RKHS Weightings of Functions

Gabriel Dubé, Mario Marchand

Journal of Machine Learning Research 23 (2022) 1-50 Submitted 1/21; Revised 5/22; Published 9/22

RKHS Weightings of Functions

Gabriel Dubé GADUB44@ULAVAL.CA
Département d'informatique et de génie logiciel
Université Laval

Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département d'informatique et de génie logiciel
Université Laval

Editor: TBD

Abstract

We examine a new general machine learning model for binary classification by considering the expected *weighted* output of a parameterized predictor, where the weighting belongs to an RKHS of functions over the parameters. This weighting can be learned using various algorithms. One of them is Stochastic Functional Gradient Descent (SFGD), which iteratively samples a parameter and some training data, and calculates an approximation of the functional gradient of the loss. Using the stability properties of the algorithm, we show that convergence is guaranteed, under mild assumptions, with rate $\mathcal{O}(1/\sqrt{m})$ on the number of examples needed for learning. Further theoretical analysis, based on the Rademacher complexity of the proposed class of predictors, provides a similar bound on the generalization error. We also present three alternate learning algorithms, and a procedure for pruning the model using the Lasso. We prove an error bound for the resulting sparse predictor. Finally, we run experiments using simple instantiations of the model to showcase its usability, and compare the learning algorithms among one another, and to the state of the art.

Keywords: Reproducing kernel Hilbert space, functional gradient, stochastic gradient descent, random features, kernel methods

1. Introduction

The simplest but not least important group of machine learning models consists of linear predictors. Linear predictors are easy to optimize, and simple to analyze, i.e. derive theoretical guarantees of their prediction performance. Kernel methods, such as Support Vector Machines (SVMs) (Steinwart and Christmann, 2008), and Kernel Ridge Regression (Vovk, 2013), build upon linear predictors using the kernel trick to easily produce highly complex predictors. These methods are powerful, but have an algorithmic complexity which is cubic in the amount of training data, meaning that they scale poorly to Big Data.

A popular way of circumventing this weakness of kernel methods is to approximate the kernel via random features (called random feature methods, see e.g. Rahimi and Recht (2008, 2009)). Indeed, it can be shown that any kernel can be expressed as the expected value of some simple random variable (Hein and Bousquet, 2004). A large enough sample will yield high fidelity, and the learning algorithm will only have linear complexity in the size of the dataset. However, there are downsides to this approach. The approximation might require

a prohibitively large sample, and the algorithmic complexity is a function of this sample size, leading to impractical learning times when a high accuracy is needed. In general, random feature methods also work poorly when the input space is high-dimensional. On the theoretical side, there is also the fact that the predictors obtained from approximating the kernel in this way do not actually belong to the intended class (the RKHS of the kernel), but only to the approximating class, leading to an approximation error between the classes.

In this paper, we introduce a new model which aims at addressing some of these shortcomings of random feature methods. Most importantly, the model does not need to be approximated, since it can be calculated exactly (under some conditions). This removes any loss of accuracy due to an imprecise approximation, and enables working directly within the target class of predictors, rather than an approximation class. It also changes the geometry of the predictor space in interesting ways, such as allowing for a simple, natural pruning method.

Perhaps the most interesting aspect of the model we propose is its generality. It is a highly flexible model, in that it can be instantiated using a variety of high-level parameters. (More precisely, an instantiation is a tuple $(\mathcal{W}, \mathcal{K}, p, \phi)$, where $\mathcal{W}$ is a parameter space, $\mathcal{K}$ is a kernel, p is a probability distribution on $\mathcal{W}$ and ϕ is a base predictor. Any combination of these four parameters will yield a different space of predictors.) In this paper, we only scratch the surface of what this model can accomplish. Leveraging this flexibility appears to be a significant challenge, however, due to a requirement for calculating complex integrals. It is the most important aspect of the model to be further investigated.

We provide four different algorithms for learning the model. The most theoretically interesting is called Stochastic Functional Gradient Descent (SFGD). It is an iterative algorithm with convergence rate of $\mathcal{O}(1/\sqrt{m})$ on the number of examples needed for learning (under mild assumptions). We also provide a bound on the generalization error based on the Rademacher complexity of the proposed class of predictors, valid for any learning algorithm. These theoretical results are obtained using less stringent assumptions than Rahimi and Recht (2009), which is an improvement over a longstanding state-of-the-art result.

Finally, we run experiments using simple instantiations of the model to showcase its usability. Prediction accuracy is equal or better than Rahimi and Recht (2009), and ties the state of the art on some datasets. These results clearly show that the model and algorithm are viable machine learning tools, though further work is needed to extract the full potential of the model.

The paper is structured as follows. We begin in Section 2 by recalling basic notions and notations of functional analysis and machine learning. We summarize the work of Rahimi and Recht (2009) in Section 3. Next, we define the model and its theoretical properties in Section 4, including some examples of model instantiations in Section 4.5. We present four learning algorithms in Section 5. We describe in Section 6 how to prune the model, and prove bounds on the error of the resulting sparse predictor. We prove theoretical guarantees in Section 7: a bound on the generalization error in Section 7.1, based on the Rademacher complexity of the class of predictors, and a bound on the rate of convergence of Algorithm 1 in Section 7.2, using the stability properties of the algorithm. We also compare these results to Rahimi and Recht (2009) in Section 7.3. Finally, we present experimental results in Section 8, demonstrating the soundness of the model and learning algorithms.

2. Basic notions

We begin in Section 2.1 by recalling the main notions and definitions of supervised machine learning, and establish the notation to be used in the remainder of the paper. Then, we recall core definitions of functional analysis in Section 2.2, and give a summary description of reproducing kernel Hilbert spaces in Section 2.3. In Section 2.4, we present the notion of the functional gradient, which is crucial to the working of the learning algorithms.

2.1 Machine learning

Consider some **instance space** $\mathcal{X}$, and a **label space** $\mathcal{Y} \subseteq \mathbb{R}$. A **predictor** or **hypothesis** is a function $h : \mathcal{X} \to \mathcal{Y}$ (we will equivalently write $h \in \mathcal{Y}^{\mathcal{X}}$, where $\mathcal{Y}^{\mathcal{X}}$ is the space of all functions from $\mathcal{X}$ to $\mathcal{Y}$). We are in the presence of a **classification problem** when $\mathcal{Y}$ is a discrete, usually finite set. The most basic and important type of classification problem, which we focus on in this paper, is **binary classification**, when $\mathcal{Y}$ contains only two labels. In that case, we can always write $\mathcal{Y} = \{-1, 1\}$ to simplify equations.

We call a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ an **example**. A **sample** or **training set** is a sequence or set of examples $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m \subset (\mathcal{X} \times \mathcal{Y})^m$. A learning algorithm $\mathcal{A}$ takes some sample $\mathcal{S}$ as input, and outputs a predictor $\mathcal{A}(\mathcal{S}) \in \mathbb{R}^{\mathcal{X}}$. The goal is for this predictor to satisfy some notion of accuracy, and the algorithm will achieve this by optimizing some criteria over the training set. To this end, we define a **loss** to be a function $\ell : \mathbb{R} \times \mathcal{Y} \to [0, \infty)$. Given some predictor h, we can calculate the loss value on example (x, y) as $\ell(h(x), y)$. This value can be seen as a penalty for predicting $h(x)$ when the correct value is y. Common examples are the **0-1 loss** $\ell(h(x), y) := \mathbb{1}[\operatorname{sign}(h(x)) \neq y]$ and the **square loss**, defined as $\ell(h(x), y) := (h(x) - y)^2$.

We define the **empirical risk** of the predictor on the sample $\mathcal{S}$ as:

$$\mathcal{L}_{\mathcal{S}}(h) := \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i). \tag{1}$$

Furthermore, we suppose that all examples have been sampled i.i.d. from some data generating probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. This allows us to define the **true risk** of the predictor h as:

$$\mathcal{L}_{\mathcal{D}}(h) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} [\ell(h(x), y)]. \tag{2}$$

In the binary classification setting, we denote the **classification error** of h as:

$$\mathcal{L}^{01}_{\mathcal{D}}(h) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathbb{1}[\operatorname{sign}(h(x)) \neq y], \tag{3}$$

and its **empirical classification error** as:

$$\mathcal{L}^{01}_{\mathcal{S}}(h) := \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[\operatorname{sign}(h(x_i)) \neq y_i]. \tag{4}$$

2.2 Functional analysis

We denote **scalar products** (or **inner products**) on a linear space X as $\langle \cdot, \cdot \rangle : X \times X \to \mathbb{R}$. For some operator $\Lambda : X \to Y$, we will often write Λx instead of $\Lambda(x)$. The **norm** of a linear

operator is defined as:

$$\|\Lambda\| := \sup_{x \in X : \|x\|_X = 1} \|\Lambda x\|_Y. \tag{5}$$

The operator is said to be **bounded** if its norm is finite. Note also that a linear operator is continuous if and only if it is bounded. In the special case where $Y = \mathbb{R}$, an operator $\Lambda : X \to \mathbb{R}$ is called a **functional**. The following theorem applies to bounded (i.e. continuous) linear functionals.

Theorem 1 (Riesz representer theorem) *Consider a Hilbert space $\mathcal{H}$, and a bounded linear functional $L : \mathcal{H} \to \mathbb{R}$. There exists a unique $h \in \mathcal{H}$ such that $\forall \alpha \in \mathcal{H}, L\alpha = \langle \alpha, h \rangle_{\mathcal{H}}$. This element h is the representative of the functional L.*

This theorem simply states that bounded linear functionals in a Hilbert space can be written as a scalar product. This is a surprisingly powerful result. It forms the basis for understanding reproducing kernel Hilbert spaces, as we will see in the next section.

(See e.g. Atkinson and Han (2005) for more details on functional analysis.)

2.3 Reproducing kernel Hilbert spaces

Formally, a **reproducing kernel Hilbert space** (RKHS) over some vector space $\mathcal{W}$ is a Hilbert space $\mathcal{H} \subset \mathbb{R}^{\mathcal{W}}$ of real-valued functions of $\mathcal{W}$ which has the property that the evaluation functionals are continuous. This means that for any $w \in \mathcal{W}$, the functional $L_w : \mathcal{H} \to \mathbb{R}$ defined by $L_w(\alpha) := \alpha(w)$ is continuous. By the Riesz representer theorem, there exists for all $w \in \mathcal{W}$ a unique element $\mathcal{K}_w \in \mathcal{H}$ such that

$$\alpha(w) = L_w(\alpha) = \langle \alpha, \mathcal{K}_w \rangle_{\mathcal{H}}, \qquad \forall \alpha \in \mathcal{H}. \tag{6}$$

Because $\mathcal{K}_w$ is itself a function of $\mathcal{W}$, we can apply it to some other $u \in \mathcal{W}$, and reapply the Riesz representer theorem, which gives us:

$$\mathcal{K}_w(u) = L_u(\mathcal{K}_w) = \langle \mathcal{K}_w, \mathcal{K}_u \rangle_{\mathcal{H}}. \tag{7}$$

We define the **reproducing kernel** of the RKHS $\mathcal{H}$ as the function:

$$\mathcal{K}(w, u) := \mathcal{K}_w(u) = \langle \mathcal{K}_w, \mathcal{K}_u \rangle_{\mathcal{H}}. \tag{8}$$

As we can see from the definition of the kernel, the element $\mathcal{K}_w$ exactly corresponds to the function $\mathcal{K}(w, \cdot) : \mathcal{W} \to \mathbb{R}$. For clarity, we will use this new notation from now on.

The reproducing kernel calculates the scalar product between two elements $w, u \in \mathcal{W}$ following the embedding $(w, u) \mapsto (\mathcal{K}(w, \cdot), \mathcal{K}(u, \cdot))$ into $\mathcal{H}$. Importantly, it is not necessary to calculate this embedding explicitely, which would be impossible when $\mathcal{H}$ is infinite-dimensional. In fact, the Moore–Aronszajn theorem (see e.g. Berlinet and Thomas-Agnan (2011)) states that any positive definite symmetric (PDS) function $\mathcal{K} : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$ is the reproducing kernel of an RKHS $\mathcal{H}$ of real-valued functions of $\mathcal{W}$, and every function $\alpha \in \mathcal{H}$ can be written as a sum or a convergent series of the form:

$$\alpha = \sum_{i=1}^{\infty} a_i \mathcal{K}(w_i, \cdot), \tag{9}$$

for some real coefficients $(a_i)_{i=1}^{\infty}$ and some $\{w_i\}_{i=1}^{\infty} \subset \mathcal{W}$. Finally, given this representation for the elements in $\mathcal{H}$, the scalar product can also be written as a sum (or series). For some $\alpha, \beta \in \mathcal{H}$ that can be written as $\alpha = \sum_{i=1}^{\infty} a_i \mathcal{K}(w_i, \cdot)$ and $\beta = \sum_{i=1}^{\infty} b_i \mathcal{K}(u_i, \cdot)$, the scalar product is equal to:

$$\langle \alpha, \beta \rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} a_i b_j \mathcal{K}(w_i, u_j). \tag{10}$$

2.4 Functional gradient

The usual notion of gradient (crucial for gradient descent algorithms) can be generalized to functionals. Consider some normed spaces X and Y. An operator $\Lambda : X \to Y$ is **Fréchet differentiable** at $x \in X$ if and only if there exists a bounded linear operator $A : X \to Y$ such that:

$$\Lambda(x + z) = \Lambda x + Az + o(\|z\|), \qquad z \to 0. \tag{11}$$

See Definition 5.3.1 of Atkinson and Han (2005). We call A the **Fréchet derivative** of Λ at x. If Λ is a bounded linear operator, then it is its own Fréchet derivative.

Notice that when Λ is a functional, in other words when $Y = \mathbb{R}$, the Fréchet derivative A is a bounded linear functional. If X is a Hilbert space, then the Riesz representer theorem applies, allowing us to write A as a scalar product of the form:

$$Az = \langle z, \nabla_x \Lambda x \rangle_X , \tag{12}$$

for some $\nabla_x \Lambda x \in X$. Equation (11) becomes:

$$\Lambda(x + z) = \Lambda x + \langle z, \nabla_x \Lambda x \rangle_X + o(\|z\|), \qquad z \to 0. \tag{13}$$

We call $\nabla_x \Lambda x$ the **functional gradient** of Λ at x.

The functional gradient is a well-defined object, but it often cannot be used in practice, due to the infinite complexity of representing arbitrary functions. A notable exception is when the input space X is an RKHS, in which case the functional gradient sometimes admits an analytical form, which can be calculated exactly in finite time. The following lemmas present the functional gradient of the evaluation functionals and the squared norm in an RKHS.

Lemma 2 *Consider an RKHS $\mathcal{H}$ of kernel $\mathcal{K}$ over a vector space $\mathcal{W}$. The functional gradient of the evaluation functional $L_w(\alpha) := \alpha(w)$ for some fixed point $w \in \mathcal{W}$ is:*

$$\boldsymbol{\nabla}_\alpha L_w(\alpha) = \boldsymbol{\nabla}_\alpha \alpha(w) = \mathcal{K}(w, \cdot).$$

Proof Notice that $\alpha(w) = L_w(\alpha) = \langle \alpha, \mathcal{K}(w, \cdot) \rangle$ is a bounded linear functional. As mentioned above, a bounded linear functional is its own Fréchet derivative. By comparing to Equation (12), we can immediately conclude that the functional gradient of the evaluation functional must be its representer $\mathcal{K}(w, \cdot)$. ■

Lemma 3 *Consider an RKHS $\mathcal{H}$ and $\alpha \in \mathcal{H}$. Then $\boldsymbol{\nabla}_\alpha \|\alpha\|_{\mathcal{H}}^2 = 2\alpha$.*

Proof Let $\varepsilon > 0$ and $\beta \in \mathcal{H}$, with $\|\beta\|_{\mathcal{H}} = 1$. We have:

$$\begin{aligned}\|\alpha + \varepsilon\beta\|_{\mathcal{H}}^2 &= \langle \alpha + \varepsilon\beta, \alpha + \varepsilon\beta \rangle_{\mathcal{H}} \\ &= \langle \alpha, \alpha \rangle_{\mathcal{H}} + 2\langle \alpha, \varepsilon\beta \rangle_{\mathcal{H}} + \langle \varepsilon\beta, \varepsilon\beta \rangle_{\mathcal{H}} \\ &= \|\alpha\|_{\mathcal{H}}^2 + \langle 2\alpha, \varepsilon\beta \rangle_{\mathcal{H}} + \varepsilon^2 \|\beta\|_{\mathcal{H}}^2 \\ &= \|\alpha\|_{\mathcal{H}}^2 + \langle 2\alpha, \varepsilon\beta \rangle_{\mathcal{H}} + \varepsilon^2.\end{aligned}$$

Comparing this to Equation (13), we can conclude that the functional gradient is 2α. ■

3. Previous work

The model introduced in the next section of this paper most closely resembles the work of Rahimi and Recht (2009), a longstanding state-of-the-art in the field of random feature methods. Given a parameter space $\mathcal{W}$ and feature function $\phi(w, x)$, as well as a probability distribution p on $\mathcal{W}$, Rahimi and Recht (2009) define the following class (cf. Equation (6) of Rahimi and Recht (2009)):

$$\mathcal{F}_p := \left\{ x \mapsto \mathop{\mathbb{E}}_{w \sim p} [\alpha(w)\phi(w, x)] \;\middle|\; \|\alpha\|_{L^\infty(\mathcal{W})} \leq C \right\}, \tag{14}$$

and requires $\sup_{w,x} |\phi(w, x)| \leq 1$. Since the functions in $\mathcal{F}_p$ can't be represented or calculated exactly, Rahimi and Recht (2009) also define the following approximation class:

$$\hat{\mathcal{F}}_w := \left\{ x \mapsto \frac{1}{T} \sum_{t=1}^{T} a_t \phi(w_t, x) \;\middle|\; \forall t, |a_t| \leq C \right\}, \tag{15}$$

for some $(w_1, \ldots, w_T)$ which have been randomly sampled according to distribution p. Further assuming that the loss is ρ-Lipschitz, the main result of Rahimi and Recht (2009) states that the output $\hat{f}$ of their Algorithm 1 is such that:

$$\mathcal{L}_{\mathcal{D}}(\hat{f}) - \min_{f \in \mathcal{F}_p} \mathcal{L}_{\mathcal{D}}(f) \leq \mathcal{O}\left(\left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{T}} \right) \rho C \sqrt{\log \frac{1}{\delta}} \right), \tag{16}$$

with probability $1 - 2\delta$ on the choice of training set and parameters $(w_1, \ldots, w_T)$. We compare this to our results in Section 7.3.

4. RKHS weightings of functions

In this section, we introduce a new mathematical model. It takes the form of an operator Λ, which takes as input an RKHS function α, and outputs a predictor $\Lambda\alpha$. The model and hypothesis class are defined in Section 4.1. Because of the importance of the operator norm in the theoretical guarantees of Section 7, we examine in detail the operator norm of Λ in Section 4.2. Then, we define all the assumptions that we make in Section 4.3. We calculate the functional gradient of the loss functionals in Section 4.4. Finally, we give explicit examples of instantiations of the model, with formulas for calculating the output, in Section 4.5.

4.1 Model definition

Let $\mathcal{W}$ be a parameter space, and $\mathcal{X}$ the instance space. Let $\phi : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$ be some function, which we will refer to as the **base predictor**. We can for example take $\mathcal{W} = \mathcal{X} = \mathbb{R}^n$, and use $\phi(w, x) = \text{sign}(\langle w, x \rangle)$ (see the examples of Section 4.5). Let p be a probability distribution over $\mathcal{W}$. We define the operator $\Lambda : \mathbb{R}^{\mathcal{W}} \to \mathbb{R}^{\mathcal{X}}$ as:

$$\Lambda\alpha := \mathop{\mathbb{E}}_{w \sim p} [\alpha(w)\phi(w, \cdot)]. \tag{17}$$

In other words, we have the following for all $x \in \mathcal{X}$:

$$\Lambda\alpha(x) := \mathop{\mathbb{E}}_{w \sim p} [\alpha(w)\phi(w, x)]. \tag{18}$$

This operator transforms a weight function α over $\mathcal{W}$ into a predictor $\Lambda\alpha$ over $\mathcal{X}$. However, calculating the prediction for a given x requires solving the expectation of Equation (18). To ensure that this model is well-defined (the expectation must always exist and be finite), we need to constrain the space of allowable weight functions (and add some further assumptions, which we will explore further below). We will pay special interest, though not exclusive, to the case where Λ operates on an RKHS.

Let's therefore consider $\mathcal{K} : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$ a positive definite symmetric (PDS) kernel over $\mathcal{W}$. Denote $\mathcal{H}$ the RKHS of kernel $\mathcal{K}$. We call the tuple $(\mathcal{W}, \mathcal{K}, p, \phi)$ an **instantiation** of the model, and define the following class of predictors:

$$\Lambda\mathcal{H} := \left\{ \Lambda\alpha \;\middle|\; \alpha \in \mathcal{H} \right\} = \left\{ x \mapsto \Lambda\alpha(x) := \mathop{\mathbb{E}}_{w \sim p} [\alpha(w)\phi(w, x)] \;\middle|\; \alpha \in \mathcal{H} \right\}. \tag{19}$$

It will be useful for theoretical guarantees to limit the norm of the weight function $\alpha \in \mathcal{H}$. Therefore, for some constant $B > 0$, we also define the set:

$$\mathcal{H}_B := \left\{ \alpha \in \mathcal{H} \;\middle|\; \|\alpha\|_{\mathcal{H}} \leq B \right\}, \tag{20}$$

and the following class of predictors:

$$\Lambda\mathcal{H}_B := \left\{ \Lambda\alpha \;\middle|\; \alpha \in \mathcal{H}, \|\alpha\|_{\mathcal{H}} \leq B \right\}. \tag{21}$$

For practical, computationally viable purposes, the weight function $\alpha \in \mathcal{H}$ will be expressed as a finite sum:

$$\alpha = \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot), \tag{22}$$

for some real coefficients $(a_t)_{t \geq 1}$ and $\{w_t\}_{t \geq 1} \subset \mathcal{W}$. (In practice, $\{w_t\}_{t \geq 1}$ will be sampled from distribution p, and the coefficients $(a_t)_{t \geq 1}$ will be learned.) The output of the model can be rewritten as:

$$\begin{aligned} \Lambda\alpha(x) &:= \mathop{\mathbb{E}}_{w \sim p} [\alpha(w)\phi(w, x)] \\ &= \mathop{\mathbb{E}}_{w \sim p} \left[\sum_{t=1}^{T} a_t \mathcal{K}(w_t, w)\phi(w, x) \right] \\ &= \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w \sim p} [\mathcal{K}(w_t, w)\phi(w, x)]. \end{aligned} \tag{23}$$

To calculate this prediction exactly, the expectation:

$$\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] \tag{24}$$

must admit an analytical form for all $u \in \mathcal{W}$ and $x \in \mathcal{X}$. We give examples in Section 4.5.

In the next section, we explore under what conditions the operator Λ is continuous by upper bounding its norm. This will ensure that the model is well-defined, and give us useful results for deriving our theoretical guarantees in Section 7. Indeed, it is important that finite movements in weight function space have finite effect in predictor space, so that learning algorithms can converge.

4.2 Norm of the operator

A crucial quantity, which appears in most guarantees that we show in Section 7, is the operator norm of Λ. The following theorem upper bounds that norm.

Theorem 4 *Consider an instance space $\mathcal{X}$, a parameter space $\mathcal{W}$, a function $\phi : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$, an RKHS $\mathcal{H} \subset \mathbb{R}^{\mathcal{W}}$ of kernel $\mathcal{K}$, a distribution p over $\mathcal{W}$ and the operator Λ defined by:*

$$\Lambda\alpha := \mathop{\mathbb{E}}_{w\sim p}[\alpha(w)\phi(w,\cdot)].$$

Consider the constant defined by:

$$\kappa := \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}}^2\right]} = \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w,w)\phi(w,x)^2]}. \tag{25}$$

Suppose that κ is finite. Then $\Lambda\alpha \in L^\infty(\mathcal{X})$ for all $\alpha \in \mathcal{H}$. Furthermore, the constant:

$$\theta := \sup_{x\in\mathcal{X}} \left\| \mathop{\mathbb{E}}_{w\sim p}[\phi(w,x)\mathcal{K}(w,\cdot)] \right\|_{\mathcal{H}} = \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\mathop{\mathbb{E}}_{u\sim p}[\mathcal{K}(u,w)\phi(u,x)\phi(w,x)]} \tag{26}$$

is well-defined and we have:

$$\|\Lambda\| := \|\Lambda\|_{\mathcal{H}\to L^\infty(\mathcal{X})} \le \theta \le \kappa. \tag{27}$$

Proof The ultimate goal is to show that $\|\Lambda\| \le \theta \le \kappa$. This can be achieved by showing that $\|\Lambda\alpha\|_\infty \le \theta\|\alpha\|_{\mathcal{H}} \le \kappa\|\alpha\|_{\mathcal{H}}$ for any $\alpha \in \mathcal{H}$. To do this, we show that $|\Lambda\alpha(x)| \le \theta \le \kappa$ for all x. Let's begin by showing that $\|\Lambda\| \le \kappa$. Consider any $\alpha \in \mathcal{H}$ and $x \in \mathcal{X}$. We have:

$$\begin{aligned}
|\Lambda\alpha(x)| &= \left| \mathop{\mathbb{E}}_{w\sim p}[\alpha(w)\phi(w,x)] \right| \\
&\le \mathop{\mathbb{E}}_{w\sim p}\left[|\alpha(w)\phi(w,x)| \right] \\
&= \mathop{\mathbb{E}}_{w\sim p}\left[|\langle \alpha, \mathcal{K}(w,\cdot)\rangle_{\mathcal{H}}\, \phi(w,x)| \right] && \text{(Reproducing property)} \\
&\le \mathop{\mathbb{E}}_{w\sim p}\left[\|\alpha\|_{\mathcal{H}} \|\mathcal{K}(w,\cdot)\|_{\mathcal{H}} |\phi(w,x)| \right] && \text{(Cauchy–Schwartz inequality)} \\
&= \mathop{\mathbb{E}}_{w\sim p}\left[\|\mathcal{K}(w,\cdot)\|_{\mathcal{H}} |\phi(w,x)| \right] \|\alpha\|_{\mathcal{H}} \\
&\le \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}}^2 \right]} \|\alpha\|_{\mathcal{H}} && \text{(Jensen inequality)} \\
&\le \kappa\|\alpha\|_{\mathcal{H}}.
\end{aligned}$$

Therefore $\Lambda\alpha$ is a bounded function of x, and we have $\|\Lambda\| := \|\Lambda\|_{\mathcal{H}\to L^\infty(\mathcal{X})} \leq \kappa$. Additionally, we can observe that $\Lambda\alpha(x)$, seen as an operator from $\mathcal{H}$ to $\mathbb{R}$ (mapping α to $\Lambda\alpha(x)$), is a bounded linear functional of norm at most κ. The Riesz representer theorem applies, and tells us that we can write:

$$\Lambda\alpha(x) = \langle \alpha, \psi(x) \rangle_{\mathcal{H}}\,, \tag{28}$$

for some $\psi(x) \in \mathcal{H}$. In fact, we have:

$$\psi(x) = \mathop{\mathbb{E}}_{w\sim p}[\phi(w,x)\mathcal{K}(w,\cdot)], \tag{29}$$

since, by the linearity of the scalar product in $\mathcal{H}$ and of the expectation, we can write:

$$\begin{aligned}
\Lambda\alpha(x) &:= \mathop{\mathbb{E}}_{w\sim p}[\alpha(w)\phi(w,x)] \\
&= \mathop{\mathbb{E}}_{w\sim p}[\langle \alpha, \mathcal{K}(w,\cdot)\rangle_{\mathcal{H}}\,\phi(w,x)] \\
&= \left\langle \alpha, \mathop{\mathbb{E}}_{w\sim p}[\phi(w,x)\mathcal{K}(w,\cdot)] \right\rangle_{\mathcal{H}} \\
&= \langle \alpha, \psi(x) \rangle_{\mathcal{H}}\,.
\end{aligned} \tag{30}$$

Using the Cauchy–Schwartz inequality, we have:

$$|\Lambda\alpha(x)| \leq \|\alpha\|_{\mathcal{H}}\|\psi(x)\|_{\mathcal{H}} = \|\psi(x)\|_{\mathcal{H}} = \left\| \mathop{\mathbb{E}}_{w\sim p}[\phi(w,x)\mathcal{K}(w,\cdot)] \right\|_{\mathcal{H}} \leq \theta. \tag{31}$$

This shows that $\|\Lambda\| \leq \theta$. Finally, we need to show that $\theta \leq \kappa$. This is a direct consequence of Jensen's inequality:

$$\begin{aligned}
\theta &= \sup_{x\in\mathcal{X}} \left\| \mathop{\mathbb{E}}_{w\sim p}[\phi(w,x)\mathcal{K}(w,\cdot)] \right\|_{\mathcal{H}} \\
&\leq \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}} \right] \\
&\leq \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}}^2 \right]} = \kappa.
\end{aligned}$$

■

We give examples in Section 4.5 of the values of θ and κ for two different instantiations of the model. A recurring pattern is that θ is more difficult to calculate, but yields a much tighter bound on the operator norm of Λ.

4.3 Assumptions

The theoretical results in this paper use some or all of the following assumptions:

Assumption 1 (A1). Consider some instance space $\mathcal{X}$, an instantiation $(\mathcal{W}, \mathcal{K}, p, \phi)$ of the model such that the constant κ defined in Equation (25) is finite, $\mathcal{H}$ the RKHS of kernel $\mathcal{K}$, the operator Λ defined by Equation (17) and the predictor class $\Lambda\mathcal{H}_B$ defined by Equation (21).

Assumption 2 (A2). Consider a label space $\mathcal{Y} \subseteq \mathbb{R}$ and let $\mathcal{D}$ be the data generating distribution over $\mathcal{X} \times \mathcal{Y}$.

Assumption 3 (A3). Suppose the loss $\ell : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ is ρ-Lipschitz, and can be written as $\ell(z, y) = \ell(yz)$.

Assumption 4 (A4). Suppose the loss $\ell : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ is convex and differentiable in its first argument. We denote $\ell'(z, y) := \frac{\partial \ell}{\partial z}(z, y)$ the partial derivative with regard to the first argument.

4.4 Unbiased approximation of the functional gradient

Iterative learning algorithms for the model of Equation 17 require an approximation of the gradient at each iteration. In this section, we calculate the functional gradient of the empirical risk functional (see Equation (1)) with regard to the model of Equation (17). We start with the following lemma, which gives the functional gradient of the model evaluation at a single point $x \in \mathcal{X}$.

Lemma 5 *Assume A1. Then* $\boldsymbol{\nabla}_\alpha \Lambda\alpha(x) = \psi(x) := \mathbb{E}_{w \sim p}\left[\phi(w, x)\mathcal{K}(w, \cdot)\right]$.

Proof See Equation (28). ■

We can use Lemma 5, the linearity of the gradient, as well as the chain rule, to calculate the gradient of the empirical risk functional. We do this in the next theorem.

Theorem 6 *Assume A1, A2 and A4. Suppose we have a sample* $\mathcal{S} \subseteq (\mathcal{X} \times \mathcal{Y})^m$. *Then:*

$$\boldsymbol{\nabla}_\alpha(\mathcal{L}_\mathcal{S}(\Lambda\alpha)) = \mathop{\mathbb{E}}_{w \sim p}\left[\frac{1}{m}\sum_{i=1}^{m} \ell'(\Lambda\alpha(x_i), y_i)\phi(w, x_i)\mathcal{K}(w, \cdot)\right]. \tag{32}$$

Proof We have:

$$\begin{aligned}
\boldsymbol{\nabla}_\alpha(\mathcal{L}_\mathcal{S}(\Lambda\alpha)) &= \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\nabla}_\alpha \ell(\Lambda\alpha(x_i), y_i) && \text{(Linearity of the gradient)} \\
&= \frac{1}{m}\sum_{i=1}^{m} \ell'(\Lambda\alpha(x_i), y_i)\boldsymbol{\nabla}_\alpha \Lambda\alpha(x_i) && \text{(Chain rule)} \\
&= \frac{1}{m}\sum_{i=1}^{m} \ell'(\Lambda\alpha(x_i), y_i) \mathop{\mathbb{E}}_{w \sim p}\left[\phi(w, x_i)\mathcal{K}(w, \cdot)\right] && \text{(Lemma 5)} \\
&= \mathop{\mathbb{E}}_{w \sim p}\left[\frac{1}{m}\sum_{i=1}^{m} \ell'(\Lambda\alpha(x_i), y_i)\phi(w, x_i)\mathcal{K}(w, \cdot)\right]. && \text{(Linearity of the expectation)}
\end{aligned}$$

■

We will see in Section 7, for example in Theorem 10, that the theoretical guarantees depend on the norm of the weight function α, namely $\|\alpha\|_\mathcal{H}$. Because of this, it is wise to add a regularization term $\frac{\lambda}{2}\|\alpha\|_\mathcal{H}^2$ (with $\lambda > 0$) to the empirical risk in order to control this norm while learning. We therefore define the **regularized empirical risk**:

$$\mathcal{L}_\mathcal{S}^{\text{reg}}(\Lambda\alpha) := \mathcal{L}_\mathcal{S}(\Lambda\alpha) + \frac{\lambda}{2}\|\alpha\|_\mathcal{H}^2, \tag{33}$$

By the linearity of the gradient, and Lemma 3, we have that:

$$\boldsymbol{\nabla}_\alpha \mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha) = \boldsymbol{\nabla}_\alpha \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \boldsymbol{\nabla}_\alpha \frac{\lambda}{2}\|\alpha\|_{\mathcal{H}}^2 = \boldsymbol{\nabla}_\alpha \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \lambda\alpha. \tag{34}$$

Because the functional gradients of Equations (32) and (34) are expectations over the choice of w, they are difficult objects to work with in practice. However, we can easily extract unbiased approximations of the gradients through simple random sampling. Indeed, consider the regularized empirical risk functional $\mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha)$. By sampling some parameter u according to distribution p, and a data batch $\mathbf{s}$ uniformly (with replacement) from $\mathcal{S}$, we can define the following unbiased approximation for the functional gradient $\boldsymbol{\nabla}_\alpha \mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha)$:

$$v(\alpha, u, \mathbf{s}) := \left(\frac{1}{|\mathbf{s}|} \sum_{(x,y)\in\mathbf{s}} \ell'(\Lambda\alpha(x), y)\phi(u, x) \right) \mathcal{K}(u, \cdot) + \lambda\alpha. \tag{35}$$

This approximation is indeed unbiased since, by Equations (32) and (34), we get that:

$$\mathop{\mathbb{E}}_{u\sim p} \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{U}(\mathcal{S})^{|\mathbf{s}|}} [v(\alpha, u, \mathbf{s})] = \boldsymbol{\nabla}_\alpha \mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha). \tag{36}$$

We show in Section 5 how to use this gradient approximation to learn the model.

4.5 Examples of instantiations of the model

The model of Equation (17) needs to be instantiated by choosing values for $(\mathcal{W}, \mathcal{K}, p, \phi)$. This is a very flexible model, and as such, the possibilities are wide. In this section, we present two generic instantiations of the model that can be used out of the box for any problem where $\mathcal{X} \subseteq \mathbb{R}^n$. These are some of the simplest instantiations possible, using basic choices for $(\mathcal{W}, \mathcal{K}, p, \phi)$ such as a Gaussian distribution p or the sign function as the base predictor ϕ. This allows calculating the analytical form for the expectation:

$$\mathop{\mathbb{E}}_{w\sim p} [\mathcal{K}(u, w)\phi(w, x)], \tag{37}$$

required to calculate the model in practice (see Equation (23)), as well as the exact value for κ (Equation (25)) and an upper bound for θ (Equation (26)). All of these values are in Table 1. (The calculus is in the appendix). In natural language:

1. Instantiation 1 considers the Gaussian distribution and kernel, and the sign function as the base predictor.

2. Instantiation 2 considers decision stumps as the base predictor. Each parameter $w \in \mathcal{W}$ is a tuple containing a variable index, and a threshold value. The distribution is uniform on the index, and Gaussian on the threshold value. The kernel is Gaussian on the threshold value.

The range of possible instantiations is virtually infinite, but each one requires rather heavy calculus to find the value for the expectation $\mathbb{E}_{w\sim p} [\mathcal{K}(u, w)\phi(w, x)]$. This is a vast area for further work, but for the purpose of this introductory paper, we will limit ourselves to these two simple instantiations.

	Instantiation 1	Instantiation 2
$\mathcal{W}$	$\mathbb{R}^n$	$\{1,\ldots,n\} \times \mathbb{R}$
$\mathcal{K}(u,w)$	$\exp\left(-\frac{\|u-w\|_2^2}{2\gamma^2}\right)$	$\mathbb{1}[u_1 = w_1] \exp\left(-\frac{(u_2-w_2)^2}{2\gamma^2}\right)$
p	$\mathcal{N}(0,\sigma^2 I)$	$\mathcal{U}(\{1,\ldots,n\}) \times \mathcal{N}(0,\sigma^2)$
$\phi(w,x)$	$\text{sign}(\langle w,x\rangle)$	$\text{sign}(x_{w_1} - w_2)$
$\mathbb{E}_{w\sim p}\left[\mathcal{K}(u,w)\phi(w,x)\right]$	$\left(1+\frac{\sigma^2}{\gamma^2}\right)^{-n/2} e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \text{erf}\left(\frac{\langle u',x\rangle}{\sqrt{2}\zeta\|x\|_2}\right)$	$\frac{\zeta}{\sigma n} e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}} \text{erf}\left(\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}\right)$
κ	1	1
θ	$\leq \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-n/4}$	$\leq \frac{1}{\sqrt{n}}\left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-1/4}$

Table 1: Two instantiations of the model. Note that ζ is defined by the relationship $\frac{1}{2\zeta^2} = \frac{1}{2\gamma^2} + \frac{1}{2\sigma^2}$, and u' is obtained from u by the transformation $u' := \left(1+\frac{\gamma^2}{\sigma^2}\right)^{-1} u$.

5. Learning the model

Multiple learning algorithms can learn the model of Equation (18). We present four of them in the following sections.

5.1 Stochastic functional gradient descent

Assuming that the loss ℓ is convex, we can apply a stochastic gradient descent algorithm using the gradient approximation given by Equation (35), with updates at iteration t of the form:

$$\alpha \leftarrow \alpha - \eta_t v(\alpha, w_t, \mathbf{s}_t), \tag{38}$$

for some stepsize η_t, and be guaranteed to converge to the optimal solution. In fact, $\mathcal{L}_S^{\text{reg}}(\Lambda\alpha)$ is λ-strongly convex in α, by convexity of ℓ and linearity of $\Lambda\alpha$. We therefore propose using a stochastic functional gradient descent algorithm for λ-strongly convex functions, similar to the one found in Section 14.4.4 of Shalev-Shwartz and Ben-David (2014). See Algorithm 1 for the pseudocode and Theorem 11 for a convergence guarantee.

The weight function α learned this way will take the form:

$$\alpha = \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot), \tag{39}$$

for some real coefficients $(a_1,\ldots,a_T)$ and where $(w_1,\ldots,w_T)$ are the sampled parameters (assuming $\alpha = 0 \in \mathcal{H}$ is the first iterate), meaning that the weight function will always remain an element of the RKHS $\mathcal{H}$. This allows us to use Equation (23) to calculate the predictions.

The bottleneck of Algorithm 1 is the projection step. Calculating the norm $\left\|\alpha^{(t-\frac{1}{2})}\right\|_{\mathcal{H}}$ can be done in $\mathcal{O}(t^2)$, as it involves doing a matrix multiplication using the $t \times t$ Gram matrix of the parameters at iteration t. Over T iterations, the projection step therefore costs $\mathcal{O}(T^3)$ in computation time. However, a simple optimization can reduce this cost by a

Algorithm 1 Stochastic functional gradient descent for learning the weight function

input Instantiation $(\mathcal{W}, \mathcal{K}, p, \phi)$, $\lambda > 0$, $B > 0$, number of iterations T, sample $\mathcal{S}$, batch size $|\mathbf{s}|$
  $\alpha^{(0)} \leftarrow 0 \in \mathcal{H}$
  for $t = 1, \dots, T$ **do**
    Sample $\mathbf{s}_t \sim \mathcal{U}(\mathcal{S})^{|\mathbf{s}|}$
    Sample $w_t \sim p$
    $\eta_t \leftarrow \frac{1}{\lambda t}$
    $v_t \leftarrow v\big(\alpha^{(t-1)}, w_t, \mathbf{s}_t\big)$ (see Equation (35))
    $\alpha^{(t-\frac{1}{2})} \leftarrow \alpha^{(t-1)} - \eta_t v_t$
    $\alpha^{(t)} \leftarrow \min\left(1, \frac{B}{\left\|\alpha^{(t-\frac{1}{2})}\right\|_{\mathcal{H}}}\right)\alpha^{(t-\frac{1}{2})}$ (projection step)
  end for
output $\bar{\alpha} := \frac{1}{T+1}\sum_{t=0}^{T}\alpha^{(t)}$

factor of T to $\mathcal{O}(T^2)$, in line with the rest of the algorithm. Indeed, notice that the update formula is:

$$\alpha^{(t-\frac{1}{2})} \leftarrow \alpha^{(t-1)} - \eta_t v_t. \tag{40}$$

Referring back to Equation (35) and using the shorthand $b_t := \Big(\frac{1}{|\mathbf{s}_t|}\sum_{(x,y)\in\mathbf{s}_t}\ell'(\Lambda\alpha(x), y)\phi(w_t, x)\Big)$, this can be rewritten as:

$$\alpha^{(t-\frac{1}{2})} \leftarrow \alpha^{(t-1)} - \eta_t\Big(b_t\mathcal{K}(w_t, \cdot) + \lambda\alpha^{(t-1)}\Big) = (1-\eta_t\lambda)\alpha^{(t-1)} - \eta_t b_t\mathcal{K}(w_t, \cdot). \tag{41}$$

Considering the squared norm, we have:

$$\begin{aligned}\left\|\alpha^{(t-\frac{1}{2})}\right\|_{\mathcal{H}}^2 &= \left\|(1-\eta_t\lambda)\alpha^{(t-1)} - \eta_t b_t\mathcal{K}(w_t, \cdot)\right\|_{\mathcal{H}}^2 \\ &= (1-\eta_t\lambda)^2\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}}^2 - 2(1-\eta_t\lambda)\eta_t b_t\left\langle\alpha^{(t-1)}, \mathcal{K}(w_t, \cdot)\right\rangle_{\mathcal{H}} + \eta_t^2 b_t^2\|\mathcal{K}(w_t, \cdot)\|_{\mathcal{H}}^2 \\ &= (1-\eta_t\lambda)^2\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}}^2 - 2(1-\eta_t\lambda)\eta_t b_t\alpha^{(t-1)}(w_t) + \eta_t^2 b_t^2\mathcal{K}(w_t, w_t).\end{aligned} \tag{42}$$

Calculating $\alpha^{(t-1)}(w_t)$ is in $\mathcal{O}(t)$, and $\mathcal{K}(w_t, w_t)$ takes constant time to compute (with regard to t). Therefore, one only needs to keep in memory the norm of the previous iterate, which is essentially costless, to reduce the learning time of the algorithm from $\mathcal{O}(T^3)$ to $\mathcal{O}(T^2)$.

5.2 Optimal stepsize of the gradient descent

From Theorem 6, we see that the unbiased approximation of the functional gradient of the regularized empirical risk with regard to some weight function α for a sampled parameter u is simply a scalar multiple of $\mathcal{K}(u, \cdot)$ plus $\lambda\alpha$:

$$\boldsymbol{\nabla}_\alpha\mathcal{L}_{\mathcal{S}}^{\mathrm{reg}}(\Lambda\alpha) \approx -\eta\mathcal{K}(u, \cdot) + \lambda\alpha, \tag{43}$$

for some $\eta \in \mathbb{R}$. Setting

$$\alpha' := \alpha + \eta\mathcal{K}(u, \cdot) - \lambda\alpha = (1-\lambda)\alpha + \eta\mathcal{K}(u, \cdot), \tag{44}$$

we can consider minimizing the regularized empirical risk with regard to the weight η:

$$\operatorname*{argmin}_{\eta} \mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha') = \operatorname*{argmin}_{\eta} \mathcal{L}_{\mathcal{S}}(\Lambda\alpha') + \frac{\lambda}{2}\|\alpha'\|_{\mathcal{H}}^2. \tag{45}$$

Assuming the loss ℓ is convex and differentiable, we can find the optimal η by finding where the derivative with regard to η is 0. We have the following partial derivatives:

$$\begin{aligned}\frac{d}{d\eta}\Lambda\alpha'(x) &= \frac{d}{d\eta}\left(\mathop{\mathbb{E}}_{w\sim p}[((1-\lambda)\alpha(w) + \eta\mathcal{K}(u,w))\phi(w,x)]\right) \\ &= \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)],\end{aligned} \tag{46}$$

$$\begin{aligned}\frac{d}{d\eta}\mathcal{L}_{\mathcal{S}}(\Lambda\alpha') &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{d}{d\eta}\ell(\Lambda\alpha'(x_i), y_i)\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left[\ell'(\Lambda\alpha'(x_i), y_i)\frac{d}{d\eta}\Lambda\alpha'(x_i)\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left[\ell'(\Lambda\alpha'(x_i), y_i)\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x_i)]\right],\end{aligned} \tag{47}$$

$$\begin{aligned}\frac{d}{d\eta}\|\alpha'\|_{\mathcal{H}}^2 &= \frac{d}{d\eta}\|(1-\lambda)\alpha + \eta\mathcal{K}(u,\cdot)\|_{\mathcal{H}}^2 \\ &= \frac{d}{d\eta}\Big((1-\lambda)^2\|\alpha\|_{\mathcal{H}}^2 + 2\eta(1-\lambda)\langle\alpha, \mathcal{K}(u,\cdot)\rangle + \eta^2\|\mathcal{K}(u,\cdot)\|_{\mathcal{H}}^2\Big) \\ &= \frac{d}{d\eta}\big(2\eta(1-\lambda)\alpha(u) + \eta^2\mathcal{K}(u,u)\big) \\ &= 2(1-\lambda)\alpha(u) + 2\eta\mathcal{K}(u,u).\end{aligned} \tag{48}$$

The derivative of the regularized empirical risk is therefore:

$$\begin{aligned}\frac{d}{d\eta}\mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha') &= \frac{d}{d\eta}\mathcal{L}_{\mathcal{S}}(\Lambda\alpha') + \frac{\lambda}{2}\frac{d}{d\eta}\|\alpha'\|_{\mathcal{H}}^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}\left[\ell'\big(\Lambda\alpha'(x_i), y_i\big)\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x_i)]\right] + \lambda(1-\lambda)\alpha(u) + \lambda\eta\mathcal{K}(u,u),\end{aligned} \tag{49}$$

where:

$$\begin{aligned}\Lambda\alpha'(x_i) &= \Lambda((1-\lambda)\alpha + \eta\mathcal{K}(u,\cdot))(x_i) \\ &= (1-\lambda)\Lambda\alpha(x_i) + \eta\Lambda\mathcal{K}(u,\cdot)(x_i).\end{aligned}$$

While the root of Equation (49) will in general not have an analytical expression due to the nonlinearity of the loss, root finding algorithms can solve the problem numerically with good precision relatively quickly. This allows calculating the optimal stepsize for any convex loss.

This gives rise to a new learning algorithm for the model (see Algorithm 2), which consists of greedily adding the new parameters $(w_1, \ldots, w_T)$ and calculating at each iteration the weights $(a_1, \ldots, a_T)$ which optimally reduce the regularized empirical risk. This learning algorithm has a $\mathcal{O}(mT^2)$ time complexity, the same as Algorithm 1, but requires fewer iterations in order to reach a small empirical risk, as seen in the experiments of Section 8.

Algorithm 2 Optimal Stepsize Descent for learning the weight function

input Instantiation $(\mathcal{W}, \mathcal{K}, p, \phi)$, $\lambda > 0$, number of iterations T, sample $\mathcal{S}$, batch size $|\mathbf{s}|$
  $\alpha^{(0)} \leftarrow 0 \in \mathcal{H}$
  for $t = 1, \ldots, T$ **do**
    Sample $\mathbf{s}_t \sim \mathcal{U}(\mathcal{S})^{|\mathbf{s}|}$
    Sample $w_t \sim p$
    Get η_t by finding the root of Equation (49) (calculated with $\mathcal{S} = \mathbf{s}_t$)
    $\alpha_t \leftarrow \alpha_{t-1} + \eta_t \mathcal{K}(w_t, \cdot)$
  end for
output α_T

Remark. Algorithm 2 uses batches to approximate the gradient, in order to accelerate learning. It is important to keep a large enough batch size, as optimizing Equation (45) on too few examples can overfit the batch, and fail to generalize to the entire dataset.

5.3 Least squares fit of the random features

Many Random Feature Methods first generate a large number of random parameters according to the sampling distribution, then learn the weight coefficients by analytically solving a convex optimization problem (Rahimi and Recht, 2009; Huang et al., 2006). This is possible when using the squared loss $\ell(h(x), y) := \frac{1}{2}(h(x) - y)^2$. We can use this same idea.

To understand how to achieve this, first recall that the model can be written as:

$$\Lambda\alpha(x) := \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_t, w)\phi(w, x)], \tag{50}$$

from some array of parameters $(w_1, \ldots, w_T)$ and vector of coefficients $a = (a_1, \ldots, a_T)$. We can define the operator φ, which embeds an instance x into a higher dimensional space:

$$\varphi(x) := \left(\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_1, w)\phi(w, x)], \ldots, \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_T, w)\phi(w, x)] \right)^\top.$$

The output of the model is then simply a linear function in the embedding space:

$$\Lambda\alpha(x) := \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_t, w)\phi(w, x)] = \langle a, \varphi(x) \rangle, \tag{51}$$

For some sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$, the regularized empirical squared loss is:

$$\mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha(x)) := \frac{1}{2m} \sum_{i=1}^{m} (\langle a, \varphi(x_i) \rangle - y_i)^2 + \frac{\lambda}{2} \|\alpha\|_{\mathcal{H}}^2. \tag{52}$$

Denoting $\Phi := (\varphi(x_1), \ldots, \varphi(x_m))^\top \in \mathbb{R}^{m \times T}$, $\mathbf{y} := (y_1, \ldots, y_m)^\top$, and G the Gram matrix of the parameters $(w_1, \ldots, w_T)$, we can simplify:

$$\mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha(x)) = \frac{1}{2m}\|\Phi a - \mathbf{y}\|_2^2 + \frac{\lambda}{2} a^\top G a. \tag{53}$$

The minimizer $a \in \mathbb{R}^T$ of this expression is the solution to the linear problem:

$$\Big(\Phi^\top \Phi + m\lambda G\Big) a = \Phi^\top \mathbf{y}. \tag{54}$$

Solving the linear system of Equation 54 requires $\mathcal{O}(T^3)$ operations, which is slower than the $\mathcal{O}(T^2)$ required by Algorithm 1. On the other hand, solving the linear system yields the optimal weights for minimizing the regularized empirical loss, requiring fewer sampled parameters to get the same accuracy. See Figure 1 in Section 8.

Algorithm 3 Least squares fit of the weight function coefficients

input Instantiation $(\mathcal{W}, \mathcal{K}, p, \phi)$, $\lambda > 0$, number of parameters T, sample $\mathcal{S}$ of size m
Sample $(w_1, \ldots, w_T) \sim p^T$
Calculate the matrix $\Phi \in \mathbb{R}^{m \times T}$, where $\Phi_{it} := \mathbb{E}_{w \sim p}\left[\mathcal{K}(w_t, w)\phi(w, x_i)\right]$
Calculate the matrix $G \in \mathbb{R}^{T \times T}$, where $G_{ij} := \mathcal{K}(w_i, w_j)$
Get $(a_1, \ldots, a_T)$ by solving Equation (54)
output $\alpha := \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot)$

5.4 Lasso fit of the random features

We can replace the Tikhonov regularizer $\frac{\lambda}{2}\|\alpha\|_{\mathcal{H}}^2$ in Equation (53) by the ℓ_1 regularizer on the norm of the coefficients, $\lambda\|a\|_1$, giving us the new minimization objective:

$$\mathcal{L}_{\mathcal{S}}^{\ell_1}(\Lambda\alpha(x)) = \frac{1}{2m}\|\Phi a - \mathbf{y}\|_2^2 + \lambda\|a\|_1. \tag{55}$$

The solution a of this problem can be obtained by applying the Lasso algorithm (Hastie et al., 2015). By the nature of minimizing with an ℓ_1 regularizer, the vector of coefficients a obtained this way will be sparse, which is an interesting advantage.

Algorithm 4 Lasso fit of the weight function coefficients

input Instantiation $(\mathcal{W}, \mathcal{K}, p, \phi)$, $\lambda > 0$, number of parameters T, sample $\mathcal{S}$ of size m
Sample $(w_1, \ldots, w_T) \sim p^T$
Calculate the matrix $\Phi \in \mathbb{R}^{m \times T}$, where $\Phi_{it} := \mathbb{E}_{w \sim p}\left[\mathcal{K}(w_t, w)\phi(w, x_i)\right]$
Get $(a_1, \ldots, a_T)$ by minimizing Equation (55) using the Lasso
output $\alpha := \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot)$

6. Pruning the model

Unlike fixed-sized models, such as neural networks, which do not grow with the number of iterations, the sum of Equation (23), reproduced here:

$$\Lambda\alpha(x) = \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_t, w)\phi(w, x)],$$

can have a limitless number of terms. Since this number impacts the computational complexity of learning, as well as the memory footprint of the model, we can be interested in being able to prune the model, similarly to how Algorithm 4 naturally outputs a model with few terms. We can indeed use the properties and structure of the model to concentrate α into its most salient components—thus pruning the sum. The sparsity of the resulting model also increases its interpretability.

Given a weight function $\alpha = \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot)$, the task is to find some $\beta = \sum_{t=1}^{T} b_t \mathcal{K}(w_t, \cdot)$ which shares its parameters $\{w_t\}_{t=1}^{T}$ with α and leads to (almost) the same predictions, but where a large number of the coefficients $\{b_t\}_{t=1}^{T}$ are zero.

The first step is to notice that the difference in error between predictors $\Lambda\alpha$ and $\Lambda\beta$ can be upper bounded as a function of $\|\alpha - \beta\|_{\mathcal{H}}$. We encapsulate this in the following lemma.

Lemma 7 *Assume A1, A2, A3. Consider any $\alpha, \beta \in \mathcal{H}$. Then:*

$$|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{D}}(\Lambda\beta)| \leq \rho\theta\|\alpha - \beta\|_{\mathcal{H}}. \tag{56}$$

Proof We have:

$$\begin{aligned}
|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{D}}(\Lambda\beta)| &= \left| \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} [\ell(\Lambda\alpha(x), y) - \ell(\Lambda\beta(x), y)] \right| \\
&\leq \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[|\ell(\Lambda\alpha(x), y) - \ell(\Lambda\beta(x), y)| \Big] \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[|\Lambda\alpha(x) - \Lambda\beta(x)| \Big] && (\ell \text{ is } \rho\text{-Lipschitz}) \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\|\Lambda(\alpha - \beta)\|_{L_\infty(\mathcal{X})} \Big] \\
&\leq \rho\|\Lambda\|\|\alpha - \beta\|_{\mathcal{H}} && (\text{Definition of } \|\Lambda\|) \\
&\leq \rho\theta\|\alpha - \beta\|_{\mathcal{H}}. && (\text{Theorem 4})
\end{aligned}$$

■

If $\|\alpha - \beta\|_{\mathcal{H}} < \frac{\varepsilon}{\rho\theta}$, then $|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{D}}(\Lambda\beta)| < \varepsilon$. Therefore, one way to achieve our objective is to minimise $\|\alpha - \beta\|_{\mathcal{H}}$ with a constraint on the number of nonzero coefficients in β. This can be framed as a simple least squares regression problem with ℓ^1 regularization by first denoting $a = (a_1, \ldots, a_T)$ and $b = (b_1, \ldots, b_T)$ the vectors of coefficients which define α and β, and $G := (\mathcal{K}(w_i, w_j))_{i,j=1}^{T}$ the Gram matrix of the parameters $(w_1, \ldots, w_T)$. Then, because the scalar product in $\mathcal{H}$ is given by:

$$\langle\alpha, \beta\rangle_{\mathcal{H}} := \sum_{i=1}^{T}\sum_{j=1}^{T} a_i b_j \mathcal{K}(w_i, w_j), \tag{57}$$

we can write:

$$\begin{aligned}\|\alpha - \beta\|_{\mathcal{H}}^2 &= \langle \alpha - \beta, \alpha - \beta \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{T} \sum_{j=1}^{T} (a_i - b_i) \mathcal{K}(w_i, w_j)(a_j - b_j) \\ &= (a-b)^\top G (a-b). \end{aligned} \tag{58}$$

Because $\mathcal{K}$ is a positive definite symmetric kernel, we can assume that G is a positive definite matrix.[1] It can thus can be written as $G = U^\top U$ for some upper triangular matrix U obtained by the Choleski decomposition of G. By replacing $G = U^\top U$ in Equation (58), we get the following expression:

$$\begin{aligned}\|\alpha - \beta\|_{\mathcal{H}}^2 &= (a-b)^\top G (a-b) \\ &= (a-b)^\top U^\top U (a-b) \\ &= \|Ua - Ub\|_2^2. \end{aligned} \tag{59}$$

We can therefore prune the model by using the Lasso (see Equation 2.5 of Hastie et al. (2015)) to solve:

$$\min_{b \in \mathbb{R}^T} \frac{1}{2T} \|Ua - Ub\|_2^2 + \lambda_{\text{Lasso}} \|b\|_1, \tag{60}$$

for some parameter $\lambda_{\text{Lasso}} > 0$.

The parameter λ_{Lasso} is a lever to control the compromise between the number of nonzero coefficients in b and the norm $\|\alpha - \beta\|_{\mathcal{H}}$. In fact, we can use Theorem 11.2 of Hastie et al. (2015) to guide the choice for λ_{Lasso}. This theorem states that, in our situation, the minimizer b of Equation (60) will be such that:

$$\|\alpha - \beta\|_{\mathcal{H}}^2 = \|Ua - Ub\|_2^2 \leq 12T \|a\|_1 \lambda_{\text{Lasso}}. \tag{61}$$

If:

$$\lambda_{\text{Lasso}} \leq \frac{\varepsilon^2}{12T\rho^2\theta^2\|a\|_1}, \tag{62}$$

then we will have $|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{D}}(\Lambda\beta)| < \varepsilon$ (by Lemma 7). This value of the parameter of the Lasso will ensure minimal degradation of the prediction accuracy. It is, however, a highly conservative value which will, in practice, yield limited pruning.

Fortunately, unlike typical applications of linear regression, the true regression vector is already known; it is simply $b = a$.[2] This allows us to calculate the norm $\|\alpha - \beta\|_{\mathcal{H}}$. This value can then be inserted directly into Lemma 7 to obtain the true bound on the error of $\Lambda\beta$, which will be tighter than the value ε guaranteed by taking λ_{Lasso} as given by Equation (62). If the bound is deemed small enough, the Lasso can be applied again with a larger λ_{Lasso} value, for better pruning. This process can be automated; all that is required is a starting value for λ_{Lasso} (for example using Equation (62)), and some stopping criterion. We summarize all of this in Algorithm 5.

1. By the properties of RKHS, G is at least positive semidefinite. If the smallest eigenvalue of G is too small or 0, adding a small (e.g. 10^{-8}) multiple of the identity matrix to G will ensure that it is positive definite, and will solve numerical instability problems that would have arisen otherwise.
2. In fact, this observation can be used to accelerate the convergence of the Lasso by using $b = a$ as a warm start for the algorithm.

Algorithm 5 Model pruning using the Lasso

input A weight function $\alpha \in \mathcal{H}$ to prune, a parameter λ_{Lasso} (e.g. as given by Equation (62)), an acceptation criterion $\text{ACCEPT} : \mathcal{H} \times \mathcal{H} \to \{0, 1\}$.
  $\beta, \beta_{\text{temp}} \leftarrow \alpha$
  while $\text{ACCEPT}(\alpha, \beta_{\text{temp}})$ **do**
    $\beta \leftarrow \beta_{\text{temp}}$
    Get new β_{temp} by solving Equation (60).
    $\lambda_{\text{Lasso}} \leftarrow 10\lambda_{\text{Lasso}}$
  end while
output β

Nonetheless, Lemma 7 will remain a rather loose bound, since it considers the worst case via Cauchy–Schwartz's inequality, and will not, in general, adequately reflect the true risk of $\Lambda\beta$. The following theorem is a probabilistic bound on the difference $|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{D}}(\Lambda\beta)|$. It uses a dataset to calculate a tighter guarantee with high probability. (The proof is in the appendix.)

Theorem 8 *Assume A1, A2, A3. Consider any $\alpha \in \mathcal{H}$. Then we have with probability at least $1 - \delta$ over the choice of $\mathcal{S} \sim \mathcal{D}^m$ that the following holds for all $\beta \in \mathcal{H}$ with $\|\beta - \alpha\|_{\mathcal{H}} \leq C$:*

$$\mathcal{L}_{\mathcal{D}}(\Lambda\beta) - \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\rho\theta C}{\sqrt{m}}\left(2 + \sqrt{2\log\frac{1}{\delta}}\right).$$

With this in mind, below is a list of possible ACCEPT functions that Algorithm 5 can use, in increasing order of leniency (functions further down will yield higher pruning). All of them use a user specified parameter $\varepsilon > 0$, the maximal acceptable degradation in accuracy.

- From Lemma 7:

$$\text{ACCEPT}(\alpha, \beta) = \text{TRUE} \qquad \text{if} \qquad \rho\theta\|\alpha - \beta\|_{\mathcal{H}} < \varepsilon. \tag{63}$$

- From Theorem 8:

$$\text{ACCEPT}(\alpha, \beta) = \text{TRUE} \qquad \text{if} \qquad \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\rho\theta C}{\sqrt{m}}\left(2 + \sqrt{2\log\frac{1}{\delta}}\right) < \varepsilon. \tag{64}$$

- Simple comparison of the empirical risks:

$$\text{ACCEPT}(\alpha, \beta) = \text{TRUE} \qquad \text{if} \qquad \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) < \varepsilon. \tag{65}$$

- Simple comparison of the empirical prediction accuracies (used for the experiments in Section 8.2):

$$\text{ACCEPT}(\alpha, \beta) = \text{TRUE} \qquad \text{if} \qquad \mathcal{L}^{01}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}^{01}_{\mathcal{S}}(\Lambda\alpha) < \varepsilon. \tag{66}$$

7. Theoretical guarantees

In this section, we demonstrate various theoretical guarantees of the model, most importantly a bound on the generalization error in the first section, and rate of convergence of the stochastic functional gradient descent afterward. We then offer an in-depth comparison to Rahimi and Recht (2009), a longstanding state-of-the-art in the field.

7.1 Bounding the generalization error using Rademacher complexity

We call the **generalization error** of a predictor the difference between its true risk and its empirical risk. High generalization error is a result of overfitting, which is usually a sign that the complexity of the class of predictors is inappropriately high. On the other hand, knowing this complexity allows us to limit overfitting. Our first theorem upper bounds the empirical Rademacher complexity of class $\Lambda\mathcal{H}_B$:

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B) := \frac{1}{m} \mathop{\mathbb{E}}_{\sigma\sim\{\pm 1\}^m} \left[\sup_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \sum_{i=1}^{m} \sigma_i \Lambda\alpha(x_i) \right], \tag{67}$$

and its expected Rademacher complexity:

$$\mathcal{R}_m(\Lambda\mathcal{H}_B) := \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m} \left[\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B) \right]. \tag{68}$$

(See e.g. Shalev-Shwartz and Ben-David (2014) or Mohri et al. (2012) for a rigorous exposition of the Rademacher theory for bounding the generalization error.)

Theorem 9 *Given assumptions A1 and A2, we have for any sample* $\mathcal{S} := \{(x_i, y_i)\}_{i=1}^m \subset (\mathcal{X}\times\mathcal{Y})^m$ *that:*

$$\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B) \leq \frac{B\theta}{\sqrt{m}}, \tag{69}$$

and:

$$\mathcal{R}_m(\Lambda\mathcal{H}_B) \leq \frac{B\theta}{\sqrt{m}}. \tag{70}$$

Proof Starting from the definition of the sample Rademacher complexity of $\Lambda\mathcal{H}_B$, we have:

$$\begin{aligned}
m\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B) &:= \mathop{\mathbb{E}}_{\sigma\sim\{\pm 1\}^m} \left[\sup_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \sum_{i=1}^{m} \sigma_i \Lambda\alpha(x_i) \right] \\
&= \mathop{\mathbb{E}}_{\sigma} \left[\sup_{\alpha\in\mathcal{H}_B} \sum_{i=1}^{m} \sigma_i \left\langle \alpha, \psi(x_i) \right\rangle_{\mathcal{H}} \right] && \text{(Equation (28))} \\
&= \mathop{\mathbb{E}}_{\sigma} \left[\sup_{\alpha\in\mathcal{H}_B} \left\langle \alpha, \sum_{i=1}^{m} \sigma_i \psi(x_i) \right\rangle_{\mathcal{H}} \right] \\
&\leq \mathop{\mathbb{E}}_{\sigma} \left[\sup_{\alpha\in\mathcal{H}_B} \|\alpha\|_{\mathcal{H}} \left\| \sum_{i=1}^{m} \sigma_i \psi(x_i) \right\|_{\mathcal{H}} \right] && \text{(Cauchy–Schwartz)} \\
&= B \mathop{\mathbb{E}}_{\sigma} \left[\left\| \sum_{i=1}^{m} \sigma_i \psi(x_i) \right\|_{\mathcal{H}} \right].
\end{aligned}$$

We can then apply Jensen's inequality to get:

$$\begin{aligned}
m\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B) &\leq B \mathop{\mathbb{E}}_{\sigma}\left[\left\|\sum_{i=1}^{m}\sigma_i\psi(x_i)\right\|_{\mathcal{H}}\right] \\
&\leq B\sqrt{\mathop{\mathbb{E}}_{\sigma}\left[\left\|\sum_{i=1}^{m}\sigma_i\psi(x_i)\right\|_{\mathcal{H}}^2\right]} && \text{(Jensen)} \\
&= B\sqrt{\mathop{\mathbb{E}}_{\sigma}\left[\left\langle\sum_{i=1}^{m}\sigma_i\psi(x_i),\sum_{i=1}^{m}\sigma_i\psi(x_i)\right\rangle_{\mathcal{H}}^2\right]} \\
&= B\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}\langle\psi(x_i),\psi(x_j)\rangle_{\mathcal{H}}\mathop{\mathbb{E}}_{\sigma}[\sigma_i\sigma_j]} \\
&= B\sqrt{\sum_{i=1}^{m}\|\psi(x_i)\|_{\mathcal{H}}^2} && (\mathbb{E}_\sigma[\sigma_i\sigma_j] = \mathbb{1}[i=j]) \\
&\leq \sqrt{m}B\theta.
\end{aligned}$$

We get the expected Rademacher complexity of $\Lambda\mathcal{H}_B$ by taking the expectation over the choice of sample. ■

We see that the Rademacher complexity is characterized by two model-dependent constants. The first is θ, which is defined by the instantiation of the model. The second is B, the maximal RKHS norm for the weight function, which acts as a hyperparameter of the algorithm, and can be used to control overfitting. Using standard results (principally Theorem 3.1 of Mohri et al. (2012)), we can convert the Rademacher complexity of $\Lambda\mathcal{H}_B$ into the following uniform bound on the generalization error.

Theorem 10 *Given assumptions A1, A2, and A3, and assuming that the loss ℓ takes values in $[0, M]$, we have with probability at least $1-\delta$ over the choice of $\mathcal{S} \sim \mathcal{D}^m$ that the following holds for all $\Lambda\alpha \in \Lambda\mathcal{H}_B$:*

$$\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{2B\rho\theta}{\sqrt{m}} + M\sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \tag{71}$$

Proof Assumption A3 and Theorem 9 allow us to use Talagrand's lemma (Lemma 4.2 of Mohri et al. (2012)) to get:

$$\begin{aligned}
\mathcal{R}_m(\ell\circ\Lambda\mathcal{H}_B) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\widehat{\mathcal{R}}_{\mathcal{S}}(\ell\circ\Lambda\mathcal{H}_B)\right] \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\widehat{\mathcal{R}}_{\mathcal{S}}(\Lambda\mathcal{H}_B)\right] \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{B\theta}{\sqrt{m}}\right] \\
&\leq \frac{B\rho\theta}{\sqrt{m}}.
\end{aligned} \tag{72}$$

A straightforward application of Theorem 3.1 of Mohri et al. (2012) with this bound on $\mathcal{R}_m(\ell \circ \Lambda\mathcal{H}_B)$ gives us the result. (Note that Mohri et al. (2012) assumes that the loss takes its values in $[0, 1]$. However, the proof of Theorem 3.1 of Mohri et al. (2012) can directly be generalized to losses taking values in $[0, M]$ by adjusting the application of McDiarmid's inequality. The bounded difference condition will be satisfied with constant $\frac{M}{m}$ rather than $\frac{1}{m}$.) ■

The bound of Theorem 10 can be calculated in practice. It can be made as tight as possible by choosing $B := \|\alpha\|_{\mathcal{H}}$ a posteriori, where α is the weight function learned using Algorithm 1 or any other learning method. Note however that the bound suggests that B can only grow as $o(\sqrt{m})$ in order to avoid overfitting. This justifies minimizing the regularized empirical risk, in order to control the norm of the weight function.

For loss functions which are not naturally bounded by some constant M, Theorem 10 can still be used. By Theorem 4, predictors in $\Lambda\mathcal{H}_B$ are bounded by $B\theta$, i.e. $|\Lambda\alpha(x)| \leq B\theta$ for all $x \in \mathcal{X}$ and for all $\Lambda\alpha \in \Lambda\mathcal{H}_B$. It is generally easy to use this fact to derive a maximal possible value for a given loss, especially in the binary classification setting, where $\mathcal{Y} = \{-1, 1\}$. In the case of the square loss:

$$\ell(y', y) := (1 - yy')^2 = (y - y')^2. \tag{73}$$

Since the model is bounded by $B\theta$, we can use $M = (B\theta + 1)^2$.

7.2 Stability analysis of the stochastic functional gradient descent

The following theorem describes the convergence in expectation of Algorithm 1 with regard to the number of examples in the sample, and the number of iterations. (The proof can be found in the appendix.)

Theorem 11 *Assume A1, A2, A3, A4. The output $\bar{\alpha}$ of Algorithm 1 is such that:*

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] - \min_{\Lambda\alpha \in \Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \frac{2\rho\theta B}{\sqrt{m}} + \lambda B^2 + \frac{8\rho^2}{\lambda m} + \frac{(\rho\kappa + \lambda B)^2}{2\lambda T}(1 + \log(T)), \tag{74}$$

where the expectation is taken over the choice of sample and all sampled parameters and batches (i.e. $\mathcal{S} \sim \mathcal{D}^m, (w_1, \ldots, w_T) \sim p^T, (\mathbf{s}_1, \ldots, \mathbf{s}_T) \sim \mathcal{U}(\mathcal{S})^{T \times |\mathbf{s}|}$). By choosing $\lambda = \sqrt{\frac{8\rho^2}{B^2 m}}$, we obtain:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] - \min_{\Lambda\alpha \in \Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \frac{\rho B}{\sqrt{m}}\left(\sqrt{32} + 2\theta + \frac{m}{\sqrt{32}T}\left(\kappa + \sqrt{\frac{8}{m}}\right)^2(1 + \log(T))\right) \tag{75}$$

$$\in \mathcal{O}\left(\frac{1}{\sqrt{m}} + \frac{\sqrt{m}}{T}\log T\right).$$

This bound guarantees convergence to the optimal given enough data and iterations. While it is a result on the expected risk, it can be turned into a probabilistic bound via Markov's inequality or other more sophisticated methods, such as the one given in exercise 13.1 of Shalev-Shwartz and Ben-David (2014).

Note also that, in practice, it is easier to use Equation (74) rather than Equation (75), since it allows using an arbitrary value for λ (chosen by cross-validation, for instance). It is also slightly less constraining, due to the ability not to specify the maximal norm B before learning, thus skipping the projection step of Algorithm 1 if desired while maintaining the ability to calculate, a posteriori, a bound on the convergence (replacing B in Equation (74) by the largest iterate norm).

7.3 Comparison with the Random Kitchen Sinks

In this section, we compare our hypothesis class, assumptions and theoretical results to Rahimi and Recht (2009). Recall from Section 3 the two prediction classes from Rahimi and Recht (2009), as well as the class introduced in this paper:

$$\mathcal{F}_p := \left\{ x \mapsto \Lambda\alpha(x) := \mathop{\mathbb{E}}_{w\sim p}\left[\alpha(w)\phi(w,x)\right] \;\middle|\; \|\alpha\|_{L^\infty(\mathcal{W})} \leq C \right\}, \tag{14}$$

$$\hat{\mathcal{F}}_w := \left\{ x \mapsto \frac{1}{K}\sum_{k=1}^{K} a_k \phi(w_k, x) \;\middle|\; \forall k, |a_k| \leq C \right\}, \tag{15}$$

$$\Lambda\mathcal{H}_B := \left\{ x \mapsto \Lambda\alpha(x) := \mathop{\mathbb{E}}_{w\sim p}\left[\alpha(w)\phi(w,x)\right] \;\middle|\; \alpha \in \mathcal{H}, \|\alpha\|_{\mathcal{H}} \leq B \right\}, \tag{21}$$

Because the functions in $\mathcal{F}_p$ can't be represented or calculated exactly, Rahimi and Recht (2009) instead approximate the expectation by an average, giving rise to the class $\hat{\mathcal{F}}_w$.

The predictors in $\mathcal{F}_p$ and $\Lambda\mathcal{H}_B$ have the same form $\Lambda\alpha(x) = \mathbb{E}_{w\sim p}\left[\alpha(w)\phi(w,x)\right]$. (They are a weighted expectation of the base predictor ϕ according to distribution p and a weight function α.) Furthermore, looking at Equation (23):

$$\Lambda\alpha(x) = \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w\sim p}\left[\mathcal{K}(w_t, w)\phi(w,x)\right],$$

we see that the functions that can be represented in practice in $\Lambda\mathcal{H}_B$ (when α is a finite sum) have the same form as the predictors in $\hat{\mathcal{F}}_w$, the approximation class of Rahimi and Recht (2009), where $\phi(w_k, w)$ has been replaced by $\mathbb{E}_{w\sim p}\left[\mathcal{K}(w_k, w)\phi(w,x)\right]$. In fact, this observation is what underlies Algorithms 3 and 4. Both algorithms solve a regularized least squares problem, precisely like Rahimi and Recht (2009). However, the regularization term itself is different. Whereas Rahimi and Recht (2009) regularize the euclidean norm of the coefficients[3], Algorithm 4 uses an ℓ_1 regularizer, and Algorithm 3 regularizes the RKHS norm of the weight function. This last regularizer is directly adapted to the form our model takes, making it an interesting new aspect of our model.

The most crucial difference between the models is the provenance of the weight function. In $\mathcal{F}_p$, α must be a bounded function of $\mathcal{W}$, whereas in $\Lambda\mathcal{H}_B$, we restrict α to the ball of radius B of an RKHS. Let's unpack the consequences of this difference.

3. We point to an important fact, which is not necessarily obvious from a brief look at Rahimi and Recht (2009). The algorithm that they use in their experiments is not their Algorithm 1, but rather they analytically minimize the ℓ_2 regularized version of their optimization objective.

Consequence 1. As explained in Section 4, and exemplified in Table 1, the properties of the RKHS allow calculating the model exactly through Equation (23), as long as α is a finite sum and the expectation $\mathbb{E}_{w\sim p}\left[\mathcal{K}(w_t, w)\phi(w, x)\right]$ has an analytical form. This means that the predictors $\Lambda\alpha$ we work with are always within the target class $\Lambda\mathcal{H}_B$. This constrasts Rahimi and Recht (2009), which can only use approximations of the functions in $\mathcal{F}_p$. While we do also technically work within an approximation class in practice (the set of all the weight functions that can be written using the $(w_1, \ldots, w_T)$ that have been sampled), this approximation class is in fact a subset of $\Lambda\mathcal{H}_B$. Class $\Lambda\mathcal{H}_B$ therefore forgoes the loss of precision due to the approximation of its functions.

Consequence 2. The structure imparted by the RKHS also allows softening the condition on ϕ. Where Rahimi and Recht (2009) require that $\phi(w, x)$ be bounded, class $\Lambda\mathcal{H}_B$ only supposes that $\kappa^2 := \sup_x \mathbb{E}_{w\sim p}\left[\mathcal{K}(w, w)\phi(w, x)^2\right]$ is finite (see Section 4.2). This is a more lenient condition, as κ can be finite despite ϕ being unbounded, which means that ϕ can be chosen from a wider range of functions. This allows instantiating the model more flexibly, allowing entirely new classes of predictors.

Consequence 3. Even when distribution p and base predictor ϕ are the same with both models, the additional choice of a kernel $\mathcal{K}$ in our model creates a wider range of options. For instance, if $\mathcal{K}$ is unbounded, then the functions of its RKHS (the weight functions α) can also be unbounded[4]. Such a choice of kernel therefore gives us access to weight functions that are disallowed in Rahimi and Recht (2009), which require $\|\alpha\|_{L^\infty(\mathcal{W})} \leq C$.

As we can see, class $\Lambda\mathcal{H}_B$ presents advantages over $\mathcal{F}_p$. The remaining task is to assess the expressivity of $\Lambda\mathcal{H}_B$ in comparison to $\mathcal{F}_p$ when they share the same distribution p and base predictor ϕ. We do so in the next lemma.

Lemma 12 *Assume A1, A2, A3. Further assume that* $\sup_{w,x}|\phi(w, x)| \leq 1$*, and that* $\mathcal{K}$ *is a universal kernel on* $L^2(p)$ *(the Gaussian or exponential kernels, for example, satisfy this condition; see (Steinwart and Christmann, 2008)). Consider some* $\Lambda\alpha^\star \in \mathcal{F}_p$*. Then for all* $\varepsilon > 0$*, there exists* $\alpha \in \mathcal{H}$ *such that:*

$$|\mathcal{L}_\mathcal{D}(\Lambda\alpha^\star) - \mathcal{L}_\mathcal{D}(\Lambda\alpha)| < \varepsilon. \tag{76}$$

Proof Since $\mathcal{K}$ is universal, its RKHS $\mathcal{H}$ is dense in $L^2(p)$. Therefore, because $\alpha^\star \in L^2(p)$, we can find $\alpha \in \mathcal{H}$ such that:

$$\|\alpha^\star - \alpha\|_{L^2(p)} := \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[|\alpha^\star(w) - \alpha(w)|^2\right]} < \frac{\varepsilon}{\rho}, \tag{77}$$

4. Take for example $\mathcal{K}(w, u) := \exp(\langle w, u\rangle)$, and $\alpha = \mathcal{K}(w, \cdot)$ for some $w \in \mathcal{W} = \mathbb{R}^n$. Then α is not bounded as a function of $\mathcal{W}$, since $\alpha(cw) = \exp\big(c\|w\|_2^2\big) \to \infty$ when $c \to \infty$.

where ρ is the Lipschitz constant of the loss. Consequently, we have:

$$\begin{aligned}
|\mathcal{L}_{\mathcal{D}}(\Lambda\alpha^\star) - \mathcal{L}_{\mathcal{D}}(\Lambda\alpha)| &= \left| \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\ell(\Lambda\alpha^\star(x), y) - \ell(\Lambda\alpha(x), y) \Big] \right| \\
&\leq \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[|\ell(\Lambda\alpha^\star(x), y) - \ell(\Lambda\alpha(x), y)| \Big] \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[|\Lambda\alpha^\star(x) - \Lambda\alpha(x)| \Big] && (\ell \text{ is } \rho\text{-Lipschitz}) \\
&= \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\left| \mathop{\mathbb{E}}_{w\sim p} \Big[(\alpha^\star - \alpha)(w)\phi(w, x) \Big] \right| \right] \\
&= \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\big| \langle \alpha^\star - \alpha, \phi(\cdot, x) \rangle_{L_2(p)} \big| \Big] \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\|\alpha^\star - \alpha\|_{L_2(p)} \|\phi(\cdot, x)\|_{L_2(p)} \Big] && (\text{Cauchy–Schwartz}) \\
&\leq \rho \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\|\alpha^\star - \alpha\|_{L_2(p)} \Big] && (\sup_{\omega,x} |\phi(\omega, x)| \leq 1) \\
&< \rho \frac{\varepsilon}{\rho} \\
&= \varepsilon.
\end{aligned}$$

■

Lemma 12 means that any predictor in $\mathcal{F}_p$ can be approximated accurately by the class $\Lambda\mathcal{H}$. There is not, however, a guarantee on how large the norm of the weight function might be required to be.

Finally, Theorem 11 shows a convergence rate of $\mathcal{O}\Big(\frac{1}{\sqrt{m}} + \frac{\sqrt{m}}{T} \log T\Big)$ for Algorithm 1, compared to $\mathcal{O}\Big(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{T}}\Big)$ for Algorithm 1 of Rahimi and Recht (2009). When T is large (commensurate to m), these bounds have the same complexity, up to a log factor. Further work to derive a tighter bound is warranted, as well as bounds for the other algorithms, especially Algorithms 3 and 4, which are similar to the algorithm of Rahimi and Recht (2009), and perform better in practice, as we will see in the next section.

In short, the predictor class $\Lambda\mathcal{H}_B$ examined in this work appears expressive and flexible, though a direct comparison with Rahimi and Recht (2009) remains challenging, since the different constraints provide different characteristics to the predictors, and neither method appears unambiguously superior to the other. We do note that the particular form of the predictors in $\Lambda\mathcal{H}_B$, with the weight function belonging to an RKHS, has the potential to confer new advantages. For example, the choice of kernel could be a way to infuse prior knowledge into the model. While universal kernels lead to highly expressive classes of predictors, as seen in Lemma 12, this can also cause overfitting and be counterproductive. One might instead want to use a kernel which is tailored to a specific problem, thus reducing the complexity of $\Lambda\mathcal{H}_B$ and the risk of overfitting, without actually sacrificing the prediction accuracy. This is an open and promising avenue for further research.

8. Experiments

In this section, we present three experiments for testing the model and its learning algorithms. The first one (Section 8.1) compares the prediction accuracy and training time of all four learning algorithms with regard to the sampling size T. The second (Section 8.2) tests pruning the model. The third is a comparison to other models (AdaBoost and the original Random Kitchen Sinks algorithm).

In every case, the algorithms used the mean square error as the learning loss, and the regularization parameter λ was chosen through 5-fold cross-validation among the set:

$$\{0.0000001, 0.000001, 0.00001, 0.0001, 0.001\}.$$

The two iterative algorithms, Algorithm 1 and Algorithm 2, used a sample minibatch size $|\mathbf{s}|$ of 50 to approximate the gradient. We also set $B = 1000$ for Algorithm 1. We've used four datasets for these experiments, chosen for their different sizes and dimensionalities:

MNIST17. Digits 1 and 7 of the widely used handwritten digits recognition dataset. Data is found at: `http://yann.lecun.com/exdb/mnist/`. This dataset has 13007 training examples, and 2163 test examples. The dimensionality of the data is 784.

Adults. The task is to predict yearly income based on various information about an individual. Data is found at: `https://archive.ics.uci.edu/ml/datasets/adult`. Note that we changed all categorical variables (such as occupation, or native country) to numerical variables via a one-hot-encoding scheme. This dataset has 32561 training examples, and 16281 test examples. After preprocessing, the dimensionality of the data is 108.

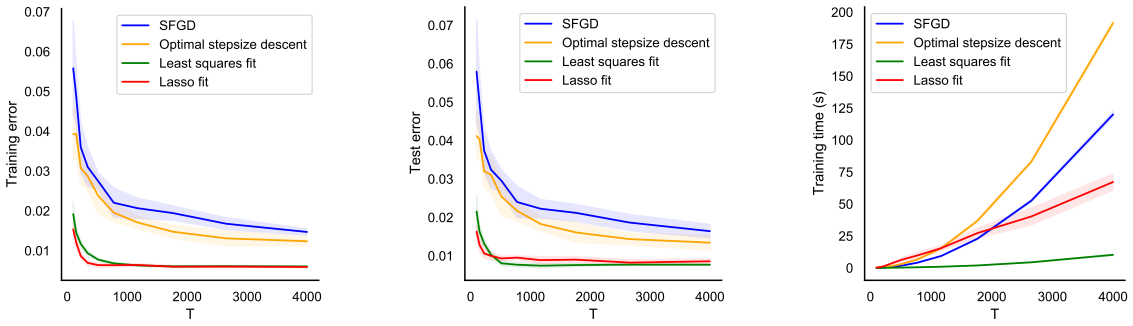
Breast Cancer. The task is to classify whether a cell sample corresponds to a malignant cancer cell or not. Data is found at: `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`. The training set has 426 examples, and the test set has 143. Dimensionality is 30.

Skin Segmentation. The task is to classify whether an RGB value corresponds to a skin or nonskin sample. Data is found at: `https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation`. The training set has 183792 examples, and the test set has 61265. Dimensionality is 3.

The datasets were all scaled to have variance of 1 and mean of 0 on each variable, and a dummy variable of value 1 was added to all instances of all datasets to simulate bias. For the datasets which do not come as a (training set, test set) pair, the dataset was randomly split on a 75:25 ratio. Additional implementation details can be found in the Appendix. The full code can be found at `https://github.com/gadub44/jmlr2023`.

8.1 Comparison of the learning algorithms

In Figure 1, we compare the learning rates and training time of all four algorithms with regards to the sampling size T (the number of sampled parameters). The results are expected: the two algorithms which can directly optimize the coefficients (Algorithms 3 and 4) have significantly better accuracy than the two iterative algorithms (Algorithms 1 and 2). The Lasso fit appears to be the best algorithm, as it has equal or better performance than the Least squares fit, but is faster. Between the two iterative algorithms, the Optimal stepsize descent converges faster than SFGD, which is the obvious consequence of calculating bet-



(a) (b) (c)

Figure 1: Comparison of four learning algorithms with regards to the number of sampled parameters: **Stochastic Functional Gradient Descent** (Algorithm 1), **Optimal Stepsize Descent** (Algorithm 2), **Least squares fit** of the parameters (Algorithm 3), and **Lasso fit** of the parameters (Algorithm 4). All four algorithms were tasked with learning the weight function for Instantiation 1 on MNIST17. Every point is the average of 10 independently seeded runs, with the standard deviation shown as a shaded area. The Lasso and Least squares fit of the parameters clearly provide the best performance. The Least squares fit is by far the fastest method.

Table 2: Prediction accuracy and pruning ratio of various algorithms on MNIST17. **Lasso fit** is Algorithm 4 alone, while **Least squares fit**, **Optimal stepsize descent** and **SFGD** are the respective algorithm followed by Algorithm 5 for pruning the learned model. $\|\beta\|_{\mathcal{H}}$ is the norm of the weight function after pruning. In every case, $T = 1000$. Every line is the average of 10 independently seeded runs.

algo	$\mathcal{L}_{\mathcal{S}}^{01}$	$\mathcal{L}_{\mathcal{D}}^{01}$	%pruning	pruning+$\mathcal{L}_{\mathcal{S}}^{01}$	pruning+$\mathcal{L}_{\mathcal{D}}^{01}$	$\|\beta\|_{\mathcal{H}}$
Lasso fit	0.006	0.009	63.99	0.006	0.009	1053.973
Least squares fit	0.007	0.007	59.90	0.007	0.009	248.396
Optimal stepsize descent	0.018	0.019	51.42	0.022	0.024	534.325
SFGD	0.025	0.027	73.40	0.027	0.029	310.779

ter coefficients at each iteration. However, the additional computation results in a longer training time.

8.2 Pruning experiments

The different algorithms of Section 5 lead to different learned models, which can react differently to the pruning method explained in Section 6. When the number of terms in the model is important, it is best to use the learning and pruning method that lead to the sparsest model possible. In Table 2, we examine the different pruning ratios that can be obtained. The options are either to directly use Algorithm 4 (the Lasso fit of the coefficients) to get a sparse model, or learn the model using Algorithm 1, 2 or 3, followed by Algorithm 5

for pruning the model. The acceptation criterion used for Algorithm 5 was whether the empirical 0-1 error degraded by at most ε, with $\varepsilon = 0.01$.

Expectedly, directly optimizing with the Lasso using an ℓ_1 regularizer yields both a high pruning ratio and an excellent prediction accuracy, though the Least squares fit followed by pruning yields a very similar performance and pruning ratio. We do observe that the best pruning was reached after using SFGD, but the performance was also the worst. The reason for the high pruning ratio in that case can possibly be attributed to the ever decreasing coefficients as the gradient descent progresses, which can more easily be set to 0.

8.3 Comparison to other models

Table 3 shows the prediction accuracy and training time of the model in comparison to three well-known algorithms: AdaBoost, an RBF kernel SVM, and the Random Kitchen Sinks algorithm.

AdaBoost achieves the best performance on `adults`, `breast cancer` and `mnist`, with our model (the least squares fit using Instantiation 2) achieving similar accuracy on `breast cancer`. SVM is the best algorithm on `skin segmentation`, with our model (the Lasso fit using Instantiation 1) reaching comparable performance. Notably, our model outperforms the Random Kitchen Sinks algorithm on all datasets (tying on `mnist`). This is most relevant when using Instantiation 2, since it uses exactly the same distribution p and base predictor ϕ as the RKS algorithm, meaning that the comparison is direct and shows that our model can outperform Rahimi and Recht (2009).

The two algorithms tested for learning our model, Algorithms 3 and 4, achieve very similar prediction accuracy. They do however have different properties on the other metrics. Most importantly, the Lasso fit has a longer training time, especially on the large `skin segmentation` dataset. The Least squares fit also consistenly leads to a lower generalization bound. It can therefore considered to be the best learning algorithm for the model, except when a sparse model is desired and a longer training time is acceptable.

As for the instantiations, Instantiation 1 leads to better performance on `skin segmentation`, while Instantiation 2 is the best on `adults` and `breast cancer`. The performance of the model therefore depends on both the instantiation and on the dataset. In fact, we can easily imagine that some other instantiations of the model would have proven better for each of these datasets, improving the empirical results.

Finally, we do notice that the guarantee offered by Theorem 10 appears loose, meaning that further work to derive a tighter bound is warranted.

9. Future and limitations

This paper provides a first study of a new model, and several directions merit further investigation. In particular, the following aspects appear to us to be the most important issues.

Difficult integrals. While the model is flexible in theory, the expectation $\mathbb{E}_{w\sim p}\left[\mathcal{K}(u,w)\phi(w,x)\right]$ should ideally have an analytical form, as in Table 1, in order for the output of the model to be calculated exactly via Equation (23). In practice, these integrals require rather hefty calculus to solve, and might well not have an exact formula for a given instantiation of the

Table 3: Performance comparison of the model to AdaBoost (**AB**), **SVM** and the Random Kitchen Sinks (**RKS**) algorithm on various binary classification datasets. Instantiations 1 and 2 of the model (**I1** and **I2**) were learned using Algorithm 3 (**LS fit**) and Algorithm 4 (**Lasso fit**) with $T = 1000$. $\mathcal{L}_\mathcal{S}$ and $\mathcal{L}_\mathcal{D}$ are the empirical and test mean square error. $\mathcal{L}^{01}_\mathcal{S}$ and $\mathcal{L}^{01}_\mathcal{D}$ are the empirical and test classification errors. $\mathcal{R}_m$ is the right-hand side of Equation (71), with $\delta = 0.05$. Every line (except SVM) is the average of 10 independent runs.

dataset	algo	$\mathcal{L}_\mathcal{S}$	$\mathcal{L}_\mathcal{D}$	$\mathcal{R}_m$	$\mathcal{L}^{01}_\mathcal{S}$	$\mathcal{L}^{01}_\mathcal{D}$	training time (s)
adults	**AdaBoost**				**0.130**	**0.133**	6.585
	LS fit w/ I1	0.488	0.487	61.4	0.157	0.155	0.863
	LS fit w/ I2	0.429	0.429	**7.0**	0.147	0.146	0.737
	Lasso fit w/ I1	0.454	0.454	$>10^3$	0.158	0.156	9.422
	Lasso fit w/ I2	0.425	0.427	155.2	0.145	0.145	49.484
	RKS				0.148	0.148	0.513
	SVM				0.137	0.148	79.222
breast cancer	**AdaBoost**				**0.000**	**0.021**	0.109
	LS fit w/ I1	0.162	0.191	13.2	0.021	0.036	0.069
	LS fit w/ I2	0.135	0.145	**11.7**	**0.018**	**0.022**	0.049
	Lasso fit w/ I1	0.073	0.147	$>10^5$	0.012	0.043	0.872
	Lasso fit w/ I2	0.087	0.158	853.9	0.009	0.035	0.768
	RKS				0.000	0.076	0.029
	SVM				0.014	0.035	0.004
mnist17	**AdaBoost**				**0.000**	**0.005**	38.527
	LS fit w/ I1	0.057	0.061	$>10^3$	0.007	0.010	0.822
	LS fit w/ I2	0.066	0.071	**5.5**	0.008	0.013	0.494
	Lasso fit w/ I1	0.051	0.056	$>10^4$	0.007	0.010	18.259
	Lasso fit w/ I2	0.054	0.059	112.8	0.006	0.010	24.711
	RKS				0.007	0.012	0.325
	SVM				0.000	0.010	6.635
skin segmentation	AdaBoost				0.039	0.039	7.278
	LS fit w/ I1	0.03	0.03	15.7	0.005	0.005	5.561
	LS fit w/ I2	0.172	0.173	**7.4**	0.042	0.042	6.288
	Lasso fit w/ I1	0.015	0.015	693.8	**0.002**	**0.002**	411.723
	Lasso fit w/ I2	0.17	0.171	26.8	0.040	0.040	353.341
	RKS				0.040	0.040	2.865
	SVM				**0.001**	**0.001**	7.495

model. In that case, these integrals can be approximated via Monte Carlo methods, at the cost of computation time and accuracy.

Curse of dimensionality. Whenever the distribution p covers a high-dimensional space, the model's output tends to vanish to zero (as can be seen from the $\left(1+\sigma^2/\gamma^2\right)^{-n/2}$ coefficient in Table 1). A commensurably large norm of the weight function could be required, leading to poor guarantees. Care needs to be applied when choosing the parameter space $\mathcal{W}$, kernel $\mathcal{K}$, distribution p, and base predictor ϕ.

Multiclass classification. The method we have examined works in the binary classification setting. By using vector-valued reproducing kernel Hilbert spaces (Micchelli and Pontil, 2005), it is possible to extend the model to vector-valued outputs, and the stochastic functional gradient descent algorithm will be able to learn the weight function. However, the theoretical guarantees we present in this paper do not apply to that context. New bounds need to be derived.

Instantiating the model. We saw in Table 3 that different instantiations can lead to different performance on different datasets. This means that the choice of instantiation is crucial for solving a given problem. How to choose or construct the best instantiation for the problem at hand is an important aspect to study.

In summary, the model we have introduced in this paper is usable in practice, and has multiple valid learning algorithms, but its full potential has not yet been fully realized. More work needs to be done in understanding how to choose or construct high performance instantiations, especially in the case of high-dimensional data. Tighter theoretical guarantees should also be developped. In addition to pursuing these avenues, we will continue seeking ways of leveraging the RKHS structure of the weight function in order to extract out of the model such properties as sparsity or interpretability, as we believe this to be the model's greatest strength.

10. Conclusion

In this paper, we examined a novel general machine learning model for binary classification. We have shown how the model can be learned using stochastic functional gradient descent (and other algorithms), and proved convergence guarantees using the stability properties of the algorithm. We have also proven a bound on the generalization error of the proposed class of predictors via Rademacher complexity theory. We have shown how the model can be pruned using the Lasso, and demonstrated a bound on the error of the pruned predictor. We ran experiments using simple instantiations of the model to showcase its usability.

This by no means constitutes a complete exploration of the applications and theoretical properties of the model. Indeed, several challenges remain, such as understanding how to successfully instantiate the model for a given problem, and how to leverage the model's flexibility despite a requirement on solving difficult integrals. It is yet unclear what family of problems this model will excel at solving. We hope to use it in particular to build interpretable predictors, which is crucial for the widespread societal adoption of artificial intelligence solutions.

Acknowledgments and Disclosure of Funding

The authors wish to thank the DEEL project CRDPJ 537462-18 funded by the National Science and Engineering Research Council of Canada (NSERC) and the Consortium for Research and Innovation in Aerospace in Québec (CRIAQ), together with its industrial partners Thales Canada inc, Bell Textron Canada Limited, CAE inc, and Bombardier inc.

Further thanks to Yann Pequignot for his diligent proof reading of the manuscript and valuable insights.

References

Kendall Atkinson and Weimin Han. *Theoretical numerical analysis*, volume 39. Springer, 2005.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 1498712169.

Matthias Hein and Olivier Bousquet. Kernels, associated structures and generalizations. 2004.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

Charles A Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural computation*, 17(1):177–204, 2005.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008. URL `https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf`.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf`.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: from theory to learning algorithms*. Cambridge University Press, New York, NY, USA, 2014. ISBN 978-1-107-05713-5.

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Vladimir Vovk. Kernel ridge regression. In *Empirical inference*, pages 105–116. Springer, 2013.

Appendix A. Proofs

Theorem 8 Assume A1, A2, A3. Consider any $\alpha \in \mathcal{H}$. Then we have with probability at least $1-\delta$ over the choice of $\mathcal{S} \sim \mathcal{D}^m$ that the following holds for all $\beta \in \mathcal{H}$ with $\|\beta - \alpha\|_{\mathcal{H}} \leq C$:

$$\mathcal{L}_{\mathcal{D}}(\Lambda\beta) - \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\rho\theta C}{\sqrt{m}}\left(2 + \sqrt{2\log\frac{1}{\delta}}\right).$$

Proof First, notice that:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(\Lambda\beta) &= \mathcal{L}_{\mathcal{D}}(\Lambda\beta) - \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\ell(\Lambda\beta(x), y) - \ell(\Lambda\alpha(x), y)\Big] + \mathcal{L}_{\mathcal{D}}(\Lambda\alpha). \end{aligned} \tag{78}$$

We can bound the expectation of Equation (78) using Rademacher theory. Define the following class of functions:

$$\mathcal{H}_\alpha := \Big\{(x,y) \mapsto \ell(\Lambda\beta(x), y) - \ell(\Lambda\alpha(x), y) \;\Big|\; \beta \in \mathcal{H} \wedge \|\beta - \alpha\|_{\mathcal{H}} \leq C\Big\}. \tag{79}$$

By Lemma 7 and the ρ-Lipschitzness of ℓ, we have:

$$\begin{aligned}|\ell(\Lambda\beta(x), y) - \ell(\Lambda\alpha(x), y)| &\leq \rho|\Lambda\beta(x) - \Lambda\alpha(x)| \\ &\leq \rho\theta\|\beta - \alpha\|_{\mathcal{H}} \\ &\leq \rho\theta C.\end{aligned}$$

Therefore, the functions in $\mathcal{H}_\alpha$ take their values in $[-\rho\theta C, \rho\theta C]$. By Theorem 3.1 of Mohri et al. (2012) (generalized to losses taking values in $[-M, M]$), we have with probability at least $1-\delta$ on the choice of sample $\mathcal{S} \sim \mathcal{D}^m$ that the following expression is valid for all β such that $\|\beta - \alpha\|_{\mathcal{H}} \leq C$:

$$\begin{aligned}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\ell(\Lambda\beta(x), y) - \ell(\Lambda\alpha(x), y)\Big] &\leq \frac{1}{m}\sum_{i=1}^{m}\Big(\ell(\Lambda\beta(x_i), y_i) - \ell(\Lambda\alpha(x_i), y_i)\Big) + 2\mathcal{R}_m(\mathcal{H}_\alpha) + 2\rho\theta C\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}} \\ &= \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + 2\mathcal{R}_m(\mathcal{H}_\alpha) + 2\rho\theta C\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}}.\end{aligned} \tag{80}$$

All that is left if to calculate the Rademacher complexity of $\mathcal{H}_\alpha$:

$$\begin{aligned}
\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{H}_\alpha) &= \widehat{\mathcal{R}}_{\mathcal{S}}\Big(\Big\{(x,y) \mapsto \ell(\Lambda\beta(x),y) - \ell(\Lambda\alpha(x),y) \;\Big|\; \beta \in \mathcal{H} \wedge \|\beta-\alpha\|_{\mathcal{H}} \leq C\Big\}\Big) \\
&= \widehat{\mathcal{R}}_{\mathcal{S}}\Big(\Big\{(x,y) \mapsto \ell(\Lambda\beta(x),y) \;\Big|\; \beta \in \mathcal{H} \wedge \|\beta-\alpha\|_{\mathcal{H}} \leq C\Big\}\Big) \\
&\qquad\qquad \text{(Lemma 26.6 of Shalev-Shwartz and Ben-David (2014))} \\
&\leq \rho\widehat{\mathcal{R}}_{\mathcal{S}}\Big(\Big\{\Lambda\beta \;\Big|\; \beta \in \mathcal{H} \wedge \|\beta-\alpha\|_{\mathcal{H}} \leq C\Big\}\Big) \qquad \text{(Talagrand's lemma)} \\
&= \rho\widehat{\mathcal{R}}_{\mathcal{S}}\Big(\Big\{\Lambda\beta - \Lambda\alpha \;\Big|\; \beta \in \mathcal{H} \wedge \|\beta-\alpha\|_{\mathcal{H}} \leq C\Big\}\Big) \\
&\qquad\qquad \text{(Lemma 26.6 of Shalev-Shwartz and Ben-David (2014))} \\
&= \rho\widehat{\mathcal{R}}_{\mathcal{S}}\Big(\Big\{\Lambda\beta' \;\Big|\; \beta' \in \mathcal{H} \wedge \|\beta'\|_{\mathcal{H}} \leq C\Big\}\Big) \\
&\leq \frac{\rho\theta C}{\sqrt{m}}. \qquad \text{(Theorem 9)}
\end{aligned}$$

By taking the expectation over $\mathcal{S}$, we get $\mathcal{R}_m(\mathcal{H}_\alpha) = \frac{\rho\theta C}{\sqrt{m}}$.

Because the β resulting from solving Equation (60) is the one that we are interested in, we can take $C = \|\beta-\alpha\|_{\mathcal{H}}$ to get the tightest possible bound. Assembling Equations (78), (80) and the Rademacher complexity of $\mathcal{H}_\alpha$, we obtain the following bound on the risk of $\Lambda\beta$, valid with probability at least $1-\delta$ over the choice of sample:

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}(\Lambda\beta) &\leq \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{2\rho\theta\|\beta-\alpha\|_{\mathcal{H}}}{\sqrt{m}} + 2\rho\theta\|\beta-\alpha\|_{\mathcal{H}}\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}} \\
&= \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \mathcal{L}_{\mathcal{S}}(\Lambda\beta) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\rho\theta\|\beta-\alpha\|_{\mathcal{H}}}{\sqrt{m}}\left(2 + \sqrt{2\log\left(\frac{1}{\delta}\right)}\right). \qquad (81)
\end{aligned}$$

■

Lemma 13 *Consider some independent identically distributed variables $(z_1, \ldots, z_m)$ taken from a Hilbert space. Then:*

$$\mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m} z_i\right\|^2\right] \leq \mathbb{E}\left[\|z_1\|^2\right].$$

Proof We have:

$$\begin{aligned}
\mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m} z_i\right\|^2\right] &= \frac{1}{m^2}\,\mathbb{E}\left[\left\langle\sum_{i=1}^{m} z_i, \sum_{j=1}^{m} z_j\right\rangle\right] \\
&= \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}\left[\langle z_i, z_j\rangle\right] && \text{(Linearity)} \\
&\leq \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}\left[\|z_i\|\|z_j\|\right] && \text{(Cauchy–Schwartz)} \\
&= \frac{1}{m^2}\sum_{i=1}^{m}\left[\mathbb{E}\left[\|z_i\|^2\right] + \sum_{j\neq i}\mathbb{E}\left[\|z_i\|\right]\mathbb{E}\left[\|z_j\|\right]\right] && \text{(Independence)} \\
&= \frac{1}{m^2}\sum_{i=1}^{m}\left[\mathbb{E}\left[\|z_1\|^2\right] + (m-1)\Big(\mathbb{E}\left[\|z_1\|\right]\Big)^2\right] && \text{(Identical distribution)} \\
&\leq \frac{1}{m^2}\sum_{i=1}^{m}\left[\mathbb{E}\left[\|z_1\|^2\right] + (m-1)\,\mathbb{E}\left[\|z_1\|^2\right]\right] && \text{(Jensen)} \\
&= \mathbb{E}\left[\|z_1\|^2\right].
\end{aligned}$$

■

Theorem 11 Assume A1, A2, A3, A4. The output $\bar{\alpha}$ of Algorithm 1 is such that:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] - \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \frac{2\rho\theta B}{\sqrt{m}} + \lambda B^2 + \frac{8\rho^2}{\lambda m} + \frac{(\rho\kappa+\lambda B)^2}{2\lambda T}(1+\log(T)), \tag{82}$$

where the expectation is taken over the choice of sample and all sampled parameters and batches (i.e. $\mathcal{S}\sim\mathcal{D}^m, (w_1,\ldots,w_T)\sim p^T, (\mathbf{s}_1,\ldots,\mathbf{s}_T)\sim\mathcal{U}(\mathcal{S})^{T\times|\mathbf{s}|}$). By choosing $\lambda = \sqrt{\frac{8\rho^2}{B^2 m}}$, we obtain:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] - \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) \leq \frac{\rho B}{\sqrt{m}}\left(\sqrt{32} + 2\theta + \frac{m}{\sqrt{32T}}\left(\kappa + \sqrt{\frac{8}{m}}\right)^2(1+\log(T))\right). \tag{83}$$

Proof First, consider any sample $\mathcal{S}$ and denote:

$$A_\alpha(\mathcal{S}) := \operatorname*{argmin}_{\alpha\in\mathcal{H}_B}\left(\mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\lambda}{2}\|\alpha\|_{\mathcal{H}}^2\right), \tag{84}$$

and:

$$A(\mathcal{S}) := \Lambda A_\alpha(\mathcal{S}) = \operatorname*{argmin}_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\left(\mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\lambda}{2}\|\alpha\|_{\mathcal{H}}^2\right), \tag{85}$$

the regularized empirical risk minimizer. Then, write:

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] = &\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha})\right] \\ &+ \mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(A(\mathcal{S}))\right] \\ &+ \mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(A(\mathcal{S})) - \mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right] \\ &+ \mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right]. \end{aligned} \tag{86}$$

(Every expectation in this proof is over the sample and all sampled parameters and batches, unless specified otherwise.) We can bound each of these terms separately. First, Lemma 26.2 of Shalev-Shwartz and Ben-David (2014) applied to Equation (72) gives us:

$$\mathop{\mathbb{E}}_{\mathcal{S}\sim\mathcal{D}^m}\left[\sup_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\left(\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) - \mathcal{L}_{\mathcal{S}}(\Lambda\alpha)\right)\right] \leq \frac{2B\rho\theta}{\sqrt{m}}. \tag{87}$$

Being a bound on the supremum, it is also valid for $\Lambda\bar{\alpha}$, independently of the sampled parameters and batches, which gives us:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha})\right] \leq \frac{2\rho\theta B}{\sqrt{m}}. \tag{88}$$

Next, the proof of Corollary 13.6 of Shalev-Shwartz and Ben-David (2014) can be modified by swapping the two terms on the left-hand side of equation 13.11, which allows us to get :

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(A(\mathcal{S})) - \mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right] = \mathop{\mathbb{E}}_{\mathcal{S}\sim\mathcal{D}^m}\left[\mathcal{L}_{\mathcal{S}}(A(\mathcal{S})) - \mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right] \leq \frac{4\rho^2}{\lambda m}. \tag{89}$$

Also, Corollary 13.8 of Shalev-Shwartz and Ben-David (2014) directly gives us :

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right] = \mathop{\mathbb{E}}_{\mathcal{S}\sim\mathcal{D}^m}\left[\mathcal{L}_{\mathcal{D}}(A(\mathcal{S}))\right] \leq \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{\lambda}{2}B^2 + \frac{4\rho^2}{\lambda m}. \tag{90}$$

Bounding the term $\mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(A(\mathcal{S}))\right]$ requires more work. We seek to apply Theorem 14.11 of Shalev-Shwartz and Ben-David (2014) to the regularized empirical risk $\mathcal{L}_{\mathcal{S}}(\Lambda\alpha) + \frac{\lambda}{2}\|\alpha\|^2_{\mathcal{H}}$, which is λ-strongly convex (by convexity of ℓ and linearity in α of $\Lambda\alpha$). At iteration t of Algorithm 1, the subgradient (unbiased gradient approximation) is given by Equation (35):

$$v_t := v(\alpha^{(t-1)}, w_t, \mathbf{s}_t) := \left(\frac{1}{|\mathbf{s}_t|}\sum_{(x,y)\in\mathbf{s}_t}\ell'(\Lambda\alpha^{(t-1)}(x), y)\phi(w_t, x)\right)\mathcal{K}(w_t, \cdot) + \lambda\alpha^{(t-1)}.$$

We need to bound $\mathbb{E}\left[\|v_t\|^2\right]$. First, write:

$$u_t := v_t - \lambda\alpha^{(t-1)} = \frac{1}{|\mathbf{s}_t|}\sum_{(x,y)\in\mathbf{s}_t}\ell'(\Lambda\alpha^{(t-1)}(x), y)\phi(w_t, x)\mathcal{K}(w_t, \cdot).$$

For the purpose of finding an upper bound, we can assume that the batch size is 1, using Lemma 13. Writing $\mathbf{s}_t = \{(x_t, y_t)\}$, we have:

$$\begin{aligned}
\mathbb{E}\left[\|u_t\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{|\mathbf{s}_t|}\sum_{(x,y)\in\mathbf{s}_t}\ell'(\Lambda\alpha^{(t-1)}(x), y)\phi(w_t, x)\mathcal{K}(w_t, \cdot)\right\|_{\mathcal{H}}^2\right] \\
&= \mathbb{E}\left[\left\|\ell'(\Lambda\alpha^{(t-1)}(x_t), y_t)\phi(w_t, x_t)\mathcal{K}(w_t, \cdot)\right\|_{\mathcal{H}}^2\right] \\
&\leq \mathbb{E}\left[\rho^2\|\mathcal{K}(w_t, \cdot)\phi(w_t, x_t)\|_{\mathcal{H}}^2\right] \\
&= \rho^2\,\mathbb{E}\left[\mathcal{K}(w_t, w_t)\phi(w_t, x_t)^2\right] \\
&\leq \rho^2\kappa^2. \qquad (91)
\end{aligned}$$

Next, because of the projection step in Algorithm 1, we have $\alpha^{(t-1)} \in \Lambda\mathcal{H}_B$ for every t, i.e. $\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}} \leq B$. This allows us to write:

$$\begin{aligned}
\mathbb{E}\left[\|v_t\|^2\right] &= \mathbb{E}\left[\left\|u_t + \lambda\alpha^{(t-1)}\right\|_{\mathcal{H}}^2\right] \\
&\leq \mathbb{E}\left[\left(\|u_t\|_{\mathcal{H}} + \lambda\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}}\right)^2\right] && \text{(Triangle inequality)} \\
&= \mathbb{E}\left[\|u_t\|_{\mathcal{H}}^2 + 2\lambda\|u_t\|_{\mathcal{H}}\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}} + \lambda^2\left\|\alpha^{(t-1)}\right\|_{\mathcal{H}}^2\right] \\
&\leq \mathbb{E}\left[\|u_t\|_{\mathcal{H}}^2\right] + 2\lambda B\,\mathbb{E}\left[\|u_t\|_{\mathcal{H}}\right] + \lambda^2B^2 \\
&\leq \rho^2\kappa^2 + 2\lambda B\sqrt{\mathbb{E}\left[\|u_t\|_{\mathcal{H}}^2\right]} + \lambda^2B^2 && \text{(Jensen)} \\
&\leq \rho^2\kappa^2 + 2\lambda B\rho\kappa + \lambda^2B^2 \\
&= (\rho\kappa + \lambda B)^2.
\end{aligned}$$

Now, denote:

Applying Theorem 14.11 of Shalev-Shwartz and Ben-David (2014), we get:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(A(\mathcal{S}))\right] \leq \mathbb{E}\left[\frac{\lambda}{2}\|\bar{\alpha}\|_{\mathcal{H}}^2 - \frac{\lambda}{2}\|A_\alpha(\mathcal{S})\|_{\mathcal{H}}^2\right] + \frac{(\rho\kappa + \lambda B)^2}{2\lambda T}(1 + \log(T)). \qquad (92)$$

We can simplify this expression by noticing that $\frac{\lambda}{2}\|\bar{\alpha}\|_{\mathcal{H}}^2 - \frac{\lambda}{2}\|A_\alpha(\mathcal{S})\|_{\mathcal{H}}^2$ is at most $\frac{\lambda}{2}B^2$:

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{S}}(\Lambda\bar{\alpha}) - \mathcal{L}_{\mathcal{S}}(A(\mathcal{S}))\right] \leq \frac{\lambda}{2}B^2 + \frac{(\rho\kappa + \lambda B)^2}{2\lambda T}(1 + \log(T)). \qquad (93)$$

Inserting Equations (88), (89), (90) and (93) into Equation (86), we obtain the first part of the theorem :

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] \leq \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B}\mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{2\rho\theta B}{\sqrt{m}} + \lambda B^2 + \frac{8\rho^2}{\lambda m} + \frac{(\rho\kappa + \lambda B)^2}{2\lambda T}(1 + \log(T)). \qquad (94)$$

Taking $\lambda = \sqrt{\frac{8\rho^2}{B^2 m}}$, we get the second part of the theorem :

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}_{\mathcal{D}}(\Lambda\bar{\alpha})\right] &\leq \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{2\rho\theta B}{\sqrt{m}} + \lambda B^2 + \frac{8\rho^2}{\lambda m} + \frac{(\rho\kappa+\lambda B)^2}{2\lambda T}(1+\log(T)) \\
&= \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{2\rho\theta B}{\sqrt{m}} + 2\sqrt{\frac{8\rho^2 B^2}{m}} + \sqrt{\frac{B^2 m}{32\rho^2 T^2}}\left(\rho\kappa + \sqrt{\frac{8\rho^2}{m}}\right)^2 (1+\log(T)) \\
&= \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{2\rho\theta B}{\sqrt{m}} + \sqrt{\frac{32\rho^2 B^2}{m}} + \sqrt{\frac{\rho^2 B^2}{m}}\left(\frac{m}{\sqrt{32}T}\left(\kappa + \sqrt{\frac{8}{m}}\right)^2 (1+\log(T))\right) \\
&= \min_{\Lambda\alpha\in\Lambda\mathcal{H}_B} \mathcal{L}_{\mathcal{D}}(\Lambda\alpha) + \frac{\rho B}{\sqrt{m}}\left(\sqrt{32} + 2\theta + \frac{m}{\sqrt{32}T}\left(\kappa + \sqrt{\frac{8}{m}}\right)^2 (1+\log(T))\right).
\end{aligned}$$

■

Appendix B. Calculus

In this appendix, we present all the calculus required to calculate the expectation $\mathbb{E}_{w\sim p}\left[\mathcal{K}(u,w)\phi(w,x)\right]$ (Equation (24)), as well as upper bound the constant θ (Equation (26)).

Lemma 14 *Consider a Hilbert space $\mathcal{W}$. Let $u, w \in \mathcal{W}$ and $a, b > 0$. Then:*

$$a\|w-u\|^2 + b\|w\|^2 = (a+b)\left\|w - \frac{a}{a+b}u\right\|^2 + \frac{ab}{a+b}\|u\|^2.$$

Proof We have:

$$\begin{aligned}
a\|w-u\|^2 + b\|w\|^2 &= a\|w\|^2 - 2a\langle w,u\rangle + a\|u\|^2 + b\|w\|^2 \\
&= (a+b)\|w\|^2 - 2a\langle w,u\rangle + a\|u\|^2 \\
&= (a+b)\|w\|^2 - 2a\langle w,u\rangle + \frac{a^2}{a+b}\|u\|^2 - \frac{a^2}{a+b}\|u\|^2 + a\|u\|^2 \\
&= (a+b)\left[\|w\|^2 - 2\frac{a}{a+b}\langle w,u\rangle + \frac{a^2}{(a+b)^2}\|u\|^2\right] - \frac{a^2}{a+b}\|u\|^2 + a\|u\|^2 \\
&= (a+b)\left\|w - \frac{a}{a+b}u\right\|^2 + \frac{ab}{a+b}\|u\|^2.
\end{aligned}$$

■

Lemma 15 *Consider a Hilbert space $\mathcal{W}$. Let $u, w, w_0 \in \mathcal{W}$ and $a, b > 0$. Then:*

$$\frac{\|w-u\|^2}{a} + \frac{\|w\|^2}{b} = \left(\frac{1}{a}+\frac{1}{b}\right)\left\|w - \frac{1}{1+\frac{a}{b}}u\right\|^2 + \frac{1}{a+b}\|u\|^2.$$

Proof We have:

$$\begin{aligned}\frac{\|w-u\|^2}{a}+\frac{\|w\|^2}{b} &= \left(\frac{1}{a}+\frac{1}{b}\right)\left\|w-\frac{1}{a\left(\frac{1}{a}+\frac{1}{b}\right)}u\right\|^2+\frac{1}{ab\left(\frac{1}{a}+\frac{1}{b}\right)}\|u\|^2 \qquad \text{(Lemma 14)}\\ &= \left(\frac{1}{a}+\frac{1}{b}\right)\left\|w-\frac{1}{1+\frac{a}{b}}u\right\|^2+\frac{1}{a+b}\|u\|^2.\end{aligned}$$

■

Lemma 16 *We have:*

$$\int_{\mathbb{R}^n} e^{-a\|w-u\|^2}\mathrm{d}w = \left(\frac{\pi}{a}\right)^{\frac{n}{2}}.$$

Proof This is the unnormalized integral of a Gaussian density with variance $\frac{1}{2a}I$ and mean u. ■

Instantiation 1

Consider instantiation 1. We will work up to the full integral in $\mathbb{R}^n$ through a series of lemmas.

Lemma 17 *We have:*

$$\int_{-\infty}^{\infty} e^{-(w-u)^2/2\gamma^2}\operatorname{sign}(wx)\mathrm{d}w = \sqrt{2\pi}\gamma\operatorname{sign}(x)\operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right).$$

Proof Applying an adequate change of variable causes the error function to appear:

$$\begin{aligned}
&\int_{-\infty}^{\infty} e^{-(w-u)^2/2\gamma^2} \operatorname{sign}(wx)\mathrm{d}w \\
&\qquad = \operatorname{sign}(x) \int_{-\infty}^{\infty} e^{-(w-u)^2/2\gamma^2} \operatorname{sign}(w)\mathrm{d}w \\
&\qquad = \operatorname{sign}(x)\left(\int_{0}^{\infty} e^{-(w-u)^2/2\gamma^2}\mathrm{d}w - \int_{-\infty}^{0} e^{-(w-u)^2/2\gamma^2}\mathrm{d}w\right) \\
&\qquad = \sqrt{2}\gamma \operatorname{sign}(x)\left(\int_{\frac{-u}{\sqrt{2}\gamma}}^{\infty} e^{-t^2}\mathrm{d}w - \int_{-\infty}^{\frac{-u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w\right) \qquad (t := \tfrac{w-u}{\sqrt{2}\gamma},\ \mathrm{d}w = \tfrac{\mathrm{d}w}{\sqrt{2}\gamma}) \\
&\qquad = \sqrt{2}\gamma \operatorname{sign}(x)\left(\int_{0}^{\infty} e^{-t^2}\mathrm{d}w + \int_{\frac{-u}{\sqrt{2}\gamma}}^{0} e^{-t^2}\mathrm{d}w - \int_{-\infty}^{0} e^{-t^2}\mathrm{d}w - \int_{0}^{\frac{-u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w\right) \\
&\qquad = \sqrt{2}\gamma \operatorname{sign}(x)\left(\int_{\frac{-u}{\sqrt{2}\gamma}}^{0} e^{-t^2}\mathrm{d}w - \int_{0}^{\frac{-u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w\right) \\
&\qquad = \sqrt{2}\gamma \operatorname{sign}(x)\left(\int_{0}^{\frac{u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w + \int_{0}^{\frac{u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w\right) \\
&\qquad = 2\sqrt{2}\gamma \operatorname{sign}(x) \int_{0}^{\frac{u}{\sqrt{2}\gamma}} e^{-t^2}\mathrm{d}w \\
&\qquad = 2\sqrt{2}\gamma \operatorname{sign}(x)\frac{\sqrt{\pi}}{2} \operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right) \\
&\qquad = \sqrt{2\pi}\gamma \operatorname{sign}(x) \operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right).
\end{aligned}$$

■

Lemma 18 *We have:*

$$\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} \operatorname{sign}(\langle w, x\rangle)\mathrm{d}w = \left(\sqrt{2\pi}\gamma\right)^n \operatorname{erf}\left(\frac{\langle u, x\rangle}{\sqrt{2}\gamma\|x\|}\right).$$

Proof Calculate the integral using an orthonormal basis $\{v_1, \ldots, v_n\}$ of $\mathbb{R}^n$ such that $v_n := \frac{x}{\|x\|}$. Write $w = (w_1, \ldots, w_n)$ in this new basis (i.e. $w_i := \langle w, v_i\rangle$ for all i), and similarly $(u_1, \ldots, u_n)$ for u. Under this change of coordinates, the integral becomes:

$$\begin{aligned}
&\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} \operatorname{sign}(\langle w, x\rangle)\mathrm{d}w \\
&\qquad = \int_{\mathbb{R}} \int_{\mathbb{R}^{n-1}} e^{-\left[\sum_{i=1}^{n-1}(w_i-u_i)^2 + (w_n-u_n)^2\right]/2\gamma^2} \operatorname{sign}(w_n\|x\|)\mathrm{d}w_1, \ldots, \mathrm{d}w_{n-1}\mathrm{d}w_n.
\end{aligned}$$

We are left with a product of n independent integrals:

$$\begin{aligned}\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2}\operatorname{sign}(\langle w,x\rangle)\mathrm{d}w \\ &= \int_{\mathbb{R}^{n-1}} e^{-\sum_{i=1}^{n-1}(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_1\ldots\mathrm{d}w_{n-1}\int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2}\operatorname{sign}(w_n\|x\|)\mathrm{d}w_n \\ &= \int_{\mathbb{R}^{n-1}} \prod_{i=1}^{n-1} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_1\ldots\mathrm{d}w_{n-1}\int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2}\operatorname{sign}(w_n\|x\|)\mathrm{d}w_n \\ &= \prod_{i=1}^{n-1}\int_{\mathbb{R}} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_i\int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2}\operatorname{sign}(w_n\|x\|)\mathrm{d}w_n.\end{aligned}$$

For each i, Lemma 16 gives us:

$$\int_{\mathbb{R}} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_i = \sqrt{2\pi}\gamma.$$

Also, Lemma 17 gives us:

$$\int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2}\operatorname{sign}(w_n\|x\|)\mathrm{d}w_n = \sqrt{2\pi}\gamma\operatorname{erf}\left(\frac{u_n}{\sqrt{2}\gamma}\right).$$

Finally, since $u_n = \frac{\langle u,x\rangle}{\|x\|}$, we have the result. ■

Theorem 19 *Considering instantiation 1, we have:*

$$\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] = \left(1+\tfrac{\sigma^2}{\gamma^2}\right)^{-n/2} e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}}\operatorname{erf}\left(\frac{\langle u',x\rangle}{\sqrt{2}\zeta\|x\|_2}\right),$$

where ζ is defined by the relationship:

$$\frac{1}{2\zeta^2} = \frac{1}{2\gamma^2}+\frac{1}{2\sigma^2},$$

and:

$$u' := \left(1+\tfrac{\gamma^2}{\sigma^2}\right)^{-1}u.$$

Proof The proof is simply completing the square at the exponent and applying Lemma 18. We have :

$$\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2}e^{-\|w\|^2/2\sigma^2}\operatorname{sign}(\langle w,x\rangle)\mathrm{d}w.$$

Lemma 15 gives us :

$$\begin{aligned}-\|w-u\|^2/2\gamma^2-\|w\|^2/2\sigma^2 &= -\frac{1}{2\zeta^2}\left\|w-\left(1+\frac{\gamma^2}{\sigma^2}\right)^{-1}u\right\|^2-\left(\frac{1}{2\sigma^2+2\gamma^2}\right)\|u\|^2 \\ &= -\frac{1}{2\zeta^2}\|w-u'\|^2-\left(\frac{1}{2\sigma^2+2\gamma^2}\right)\|u\|^2.\end{aligned}$$

Therefore, we have:

$$\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\|u\|^2/(2\gamma^2+2\sigma^2)} \int_{\mathbb{R}^n} e^{-\|w-u'\|^2/2\zeta^2}\,\mathrm{sign}(\langle w,x\rangle)\mathrm{d}w.$$

Applying Lemma 18, we get:

$$\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] = \left(\frac{\zeta}{\sigma}\right)^n e^{-\|u\|^2/(2\gamma^2+2\sigma^2)}\,\mathrm{erf}\left(\frac{\langle u',x\rangle}{\sqrt{2}\zeta\|x\|}\right).$$

We obtain the final result by noticing that.

$$\left(\frac{\zeta}{\sigma}\right)^n = \left(1+\frac{\sigma^2}{\gamma^2}\right)^{-n/2}.$$

■

Before calculating θ, we start by a few lemmas.

Lemma 20 *Consider $\sigma > 0$ and $\gamma > 0$. Then:*

$$\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\|u-w\|^2/2\gamma^2} e^{-\langle u,w\rangle/\sigma^2}\mathrm{d}u\mathrm{d}w = (2\pi)^n\left(\frac{\gamma^2\sigma^4}{2\sigma^2-\gamma^2}\right)^{n/2}. \tag{95}$$

Proof

$$\begin{aligned}
\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w &= \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\langle t+w,w\rangle}{\sigma^2}}\mathrm{d}t\mathrm{d}w \qquad (t := u-w,\ \mathrm{d}t = \mathrm{d}u)\\
&= \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\|w\|^2}{\sigma^2}} e^{-\frac{\langle t,w\rangle}{\sigma^2}}\mathrm{d}t\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\langle t,w\rangle}{\sigma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\left\|t+\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\left\|t+\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \left(\sqrt{2\pi\gamma^2}\right)^n \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}w.
\end{aligned}$$

Then, simplifying the exponent:

$$-\frac{\|w\|^2}{\sigma^2} + \frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2} = -\frac{\|w\|^2}{2\sigma^2}\left(2-\frac{\gamma^2}{\sigma^2}\right) = -\frac{\|w\|^2}{2\sigma^2}\left(\frac{2\sigma^2-\gamma^2}{\sigma^2}\right) = -\frac{\|w\|^2}{2\sigma^4}(2\sigma^2-\gamma^2),$$

we get:

$$\begin{aligned}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}} \mathrm{d}u\mathrm{d}w &= \left(\sqrt{2\pi\gamma^2}\right)^n \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{2\sigma^4}\left(2\sigma^2-\gamma^2\right)} \mathrm{d}w \\ &= \left(\sqrt{2\pi\gamma^2}\right)^n \left(\sqrt{2\pi\frac{\sigma^4}{2\sigma^2-\gamma^2}}\right)^n \\ &= (2\pi)^n \left(\frac{\gamma^2\sigma^4}{2\sigma^2-\gamma^2}\right)^{n/2}.\end{aligned}$$

■

Lemma 21 *Consider $\sigma > 0$ and $\gamma > 0$. Denote I_n the identity matrix in $\mathbb{R}^n$. Then:*

$$\mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,\sigma^2 I_n)} \mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,\sigma^2 I_n)} \left[e^{-\|u-w\|^2/2\gamma^2}\right] = \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-n/2}. \tag{96}$$

Proof The expectation is a straightforward integral:

$$\begin{aligned}\mathop{\mathbb{E}}_{w\sim p} \mathop{\mathbb{E}}_{u\sim p} [\mathcal{K}(u,w)] &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n} \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\|u\|^2}{2\sigma^2}} e^{-\frac{\|w\|^2}{2\sigma^2}} \mathrm{d}u\mathrm{d}w \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n} \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\|u\|^2}{2\sigma^2}} e^{\frac{\langle u,w\rangle}{\sigma^2}} e^{-\frac{\|w\|^2}{2\sigma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}} \mathrm{d}u\mathrm{d}w \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n} \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\|u-w\|^2}{2\sigma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}} \mathrm{d}u\mathrm{d}w \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n} \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\zeta^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}} \mathrm{d}u\mathrm{d}w \ \left(\tfrac{1}{2\zeta^2} = \tfrac{1}{2\gamma^2} + \tfrac{1}{2\sigma^2} = \tfrac{\sigma^2+\gamma^2}{2\sigma^2\gamma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n} (2\pi)^n \left(\frac{\zeta^2\sigma^4}{2\sigma^2-\zeta^2}\right)^{n/2} \qquad \text{(Lemma 20)} \\ &= \left(\frac{\zeta^2}{2\sigma^2-\zeta^2}\right)^{n/2} \\ &= \left(\frac{2\sigma^2}{\zeta^2}-1\right)^{-n/2} \\ &= \left(\frac{2\sigma^2}{\sigma^2}+\frac{2\sigma^2}{\gamma^2}-1\right)^{-n/2} \\ &= \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-n/2}.\end{aligned}$$

■

Lemma 22 *Considering instantiation 1, we have:*

$$\theta \leq \left(1 + \tfrac{2\sigma^2}{\gamma^2}\right)^{-n/4}. \tag{97}$$

Proof We have:

$$\begin{aligned}
\theta^2 &= \sup_{x \in \mathcal{X}} \|\psi(x)\|_{\mathcal{H}}^2 \\
&= \sup_{x \in \mathcal{X}} \mathop{\mathbb{E}}_{w \sim p} \mathop{\mathbb{E}}_{u \sim p} [\mathcal{K}(u,w)\phi(u,x)\phi(w,x)] \\
&\leq \sup_{x \in \mathcal{X}} \mathop{\mathbb{E}}_{w \sim p} \mathop{\mathbb{E}}_{u \sim p} [|\mathcal{K}(u,w)\phi(u,x)\phi(w,x)|] \\
&= \mathop{\mathbb{E}}_{w \sim p} \mathop{\mathbb{E}}_{u \sim p} [\mathcal{K}(u,w)]. \qquad (|\phi(w,x)| = 1 \text{ for all } w \text{ and } x)
\end{aligned}$$

The result is given by Lemma 21 (and taking the square root). ■

Instantiation 2

Consider instantiation 2. Let's calculate the expectation.

Lemma 23 *Considering instantiation 2, we have:*

$$\mathop{\mathbb{E}}_{w \sim p} [\mathcal{K}(u,w)\phi(w,x)] = \frac{\zeta}{\sigma n} e^{\frac{-u_2^2}{2\sigma^2 + 2\gamma^2}} \operatorname{erf}\left(\frac{x_{u_1} - u_2'}{\sqrt{2}\zeta}\right),$$

where ζ is defined by the relationship:

$$\frac{1}{2\zeta^2} = \frac{1}{2\gamma^2} + \frac{1}{2\sigma^2},$$

and:

$$u_2' := \left(1 + \tfrac{\gamma^2}{\sigma^2}\right)^{-1} u_2.$$

Proof We have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] &= \mathop{\mathbb{E}}_{w_1\sim\mathcal{U}(\{1,\dots,n\})}\mathop{\mathbb{E}}_{w_2\sim\mathcal{N}(0,\sigma^2)}[\mathcal{K}(u,w)\phi(w,x)] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{w_2\sim\mathcal{N}(0,\sigma^2)}\Big[\mathbb{1}[i=u_1]e^{-(w_2-u_2)^2/2\gamma^2}\operatorname{sign}(x_i-w_2)\Big] \\
&= \frac{1}{n}\mathop{\mathbb{E}}_{w_2\sim\mathcal{N}(0,\sigma^2)}\Big[e^{-(w_2-u_2)^2/2\gamma^2}\operatorname{sign}(x_{u_1}-w_2)\Big] \\
&= \frac{1}{n}\frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{\infty}e^{-(w_2-u_2)^2/2\gamma^2}e^{-w^2/2\sigma^2}\operatorname{sign}(x_{u_1}-w_2)\mathrm{d}w_2 \\
&= \frac{1}{n\sqrt{2\pi\sigma^2}}e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}\int_{-\infty}^{\infty}e^{-(w_2-u_2')^2/2\zeta^2}\operatorname{sign}(x_{u_1}-w_2)\mathrm{d}w_2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(Lemma 15)} \\
&= \frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n\sqrt{2\pi\sigma^2}}\left[\int_{-\infty}^{x_{u_1}}e^{-(w_2-u_2')^2/2\zeta^2}\mathrm{d}w_2-\int_{x_{u_1}}^{\infty}e^{-(w_2-u_2')^2/2\zeta^2}\mathrm{d}w_2\right] \\
&= \frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n\sqrt{2\pi\sigma^2}}\sqrt{2}\zeta\left[\int_{-\infty}^{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}e^{-t^2}\mathrm{d}w-\int_{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}^{\infty}e^{-t^2}\mathrm{d}w\right] \\
&\qquad\qquad\qquad\qquad\qquad\left(t:=\tfrac{w_2-u_2'}{\sqrt{2}\zeta},\ \mathrm{d}w=\tfrac{\mathrm{d}w_2}{\sqrt{2}\zeta}\right) \\
&= \frac{\zeta}{\sigma}\frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n\sqrt{\pi}}\left[\int_{-\infty}^{0}e^{-t^2}\mathrm{d}w+\int_{0}^{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}e^{-t^2}\mathrm{d}w-\int_{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}^{0}e^{-t^2}\mathrm{d}w-\int_{0}^{\infty}e^{-t^2}\mathrm{d}w\right] \\
&= \frac{\zeta}{\sigma}\frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n\sqrt{\pi}}\left[\int_{0}^{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}e^{-t^2}\mathrm{d}w-\int_{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}^{0}e^{-t^2}\mathrm{d}w\right].
\end{aligned}$$

Finally, we have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] &= \frac{\zeta}{\sigma}\frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n\sqrt{\pi}}\left[\int_{0}^{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}e^{-t^2}\mathrm{d}w-\int_{\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}}^{0}e^{-t^2}\mathrm{d}w\right] \\
&= \frac{\zeta}{\sigma}\frac{e^{\frac{-u_2^2}{2\sigma^2+2\gamma^2}}}{n}\operatorname{erf}\left(\frac{x_{u_1}-u_2'}{\sqrt{2}\zeta}\right).
\end{aligned}$$

■

Lemma 24 *Considering instantiation 2, we have:*

$$\theta \leq \frac{1}{\sqrt{n}}\Big(1+\tfrac{2\sigma^2}{\gamma^2}\Big)^{-1/4}. \tag{98}$$

Proof We have:

$$\begin{aligned}
\theta^2 &= \sup_{x\in\mathcal{X}} \|\psi(x)\|_{\mathcal{H}}^2 \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{w\sim p} \mathop{\mathbb{E}}_{u\sim p} [\mathcal{K}(u,w)\phi(u,x)\phi(w,x)] \\
&\leq \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{w\sim p} \mathop{\mathbb{E}}_{u\sim p} [|\mathcal{K}(u,w)\phi(u,x)\phi(w,x)|] \\
&= \mathop{\mathbb{E}}_{w\sim p} \mathop{\mathbb{E}}_{u\sim p} [\mathcal{K}(u,w)] && (|\phi(w,x)| = 1 \text{ for all } w \text{ and } x) \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,\sigma^2)} \mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,\sigma^2)} \left[\mathbb{1}[i=j] e^{-(u-w)/2\gamma^2}\right] \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,\sigma^2)} \mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,\sigma^2)} \left[e^{-(u-w)/2\gamma^2}\right] \\
&= \frac{1}{n^2} \sum_{i=1}^{n} \left(1 + \tfrac{2\sigma^2}{\gamma^2}\right)^{-1/2} && \text{(See proof of Lemma 22)} \\
&= \frac{1}{n} \left(1 + \tfrac{2\sigma^2}{\gamma^2}\right)^{-1/2}.
\end{aligned}$$

■

Appendix C. Details of experimentation

Lasso

Both Algorithms 4 and 5 use the Lasso. We've used the implementation of the Lasso found in the `scikit-learn` package.[5]. Being a coordinate descent algorithm, the Lasso requires us to specify a maximal number of iterations. We've allowed $5n$ iterations, where n is the number of variables, meaning that each variable is seen up to 5 times. For Algorithm 5 specifically, because the unregularized minimizer of Equation (60) is simply $b = a$, we gave this value as warm start to the algorithm to accelerate convergence. Finally, we gave the Lasso a tolerance of 10^{-4}.

Hyperparameter selection

Choosing the hyperparameters for instantiation 1 turns out to be a challenging task. The values of the expectation for instantiation 1 (see Table 1 or Theorem 19) has an exponential dependence on the dimensionality of the instance space, seen from the $\left(1 + \sigma^2/\gamma^2\right)^{-n/2}$ coefficient. To decide on reasonable values for the experiment of Section 8.3, we have opted to use this coefficient to calculate γ from σ. Specifically, given some value for σ, and

5. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`

$c \in (0,1)$, we can take:

$$\gamma = \frac{\sigma}{\sqrt{c^{-2/n} - 1}}, \tag{99}$$

in order to get $\left(1 + \sigma^2/\gamma^2\right)^{-n/2} = c$. This is a rather rudimentary method for selecting the hyperparameters of the model; a better way to do so likely exists, though requiring further theoretical work to develop.

The problem resides mostly in the relationship between parameters σ and γ in high dimensions. Because instantiation 2 considers stumps on individual variables, it does not suffer from the curse of dimensionality. Reasonable parameter values were chosen directly (see below).

In the end, we applied 5-fold cross-validation[6] to find the best hyperparameters among the following values (separately for each combination of instantiation, dataset, and random seed):

Instantiation 1:

- $\sigma = 1$
- $c \in \{0.1, 0.5, 0.9\}$
- $\lambda \in \{0.0000001, 0.000001, 0.00001, 0.0001, 0.001\}$

instantiation 2:

- $\sigma \in \{0.01, 0.1, 1\}$
- $\gamma \in \{0.01, 0.1, 1\}$
- $\lambda \in \{0.0000001, 0.000001, 0.00001, 0.0001, 0.001\}$

We also chose the number of estimators (decision stumps) of AdaBoost in $\{10, 25, 50, 100, 150, 200\}$. (We used the implementation of AdaBoost found in the `scikit-learn` package.[7]) We chose the C parameter of SVM in $\{0.01, 0.1, 1, 10, 100\}$. (We used the implementation of SVM found in the `scikit-learn` package.[8])

Calculating the bounds

Theorem 10 is a uniform bound, and therefore does not depend on the actual value of B used for training the model. To get the tightest possible bound, we can simply take $B = \|\bar{\alpha}\|_{\mathcal{H}}$, where $\bar{\alpha}$ is the output of Algorithm 1. This is how we obtained the values in Table 3.

The theorem also uses the Lipschitz constant of the loss. In the case of the square loss, we can use the boundedness of the model ($|\Lambda(x)| \leq \theta\|\alpha\|_{\mathcal{H}}$) to get $\rho \leq 2(\theta\|\alpha\|_{\mathcal{H}} + 1)$.

6. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`
7. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`
8. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

Reproducing the results

The code can be found at `https://github.com/gadub44/jmlr2023`. The experimental results were achieved on a server running Ubuntu 20.04.5 LTS, with an Intel Core i7-5930K CPU, and 64 GB of RAM. The code's execution took a little over two weeks. We used an up-to-date installation of Anaconda, as of April 2023. See `http://anaconda.com` for information. To install Anaconda, run the following command to download the package:

```
wget https://repo.anaconda.com/archive/Anaconda3-2023.03-1-Linux-x86_64.sh
```

Next, run the installer:

```
sudo sh Anaconda3-2021.05-Linux-x86_64.sh
```

Once the installation is completed, update with conda to make sure all the relevant packages are up-to-date:

```
conda update --all
```

The package `idx2numpy` is also required (for loading the MNIST dataset). It can be installed by running the command:

```
pip install idx2numpy
```

Finally, to reproduce our experiments, simply run the commands:

```
python fig_jmlr2023_algo_compar.py
python fig_jmlr2023_pruning.py
python fig_jmlr2023_sota.py
```

in the `/jmlr2023/` folder. This will generate the files:

- `jmlr2023_pruning.csv` in the `/jmlr2023/results/` subfolder containing the results used to generate Table 2, and `jmlr2023_pruning.tex`, the LaTex table itself, in the `/jmlr2023/tables/` subfolder.
- `jmlr2023-algo-compar.csv` in the `/jmlr2023/results/` subfolder containing the results used to generate Figure 1, and the three subfigures can then be found in the subfolder `/jmlr2023/figures/`.
- `jmlr2023-sota0.csv` in the `/jmlr2023/results/` subfolder containing the results used to generate Table 2, and `jmlr2023-sota.tex`, the LaTex table itself, in the `/jmlr2023/tables/` subfolder.

Note that all datasets are provided in the `/jmlr2023/datasets/` folder, though the code will automatically download them if they are absent.

Appendix D. Additional results

On the next pages, we have reproduced Table 2 and Table 3 with standard deviations.

Table 4: Prediction accuracy and pruning ratio of various algorithms on MNIST17. **Lasso fit** is Algorithm 4 alone, while **Least squares fit**, **Optimal stepsize descent** and **SFGD** are the respective algorithm followed by Algorithm 5 for pruning the learned model. $\|\beta\|_{\mathcal{H}}$ is the norm of the weight function after pruning. In every case, $T = 1000$. Every line is the average of 10 independently seeded runs.

algo	$\mathcal{L}_{\mathcal{S}}^{01}$	$\mathcal{L}_{\mathcal{D}}^{01}$	%pruning	pruning+$\mathcal{L}_{\mathcal{S}}^{01}$	pruning+$\mathcal{L}_{\mathcal{D}}^{01}$	$\|\beta\|_{\mathcal{H}}$
Lasso fit	0.006 ± 0.001	0.009 ± 0.001	63.99 ± 13.66	0.006 ± 0.001	0.009 ± 0.001	1053.973 ± 264.878
Least squares fit	0.007 ± 0.001	0.007 ± 0.001	59.9 ± 0.91	0.007 ± 0.001	0.009 ± 0.001	248.396 ± 3.05
Optimal stepsize descent	0.018 ± 0.002	0.019 ± 0.002	51.42 ± 1.881	0.022 ± 0.003	0.024 ± 0.002	534.325 ± 9.74
SFGD	0.025 ± 0.003	0.027 ± 0.004	73.4 ± 3.367	0.027 ± 0.005	0.029 ± 0.005	310.779 ± 57.283

Table 5: Performance comparison of the model to AdaBoost (**AB**), **SVM** and the Random Kitchen Sinks (**RKS**) algorithm on various binary classification datasets. Instantiations 1 and 2 of the model (**I1** and **I2**) were learned using Algorithm 3 (**LS fit**) and Algorithm 4 (**Lasso fit**) with $T = 1000$. $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{L}_{\mathcal{D}}$ are the empirical and test mean square error. $\mathcal{L}_{\mathcal{S}}^{01}$ and $\mathcal{L}_{\mathcal{D}}^{01}$ are the empirical and test classification errors. $\mathcal{R}_m$ is the right-hand side of Equation (71), with $\delta = 0.05$. Every line (except SVM) is the average of 10 independent runs.

dataset	algo	$\mathcal{L}_{\mathcal{S}}$	$\mathcal{L}_{\mathcal{D}}$	$\mathcal{R}_m$	$\mathcal{L}_{\mathcal{S}}^{01}$	$\mathcal{L}_{\mathcal{D}}^{01}$	training time (s)
adults	AdaBoost				0.13 ± 0.0	0.133 ± 0.0	6.585 ± 0.065
	LS fit w/ I1	0.488 ± 0.011	0.487 ± 0.011	61.4 ± 9.3	0.157 ± 0.0	0.155 ± 0.001	0.863 ± 0.013
	LS fit w/ I2	0.429 ± 0.0	0.429 ± 0.0	7.0 ± 0.1	0.147 ± 0.0	0.146 ± 0.0	0.737 ± 0.01
	Lasso fit w/ I1	0.454 ± 0.002	0.454 ± 0.002	$>10^3 \pm >10^3$	0.158 ± 0.001	0.156 ± 0.001	9.422 ± 13.499
	Lasso fit w/ I2	0.425 ± 0.002	0.427 ± 0.001	155.2 ± 139.9	0.145 ± 0.0	0.145 ± 0.001	49.484 ± 5.081
	RKS				0.148 ± 0.003	0.148 ± 0.003	0.513 ± 0.007
	SVM				0.137	0.148	79.222
breast cancer	AdaBoost				0.0 ± 0.0	0.021 ± 0.0	0.109 ± 0.014
	LS fit w/ I1	0.162 ± 0.0	0.191 ± 0.001	13.2 ± 0.1	0.021 ± 0.0	0.036 ± 0.002	0.069 ± 0.01
	LS fit w/ I2	0.135 ± 0.007	0.145 ± 0.001	11.7 ± 2.9	0.018 ± 0.002	0.022 ± 0.002	0.049 ± 0.003
	Lasso fit w/ I1	0.073 ± 0.032	0.147 ± 0.019	$>10^5 \pm >10^4$	0.012 ± 0.005	0.043 ± 0.005	0.872 ± 0.277
	Lasso fit w/ I2	0.087 ± 0.009	0.158 ± 0.009	853.9 ± $>10^3$	0.009 ± 0.003	0.035 ± 0.012	0.768 ± 0.261
	RKS				0.0 ± 0.0	0.076 ± 0.02	0.029 ± 0.003
	SVM				0.014	0.035	0.004
mnist17	AdaBoost				0.0 ± 0.0	0.005 ± 0.0	38.527 ± 0.199
	LS fit w/ I1	0.057 ± 0.0	0.061 ± 0.0	$>10^3 \pm >10^3$	0.007 ± 0.001	0.01 ± 0.0	0.822 ± 0.014
	LS fit w/ I2	0.066 ± 0.002	0.071 ± 0.003	5.5 ± 0.3	0.008 ± 0.001	0.013 ± 0.001	0.494 ± 0.007
	Lasso fit w/ I1	0.051 ± 0.002	0.056 ± 0.001	$>10^4 \pm >10^4$	0.007 ± 0.001	0.01 ± 0.001	18.259 ± 4.446
	Lasso fit w/ I2	0.054 ± 0.005	0.059 ± 0.006	112.8 ± 50.9	0.006 ± 0.001	0.01 ± 0.002	24.711 ± 3.45
	RKS				0.007 ± 0.001	0.012 ± 0.001	0.325 ± 0.015
	SVM				0.0	0.01	6.635
skin segmentation	AdaBoost				0.039 ± 0.0	0.039 ± 0.0	7.278 ± 0.056
	LS fit w/ I1	0.03 ± 0.0	0.03 ± 0.0	15.7 ± 0.1	0.005 ± 0.0	0.005 ± 0.0	5.561 ± 0.075
	LS fit w/ I2	0.172 ± 0.0	0.173 ± 0.0	7.4 ± 0.0	0.042 ± 0.0	0.042 ± 0.0	6.288 ± 0.038
	Lasso fit w/ I1	0.015 ± 0.0	0.015 ± 0.0	693.8 ± 195.8	0.002 ± 0.0	0.002 ± 0.0	411.723 ± 35.335
	Lasso fit w/ I2	0.17 ± 0.002	0.171 ± 0.001	26.8 ± 23.9	0.04 ± 0.0	0.04 ± 0.0	353.341 ± 38.872
	RKS				0.04 ± 0.0	0.04 ± 0.001	2.865 ± 0.091
	SVM				0.001	0.001	7.495