_{z\to0}\frac{z}{1+e^{-i\pi-z/\sigma}} = \lim_{z\to0}\frac{z}{1-e^{-z/\sigma}} = \lim_{z\to0}\frac{\sigma}{e^{-z/\sigma}} = \sigma,$$

where we used l'Hôpital's rule to calculate the limit. The same residue also apply to all the other singularities.

For points in the complex plane uniformly bounded away from these singularities, $|\phi_{\text{sig}}(z)|$ is bounded, and therefore it can be shown that the

[2]We note that such calculations also appear in standard textbooks on the subject, but they are usually carried under asymptotic assumptions and disregarding coefficients which are important for our purposes.

integral in Equation (17) will tend to zero as we let C become an arbitrarily large ellipse (not passing too close to any of the singularities) by taking $\rho \to \infty$. However, as ρ varies smoothly, the ellipse does cross over singularity points, and these contribute to the integral. For meromorphic functions, with a discrete set of isolated singularities, we can simply sum over all contributions, and it can be shown (see equation 10 in [12] and the subsequent discussion) that

$$\alpha_n = -2 \sum_{k=-\infty}^{\infty} \frac{r_k}{\sqrt{z_k^2 - 1}\left(z_k \pm \sqrt{z_k^2 - 1}\right)^n},$$

where z_k is the singularity point $i(\pi + 2\pi k)\sigma$ with corresponding residue r_k. Substituting the results for our chosen function, we have

$$\alpha_n = \sum_{k=-\infty}^{\infty} \frac{\sigma}{\sqrt{(i(\pi + 2\pi k)\sigma)^2 - 1}\left(i(\pi + 2\pi k)\sigma \pm \sqrt{(i(\pi + 2\pi k)\sigma)^2 - 1}\right)^n}.$$

A routine simplification leads to the following[3]:

$$\alpha_n = \sum_{k=-\infty}^{\infty} \frac{\sigma}{i^{n+1}\sqrt{((\pi + 2\pi k)\sigma)^2 + 1}\left((\pi + 2\pi k)\sigma \pm \sqrt{((\pi + 2\pi k)\sigma)^2 + 1}\right)^n}.$$

It can be verified that $\pm$ should be chosen according to $\mathbf{1}(k \geq 0)$. Therefore,

$$\begin{aligned}
|\alpha_n| &= \sum_{k=-\infty}^{\infty} \frac{\sigma}{\sqrt{((\pi + 2\pi k)\sigma)^2 + 1}\left(|\pi + 2\pi k|\sigma + \sqrt{((\pi + 2\pi k)\sigma)^2 + 1}\right)^n} \\
&\leq \sum_{k=-\infty}^{\infty} \frac{\sigma}{(|\pi + 2\pi k|\sigma + 1)^n} \leq \frac{\sigma}{(1 + \pi\sigma)^n} + 2\sum_{k=1}^{\infty} \frac{\sigma}{(1 + \pi(1 + 2k)\sigma)^n} \\
&\leq \frac{\sigma}{(1 + \pi\sigma)^n} + \int_{k=0}^{\infty} \frac{2\sigma}{(1 + \pi(1 + 2k)\sigma)^n} dk
\end{aligned}$$

[3]On first look, it might appear that α_n takes imaginary values for even n, due to the i^{n+1} factor, despite α_n being equal to a real-valued integral. However, it can be shown that $\alpha_n = 0$ for even n. This additional analysis can also be used to slightly tighten our final results in terms of constants in the exponent, but it was not included for simplicity.

Solving the integral and simplifying gives us

$$|\alpha_n| \leq \frac{1}{(1+\pi\sigma)^n}\left(\sigma + \frac{1+\pi\sigma}{\pi(n-1)}\right).$$

Since $n \geq 2$, the result in the lemma follows. ■

Lemma 9 *For any non-negative integer n and $j = 0, 1, \ldots, n$, let $t_{n,j}$ be the coefficient of x^j in $T_n(x)$. Then $t_{n,j} = 0$ for any j with a different parity than n, and for any $j > 0$,*

$$|t_{n,j}| \leq \frac{e^{n+j}}{\sqrt{2\pi}}$$

Proof The fact that $t_{n,j} = 0$ for j, n with different parities, and $|t_{n,0}| \leq 1$ is standard. Using an explicit formula from the literature (see [21], pg. 24), as well as Stirling approximation, we have that

$$\begin{aligned}
|t_{n,j}| &= 2^{n-(n-j)-1}\frac{n}{n-\frac{n-j}{2}}\binom{n-\frac{n-j}{2}}{\frac{n-j}{2}} = \frac{2^j n}{n+j}\frac{\left(\frac{n+j}{2}\right)!}{\left(\frac{n-j}{2}\right)!j!} \\
&\leq \frac{2^j n}{j!(n+j)}\left(\frac{n+j}{2}\right)^j = \frac{n(n+j)^j}{(n+j)j!} \leq \frac{n(n+j)^j}{(n+j)\sqrt{2\pi j}(j/e)^j} \\
&= \frac{ne^j}{(n+j)\sqrt{2\pi j}}\left(1+\frac{n}{j}\right)^j \\
&\leq \frac{ne^j}{(n+j)\sqrt{2\pi j}}e^n.
\end{aligned}$$

from which the lemma follows. ■

We are now in a position to prove a bound on B. As discussed earlier, $\phi_{\text{sig}}(x)$ in the domain $[-1, +1]$ equals the expansion $\sum_{n=0}^{\infty}\alpha_n T_x$. The error resulting from truncating the Chebyshev expanding at index N, for any $x \in [-1, +1]$, equals

$$\left|\phi_{\text{sig}}(x) - \sum_{n=0}^{N}\alpha_n T_n(x)\right| = \left|\sum_{n=N+1}^{\infty}\alpha_n T_n(x)\right| \leq \sum_{n=N+1}^{\infty}|\alpha_n|,$$

where in the last transition we used the fact that $|T_n(x)| \leq 1$. Using Lemma 8 and assuming $N > 0$, this is at most

$$\sum_{n=N+1}^{\infty}\frac{2\sigma + 1/\pi}{(1+\pi\sigma)^n} = \frac{2\sigma+1/\pi}{\pi\sigma(1+\pi\sigma)^N}.$$

In order to achieve accuracy of less than ϵ in the approximation, we need to equate this to ϵ and solve for N, resulting in

$$N \geq \log_{1+\pi\sigma}\left(\frac{2\sigma + 1/\pi}{\pi\sigma\epsilon}\right). \tag{18}$$

The series left after truncation is $\sum_{n=0}^{N} \alpha_n T_n(x)$, which we can write as $\sum_{j=0}^{N} \beta_j x^j$. Using Lemma 8 and Lemma 9, the absolute value of the coefficient β_j for $j > 1$ can be upper bounded by

$$\begin{aligned}
&\sum_{n=j..N, n=j \mod 2} |a_n||t_{n,j}| \leq \sum_{n=j..N, n=j \mod 2} \frac{2\sigma + 1/\pi}{(1+\pi\sigma)^n} \frac{e^{n+j}}{\sqrt{2\pi}} \\
&= \frac{(2\sigma + 1/\pi)e^j}{\sqrt{2\pi}} \sum_{n=j..N, n=j \mod 2} \left(\frac{e}{1+\pi\sigma}\right)^n \\
&= \frac{(2\sigma + 1/\pi)e^j}{\sqrt{2\pi}} \left(\frac{e}{1+\pi\sigma}\right)^j \sum_{n=0}^{\lfloor \frac{N-j}{2} \rfloor} \left(\frac{e}{1+\pi\sigma}\right)^{2n} \\
&\leq \frac{(2\sigma + 1/\pi)e^j}{\sqrt{2\pi}} \left(\frac{e}{1+\pi\sigma}\right)^j \frac{(e/(1+\pi\sigma))^{N-j+2} - 1}{(e/(1+\pi\sigma))^2 - 1}.
\end{aligned}$$

Since we assume $\sigma \leq 1/4$, we have in particular $e > 1 + \pi\sigma$, so we can upper bound the expression above by dropping the 1 in the numerator, to get

$$\frac{2\sigma + 1/\pi}{\sqrt{2\pi}((e/(1+\pi\sigma))^2 - 1)} \left(\frac{e}{1+\pi\sigma}\right)^{N+2} e^j.$$

The cases β_0, β_1 need to be treated separately, due to the different form of the bounds on α_0, α_1. Repeating a similar analysis (using the actual values of $t_{n,1}, t_{n,0}$ for any n), we get

$$\begin{aligned}
\beta_0 &\leq 1 + \frac{1}{\pi} + \frac{1}{2\pi^2\sigma} \\
\beta_1 &\leq 2 + \frac{3(2\sigma + 1/\pi)(1+\pi\sigma)}{4\pi^2\sigma^2}.
\end{aligned}$$

Now that we got a bound on the β_j, we can plug it into the bound on

B, and get

$$B = \sum_{j=0}^{N} 2^j \beta_j^2 \leq \beta_0^2 + 2\beta_1^2 + \sum_{j=2}^{N} \left(\tfrac{2\sigma+1/\pi}{\sqrt{2\pi}((e/(1+\pi\sigma))^2-1)}\right)^2 \left(\tfrac{e}{1+\pi\sigma}\right)^{2N+4} e^{2j}$$

$$\leq \beta_0^2 + 2\beta_1^2 + \left(\tfrac{2\sigma+1/\pi}{\sqrt{2\pi}((e/(1+\pi\sigma))^2-1)}\right)^2 \left(\tfrac{e}{1+\pi\sigma}\right)^{2N+4} \tfrac{e^{2N+1}}{e^2-1}$$

$$= \beta_0^2 + 2\beta_1^2 + \tfrac{(2\sigma+1/\pi)^2 e^5}{(e^2-1)2\pi((e/(1+\pi\sigma))^2-1)^2(1+\pi\sigma)^4} \left(\tfrac{e^2}{1+\pi\sigma}\right)^{2N}.$$

To make the expression more readable, we make the (rather arbitrary) assumption that $\sigma \leq 1/4$. In that case, it is not difficult to show that we can upper bound the above by

$$10 + \frac{1}{12\sigma^4} + \frac{3}{2}\left(\frac{e^2}{1+\pi\sigma}\right)^{2N} \leq 10 + \frac{1}{12\sigma^4} + \frac{3}{2}e^{4N}.$$

Combining this with Equation (18), we get the result stated in the Lemma 2.

Proof of Theorem 2 We first need to understand, for given ϵ, μ, what σ in ϕ_{sig} we need to pick. It is easy to verify that we need to pick σ such that $\epsilon \geq 1/(1+e^{\mu/\sigma})$, or $\sigma \leq \mu/(\log(1/\epsilon - 1)$. For simplicity, we will pick the somewhat smaller value $\sigma = \mu/(\log(1/\epsilon))$. Using Lemma 2, and the assumption $\sigma \leq 1/4$, we get that the appropriate value of B is

$$10+\frac{1}{12\sigma^4}+\frac{3}{2}\exp\left(4\log_{1+\pi\sigma}\left(\tfrac{2\sigma+1/\pi}{\pi\sigma\epsilon}\right)\right) \leq 10+\frac{1}{12\sigma^4}+\frac{3}{2}\exp\left(4\frac{\log\left(\frac{1/2+1/\pi}{\pi\sigma\epsilon}\right)}{\log(1+\pi\sigma)}\right)$$

Concentrating on the exponent, we have by the inequality $\log(1+x) \geq x - x^2/2$ for $x \geq 0$ and the assumption $\sigma \leq 1/4$ that it is at most

$$4\frac{\log\left(\frac{1/2+1/\pi}{\pi\sigma\epsilon}\right)}{\log(1+\pi\sigma)} \leq 4\frac{\log\left(\frac{1/2+1/\pi}{\pi\sigma\epsilon}\right)}{\pi\sigma(1-\pi\sigma/2)} \leq 4\frac{\log\left(\frac{1/2+1/\pi}{\pi\sigma\epsilon}\right)}{(1-\pi/8)\pi\sigma} \leq 4\frac{\log\left(\frac{1}{\pi\sigma\epsilon}\right)}{(1-\pi/8)\pi\sigma}$$
$$\leq \frac{2.1\log\left(\frac{1}{\pi\sigma\epsilon}\right)}{\sigma}$$

Plugging the chosen value $\sigma = \mu/(\log(1/\epsilon))$ and simplifying, the theorem follows.