Journal of Machine Learning Research 20 (2019) 1-44 Submitted 11/16; Revised 1/19; Published 1/19

Parsimonious Online Learning with Kernels via Sparse Projections in Function Space*

Alec Koppel AKOPPEL@SEAS.UPENN.EDU
Computational and Information Sciences Directorate
U.S. Army Research Laboratory
Adelphi, MD 20783, USA

Garrett Warnell GARRETT.A.WARNELL.CIV@MAIL.MIL
Computational and Information Sciences Directorate
U.S. Army Research Laboratory
Adelphi, MD 20783, USA

Ethan Stump ETHAN.A.STUMP2.CIV@MAIL.MIL
Computational and Information Sciences Directorate
U.S. Army Research Laboratory
Adelphi, MD 20783, USA

Alejandro Ribeiro ARIBEIRO@SEAS.UPENN.EDU
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Le Song

Abstract

Despite their attractiveness, popular perception is that techniques for nonparametric function approximation do not scale to streaming data due to an intractable growth in the amount of storage they require. To solve this problem in a memory-affordable way, we propose an online technique based on functional stochastic gradient descent in tandem with supervised sparsification based on greedy function subspace projections. The method, called *parsimonious online learning with kernels* (POLK), provides a controllable tradeoff between its solution accuracy and the amount of memory it requires. We derive conditions under which the generated function sequence converges almost surely to the optimal function, and we establish that the memory requirement remains finite. We evaluate POLK for kernel multi-class logistic regression and kernel hinge-loss classification on three canonical data sets: a synthetic Gaussian mixture model, the MNIST hand-written digits, and the Brodatz texture database. On all three tasks, we observe a favorable trade-off of objective function evaluation, classification performance, and complexity of the nonparametric regressor extracted by the proposed method.

Keywords: kernel methods, online learning, stochastic optimization, supervised learning, orthogonal matching pursuit, nonparametric regression

*. This work in this paper is supported by NSF CCF-1017454, NSF CCF-0952867, ONR N00014-12-1-0997, ARL MAST CTA, and ASEE SMART. Part of the results in this paper appeared in Koppel et al. (2017).

1. Introduction

Reproducing kernel Hilbert spaces (RKHS) provide the ability to approximate functions using nonparametric functional representations. Although the structure of the space is determined by the choice of kernel, the set of functions that can be represented is still sufficiently rich so as to permit close approximation of a large class of functions. This resulting expressive power makes RKHS an appealing choice in many learning problems where we want to estimate an unknown function that is specified as optimal with respect to some empirical risk. When learning these optimal function representations in a RKHS, the representer theorem is used to transform the search over functions into a search over parameters, where the number of parameters grows with each new observation that is processed (Wheeden et al., 1977; Norkin and Keyzer, 2009). This growth is what endows the representation with expressive power. However, this growth also results in function descriptions that are as complex as the number of processed observations, and, more importantly, in training algorithms that exhibit a cost per iteration that grows with each new iterate (Pontil et al., 2005; Kivinen et al., 2004). The resulting unmanageable training cost renders RKHS learning approaches inefficient for large data sets and outright inapplicable in streaming applications. This is a well-known limitation which has motivated the development of several heuristics to reduce the growth in complexity. These heuristics typically result in suboptimal functional approximations (Richard et al., 2009).

This paper proposes a new technique for learning nonparametric function approximations in a RKHS that respects optimality and ameliorates the complexity issues described above. We accomplish this by: (i) shifting the goal from that of finding an approximation that is optimal to that of finding an approximation that is optimal within a class of parsimonious (sparse) kernel representations; (ii) designing a training method that follows a trajectory of intermediate representations that are also parsimonious. The proposed technique, *parsimonious online learning with kernels* (POLK), provides a controllable tradeoff between complexity and optimality and we provide theoretical guarantees that neither factor becomes untenable.

Assumptions

Formally, we propose solving expected risk minimization problems, where the goal is to learn a regressor that minimizes a loss function quantifying the merit of a statistical model averaged over a data set. We focus on the case when the number of training examples, N, is either very large, or the training examples arrive sequentially. Further, we assume that these input-output examples, $(\mathbf{x}_n, \mathbf{y}_n)$, are i.i.d. realizations drawn from a stationary joint distribution over the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. This problem class is popular in many fields and particularly ubiquitous in text (Sampson et al., 1990), image (Mairal et al., 2007), and genomic (Taşan et al., 2014) processing. Here, we consider finding regressors that are not vector valued parameters, but rather functions $f \in \mathcal{H}$ in a hypothesized function class $\mathcal{H}$. This function estimation task allows one to learn nonlinear statistical models and is known to yield better results in applications where linearity of a given statistical model is overly restrictive such as computer vision and object recognition (Mukherjee and Nayar, 1996; Li et al., 2014). The adequacy of the regressor function f is evaluated by the convex loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the merit of the estimator $f(\mathbf{x})$ evaluated at feature vector $\mathbf{x}$. This loss is averaged over all possible training examples to define the statistical loss $L(f) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)]$, which we combine with a Tikhonov regularizer to

construct the regularized loss $R(f) := \operatorname{argmin}_{f\in\mathcal{H}} L(f) + (\lambda/2)\|f\|_{\mathcal{H}}^2$ (Shalev-Shwartz et al., 2010; Evgeniou et al., 2000). We then define the optimal function as

$$f^* = \operatorname*{argmin}_{f\in\mathcal{H}} R(f) := \operatorname*{argmin}_{f\in\mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big[\ell\big(f(\mathbf{x}), y\big)\Big] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{1}$$

The optimization problem in (1) is intractable in general. However, when $\mathcal{H}$ is equipped with a *reproducing kernel* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, a nonparametric function estimation problem of the form (1) may be reduced to a parametric form via the representer theorem (Wheeden et al., 1977; Norkin and Keyzer, 2009). This theorem states that the optimal argument of (1) is in the span of kernel functions that are centered at points in the given training data set, and it reduces the problem to that of determining the N coefficients of the resulting linear combination of kernels (Section 2). This results in a function description that is data driven and flexible, alas very complex. As we consider problems with larger training sets, the representation of f requires a growing number of kernels (Norkin and Keyzer, 2009). In the case of streaming applications this number would grow unbounded and the kernel matrix as well as the coefficient vector would grow to infinite dimension and an infinite amount of memory would be required to represent f. It is therefore customary to reduce this complexity by forgetting training points or otherwise requiring that f^* admit a parsimonious representation in terms of a sparse subset of kernels. This overcomes the difficulties associated with a representation of unmanageable complexity but a steeper difficulty is the *determination* of this optimal parsimonious representation as we explain in the following section

1.1 Context

To understand the challenge in determining optimal parsimonious representations, recall that kernel optimization methods borrow techniques from vector valued (i.e., without the use of kernels) stochastic optimization in the sense that they seek to optimize (1) by replacing the descent direction of the objective with that of a stochastic estimate (Bottou, 1998; Robbins and Monro, 1951). Stochastic optimization is well understood in vector valued problems to the extent that recent efforts are concerned with improving convergence properties through the use of variance reduction (Schmidt et al., 2013; Johnson and Zhang, 2013; Defazio et al., 2014), or stochastic approximations of Newton steps (Schraudolph et al., 2007; Bordes et al., 2009; Mokhtari and Ribeiro, 2014, 2015). Stochastic optimization in kernel spaces, however, exhibits two peculiarities that make it more challenging:

i. The implementation of stochastic methods for expected risk minimization in a RKHS requires storage of kernel matrices and weight vectors that together are cubic in the iteration index. This is true even if we require that the solution f^* admit a sparse representation because, while it may be true that the asymptotic solution admits a sparse representation, the intermediate iterates are not necessarily sparse; see, e.g., Kivinen et al. (2004).

ii. The problem in (i) makes it necessary to use sparse approximations of descent directions. However, these sparse approximates are not guaranteed to be valid stochastic descent directions. Consequently, there are no guarantees that a path of sparse approximation learns the optimal sparse approximation.

Issue (i) is a key point of departure between kernel stochastic optimization and its vector valued counterpart. It implies that redefining f^* to encourage sparsity may make it easier to work with the RKHS representation after it has been learnt. However, the stochastic gradient iterates that need to be computed to find such representation have a complexity that grows with the order of the iteration index (Kivinen et al., 2004), precluding the use of sparsity-encouraging penalties (Fu, 1998). Works on stochastic optimization in a RKHS have variously ignored the intractable growth of the parametric representation of $f \in \mathcal{H}$ (Ying and Zhou, 2006; Liu et al., 2008; Pontil et al., 2005; Dieuleveut and Bach, 2014), addressed it indirectly through probabilistic (Rahimi and Recht, 2008, 2009; Dai et al., 2014) or computational (Williams and Seeger, 2001; Zhang et al., 2008) approximations of the kernel, or have augmented the learned function to limit the memory issues associated with kernelization using online sparsification procedures. These approaches focus on limiting the growth of the kernel dictionary through forgetting factors (Kivinen et al., 2004), random dropping (Zhang et al., 2013; Le et al., 2016b), compressive sensing (Engel et al., 2004; Richard et al., 2009; Honeine, 2012), greedy search (Bordes et al., 2005) and projection (Zhao et al., 2012; Zhao and Hoi, 2012; Wang et al., 2012). These approaches overcome Issue (i) but they do so at the cost of dropping optimality [cf. Issue (ii)]. This is because these sparsification techniques introduce a bias in the stochastic gradient which nullifies convergence guarantees, and empirically leads to instability.

Past works that have considered *supervised* sparsification (addressing issues (i)-(ii)) have only been developed for special cases such as online support vector machines (SVM) (Wang et al., 2012), off-line logistic regression Zhu and Hastie (2005), and off-line SVM (Joachims and Yu, 2009). The works perhaps most similar to ours, but developed only for SVM (Wang et al., 2012)) fixes the number of kernel dictionary elements, or the model order, in advance rather tuning the kernel dictionary to guarantee stochastic descent, i.e. determining which kernel dictionary elements are most important for representing f^*. Further, the analysis of the resulting bias induced by sparsification requires overly restrictive assumptions and is conducted in terms of time-average objective sub-optimality, a looser criterion than almost sure convergence. For specialized classes of loss functions, the bias of the descent direction induced by unsupervised sparsification techniques using random sub-sampling does not prevent the derivation of bounds on the time-average sub-optimality (regret) (Zhang et al., 2013); however, this analysis omits important cases such as support vector machines and kernel ridge regression.

1.2 Contributions

In this work, we build upon past works which combine functional generalizations of first-order stochastic optimization methods operating in tandem with supervised sparsification. In particular, descending along the gradient of the objective in (1) is intractable when the sample size N is not necessarily finite, and thus stochastic methods are necessary. In Section 3, we build upon Kivinen et al. (2004) in deriving the generalization of stochastic gradient method called functional SGD (Section 3.1). The complexity of this online functional iterative optimization is on the order of the iteration index, a complicating factor of kernel methods which is untenable for streaming settings.

Thus, we project the FSGD iterates onto sparse subspaces, which, rather than being fixed independent of the problem (Zhu et al., 2009), are adaptively constructed from the span of a small number of kernel dictionary elements (Section 3.2). To find these sparse subspaces of the RKHS, we make use of greedy sampling methods based on matching pursuit (Pati et al., 1993). The use of this technique is motivated by: (i) The fact that kernel matrices induced by arbitrary data streams will not, in general, satisfy requisite conditions for methods that enforce sparsity through convex relaxation (Candes, 2008); (ii) That having function iterates that exhibit small model order is of greater importance than exact recovery since SGD iterates are not the goal signal but just a noisy stepping stone to the optimal f^*. Therefore, we construct these instantaneous sparse subspaces by making use of kernel orthogonal matching pursuit (Vincent and Bengio, 2002), a greedy search routine which, given a function and an approximation budget ϵ, returns its a sparse approximation and guarantees its output to be in a specific Hilbert-norm neighborhood of its function input. Greedy sampling methods appear in prior works for constructing RKHS subspaces in recursive mini-batching procedures (Keerthi et al., 2006) or with FSGD (Zhao et al., 2012), but in a manner that avoids addressing how to merge greedy compression with obtaining optimal function representations.

To guarantee stochastic descent, we tie the size of the error neighborhood induced by sparse projections to the magnitude of the functional stochastic gradient and other problem parameters, thereby keeping only those kernel dictionary elements necessary for convergence (Section 4). The result is that we are able to conduct stochastic gradient descent using only sparse projections of the stochastic gradient, maintaining a convergence path of moderate complexity towards the optimal f^* (1). When the data and target domains ($\mathcal{X}$ and $\mathcal{Y}$, respectively) are compact, for a certain approximation budget depending on the stochastic gradient algorithm step-size, we show that the sparse stochastically projected FSGD sequence still converges almost surely to the optimum of (5) under both attenuating and constant learning rate schemes. Moreover, the model order of this sequence remains finite for a given choice of constant step-size and approximation budget, and is, in the worst-case, comparable to the covering number of the data domain (Zhou, 2002; Pontil, 2003).

In Section 5 we present numerical results on synthetic and empirical data for large-scale kernelized supervised learning tasks. We observe stable convergence behavior of POLK comparable to vector-valued first-order stochastic methods in terms of objective function evaluation, punctuated by a state of the art trade-off between test set error and number of samples processed. Further, the proposed method reduces the complexity of training kernel regressors by orders of magnitude. In Section 6 we discuss our main findings. In particular, we suggest that there is a path forward for kernel methods as an alternative to neural networks that provides a more interpretable mechanism for inference with nonlinear statistical models and that one may achieve high generalization capability without losing convexity, an essential component for efficient training.

Relative to (Koppel et al., 2017), this paper provides detailed derivations of the algorithms and proofs to all theorems, including auxiliary and technical lemmas. Moreover, a thorough accounting of computational complexity and extensive experimentation with several state of the art alternatives, both offline and online, are provided.

2. Statistical Optimization in Reproducing Kernel Hilbert Spaces

Supervised learning is often formulated as an optimization problem that computes a set of parameters $\theta \in \Theta$ to minimize the average of a loss function $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for training examples $(\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$. When the number of training examples N is finite, this goal is referred to as *empirical risk minimization* (Tewari and Bartlett, 2014), and may be solved using batch optimization techniques. The optimal $\boldsymbol{\theta}$ is the one that minimizes the regularized average loss, $\tilde{R}(\theta; \{\mathbf{x}_n, y_n\}_{n=1}^N) = \frac{1}{N}\sum_{n=1}^N l(\boldsymbol{\theta}; (\mathbf{x}_n, y_n))$, over the set of training data $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, i.e.,

$$\theta^* = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \tilde{R}(\boldsymbol{\theta}; \mathcal{S}) = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^N l(\boldsymbol{\theta}; \mathbf{x}_n, y_n) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \,. \tag{2}$$

We focus on the case when the inputs are vectors $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p$ and the target domain $\mathcal{Y} \subseteq \{0, 1\}$ in the case of classification or $\mathcal{Y} \subseteq \mathbb{R}$ in the case of regression.

2.1 Supervised Kernel Learning

In the case of supervised kernel learning (Müller et al., 2013; Li et al., 2014), Θ is taken to be a Hilbert space, denoted here as $\mathcal{H}$. Elements of $\mathcal{H}$ are *functions*, $f : \mathcal{X} \to \mathcal{Y}$, that admit a representation in terms of elements of $\mathcal{X}$ when $\mathcal{H}$ has a special structure. In particular, equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\,, \qquad (ii)\ \mathcal{H} = \overline{\operatorname{span}\{\kappa(\mathbf{x}, \cdot)\}} \quad \text{for all } \mathbf{x} \in \mathcal{X}\,. \tag{3}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Hilbert inner product for $\mathcal{H}$. We further assume that the kernel is positive semidefinite, i.e. $\kappa(\mathbf{x}, \mathbf{x}') \geq 0$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. Function spaces with this structure are called reproducing kernel Hilbert spaces (RKHS).

In (3), property (i) is called the reproducing property of the kernel, and is a consequence of the Riesz Representation Theorem (Wheeden et al., 1977). Replacing f by $\kappa(\mathbf{x}', \cdot)$ in (3) (i) yields the expression $\langle \kappa(\mathbf{x}', \cdot), \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = \kappa(\mathbf{x}, \mathbf{x}')$, which is the origin of the term "reproducing kernel." This property provides a practical means by which to access a nonlinear transformation of the input space $\mathcal{X}$. Specifically, denote by $\phi(\cdot)$ a nonlinear map of the feature space that assigns to each $\mathbf{x}$ the kernel function $\kappa(\cdot, \mathbf{x})$. Then the reproducing property of the kernel allows us to write the inner product of the image of distinct feature vectors $\mathbf{x}$ and $\mathbf{x}'$ under the map ϕ in terms of kernel evaluations only: $\langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle_{\mathcal{H}} = \kappa(\mathbf{x}, \mathbf{x}')$. This is commonly referred to as the *kernel trick*, and it provides a computationally efficient tool for learning nonlinear functions.

Moreover, property (3) (ii) states that any function $f \in \mathcal{H}$ may be written as a linear combination of kernel evaluations. For kernelized and regularized empirical risk minimization, the Representer Theorem (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001) establishes that the optimal f in the hypothesis function class $\mathcal{H}$ may be written as an expansion of kernel evaluations *only* at elements of the training set as

$$f(\mathbf{x}) = \sum_{n=1}^N w_n \kappa(\mathbf{x}_n, \mathbf{x}) \,. \tag{4}$$

where $\mathbf{w} = [w_1, \cdots, w_N]^T \in \mathbb{R}^N$ denotes a set of weights. The upper summand index N in (4) is henceforth referred to as the model order. Common choices κ include the polynomial kernel and the radial basis kernel, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$ and $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$, respectively, where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

We may now formulate the kernel variant of the empirical risk minimization problem as the one that minimizes the loss functional $L : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ plus a complexity-reducing penalty. The loss functional L may be written as an average over instantaneous losses $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, each of which penalizes the average deviation between $f(\mathbf{x}_n)$ and the associated output y_n over the training set $\mathcal{S}$. We denote the data loss and complexity loss as $R : \mathcal{H} \to \mathbb{R}$, and consider the problem

$$f^* = \underset{f\in\mathcal{H}}{\operatorname{argmin}}\, R(f;\mathcal{S}) = \underset{f\in\mathcal{H}}{\operatorname{argmin}}\, \frac{1}{N}\sum_{n=1}^{N} \ell(f(\mathbf{x}_n), y_n) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \,. \tag{5}$$

The above problem, referred to as Tikhonov regularization (Evgeniou et al., 2000), is one in which we aim to learn a general nonlinear relationship between $\mathbf{x}_n$ and y_n through a function f. Throughout, we assume ℓ is convex with respect to its first argument $f(\mathbf{x})$. By substituting the Representer Theorem expansion in (4) into (5), the optimization problem amounts to finding an optimal set of coefficients $\mathbf{w}$ as

$$\begin{aligned} f^* &= \underset{\mathbf{w}\in\mathbb{R}^N}{\operatorname{argmin}}\, \frac{1}{N}\sum_{n=1}^{N} \ell\Big(\sum_{m=1}^{N} w_m\kappa(\mathbf{x}_m, \mathbf{x}_n), y_n\Big) + \frac{\lambda}{2}\Big\|\sum_{n,m=1}^{N} w_n w_m \kappa(\mathbf{x}_m, \mathbf{x}_n)\Big\|_{\mathcal{H}}^2 \\ &= \underset{\mathbf{w}\in\mathbb{R}^N}{\operatorname{argmin}}\, \frac{1}{N}\sum_{n=1}^{N} \ell(\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n), y_n) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{w} \end{aligned} \tag{6}$$

where we have defined the Gram matrix (variously referred to as the *kernel matrix*) $\mathbf{K}_{\mathbf{X},\mathbf{X}} \in \mathbb{R}^{N\times N}$, with entries given by the kernel evaluations between $\mathbf{x}_m$ and $\mathbf{x}_n$ as $[\mathbf{K}_{\mathbf{X},\mathbf{X}}]_{m,n} = \kappa(\mathbf{x}_m, \mathbf{x}_n)$. We further define the vector of kernel evaluations $\boldsymbol{\kappa}_{\mathbf{X}}(\cdot) = [\kappa(\mathbf{x}_1, \cdot)\ldots\kappa(\mathbf{x}_N, \cdot)]^T$, which are related to the kernel matrix as $\mathbf{K}_{\mathbf{X},\mathbf{X}} = [\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_1)\ldots\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_N)]$. The dictionary of training points associated with the kernel matrix is defined as $\mathbf{X} = [\mathbf{x}_1, \ \ldots \ , \mathbf{x}_N]$.

Observe that by exploiting the Representer Theorem, we transform a nonparametric infinite dimensional optimization problem in $\mathcal{H}$ (5) into a finite N-dimensional parametric problem (6). Thus, for empirical risk minimization, the RKHS provides a principled framework to solve nonparametric regression problems as via search over $\mathbb{R}^N$ for an optimal set of coefficients. A motivating example is presented next to clarify the setting of supervised kernel learning.

Example 1 *(Kernel Logistic Regression)* Consider the case of *kernel logistic regression* (KLR), with feature vectors $\mathbf{x}_n \in \mathcal{X} \subseteq \mathbb{R}^p$ and binary class labels $y_n \in \{0, 1\}$. We seek to learn a function $f \in \mathcal{H}$ that allows us to best approximate the distribution of an unknown class label given a training example $\mathbf{x}$ under the assumed model

$$\mathbb{P}\left(y = 0 \mid \mathbf{x}\right) = \frac{\exp\left\{f(\mathbf{x})\right\}}{1 + \exp\left\{f(\mathbf{x})\right\}} \,. \tag{7}$$

In classical logistic regression, we assume that f is linear, i.e., $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + b$. In KLR, on the other hand, we instead seek a nonlinear function of the form given in (4). By making use of (7) and (4), we may formulate a maximum-likelihood estimation (MLE) problem to find the optimal function f on the basis of $\mathcal{S}$ by solving for the $\mathbf{w}$ that maximizes the λ-regularized average negative log likelihood over $\mathcal{S}$, i.e.,

$$\begin{aligned} f^* &= \operatorname*{argmin}_{f\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N}\left[-\log \mathbb{P}(y = y_n \mid \mathbf{x} = \mathbf{x}_n) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2\right] \\ &= \operatorname*{argmin}_{f\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N}\left[\log\left(1+\exp\{f(\mathbf{x}_n)\}\right) - \mathbb{1}(y_n = 1) - f(\mathbf{x}_n)\mathbb{1}(y_n = 0) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2\right] \\ &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^N} \frac{1}{N}\sum_{n=1}^{N}\left[\log\left(1+\exp\{\mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n)\}\right) - \mathbb{1}(y_n = 1) - \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{X}}(\mathbf{x}_n)\mathbb{1}(y_n = 0) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{w}\right], \end{aligned} \tag{8}$$

where $\mathbb{1}(\cdot)$ represents the indicator function. Solving (8) amounts to finding a function f that, given a feature vector $\mathbf{x}$ and the model outlined by (7), best represents the class-conditional probabilities that the corresponding label y is either 0 or 1.

2.2 Online Kernel Learning

The goal of this paper is to solve problems of the form (5) when training examples $(\mathbf{x}_n, \mathbf{y}_n)$ either become sequentially available or their total number is not necessarily finite. To do so, we consider the case where $(\mathbf{x}_n, \mathbf{y}_n)$ are independent realizations from a stationary joint distribution of the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ (Slavakis et al., 2013). In this case, the objective in (5) may be written as an expectation over this random pair as

$$\begin{aligned} f^* = \operatorname*{argmin}_{f\in\mathcal{H}} R(f) :&= \operatorname*{argmin}_{f\in\mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \\ &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{\mathcal{I}}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\sum_{n\in\mathcal{I}} w_n\kappa(\mathbf{x}_n, \mathbf{x}), y)] + \frac{\lambda}{2}\|\sum_{n,m\in\mathcal{I}} w_n w_m \kappa(\mathbf{x}_m, \mathbf{x}_n)\|_{\mathcal{H}}^2 \quad . \end{aligned} \tag{9}$$

where we define the average loss as $L(f) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)]$. In the second equality in (1), we substitute in the expansion of f given by the Representer Theorem generalized to the infinite sample-size case established in (Norkin and Keyzer, 2009), with $\mathcal{I}$ as some countably infinite indexing set.

3. Algorithm Development

We turn to deriving an algorithmic solution to the kernelized expected risk minimization problem stated in (1). To do so, two complexity bottlenecks must be overcome. The first is that in order to develop a numerical optimization scheme such as gradient descent, we must compute the functional gradient (Frechét derivative) of the expected risk $L(f)$ with respect to f, which requires infinitely many realizations of the random pair $(\mathbf{x}, y)$. This bottleneck is handled via stochastic approximation, as detailed in Section 3.1. The second issue is that when making use of the stochastic gradient method in the RKHS setting,

the resulting parametric updates require memory storage whose complexity is cubic in the iteration index (the curse of kernelization), which rapidly becomes unaffordable. To alleviate this memory explosion, we introduce our sparse stochastic projection scheme based upon kernel orthogonal matching pursuit in Section 3.2.

3.1 Functional Stochastic Gradient Descent

Following Kivinen et al. (2004), we derive the generalization of the stochastic gradient method for the RKHS setting. The resulting procedure is referred to as *functional stochastic gradient descent* (FSGD). First, given an independent realization $(\mathbf{x}_t, y_t)$ of the random pair $(\mathbf{x}, y)$, we compute the stochastic functional gradient (Frechét derivative) of $L(f)$, stated as

$$\nabla_f \ell(f(\mathbf{x}_t), y_t)(\cdot) = \frac{\partial \ell(f(\mathbf{x}_t), y_t)}{\partial f(\mathbf{x}_t)} \frac{\partial f(\mathbf{x}_t)}{\partial f}(\cdot) \tag{10}$$

where we have applied the chain rule. Now, define the short-hand notation $\ell'(f(\mathbf{x}_t), y_t) := \partial \ell(f(\mathbf{x}_t), y_t)/\partial f(\mathbf{x}_t)$ for the derivative of $\ell(\mathbf{f}(\mathbf{x}_t), y_t)$ with respect to its first scalar argument $f(\mathbf{x}_t)$ evaluated at $\mathbf{x}_t$. To evaluate the second term on the right-hand side of (10), differentiate both sides of the expression defining the reproducing property of the kernel [cf. (3)(i)] with respect to f to obtain

$$\frac{\partial f(\mathbf{x}_t)}{\partial f} = \frac{\partial \langle f, \kappa(\mathbf{x}_t, \cdot)) \rangle_{\mathcal{H}}}{\partial f} = \kappa(\mathbf{x}_t, \cdot) \tag{11}$$

With this computation in hand, we present the stochastic gradient method for the kernelized λ-regularized expected risk minimization problem in (1) as

$$f_{t+1} = (1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) = (1 - \eta_t \lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), y_t) \kappa(\mathbf{x}_t, \cdot) \, , \tag{12}$$

where $\eta_t > 0$ is an algorithm step-size either chosen as diminishing with $\mathcal{O}(1/t)$ or a small constant – see Section 4. We further require that, given $\lambda > 0$, the step-size satisfies $\eta_t < 1/\lambda$ and the sequence is initialized as $f_0 = 0 \in \mathcal{H}$. Given this initialization, by making use of the Representer Theorem (4), at time t, the function f_t may be expressed as an expansion in terms of feature vectors $\mathbf{x}_t$ observed thus far as

$$f_t(\mathbf{x}) = \sum_{n=1}^{t-1} w_n \kappa(\mathbf{x}_n, \mathbf{x}) = \mathbf{w}_t^T \boldsymbol{\kappa}_{\mathbf{X}_t}(\mathbf{x}) \, . \tag{13}$$

On the right-hand side of (13) we have introduced the notation $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}] \in \mathbb{R}^{p \times (t-1)}$ and $\boldsymbol{\kappa}_{\mathbf{X}_t}(\cdot) = [\kappa(\mathbf{x}_1, \cdot), \, \ldots \, , \kappa(\mathbf{x}_{t-1}, \cdot)]^T$. Moreover, observe that the kernel expansion in (13), taken together with the functional update (12), yields the fact that performing the stochastic gradient method in $\mathcal{H}$ amounts to the following parametric updates on the kernel dictionary $\mathbf{X}$ and coefficient vector $\mathbf{w}$:

$$\mathbf{X}_{t+1} = [\mathbf{X}_t, \;\; \mathbf{x}_t], \quad \mathbf{w}_{t+1} = [(1 - \eta_t \lambda) \mathbf{w}_t, \;\; -\eta_t \ell'(f_t(\mathbf{x}_t), y_t)] \, , \tag{14}$$

Observe that this update causes $\mathbf{X}_{t+1}$ to have one more column than $\mathbf{X}_t$. We define the *model order* as number of data points M_t in the dictionary at time t (the number of columns of $\mathbf{X}_t$). FSGD is such that $M_t = t - 1$, and hence grows unbounded with iteration index t.

3.2 Model Order Control via Stochastic Projection

To mitigate the model order issue described above, we shall generate an approximate sequence of functions by orthogonally projecting functional stochastic gradient updates onto subspaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ that consist only of functions that can be represented using some dictionary $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_M] \in \mathbb{R}^{p \times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \{f \ : \ f(\cdot) = \sum_{n=1}^{M} w_n \kappa(\mathbf{d}_n, \cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$. For convenience we have defined $[\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = \kappa(\mathbf{d}_1, \cdot) \ldots \kappa(\mathbf{d}_M, \cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We will enforce parsimony in function representation by selecting dictionaries $\mathbf{D}$ that $M_t \ll t$.

We first show that, by selecting $\mathbf{D} = \mathbf{X}_{t+1}$ at each iteration, the sequence (12) derived in Section 3.1 may be interpreted as carrying out a sequence of orthogonal projections. To see this, rewrite (12) as the quadratic minimization

$$\begin{aligned} f_{t+1} &= \underset{f \in \mathcal{H}}{\text{argmin}} \left\| f - \Big((1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big) \right\|_{\mathcal{H}}^2 \\ &= \underset{f \in \mathcal{H}_{\mathbf{X}_{t+1}}}{\text{argmin}} \left\| f - \Big((1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big) \right\|_{\mathcal{H}}^2, \end{aligned} \tag{15}$$

where the first equality in (15) comes from ignoring constant terms which vanish upon differentiation with respect to f, and the second comes from observing that f_{t+1} can be represented using only the points $\mathbf{X}_{t+1}$, using (14). Notice now that (15) expresses f_{t+1} as the orthogonal projection of the update $(1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t)$ onto the subspace defined by dictionary $\mathbf{X}_{t+1}$.

Rather than select dictionary $\mathbf{D} = \mathbf{X}_{t+1}$, we propose instead to select a different dictionary, $\mathbf{D} = \mathbf{D}_{t+1}$, which is extracted from the data points observed thus far, at each iteration. The process by which we select $\mathbf{D}_{t+1}$ will be discussed shortly, but is of dimension $p \times M_{t+1}$, with $M_{t+1} \ll t$. As a result, we shall generate a function sequence f_t that differs from the functional stochastic gradient method presented in Section 3.1. The function f_{t+1} is parameterized by dictionary $\mathbf{D}_{t+1}$ and weight vector $\mathbf{w}_{t+1}$. We denote columns of $\mathbf{D}_{t+1}$ as $\mathbf{d}_n$ for $n = 1, \ldots, M_{t+1}$, where the time index is dropped for notational clarity but may be inferred from the context.

To be specific, we propose replacing the update (15) in which the dictionary grows at each iteration by the stochastic projection of the functional stochastic gradient sequence onto the subspace $\mathcal{H}_{\mathbf{D}_{t+1}} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M_{t+1}}$ as

$$\begin{aligned} f_{t+1} &= \underset{f \in \mathcal{H}_{\mathbf{D}_{t+1}}}{\text{argmin}} \left\| f - \Big((1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big) \right\|_{\mathcal{H}}^2 \\ &:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[(1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) \Big]. \end{aligned} \tag{16}$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{t+1}} \subset \mathcal{H}$ by the update (16).

Coefficient update The update (16), for a fixed dictionary $\mathbf{D}_{t+1} \in \mathbb{R}^{p \times M_{t+1}}$, may be expressed in terms of the parameter space of coefficients only. In order to do so, we first define the stochastic gradient update without projection, given function f_t parameterized by dictionary $\mathbf{D}_t$ and coefficients $\mathbf{w}_t$, as

$$\tilde{f}_{t+1} = (1 - \eta_t \lambda) f_t - \eta_t \nabla_f \ell(f_t; \mathbf{x}_t, \mathbf{y}_t). \tag{17}$$

This update may be represented using dictionary and weight vector

$$\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \mathbf{x}_t], \quad \tilde{\mathbf{w}}_{t+1} = [(1-\eta_t\lambda)\mathbf{w}_t, \ -\eta_t \ell'(f_t(\mathbf{x}_t), y_t)] \ . \tag{18}$$

Observe that $\tilde{\mathbf{D}}_{t+1}$ has $\tilde{M} = M_t + 1$ columns, which is also the length of $\tilde{\mathbf{w}}_{t+1}$. For a fixed dictionary $\mathbf{D}_{t+1}$, the stochastic projection in (16) amounts to a least-squares problem on the coefficient vector. To see this, make use of the Representer Theorem to rewrite (16) in terms of kernel expansions, and that the coefficient vector is the only free parameter to write

$$\begin{aligned} &\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\Big\| \sum_{n=1}^{M_{t+1}} w_n \kappa(\mathbf{d}_n, \cdot) - \sum_{m=1}^{\tilde{M}} \tilde{w}_m \kappa(\tilde{\mathbf{d}}_m, \cdot)\Big\|_{\mathcal{H}}^2 \qquad (19) \\ &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\left(\sum_{n,n'=1}^{M_{t+1}} w_n w_{n'} \kappa(\mathbf{d}_n, \mathbf{d}_{n'}) - 2\sum_{n,m=1}^{M_{t+1},\tilde{M}} w_n \tilde{w}_m \kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m) + \sum_{m,m'=1}^{\tilde{M}} \tilde{w}_m \tilde{w}_{m'} \kappa(\tilde{\mathbf{d}}_m, \tilde{\mathbf{d}}_{m'}) \right) \\ &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{M_{t+1}}} \frac{1}{2\eta_t}\Big(\mathbf{w}^T \mathbf{K}_{\mathbf{D}_{t+1},\mathbf{D}_{t+1}} \mathbf{w} - 2\mathbf{w}^T \mathbf{K}_{\mathbf{D}_{t+1},\tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} + \tilde{\mathbf{w}}_{t+1} \mathbf{K}_{\tilde{\mathbf{D}}_{t+1},\tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} \Big) := \mathbf{w}_{t+1} \ . \end{aligned}$$

In (19), the first equality comes from expanding the square, and the second comes from defining the cross-kernel matrix $\mathbf{K}_{\mathbf{D}_{t+1},\tilde{\mathbf{D}}_{t+1}}$ whose $(n,m)^{\text{th}}$ entry is given by $\kappa(\mathbf{d}_n, \tilde{\mathbf{d}}_m)$. The other kernel matrices $\mathbf{K}_{\tilde{\mathbf{D}}_{t+1},\tilde{\mathbf{D}}_{t+1}}$ and $\mathbf{K}_{\mathbf{D}_{t+1},\mathbf{D}_{t+1}}$ are similarly defined. Note that M_{t+1} is the number of columns in $\mathbf{D}_{t+1}$, while $\tilde{M} = M_t + 1$ is the number of columns in $\tilde{\mathbf{D}}_{t+1}$ [cf. (18)]. The explicit solution of (19) may be obtained by noting that the last term is a constant independent of $\mathbf{w}$, and thus by computing gradients and solving for $\mathbf{w}_{t+1}$ we obtain

$$\mathbf{w}_{t+1} = \mathbf{K}_{\mathbf{D}_{t+1}\mathbf{D}_{t+1}}^{-1} \mathbf{K}_{\mathbf{D}_{t+1}\tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} \ , \tag{20}$$

Given that the projection of $\tilde{f}_{t+1}$ onto the stochastic subspace $\mathcal{H}_{\mathbf{D}_{t+1}}$, for a fixed dictionary $\mathbf{D}_{t+1}$, amounts to a simple least-squares multiplication, we turn to detailing how the kernel dictionary $\mathbf{D}_{t+1}$ is selected from the data sample path $\{\mathbf{x}_u, y_u\}_{u\le t}$.

Dictionary Update The selection procedure for the kernel dictionary $\mathbf{D}_{t+1}$ is based upon greedy sparse approximation, a topic studied extensively in the compressive sensing community (Needell et al., 2008). The function $\tilde{f}_{t+1} = (1-\eta_t) f_t - \eta_t \nabla_f \ell(f_t; \mathbf{x}_t, \mathbf{y}_t)$ defined by stochastic gradient method without projection is parameterized by dictionary $\tilde{\mathbf{D}}_{t+1}$ [cf. (18)], whose model order is $\tilde{M} = M_t + 1$. We form $\mathbf{D}_{t+1}$ by selecting a subset of M_{t+1} columns from $\tilde{\mathbf{D}}_{t+1}$ that are best for approximating $\tilde{f}_{t+1}$ in terms of error with respect to the Hilbert norm. As previously noted, numerous approaches are possible for seeking a sparse representation. We make use of *kernel orthogonal matching pursuit* (KOMP) (Vincent and Bengio, 2002) with allowed error tolerance ϵ_t to find a kernel dictionary matrix $\mathbf{D}_{t+1}$ based on the one which adds the latest sample point $\tilde{\mathbf{D}}_{t+1}$. This choice is due to the fact that we can tune its stopping criterion to guarantee stochastic descent, and guarantee the model order of the learned function remains finite – see Section 4 for details. Similar ideas have independently appeared in a prior work but without any analysis Zhao et al. (2012).

We now describe the variant of KOMP we propose using, called Destructive KOMP with Pre-Fitting (see Vincent and Bengio (2002), Section 2.3), which is summarized in

Algorithm 1 Destructive Kernel Orthogonal Matching Pursuit (KOMP)

Require: function $\tilde{f}$ defined by dict. $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$, coeffs. $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$, approx. budget $\epsilon_t > 0$
initialize $f = \tilde{f}$, dictionary $\mathbf{D} = \tilde{\mathbf{D}}$ with indices $\mathcal{I}$, model order $M = \tilde{M}$, coeffs. $\mathbf{w} = \tilde{\mathbf{w}}$.

while candidate dictionary is non-empty $\mathcal{I} \neq \emptyset$ **do**
for $j = 1, \dots, \tilde{M}$ **do**
Find minimal approximation error with dictionary element $\mathbf{d}_j$ removed

$$\gamma_j = \min_{\mathbf{w}_{\mathcal{I}\setminus\{j\}} \in \mathbb{R}^{M-1}} \|\tilde{f}(\cdot) - \sum_{k \in \mathcal{I}\setminus\{j\}} w_k \kappa(\mathbf{d}_k, \cdot)\|_{\mathcal{H}} \, .$$

end for
Find dictionary index minimizing approximation error: $j^* = \operatorname{argmin}_{j \in \mathcal{I}} \gamma_j$
if minimal approximation error exceeds threshold $\gamma_{j^*} > \epsilon_t$
stop
else
Prune dictionary $\mathbf{D} \leftarrow \mathbf{D}_{\mathcal{I}\setminus\{j^*\}}$
Revise set $\mathcal{I} \leftarrow \mathcal{I} \setminus \{j^*\}$ and model order $M \leftarrow M - 1$.
Compute updated weights $\mathbf{w}$ defined by current dictionary $\mathbf{D}$

$$\mathbf{w} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^M} \|\tilde{f}(\cdot) - \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$$

end
end while
return $f, \mathbf{D}, \mathbf{w}$ of model order $M \leq \tilde{M}$ such that $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$

Algorithm 1. This flavor of KOMP takes as an input a candidate function $\tilde{f}$ of model order $\tilde{M}$ parameterized by its kernel dictionary $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$ and coefficient vector $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$. The method then seeks to approximate $\tilde{f}$ by a parsimonious function $f \in \mathcal{H}$ with a lower model order. Initially, this sparse approximation is the original function $f = \tilde{f}$ so that its dictionary is initialized with that of the original function $\mathbf{D} = \tilde{\mathbf{D}}$, with corresponding coefficients $\mathbf{w} = \tilde{\mathbf{w}}$. Then, the algorithm sequentially removes dictionary elements from the initial dictionary $\tilde{\mathbf{D}}$, yielding a sparse approximation f of $\tilde{f}$, until the error threshold $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ is violated, in which case it terminates.

At each stage of KOMP, a single dictionary element j of $\mathbf{D}$ is selected to be removed which contributes the least to the Hilbert-norm approximation error $\min_{f \in \mathcal{H}_{\mathbf{D}\setminus\{j\}}} \|\tilde{f} - f\|_{\mathcal{H}}$ of the original function $\tilde{f}$, when dictionary $\mathbf{D}$ is used. Since at each stage the kernel dictionary is fixed, this amounts to a computation involving weights $\mathbf{w} \in \mathbb{R}^{M-1}$ only; that is, the error of removing dictionary point $\mathbf{d}_j$ is computed for each j as $\gamma_j = \min_{\mathbf{w}_{\mathcal{I}\setminus\{j\}} \in \mathbb{R}^{M-1}} \|\tilde{f}(\cdot) - \sum_{k \in \mathcal{I}\setminus\{j\}} w_k \kappa(\mathbf{d}_k, \cdot)\|$. We use the notation $\mathbf{w}_{\mathcal{I}\setminus\{j\}}$ to denote the entries of $\mathbf{w} \in \mathbb{R}^M$ restricted to the sub-vector associated with indices $\mathcal{I} \setminus \{j\}$. Then, we define the dictionary element which contributes the least to the approximation error as $j^* = \operatorname{argmin}_j \gamma_j$. If the error associated with removing this kernel dictionary element exceeds the given approximation budget $\gamma_{j^*} > \epsilon_t$, the algorithm terminates. Otherwise, this dictionary element $\mathbf{d}_{j^*}$ is removed, the weights $\mathbf{w}$ are revised based on the pruned dictionary as

Algorithm 2 Parsimonious Online Learning with Kernels (POLK)

Require: $\{\mathbf{x}_t, \mathbf{y}_t, \eta_t, \epsilon_t\}_{t=0,1,2,\ldots}$

initialize $f_0(\cdot) = 0, \mathbf{D}_0 = [], \mathbf{w}_0 = []$, i.e. initial dictionary, coefficient vectors are empty

for $t = 0, 1, 2, \ldots$ **do**

Obtain independent training pair realization $(\mathbf{x}_t, y_t)$

Compute unconstrained functional stochastic gradient step [cf. (17)]

$$\tilde{f}_{t+1}(\cdot) = (1 - \eta_t\lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot)$$

Revise dictionary $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \mathbf{x}_t]$ and weights $\tilde{\mathbf{w}}_{t+1} \leftarrow [(1-\eta_t\lambda)\mathbf{w}_t, \ -\eta_t\ell'(f_t(\mathbf{x}_t), y_t)]$

Compute sparse function approximation via Algorithm 1

$$(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \mathbf{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$$

end for

$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}\in\mathbb{R}^M} \|\tilde{f}(\cdot) - \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$, and the process repeats as long as the current function approximation is defined by a nonempty dictionary.

With Algorithm 1 stated, we may summarize the key steps of the proposed method in Algorithm 2 for solving (1) while maintaining a finite model order, thus breaking the "curse of kernelization." The method, Parsimonious Online Learning with Kernels (POLK), executes the stochastic projection of the functional stochastic gradient iterates onto sparse subspaces $\mathcal{H}_{\mathbf{D}_{t+1}}$ stated in (16). The initial function is set to null $f_0 = 0$, meaning that it has empty kernel dictionary $\mathbf{D}_0 = []$ and coefficient vector $\mathbf{w}_0 = []$. The notation [] is used to denote the empty matrix or vector respective size $p \times 0$ or 0. Then, at each step, given an independent training example $(\mathbf{x}_t, y_t)$ and step-size η_t, we compute the *unconstrained* functional stochastic gradient iterate $\tilde{f}_{t+1}(\cdot) = (1 - \eta_t\lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot)$ which admits the parametric representation $\tilde{\mathbf{D}}_{t+1}$ and $\tilde{\mathbf{w}}_{t+1}$ as stated in (18). These parameters are then fed into KOMP with approximation budget ϵ_t, such that $(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \text{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$.

In the next section, we discuss the analytical properties of Algorithm 2 for solving online nonparametric regression problems of the form (1). We close here with an example algorithm derivation for the kernel logistic regression problem stated in Example 1 and a remark regarding the selection of KOMP to execute the greedy compression.

Example 2 *(Kernel Logistic Regression)* Returning to the case of kernel logistic regression stated in Example 1, with feature vectors $\mathbf{x}_n \in \mathcal{X} \subseteq \mathbb{R}^p$ and binary class labels $y_n \in \{0, 1\}$, we may perform sparse function estimation in $\mathcal{H}$ that fits a a training example $\mathbf{x}$ to its associated label y under the logistic model [cf. (7)] of the odds-ratio of the given class label. The associated λ-regularized maximum-likelihood estimation (MLE) is given as (8). Provided that a particular kernel map $\kappa(\cdot, \cdot)$, regularizer λ, and step-size η_t have been chosen, the only specialization of Algorithm 2 to this case is the computation of $\tilde{f}_t$, which requires computing the stochastic gradient of (7) with respect to an instantaneous training

example $(\mathbf{x}_t, y_t)$. Doing so specializes (17) to

$$\tilde{f}_{t+1}(\cdot) = (1 - \eta_t\lambda)f_t - \eta_t \frac{\exp\{-\tilde{f}_t(\mathbf{x}_t)\}}{[1 + \exp\{-\tilde{f}_t(\mathbf{x}_t)\}]^2}\kappa(\mathbf{x}_t, \cdot) \; . \tag{21}$$

The resulting dictionary and parameter updates implied by (21), given in (18), are then fed into KOMP (Algorithm 1) which returns their greedy sparse approximation for a fixed budget ϵ_t.

Remark 1 *(On the Choice of Compression Method)* Many penalty-based compression methods (Fu, 1998; Zou and Hastie, 2005) in principle could be used to reduce the complexity of the nonparametric regressor through the introduction of sparsity-promiting terms that are added to the objective in (1). However, given the infinite sample size setting of the problem considered in this work, it is not enough to sparsify the solution, since (1) must be solved with FSGD [cf. (12)]. The per-iteration complexity of FSGD [cf. (14)] that is necessary to actually solve the optimization problem means that it will still be untenably computationally burdensome even to find a sparsified solution. Thus, it is necessary to apply sparsification in an *online manner* to the FSGD sequence, akin to Algorithm 2. However, compressing the FSGD sequence may violate the descent properties of the optimization algorithm, which motivates us to address how to compress the parameterization while preserving stochastic descent. In POLK, this is done by identifying the relationship between the projected and un-projected stochastic gradient, and noting that the directional error in this relationship may be controlled by tying compression budget ϵ_t to the choice of algorithm learning rate η_t.

Therefore, to ensure that we actually attain the minimizer of the expected risk functional (1) almost surely, we need to sparsify FSGD while preserving the validity of the descent direction in the function space. In principle, it is possible to do this with *any method* that sparsifies FSGD while guaranteeing the function sequence moves in a Hilbert-norm error neighborhood of the true functional stochastic gradient. The reason we select KOMP in particular is that it easily admits a Hilbert-norm error neighborhood stopping criterion in conjunction with its compression procedure, and we were not aware of lower complexity alternatives that meet this stringent criterion. The result is that the complexity of the function sequence is then tailored to solve the learning task at hand and the model order is directly modified on the fly by that which is necessary to represent f^*, rather than arbitrarily fixed in advance of training.

Alternative online sparsification methods based on sliding windows (Kivinen et al., 2004), fixed-size subspace projections (Wang and Vucetic, 2010a; Wang et al., 2012; Zhao and Hoi, 2012), approximate linear dependence tests (Engel et al., 2004; Richard et al., 2009), or random dropping methods Zhang et al. (2013); Le et al. (2016b) cannot directly control the directional error/bias induced by sparsification so that the algorithm approximately behaves like FSGD. This is because the descent direction is intrinsically tied to its parameterization associated with a sequentially expanding kernel dictionary (14) (see Section 3.1), and these methods are proposed to reduce the complexity of the kernel representation *independently* of the optimization sequence to which they are applied.

4. Convergence Analysis

We turn to studying the theoretical performance of Algorithm 2 developed in Section 3. In particular, we establish that the method, when a diminishing step-size is chosen, is guaranteed to converge to the optimum of (1). We further obtain that when a sufficiently small constant step-size is chosen, the limit infimum of the iterate sequence is within a neighborhood of the optimum. In both cases, the convergence behavior depends on the approximation budget used in the online sparsification procedure detailed in Algorithm 1.

We also perform a worst-case analysis of the model order of the instantaneous iterates resulting from Algorithm 2, and show that asymptotically the model order depends on that of the optimal $f^* \in \mathcal{H}$. Before proceeding with the analysis, we define key quantities to simplify the analysis and introduce standard assumptions which are necessary to establish convergence. First, define the regularized stochastic functional gradient as

$$\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) = \nabla_f \ell(f_t(\mathbf{x}_t), y_t) + \lambda f_t \tag{22}$$

Further define the projected stochastic functional gradient associated with the update in (16) as

$$\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) = \Big(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big] \Big) / \eta_t \tag{23}$$

such that the Hilbert space update of Algorithm 2 [cf. (16)] may be expressed as a stochastic descent using projected functional gradients

$$f_{t+1} = f_t - \eta_t \tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \; . \tag{24}$$

The definitions (23) - (22) will be used to analyze the convergence behavior of the algorithm. Before doing so, observe that the stochastic functional gradient in (22), based upon the fact that $(\mathbf{x}_t, y_t)$ are independent and identically distributed realizations of the random pair $(\mathbf{x}, y)$, is an unbiased estimator of the true functional gradient of the regularized expected risk $R(f)$ in (1), i.e.

$$\mathbb{E}[\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \,\big|\, \mathcal{F}_t] = \nabla_f R(f_t) \tag{25}$$

for all t. Next, we formally state technical conditions on the loss functions, data domain, and stochastic approximation errors that are necessary to establish convergence.

Assumption 1 *The feature space $\mathcal{X} \subset \mathbb{R}^p$ and target domain $\mathcal{Y} \subset \mathbb{R}$ are compact, and the reproducing kernel map may be bounded as*

$$\sup_{\mathbf{x} \in \mathcal{X}} \sqrt{\kappa(\mathbf{x}, \mathbf{x})} = X < \infty \tag{26}$$

Assumption 2 *The instantaneous loss $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is uniformly C-Lipschitz continuous for all $z \in \mathbb{R}$ for a fixed $y \in \mathcal{Y}$*

$$|\ell(z, y) - \ell(z', y)| \leq C|z - z'| \tag{27}$$

Assumption 3 *The loss function $\ell(f(\mathbf{x}), y)$ is convex and differentiable with respect to its first (scalar) argument $f(\mathbf{x})$ on $\mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

Assumption 4 *Let $\mathcal{F}_t$ denote the sigma algebra which measures the algorithm history for times $u < t$, i.e. $\mathcal{F}_t = \{\mathbf{x}_u, y_u, u_u\}_{u=1}^{t}$. The projected functional gradient of the regularized instantaneous risk in* (22) *has finite conditional second moments for each t, that is,*

$$\mathbb{E}[\|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \mid \mathcal{F}_t] \leq \sigma^2 \tag{28}$$

Assumption 1 holds in most practical settings by the data domain itself, and justifies the bounding of the loss in Assumption 2. Taken together, these conditions permit bounding the optimal function f^* in the Hilbert norm, and imply that the worst-case model order is guaranteed to be finite. Variants of Assumption 2 appear in the analysis of stochastic descent methods in the kernelized setting (Pontil et al., 2005; Ying and Zhou, 2006). Assumption 3 is satisfied for supervised learning problems such as logistic regression, support vector machines with the square-hinge-loss, the square loss, among others. Moreover, it is a standard condition in the analysis of descent methods (see Boyd and Vanderberghe (2004)). Assumption 4 is common in stochastic approximation literature, and ensures that the variance of the stochastic approximation error is finite. Under these conditions, we establish technical results in Appendix A which are used to establish our main convergence results in the following subsection.

4.1 Iterate Convergence

As is customary in the analysis of stochastic algorithms, we establish that under a diminishing algorithm step-size scheme (non-summable and square-summable), with the sparse approximation budget selection

$$\sum_{t=1}^{\infty} \eta_t = \infty \,, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty \,, \quad \epsilon_t = \eta_t^2 \,, \tag{29}$$

Algorithm 2 converges exactly to the optimal function f^* in stated in (1) almost surely. We use the abbreviation a.s. for an event that occurs almost surely, or with probability 1.

Theorem 2 *Consider the sequence generated $\{f_t\}$ by Algorithm 2 with $f_0 = 0$, and denote f^* as the minimizer of the regularized expected risk stated in* (1)*. Let Assumptions 1-4 hold and suppose the step-size rules and approximation budget are diminishing as in* (29) *with regularizer such that $\eta_t < 1/\lambda$ for all t. Then the objective function error sequence converges to null in infimum almost surely as*

$$\liminf_{t\to\infty} R(f_t) - R(f^*) = 0 \qquad a.s. \tag{30}$$

Moreover, the sequence of functions $\{f_t\}$ converges almost surely to the optimum f^* as

$$\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}}^2 = 0 \qquad a.s. \tag{31}$$

Proof: See Appendix B. ■

The result in Theorem 2 states that when a diminishing algorithm step-size is chosen as, e.g. $\eta_t = \mathcal{O}(1/t)$, and the approximation budget that dictates the size of the sparse stochastic subspaces onto which the iterates are projected is selected as $\epsilon_t = \eta_t^2$, we obtain

exact convergence to the optimizer of the regularized expected risk in (1). However, in obtaining exact convergence behavior, we require the approximation budget to approach null asymptotically, which means that the model order of the resulting function sequence may grow arbitrarily, unless f^* is sparse and the magnitude of the stochastic gradient reduces sufficiently quickly, i.e., comparable to $\epsilon_t = \mathcal{O}(1/t^2)$.

If instead we consider a constant algorithm step-size $\eta_t = \eta$ and the approximation budget $\epsilon_t = \epsilon$ is chosen as a constant which satisfies $\epsilon_t = \epsilon = \mathcal{O}(\eta^{3/2})$, we obtain that the iterates converge in infimum to a neighborhood of the optimum, as we state next.

Theorem 3 *Denote $\{f_t\}$ as the sequence generated by Algorithm 2 with $f_0 = 0$, and denote f^* as the minimizer of the regularized expected risk stated in* (1)*. Let Assumptions 1-4 hold, and given regularizer $\lambda > 0$, suppose a constant algorithm step-size $\eta_t = \eta$ is chosen such that $\eta < 1/\lambda$, and the sparse approximation budget satisfies $\epsilon = K\eta^{3/2} = \mathcal{O}(\eta^{3/2})$, where K is a positive scaler. Then the algorithm converges to a neighborhood almost surely as*

$$\liminf_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} \leq \frac{\sqrt{\eta}}{\lambda}\left(K + \sqrt{K^2 + \lambda\sigma^2}\right) = \mathcal{O}(\sqrt{\eta}) \qquad a.s. \tag{32}$$

Proof: See Appendix C. ■

Theorem 3 states that when a sufficiently small constant step-size is used together with a bias tolerance induced by sparsification chosen as $\epsilon = \mathcal{O}(\eta^{3/2})$, Algorithm 2 converges in infimum to a neighborhood of the optimum which depends on the chosen step-size, the parsimony constant K which scales the approximation budget ϵ, the regularization parameter λ, as well as the variance of the stochastic gradient σ^2. This result again is typical of convergence results in stochastic gradient methods. However, the use of a constant learning rate allows use to guarantee the model order of the resulting function sequence is always bounded, as we establish in the following subsection.

4.2 Model Order Control

In this subsection, we establish that the sequence of functions $\{f_t\}$ generated by Algorithm 2, when a constant algorithm step-size is selected, is parameterized by a kernel dictionary which is guaranteed to have finitely many elements, i.e., its the model order remains bounded. We obtain that the worst-case bound on the model order of f_t depends by the topological properties of the feature space $\mathcal{X}$, the Lipschitz constant of the instantaneous loss, and the radius of convergence $\Delta = (\sqrt{\eta}/\lambda)(K + \sqrt{K^2 + \lambda\sigma^2})$ defined in Theorem 3.

Theorem 4 *Denote f_t as the function sequence defined by Algorithm 2 with constant step-size $\eta_t = \eta < 1/\lambda$ and approximation budget $\epsilon = K\eta^{3/2}$ where $K > 0$ is an arbitrary positive scalar. Let M_t be the model order of f_t i.e., the number of columns of the dictionary $\mathbf{D}_t$ which parameterizes f_t. Then there exists a finite upper bound M^∞ such that, for all $t \geq 0$, the model order is always bounded as $M_t \leq M^\infty$. Consequently, the model order of the limiting function $f^\infty = \lim_t f_t$ is finite.*

Proof: See Appendix D.1. ■

The number of kernel dictionary elements in the function sequence f_t generated by Algorithm 2 is in the worst-case determined by the packing number of the kernel transformation

Table 1: Summary of convergence results for different parameter selections.

	Diminishing	Constant
Step-size/Learning rate	$\eta_t = \mathcal{O}(1/t)$	$\eta_t = \eta > 0$
Sparse Approximation Budget	$\epsilon_t = \eta_t^2$	$\epsilon = \mathcal{O}(\eta^{3/2})$
Regularization Condition	$\eta_t < 1/\lambda$	$\eta < 1/\lambda$
Convergence Result	$f_t \to f^*$ a.s.	$\liminf_t \|f_t - f^*\| = \mathcal{O}(\sqrt{\eta})$ a.s.
Model Order Guarantee	None	Finite

of the feature space $\phi(\mathcal{X}) = \kappa(\mathcal{X}, \cdot)$, as shown in the proof of Theorem 4. Moreover, the online sparsification procedure induced by KOMP reduces to a condition on the scale of the packing number of $\phi(\mathcal{X})$ as stated in (76). Specifically, as the radius $\frac{K\sqrt{\eta}}{C}$ increases, the packing number of the kernelized feature space decreases, and hence the required model order to fill $\phi(\mathcal{X})$ decreases. This radius depends on the constant K which scales the approximation budget selection η, the learning rate η, and the constant C bounding the gradient of the regularized instantaneous loss.

We have established that Algorithm 2 yields convergent behavior for the problem (1) in both diminishing and constant step-size regimes. When the learning rate η_t satisfies $\eta_t < 1/\lambda$, where $\lambda > 0$ is the regularization parameter, and is attenuating such that $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, i.e., $\eta_t = \mathcal{O}(1/t)$, the approximation budget ϵ_t of Algorithm 1 must satisfy $\epsilon_t = \eta_t^2$ [cf. (29)]. Practically speaking, this means that asymptotically the iterates generated by Algorithm 2 may have a very large model order in the diminishing step-size regime, since the approximation budget is vanishing as $\epsilon_t = \mathcal{O}(1/t^2)$. On the other hand, when a constant algorithm step-size $\eta_t = \eta$ is chosen to satisfy $\eta < 1/\lambda$, then we only require the constant approximation budget $\epsilon_t = \epsilon$ to satisfy $\epsilon = \mathcal{O}(\eta^{3/2})$. This means that in the constant learning rate regime, we obtain a function sequence which converges to a neighborhood of the optimal f^* defined by (1) and is guaranteed to have a finite model order. These results are summarized in Table 1.

Remark 5 *(Sparsity of f^*)* Algorithm 2 provides a method to avoid keeping an unnecessarily large number of kernel dictionary elements along the convergence path towards f^* [cf. (1)], solving the classic scalability problem of kernel methods in stochastic programming. However, if the optimal function admits a low dimensional representation $|\mathcal{I}| \ll \infty$, then in addition to extracting memory efficient instantaneous iterates, POLK will obtain the optimal function exactly. In Section 5, we illustrate this property via a multi-class classification problem where the data is generated from Gaussian mixture models.

4.3 Complexity Analysis

We now study the per-iteration computational complexity of Algorithm 2. Throughout the following discussion, we shall use M_t to denote the model order at iteration t, and p to

denote the dimension of the input (feature) space. For convenience of analysis, Algorithm 2 can be thought of as operating in two distinct phases:

i. **Append**: augment the dictionary and weights so as to take the true FSGD step (i.e., such that they represent $\tilde{f}_{t+1}$).

ii. **Prune**: use matching pursuit to reduce the model order (i.e., compute the approximation f_{t+1}).

In analyzing the append phase, there are two major operations to consider. The first is the computation of the coefficient $\ell'(f(\mathbf{x}_t), y_t)$ for the new point, which requires M_t kernel evaluations, $\{\kappa(\mathbf{d}_j, \mathbf{x}_t)\}_{j=1}^{M_t}$, to be weighted and summed. Computing the kernel values exhibits complexity $\mathcal{O}(pM_t^2)$ (the complexity of a kernel evaluation scales at least linearly with p), and the weighting and summing exhibits complexity $\mathcal{O}(M_t)$. The second major operation that takes place in the append phase is that of augmenting the kernel matrix. However, for computational efficiency, a copy of the inverse of this kernel matrix is also kept, which can be updated in $O(M_t^2)$ time using classical rank-one, block-inverse update rules, based on the Sherman-Woodbury-Morrison identity (see, e.g., KRLS in Engel et al. (2004)). Therefore, the total computational complexity of the append phase is $\mathcal{O}(M_t^2)$.

To analyze the pruning phase, we first introduce some additional notation. Let Z_t denote the total number of the outer iterations of Algorithm 1 that actually take place at time t, i.e., the number of points removed from $\tilde{\mathbf{D}}$. Using an efficient implementation of Algorithm 1 inspired by Vincent and Bengio (2002), each of these Z_t iterations exhibits a computational complexity of $\mathcal{O}(M_t^2)$. After the stopping criterion has been met, the augmented function is projected onto the subspace spanned by the pruned dictionary, which also exhibits a complexity of $\mathcal{O}(M_t^2)$. Finally, the new kernel matrix and its inverse are updated, which, too, incurs a computational cost of $\mathcal{O}(M_t^2)$, bringing the overall complexity of the pruning phase to $\mathcal{O}(Z_t M_t^2)$. A pessimistic estimate of Z_t is M_t, although in practice it is always less than the mini-batch size B.

Combining the terms from each phase, the total complexity of Algorithm 2 at iteration t comes out to $O(Z_t M_t^2)$. In the steady-state, $Z = 1$ (i.e., $M_{t+1} = M_t$), which further reduces the expression to $\mathcal{O}(M_t^2)$, exactly the complexity of KRLS (Engel et al., 2004). Observe that the complexity of FSGD is $\mathcal{O}(t)$, which means that POLK is computationally advantageous whenever $t > Z_t M_t^2$. Since $\lim_t M_t$ is guaranteed to be finite, and usually $Z_t \leq B$, the mini-batch size.Thus, POLK is justified whenever t is large. This is typical of the online methods for kernel learning that we compare against experimentally in the following section.

Remark 6 *(Cheaper Projections)* Both theoretically and in practice, it is possible to cheapen the cost of projections by replacing the argmin in KOMP by a scheme that drops kernel dictionary elements *uniformly at random* until the function hits the boundary of a Hilbert-norm error neighborhood. Such a scheme's convergence would still be guaranteed by the preceding theoretical analysis, and the cost of pruning points would be significantly cheaper. Specifically, it would be at most linear in the dictionary size $\mathcal{O}(M_t Z_t)$. However, the price of pursuing this route would be lower quality estimates of f^*. We have left the investigation of these random approximated projections to future work. However, we note that several

alternatives in the literature adopt a version of this approach, but with *fixed* subspace sizes (Wang and Vucetic, 2010a; Wang et al., 2012; Zhao and Hoi, 2012; Zhang et al., 2013; Le et al., 2016b; Lu et al., 2016) rather than error-neighborhood budgets, as is done in this work, which is not enough to establish almost sure convergence.

5. Experiments

In this section, we evaluate POLK by considering its performance on two supervised learning tasks trained for three streaming data sets. The specific tasks we consider are those of (a) training a multi-class kernel logistic regressor (KLR), and (b) training a multi-class kernel support vector machine (KSVM). The three data sets we use are (i) `multidist`, a synthetic data set we constructed using two-dimensional Gaussian mixture models; (ii) `mnist`, the MNIST handwritten digits (Lecun and Cortes, 2009); and (iii) `brodatz`, image textures drawn from a subset of the Brodatz texture database (Brodatz, 1966).

Where possible, we compare our technique with competing methods. Specifically, for the online support vector machine case, we compare with budgeted stochastic gradient descent (BSGD) Wang et al. (2012), a fixed subspace projection method, which requires a maximum model order *a priori*; Dual Space Gradient Descent (Dual) (Le et al., 2016a), which executes a hybrid combination of functional stochastic gradient method together with a hybrid random feature representation of the regression function; nonparametric budgeted stochastic gradient descent (NPBSGD) Le et al. (2016b), which combines a fixed subspace projection with random dropping, Naive Online R_{reg} Minimization Algorithm (NORMA) (Kivinen et al., 2004), which does a greedy objective function value based truncation to a fixed budget, and an instantiation of Budgeted Passive-Aggressive (BPA) algorithm based on k-nearest neighbor merging of incoming points (Wang and Vucetic, 2010a). We note that all the aforementioned methods fix the model order except Dual (Le et al., 2016a). Further, for logistic regression, from the preceding list, we omit BPA and BSGD since they are derived specifically for the SVM setting. We compare all of these online methods on our first data set but then only compare POLK and BSGD on the later benchmark data domains, since the trends we observe for the former directly scale up to the later.

For off-line (batch) KLR, we compare with the import vector machine (IVM) (Zhu and Hastie, 2005), a sparse second-order method. We also compare with the batch techniques of LIBSVM (Chang and Lin, 2011), applicable to KSVM only, an L-BFGS solver (Nocedal, 1980), and methods which approximate the kernel matrix using either Fourier transforms (Rahimi and Recht, 2008, 2009) or Nyström approximation (Williams and Seeger, 2001; Zhang et al., 2008).

5.1 Tasks

The tasks we consider are those of multi-class classification, which is a problem that admits approaches based on probabilistic and geometric criteria. In what follows, we use $\mathbf{x}_n \in \mathcal{X} \subset \mathbb{R}^p$ to denote the n^{th} feature vector in a given data set, and $y_n \in \{1, \dots, C\}$ to denote its corresponding label.

Multi-class Kernel Support Vector Machines (Multi-KSVM) The first task we consider is that of training a multi-class kernel support vector machine, in which the merit of a particular regressor is defined by its ability to maximize its classification margin. In

particular, define a set of class-specific activation functions $f_c : \mathcal{X} \to \mathbb{R}$, and denote them jointly as $\mathbf{f} \in \mathcal{H}^C$. In Multi-KSVM, points are assigned the class label of the activation function that yields the maximum response. KSVM is trained by taking the instantaneous loss ℓ to be the multi-class hinge function which defines the margin separating hyperplane in the kernelized feature space, i.e.,

$$\ell(\mathbf{f}, \mathbf{x}_n, y_n) = \max(0, 1 + f_r(\mathbf{x}_n) - f_{y_n}(\mathbf{x}_n)) + \lambda \sum_{c'=1}^{C} \|f_{c'}\|_{\mathcal{H}}^2 \,, \tag{33}$$

where $r = \operatorname{argmax}_{c' \neq y} f_{c'}(\mathbf{x})$. Further details may be found in Murphy (2012).

Multi-class Kernel Logistic Regression (Multi-KLR) The second task we consider is that of kernel logistic regression, wherein, instead of maximizing the margin which separates sample points in the kernelized feature space, we instead adopt a probabilistic model on the odds ratio that a sample point has a specific label relative to all others. Using the same notation as above for the class-specific activation functions, we adopt the probabilistic model:

$$P(y = c \,|\, \mathbf{x}) \triangleq \frac{\exp(f_c(\mathbf{x}))}{\sum_{c'} \exp(f_{c'}(\mathbf{x}))}. \tag{34}$$

which models the odds ratio of a given sample point being in class c versus all others. We use the negative log likelihood pertaining to the above model as the instantaneous loss (see, e.g., Murphy (2012)), i.e.,

$$\begin{aligned} \ell(\mathbf{f}, \mathbf{x}_n, y_n) &= -\log P(y = y_n | \mathbf{x}_n) + \frac{\lambda}{2} \sum_c \|f_c\|_{\mathcal{H}}^2 \\ &= \log \left(\sum_{c'} \exp(f_{c'}(\mathbf{x}_n)) \right) - f_{y_n}(\mathbf{x}_n) + \frac{\lambda}{2} \sum_c \|f_c\|_{\mathcal{H}}^2 \,. \end{aligned} \tag{35}$$

Observe that the loss (35) substituted into the empirical risk minimization problem in Example 1 is its generalization to multi-class classification. For a given set of activation functions, the classification decision $\tilde{c}$ for $\mathbf{x}$ is given by the class that yields the maximum likelihood, i.e., $\tilde{c} = \operatorname{argmax}_{c \in \{1,\ldots,C\}} f_c(\mathbf{x})$.

5.2 Data Sets

We evaluate Algorithm 2 for the Multi-KLR and Multi-KSVM tasks described above using the `multidist`, `mnist`, `brodatz` data sets.

`multidist`

In a manner similar to (Zhu and Hastie, 2005), we generate the `multidist` data set using a set of Gaussian mixture models. The data set consists $N = 5000$ feature-label pairs for training and 2500 for testing. Each label y_n was drawn uniformly at random from the label set. The corresponding feature vector $\mathbf{x}_n \in \mathbb{R}^p$ was then drawn from a planar ($p = 2$), equitably-weighted Gaussian mixture model, i.e., $\mathbf{x} \,\big|\, y \sim (1/3) \sum_{j=1}^{3} \mathcal{N}(\boldsymbol{\mu}_{y,j}, \sigma_{y,j}^2 \mathbf{I})$ where $\sigma_{y,j}^2 = 0.2$ for all values of y and j. The means $\boldsymbol{\mu}_{y,j}$ are themselves realizations of their own Gaussian distribution with class-dependent parameters, i.e., $\boldsymbol{\mu}_{y,j} \sim \mathcal{N}(\boldsymbol{\theta}_y, \sigma_y^2 \mathbf{I})$, where

(a) `multidist` data (b) `brodatz` example texture (c) `mnist` examples

Figure 1: Visualizations of the data sets used in experiments.

$\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C\}$ are equitably spaced around the unit circle, one for each class label, and $\sigma_y^2 = 1.0$. We fix the number of classes $C = 5$, meaning that the feature distribution has, in total, 15 distinct modes. The data points are plotted in Figure 1(a).

`mnist`

The `mnist` data set we use is the popular MNIST data set (Lecun and Cortes, 2009), which consists of $N = 60000$ feature-label pairs for training and 10000 for testing. Feature vectors are $p = 784$-dimensional, where each dimension captures a single grayscale pixel value (scaled to lie within the unit interval) that corresponds to a unique location in a 28-pixel-by-28-pixel image of a cropped, handwritten digit. Labels indicate which digit is written, i.e., there are $C = 10$ classes total, corresponding to digits $0, \ldots, 9$ – examples are given in Figure 1(c).

`brodatz`

We generated the `brodatz` data set using a subset of the images provided in Brodatz (1966). Specifically, we used 13 texture images (i.e., $C = 13$), and from them generated a set of 256 textons (Leung and Malik, 1999). Next, for each overlapping patch of size 24-pixels-by-24-pixels within these images, we took the feature to be the associated $p = 256$-dimensional texton histogram. The corresponding label was given by the index of the image from which the patch was selected. When then randomly selected $N = 10000$ feature-label pairs for training and 5000 for testing. An example texture image can be seen in Figure 1(b).

5.3 Parameter Selection

For the choice of kernel, some intuition regarding the data domain is helpful, but typically the Gaussian kernel is a good starting point, and therefore is our selection for all of the experiments. For bandwidth selection $\tilde{\sigma}^2$, we find that choosing it according to the sample variance of a cross-validation training subset to be reasonably effective, and this strategy was used in our implementation. The mini-batch size of 32 was selected based on reducing the noise of stochastic approximation error once steady state is achieved, and was done through trial and error by starting at batch sizes of 1 and doubling until the computational burden of a mini-batch exceeded its benefits in terms of noise reduction. We find the regularizer λ has little effect when the data does not have an untenably large variance, and therefore fix

it as a small constant. For problems with worse conditioning, however, larger regularizers may be useful.

The selection of the algorithm step-size and parsimony constant in our experiments is done through a simple trial and error procedure. We begin with a small step-size, between 0.01 and 0.1, to test whether the algorithm converges, with parsimony constant $K = 0.001$. Convergence, however, may occur while the model order grows untenably. So, we progressively increase the parsimony constant until the bias in the stochastic gradient becomes unmanageable and the algorithm diverges. We select the parsimony constant to be as large as possible while still preserving convergence of the algorithm. Then, we exploit knowledge of the trade-off between learning rate and step-size selection: smaller step-sizes yield slower learning and smaller asymptotic error. In light of this fact, we increment the step-size to increase the learning rate, and then ramp up the parsimony constant to its maximum value that preserves convergence. This trial and error procedure only needed to be repeated a few times for each data set/problem instance to obtain the POLK results presented in this section.

5.4 Results

For each task and data set described above, we implemented POLK (Algorithm 2) along with the competing methods described at the beginning of the section. For some of the tasks, only a subset of the competing methods are applicable, and in some cases such as online logistic regression, none are. Here, we shall describe the details of each experimental setting and the corresponding results.

`multidist` Results

Sparse SVM Due to the small size of our synthetic `multidist` data set, we were able to generate results for the Multi-KSVM task using each of the methods specified earlier except for IVM. For POLK, we used the following specific parameter values: we select the Gaussian/RBF kernel with bandwidth $\tilde{\sigma}^2 = 0.6$, constant learning rate $\eta = 6.0$, parsimony constant $K = 0.04$, and regularization constant $\lambda = 10^{-6}$. Further, we processed streaming samples in mini-batches of size 32. For BSGD, Dual, NORMA, NPBSGD, and BPA, we used the same $\tilde{\sigma}^2$ and λ, but achieved the best results with smaller constant learning rate $\eta = 1.0$ (perhaps due, in part, to the fact that it is unclear how to fold mini-batching into their implementation). For Dual, we set $k = 3$, which determines the how many model points we search over to omit from the dictionary and instead contribute to the random feature functional representation. For NPBSGD, we set the Bernoulli parameter $p = \min(1, \beta/t)$ with $\beta = (45/70)N$ as in the experiments of Le et al. (2016b). In order to compare with POLK, we set the competing methods' pre-specified model orders or budget parameters to be 16, i.e., the steady-state model orders of POLK parameterized with the value of K specified above.

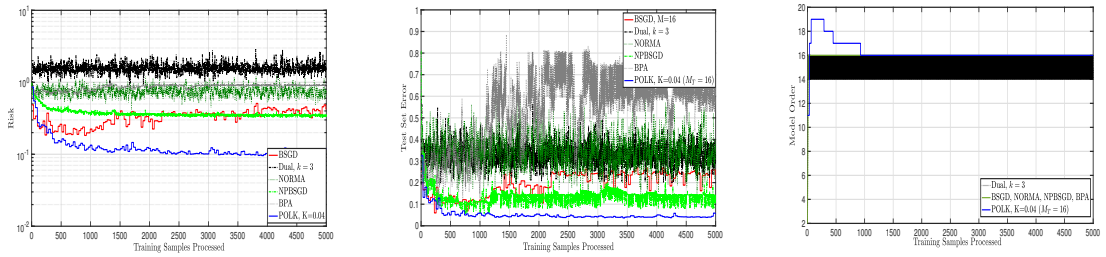
In Figure 2 we plot the empirical results of this experiment for POLK and its competitors, and observe that POLK outperforms many of its competitors by an order of magnitude in terms of objective evaluation (Fig. 2(a)) and test-set error rate (Fig 2(b)). The notable exception is NPBSGD which comes close in terms of objective evaluation but still falls short in terms of test error. Moreover, because the marginal feature density of `multidist` contains 15 modes, the optimal model order is $M^* = 15$, which is approximately *learned* by

	`multidist`	
Algorithm	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)
LIBSVM	−/3.92/656	−/−/−
L-BFGS	0.0854/4.08/5000	0.0854/4.04/5000
IVM	−/−/−	0.0894/4.08/16
Fourier	−/3.88/5000	−/3.80/5000
Nyström	−/3.67/1000	−/3.72/1000
BSGD	0.385/21.8/16	−/−/−
Dual	1.598/30.5.8/15	1.417/48.4/15
NORMA	0.751/33.1/16	1.26/29.3/16
NPBSGD	0.334/11.6/16	0.101/3.92/16
BPA	0.947/64.1/16	−/−/−
POLK	**0.0919/3.98/16**	**0.120/4.36/16**

Table 2: Comparison of different methods on the `multidist` data set for extracting the optimally sparse functional representation, i.e., $K = 0.04$ for POLK and $M = 16$ for online competitors. Reported risk and error values for POLK and BSGD were averaged over the final 5% of processed training examples. Dashes indicate where the method could not be used to generate results because it is not defined for that task. LIBSVM is used as a baseline, but note that it uses a fundamentally different model for multi-class problems (a separate one-vs-all classifier is trained for each class, and then at test time, a majority vote is executed), and so a comparable risk value can not computed.

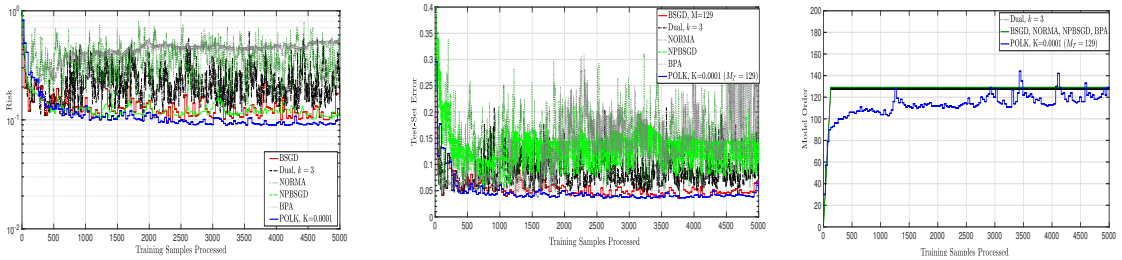
POLK for $K = 0.04$ (i.e., $M_T = 16$) (Fig. 2(c)). Several alternatives initialized with this parameter, on the other hand, do not converge. Observe that for this task POLK exhibits a state of the art trade off between test set accuracy and number of samples processed – reaching below 4% error after only 1249 samples. The final decision surface $\mathbf{f}_T$ of this trial of POLK is shown in Fig. 4(a), where it can be seen that the selected kernel dictionary elements concentrate near the modes of the marginal feature density.

We can also see from Table 2 that POLK compares favorably to the batch techniques for Multi-KSVM on the `multidist` data set. It achieves approximately the same error rate as LIBSVM with significantly fewer model points (support vectors) and even outperforms our (dense) L-BFGS batch solver in terms of test-set error, while adding the ability to process data in an online fashion. Compared to approximate batch solvers such as Random Fourier Features with $d = 1000$ features and all $N = 5000$ sample points, we observe POLK obtains a 0.1% error increase with a 99% reduction in model complexity. We also implement Nyström approximation with k-means sampled landmark points with $k = 1000$ (Zhang et al., 2008), and observe similar accuracy results to random features, which increased complexity associated with computing k means over the entire training set. We still report a model order equal to the sample size for these methods, however, since they require the full training set in order to implement their approximation procedure.



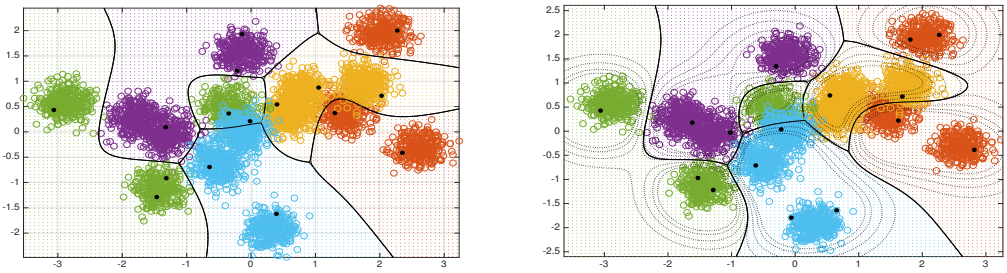
(a) Empirical risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 2: Comparison of POLK and its competitors on the `multidist` data set for the Multi-KSVM task for extracting a optimally sparse kernel regressor. Observe that POLK achieves lower risk and higher accuracy when we seek to learn a statistical model of comparable complexity to the class-conditional density ($M = 15$ total modes). Competing methods yield slower learning and higher test-set error rates.



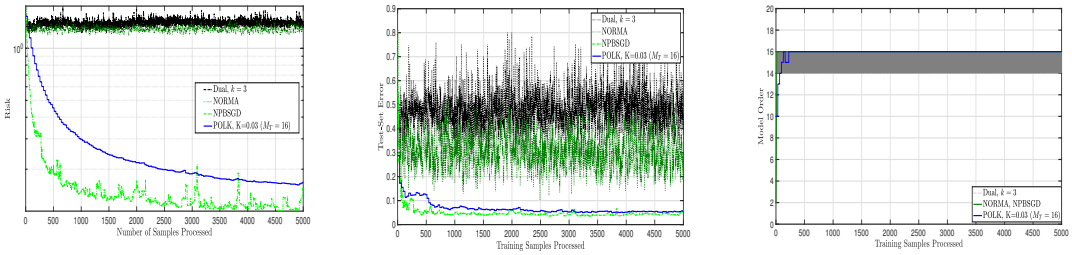
(a) Empirical risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 3: Comparison of POLK and its competitors on the `multidist` data set for the Multi-KSVM task when the model order parameter is fixed for the competitors at $M = 129$ and the parsimony constant of POLK is set to $K = 10^{-4}$. Observe that POLK achieves lower risk and higher accuracy than its competitors on this problem instance, but the gap is less relative to the sparser model extraction problem.



(a) $\mathbf{f}_T$ (hinge loss) (b) $\mathbf{f}_T$ (logistic loss)

Figure 4: Visualization of the decision surfaces yielded by POLK for the Multi-KSVM and Multi-Logisitic tasks on the `multidist` data set. Training examples from distinct classes are assigned a unique color. Grid colors represent the classification decision by $\mathbf{f}_T$. Bold black dots are kernel dictionary elements, which concentrate at the modes of the data distribution. Solid lines are drawn to denote class label boundaries, and dashed lines in 4(b) are drawn to denote confidence intervals.

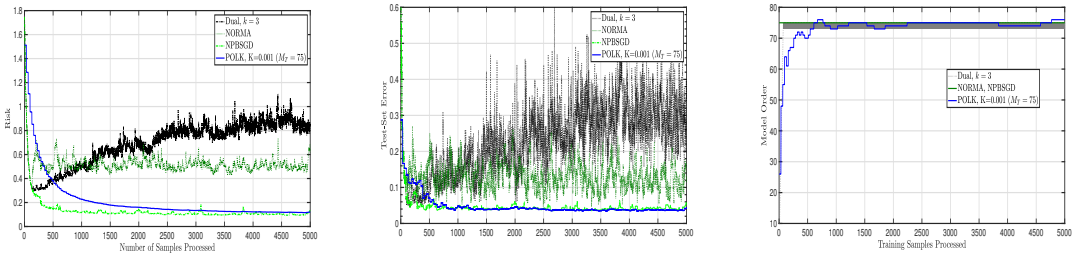


(a) Empirical risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 5: Empirical behavior of the POLK algorithm applied to the `multidist` data set for the Multi-Logistic task. Observe that the algorithm converges to a low risk value ($R(f_t) < 10^{-1}$) and achieves test set accuracy between 4% and 5% depending on choice of parsimony constant K, which respectively corresponds to a model order between 75 and 16.

Dense SVM We now repeat the previous experiment with the same parameters except for a different parsimony constant $K = 10^{-4}$ for POLK and a model order of $M = 129$ for its competitors, given that this is the value extracted by POLK for this parameter selection. The results of this experiment are given in Figure 3. Observe that NORMA and Dual appear to not converge for this experiment, while POLK converges stably, although its learning rate is exceeded by NPBSGD. This trend is observed both in terms of the risk functional (Fig. 3(a)) and test-set error (Fig 3(b)). We believe NPBSGD performs better for this case due to the smoothness of the logistic loss relative to the hinge loss. However, the test errors of these methods stabilize to similar levels of accuracy (below 4% error rate).

Sparse KLR For the Multi-Logisitic task on this data set, we were able to generate results for each method except BSGD, BPA, and LIBSVM, which are specifically tailored to the SVM task. For POLK, we used the following parameter values: Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 0.6$, constant learning rate $\eta = 6.0$, parsimony constant $K = 0.03$, and



(a) Empirical risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 6: Empirical behavior of the POLK algorithm applied to the `multidist` data set for the Multi-Logistic task. Observe that the algorithm converges to a low risk value ($R(f_t) < 10^{-1}$) and achieves test set accuracy between 4% and 5% depending on choice of parsimony constant K, which respectively corresponds to a model order between 75 and 16.

regularization constant $\lambda = 10^{-6}$. As in Multi-KSVM, we processed the streaming samples in mini-batches of size 32. The empirical behavior of POLK for the Multi-Logistic task can be seen in Figure 5 and the final decision surface is presented in Figure 4(b). Observe that POLK is exhibits comparable convergence to the SVM problem, but a smoother descent due to the differentiability of the multi-logistic loss. The competing methods, initialized with a model order of $M = 16$ do not converge, except for NPBSGD, which attains comparable test error to POLK. In Table 2 we present final accuracy and risk values on the logistic task, and note that it performs comparably, or in some cases, favorably, to the batch techniques (IVM, L-BFGS, Random Features, Nyström with k-means sampled points), while processing streaming data. The final model f_T extracted by POLK attains comparable performance to off-line methods while reducing the model complexity by several orders of magnitude.

Dense KLR Now, we re-rerun the previous experiment but modify the parsimony constant of POLK as $K = 0.001$, which yields a limiting model of order $M = 75$. The competing methods are then run with this selection. The results of this experiment are given in Figure 6: observe that NORMA and Dual diverge, while NPBSGD converges slightly faster albeit more noisily than POLK. The later two methods attain comparable test accuracy near 96% by the end of training.

`mnist` and `brodatz` Results

By construction, the `multidist` data set above yields optimal activation functions that are themselves sparse (i.e., $\mathbf{f}^*$ has a low model order due to the marginal feature density). Here, we analyze the performance of POLK on more realistic data sets where the optimal solutions are not sparse, i.e., where one might desire a sparse approximation. Due to the increased size and dimensionality of these data sets, we were unable to generate results for `mnist` using the batch L-BFGS technique, and unable to generate results for either data set using IVM. With Random Fourier Features on `mnist`, we use a bandwidth of 0.0095 with $d = 1000$ randomly chosen features for both SVM and KLR. For `brodatz`, we use a larger bandwidth of $\tilde{\sigma}^2 = 4.0$, with $d = 10^4$ Random Features both for KSVM and KLR. For Nyström approximation, we set $k = 1000$ since larger choices of k cause memory problems.

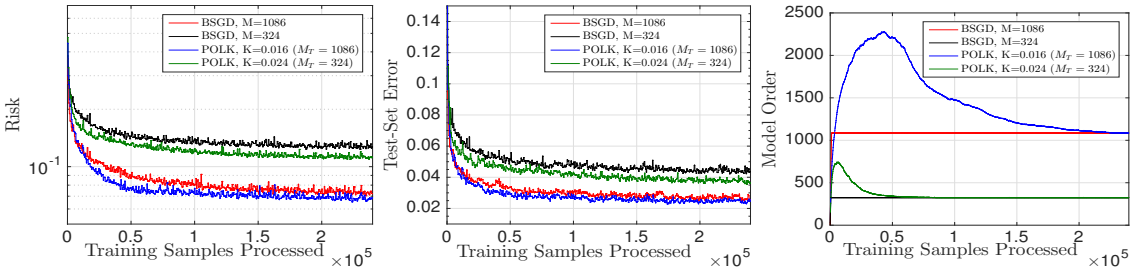
	mnist		brodatz	
Algorithm	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)
LIBSVM	$-/1.50/16118$	$-/-/-$	$-/3.72/4777$	$-/-/-$
Fourier	$-/3.83/60000$	$-/3.87/60000$	$-/5.88/10000$	$-/6.18/10000$
Nyström	$-/2.89/60000$	$-/2.94/60000$	$-/5.13/10000$	$-/5.86/10000$
L-BFGS	$-/-/-$	$-/-/-$	$0.0319/4.44/10000$	$0.0572/4.00/10000$
BSGD	$0.0731/2.67/1086$	$-/-/-$	$0.0560/4.72/1171$	$-/-/-$
POLK	$\mathbf{0.0684/2.46/1086}$	$\mathbf{0.116/2.68/2326}$	$\mathbf{0.0507/4.53/1171}$	$\mathbf{0.0871/4.41/1833}$

Table 3: Comparison of POLK, BSGD, IVM, L-BFGS, and LIBSVM results on the `mnist` and `brodatz` data sets. Reported risk and error values for POLK and BSGD were averaged over the final 5% of processed training examples. Dashes indicate where the method could not be used to generate results either because it is not defined for the task or because the size of the problem was too large for that data set. For these reasons, IVM was not able to generate results for these data sets on either task, and so is omitted here. LIBSVM is used as a baseline, but note that it uses a fundamentally different model for multi-class problems (1v1 + majority vote), and so a comparable risk value can not be computed.

For Multi-KSVM on `mnist`, we used the following parameter values for POLK: Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 4.0$, constant learning rate $\eta = 24.0$, parsimony constant $K \in \{0.16, 0.24\}$, and regularization constant $\lambda = 10^{-6}$. We again processed data in mini-batches of size 32. For `brodatz`, we used identical parameters except for changing the kernel bandwith $\tilde{\sigma}^2 = 0.1$ and parsimony constant $K \in \{0.01, 0.02\}$. For BSGD, we again found $\eta = 1.0$ to yield the best results on both datasets, and pre-specified model orders of $\{324, 1086\}$ and $\{305, 1171\}$ on `mnist` and `brodatz`, respectively, for comparison to POLK.

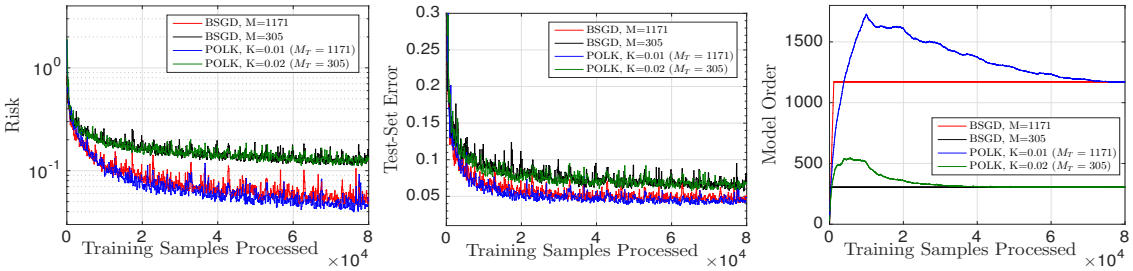
In Figures 7 and 8 we plot the empirical results of these experiments for POLK and BSGD. We observe that POLK is able to outperform the comparable BSGD trial in terms of convergence speed and steady-state risk and test-set error. The strength of the proposed technique is further demonstrated in Table 3, where we can see that POLK is able to achieve test-set error within 1-2% of LIBSVM while requiring a number of support vectors (model points) that is significantly-less than LIBSVM, *while* adding the ability to process streaming data.

For the Multi-Logistic task on `mnist`, we ran POLK using a Gaussian kernel with bandwidth $\tilde{\sigma}^2 = 4.0$, constant learning rate $\eta = 24.0$, parsimony constant $K \in \{0.08, 0.16\}$, and regularization constant $\lambda = 10^{-6}$. Data was processed in mini-batches of size 32 here as well. For `brodatz`, we again change the kernel bandwidth $\tilde{\sigma}^2 = 0.1$ and used different parsimony constants $K = \{0.005, 0.015\}$. The empirical behavior of POLK on this task can be seen in Figures 9 and 10. Observe that for this task the descent is smoother due to the differentiability of the logistic loss, although the asymptotic test accuracy is lower than that of KSVM.



(a) Expected risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 7: Comparison of POLK and BSGD on `mnist` data set for the Multi-KSVM task. Observe that POLK achieves lower risk and higher accuracy on this task, and extracts a model order directly from the feature space that yields convergence.



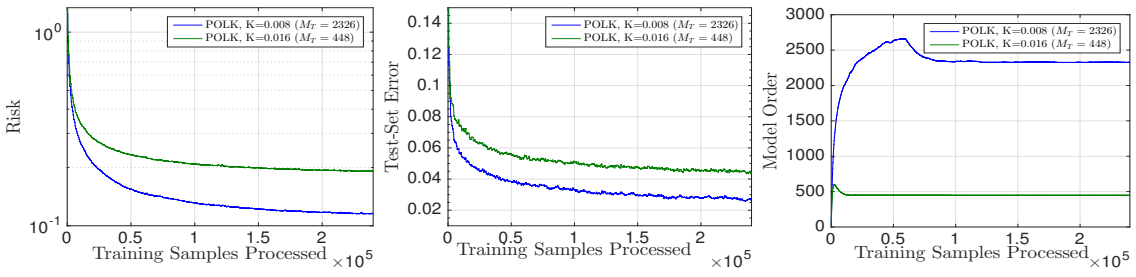
(a) Expected risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 8: Comparison of POLK and BSGD on `brodatz` data set for the Multi-KSVM task. We observe that POLK behaves similarly to BSGD for this task, stabilizing at an accuracy near 96%. For this dense data domain, larger model orders are needed to achieve convergence.

The overall performance is summarized in Table 3. Note that the only other technique that was able to generate results for this task was L-BFGS, and even there only on the `brodatz` data set, since the complexity bottleneck in the sample size for `mnist` is prohibitive for batch optimization. We see from this comparison that POLK yields a test-set error within 0.5% of the batch solution while using an order of magnitude fewer model points. Additionally, POLK is able to run *online*, with streaming data, whereas L-BFGS requires all the data points to be operated on at each step.

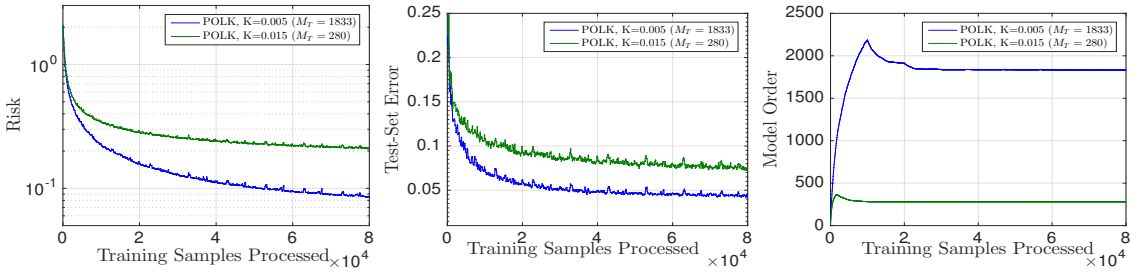
6. Discussion

Over the past several years, parametric function approximation has largely dominated the machine learning landscape. Deep learning is perhaps currently the most prominent parametric paradigm (Haykin, 1994). One must first specify a network structure, thereby fixing the parametric representation of the function to be learned, before proceeding to determine the coefficients linking neurons in different layers. Given this parametric representation, training techniques proceed by searching over the predefined parameter space for the op-



(a) Expected risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 9: Empirical behavior of the POLK algorithm applied to `mnist` data set for the Multi-Logistic task. The algorithm exhibits smoother convergence due to the differentiability of the logistic loss, and achieves asymptotic test error 2.6%. We again observe due to the dense data domain, larger model orders are needed to exhibit competitive classification performance.



(a) Expected risk $R(\mathbf{f}_t)$ (b) Error rate (c) Model order M_t

Figure 10: Empirical behavior of the POLK algorithm applied to the `brodatz` data set for the Multi-Logistic task. We observe convergent behavior, and a clear trade off between higher model order and increased accuracy. Due to this data domain being a more challenging task than the `mnist` digits, we observe asymptotic test accuracy of approximately 4.4%.

timal parameter values that minimize the error between the function and observed input-output pairs. The main reason for the popularity of parametric function approximation is its success in solving practical problems, but there are other factors that have fostered their adoption. One such factor is the availability of workable, if not necessarily efficient, optimization techniques for the determination of the optimal parameter values, in the form of stochastic gradient descent and its variants. Parametric stochastic gradient descent processes training examples sequentially and has a per-iteration complexity that is linear on the number of parameters but independent of the size of the training set.

Despite the success of parametric techniques, nonparametric function approximation has the advantage of expressive power in the sense that they are allowed to select the approximating function from a more general set of functions than those that admit a parametric form chosen *a priori*. This advantage is seen, e.g., in the improved classification accuracy of (nonparametric) kernel support vector machines (SVMs) relative to (parametric) linear SVMs (Evgeniou et al., 2000). This is not to say that nonparametric methods are necessarily better. Neural networks, e.g., have proven to be very adept parametric representations in image classification problems (Krizhevsky et al., 2012). However, it is nonetheless true that the better expressive power of nonparametric representations is of importance in some applications.

The importance of expressiveness notwithstanding, nonparametric approaches are relatively less popular. This is partly explained by the fact that, contrary to parametric approaches, workable algorithms for the minimization of functional costs are not as well-developed. Indeed, nonparametric models involve function representations that depend on an infinite number of parameters. This is a challenge not only because optimal function descriptions can become computationally intractable but, more importantly, because finding these optimal representations is itself intractable.

This work represents the first attempt at comprehensively addressing this intractability. In particular, we have proposed solving general convex expected risk minimization problems over a Hilbert space that defines nonparametric regression functions in a way that guarantees the model order of the learned function does not grow unnecessarily large. In doing so, we addressed challenges (i) - (ii) as follows: we considered the generalization of stochastic gradient descent to the kernelized expected risk setting and we compressed the learned decision function in a way that guarantees stochastic descent by tuning a greedy sparse approximation error criterion to the underlying optimization sequence. The result is an almost-sure convergent function sequence with moderate complexity that is able to operate in true online settings.

Indeed, our experiments have shown that POLK performs comparably to batch kernel methods in terms of accuracy, while its model complexity is reduced by orders of magnitude. Additionally, we observe state-of-the-art performance in terms of test-set accuracy relative to the number of samples processed. Such performance is key to achieving reasonable performance in many applications of interest, e.g., when learning on robotics platforms operating in unknown environments. In this case, the online nature of the problem is intrinsic and due to a lack of prior information on their operating domain (Koppel et al., 2016).

On the other hand, it must be noted that POLK, and even batch kernel methods, for certain large-scale supervised learning tasks, have not met the high bar of asymptotic

test set accuracy set forth by batch approaches to deep learning (Krizhevsky et al., 2012). We believe this discrepancy is on account of the single-layer nature of the nonparametric regressor, which is tied to the choice of reproducing kernel used in our experiments. Of course, more complicated multi-layer composite kernels may be used, based on the fact that a composition and positive linear combination of kernels is still a kernel (Theodoridis, 2015, Ch. 11). However, the scalable development of online nonparametric methods based on such composite kernels is not straight-forward, and left to future work.

Appendix A. Technical Results

Next we establish an auxiliary result needed to prove Theorems 2 and 3 which bounds the magnitude of the iterates of Algorithm 2 in the Hilbert norm.

Proposition 7 *Let Assumptions 1-4 hold and denote $\{f_t\}$ as the sequence generated by Algorithm 2 with $f_0 = 0$. Further denote f^* as the optimum defined by (1). Both quantities are bounded by the constant $K := CX/\lambda$ in Hilbert norm for all t as*

$$\|f_t\|_{\mathcal{H}} \leq \frac{CX}{\lambda}\,, \qquad \|f^*\|_{\mathcal{H}} \leq \frac{CX}{\lambda} \tag{36}$$

Proof: First, since we repeatedly use the Cauchy-Schwartz inequality together with the reproducing kernel property in the following analysis, we here note that for all $g \in \mathcal{H}$, $|g(\mathbf{x}_t)| \leq |\langle g, \kappa(\mathbf{x}_t, \cdot)\rangle_{\mathcal{H}}| \leq X\|g\|_{\mathcal{H}}$. Now, consider the magnitude of f_1 in the Hilbert norm, given $f_0 = 0$

$$\begin{aligned}\|f_1\|_{\mathcal{H}} &= \left\|\mathcal{P}_{\mathcal{H}_{\mathbf{D}_1}}\left[\eta_0 \nabla_f \ell(0, y_0)\right]\right\|_{\mathcal{H}} \\ &\leq \eta_0 \|\nabla_f \ell(0, y_0)\|_{\mathcal{H}} \leq \eta_0 |\ell'(0, y_0)| \|\kappa(\mathbf{x}_0, \cdot)\|_{\mathcal{H}} \\ &\leq \eta_0 CX < \frac{CX}{\lambda}\end{aligned} \tag{37}$$

The first equality comes from substituting in $f_0 = 0$ and the second inequality comes from the definition of optimality condition of the projection operator and the homogeneity of the Hilbert norm, and the third uses the derivation of the functional stochastic gradient in (10) with the Cauchy-Schwartz inequality. Lastly, we make use of Assumptions 1 and 2 to bound the scalar derivative ℓ' using the Lipschitz constant, and the boundedness of the kernel map [cf. (26)]. The final strict inequality in (37) comes from applying the step-size condition $\eta_0 < 1/\lambda$.

Now we consider the induction step. Given the induction hypothesis $\|f_t\|_{\mathcal{H}} \leq CX/\lambda$, consider the magnitude of the iterate at the time $t+1$ as

$$\begin{aligned}\|f_{t+1}\|_{\mathcal{H}} &= \left\|\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}\left[(1 - \eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t)\right]\right\|_{\mathcal{H}} \\ &\leq \|(1 - \eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}} \\ &\leq (1 - \eta_t\lambda)\|f_t\| + \eta_t \|\nabla_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}\,,\end{aligned} \tag{38}$$

where we have applied the non-expansion property of the projection operator for the first inequality on the right-hand side of (38), and the triangle inequality for the second. Now,

apply the induction hypothesis $\|f_t\|_{\mathcal{H}} \leq CX/\lambda$ to the first term on the right-hand side of (38), and the chain rule together with the triangle inequality to the second to obtain

$$\begin{aligned}\|f_{t+1}\|_{\mathcal{H}} &\leq (1-\eta_t\lambda)\frac{CX}{\lambda} + \eta_t|\ell'(f_t(\mathbf{x}_t), y_t)|\|\kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}} \\ &\leq (\frac{1}{\lambda} - \eta_t)CX + \eta_t CX = \frac{CX}{\lambda} \end{aligned} \tag{39}$$

where we have made use of Assumptions 1 and 2 to bound the scalar derivative ℓ' using the Lipschitz constant, and the boundedness of the kernel map [cf. (26)] as in the base case for f_1, as well as the fact that $\eta_t < 1/\lambda$. The same bound holds for f^* by applying the result of Section V-B of Kivinen et al. (2004) with $m \to \infty$.

■

Next we introduce a proposition which quantifies the error due to our sparse stochastic projection scheme in terms of the ratio of the sparse approximation budget to the algorithm step-size.

Proposition 8 *Given independent identical realizations $(\mathbf{x}_t, y_t)$ of the random pair $(\mathbf{x}, y)$, the difference between the projected stochastic functional gradient and the stochastic functional gradient of the regularized instantaneous risk defined by* (23) *and* (22)*, respectively, is bounded for all t as*

$$\|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}} \leq \frac{\epsilon_t}{\eta_t} \tag{40}$$

where $\eta_t > 0$ denotes the algorithm step-size and $\epsilon_t > 0$ is the approximation budget parameter of Algorithm 1.

Proof: Consider the square-Hilbert-norm difference of $\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$ and $\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$ defined in (22) and (23), respectively,

$$\begin{aligned}&\|\tilde{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \\ &\quad = \left\| \Big(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big] \Big) / \eta_t - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right\|_{\mathcal{H}}^2 \end{aligned} \tag{41}$$

Multiply and divide $\hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t)$, the last term, by η_t, and reorder terms to write

$$\begin{aligned}&\left\| \Big(f_t - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big] \Big) / \eta_t - \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \right\|_{\mathcal{H}}^2 \\ &\qquad = \left\| \frac{1}{\eta_t} \Big(f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big) - \frac{1}{\eta_t} \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[f_t - \eta_t \hat{\nabla}_f \ell(f_t(\mathbf{x}_t), y_t) \Big] \right\|_{\mathcal{H}}^2 \\ &\qquad = \frac{1}{\eta_t^2} \|\tilde{f}_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 \end{aligned} \tag{42}$$

where we have substituted the definition of $\tilde{f}_{t+1}$ and f_{t+1} in (17) and (15), respectively, and pulled the nonnegative scalar η_t outside the norm. Now, observe that the KOMP residual stopping criterion in Algorithm 1 is $\|\tilde{f}_{t+1} - f_{t+1}\|_{\mathcal{H}} \leq \epsilon_t$, which we may apply to the last term on the right-hand side of (42) to conclude (40).

■

Lemma 9 *(Stochastic Descent) Consider the sequence generated $\{f_t\}$ by Algorithm 2 with $f_0 = 0$. Under Assumptions 1-4, the following expected descent relation holds.*

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t[R(f_t) - R(f^*)] + 2\epsilon_t\|f_t - f^*\|_{\mathcal{H}} + \eta_t^2\sigma^2 \,. \quad (43)$$

Proof: Begin by considering the square of the Hilbert-norm difference between f_{t+1} and f^* defined by (1), and expand the square to write

$$\begin{aligned}\|f_{t+1} - f^*\|_{\mathcal{H}}^2 &= \|f_t - \eta_t\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \\ &= \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t\langle f_t - f^*, \tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\rangle_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2 \quad (44)\end{aligned}$$

Add and subtract the gradient of the regularized instantaneous risk $\hat{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)$ defined in (22) to the second term on the right-hand side of (44) to obtain

$$\begin{aligned}\|f_{t+1} - f^*\|_{\mathcal{H}}^2 = \|f_t - f^*\|_{\mathcal{H}}^2 &- 2\eta_t\langle f_t - f^*, \hat{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\rangle_{\mathcal{H}} \quad (45) \\ &- 2\eta_t\langle f_t - f^*, \tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t) - \hat{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\rangle_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2\end{aligned}$$

We deal with the third term on the right-hand side of (45), which represents the directional error associated with the sparse stochastic projections, by applying the Cauchy-Schwartz inequality together with Proposition 8 to obtain

$$\begin{aligned}\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \leq \|f_t - f^*\|_{\mathcal{H}}^2 &- 2\eta_t\langle f_t - f^*, \hat{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\rangle_{\mathcal{H}} \quad (46) \\ &+ 2\epsilon_t\|f_t - f^*\|_{\mathcal{H}} + \eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t), y_t)\|_{\mathcal{H}}^2\end{aligned}$$

Now compute the expectation of (46) conditional on the algorithm history $\mathcal{F}_t$

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t\langle f_t - f^*, \nabla_f R(f_t)\rangle_{\mathcal{H}} + 2\epsilon_t\|f_t - f^*\|_{\mathcal{H}} + \eta_t^2\sigma^2 \,, \quad (47)$$

where we have applied the fact that the stochastic functional gradient in (22) is an unbiased estimator [cf. (25)] for the functional gradient of the expected risk in (1), as well as the fact that the variance of the functional projected stochastic gradient is finite stated in (28) (Assumption 4). Observe that since $R(f)$ is an expectation of a convex function, it is also convex, which allows us to write

$$R(f_t) - R(f^*) \leq \langle f_t - f^*, \nabla_f R(f_t)\rangle_{\mathcal{H}} \,, \quad (48)$$

which we substitute into the second term on the right-hand side of the relation given in (47) to obtain

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t[R(f_t) - R(f^*)] + 2\epsilon_t\|f_t - f^*\|_{\mathcal{H}} + \eta_t^2\sigma^2 \,. \quad (49)$$

Thus the claim in Lemma 9 is valid. ■

Appendix B. Proof of Theorem 2

Apply the iterate bound stated in Proposition 7 to the third term on the right-hand side of (43) (Lemma 9) to write

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq \|f_t - f^*\|_{\mathcal{H}}^2 - 2\eta_t[R(f_t) - R(f^*)] + \eta_t^2\left(\frac{4CX}{\lambda} + \sigma^2\right) . \tag{50}$$

where we also have applied the approximation budget condition $\epsilon_t = \eta_t^2$. We use the relation in (50) to construct a martingale difference sequence. In particular, define the nonnegative stochastic processes α_t and β_t as

$$\alpha_t = \|f_t - f^*\|_{\mathcal{H}}^2 + \left(\frac{4CX}{\lambda} + \sigma^2\right)\sum_{u=t}^{\infty}\eta_u^2 , \qquad \beta_t = 2\eta_t[R(f_t) - R(f^*)] \tag{51}$$

Observe that α_t is finite almost surely, since $\sum_{u=t}^{\infty}\eta_u^2 \leq \sum_{u=0}^{\infty}\eta_u^2$. Given the definitions of α_t and β_t in (51), we may write

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq \alpha_t - \beta_t , \tag{52}$$

together with the fact that α_t and β_t are nonnegative, whereby the conditions of the Supermartingale Convergence Theorem (Solo and Kong, 1995) are satisfied. Therefore, we obtain that (i) α_t has a finite limit almost surely; and (ii) the series $\sum_{t=1}^{\infty}\beta_t < \infty$ is almost surely finite. The later result, taken together with the non-summability of the step-sizes stated in (29), implies that the almost surely the limit infimum of $R(f_t) - R(f^*)$ is null, i.e.

$$\liminf_{t\to\infty} R(f_t) - R(f^*) = 0 \qquad \text{a.s.} \tag{53}$$

Now, using the consequence of the Supermartingale Convergence Theorem, α_t almost surely has a limit. Observe that the sum $\sum_{u=t}^{\infty}\eta_u^2$ is a deterministic quantity whose limit is null (we sum over less and less terms over time, asymptotically summing over zero terms). Taken with the finiteness of the limit of α_t, we conclude

$$\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}}^2 = 0 \qquad \text{a.s.} \tag{54}$$

Appendix C. Proof of Theorem 3

Proof: The use of the regularizing term $(\lambda/2)\|f\|_{\mathcal{H}}^2$ in (1) implies that the regularized expected risk is λ-strongly convex with respect to $f \in \mathcal{H}$, which allows us to write

$$\frac{\lambda}{2}\|f_t - f^*\|_{\mathcal{H}}^2 \leq R(f_t) - R(f^*) \tag{55}$$

Substituting the relation (55) into the second term on the right-hand side of the expected descent relation stated in Lemma 9, with constant step-size $\eta_t = \eta$ and approximation budget $\epsilon_t = \epsilon$, yields

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq (1 - \eta\lambda)\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 . \tag{56}$$

We use the expression in (56) to construct a stopping stochastic process, which tracks the suboptimality of $\|f_t - f^*\|_{\mathcal{H}}^2$ until it reaches a specific threshold. In doing so, we obtain convergence to a neighborhood. We aim to define a stochastic process δ_t that qualifies as a supermartingale, i.e. $\mathbb{E}\left[\delta_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq \delta_t$. To do so, consider (56) and solve for the appropriate threshold by analyzing when the following holds true

$$\mathbb{E}\left[\|f_{t+1} - f^*\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t\right] \leq (1-\eta\lambda)\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq \|f_t - f^*\|_{\mathcal{H}}^2 \,. \quad (57)$$

Re-arrange the above expression to obtain the sufficient condition

$$-\eta\lambda\|f_t - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_t - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq 0 \,. \quad (58)$$

Observe that (58) defines a quadratic polynomial in $\|f_t - f^*\|_{\mathcal{H}}$, which, using the quadratic formula, has roots

$$\|f_t - f^*\|_{\mathcal{H}} = \frac{-2\epsilon \pm \sqrt{4\epsilon^2 - (-4\lambda\eta)(\eta^2\sigma^2)}}{-2\lambda\eta} = \frac{\epsilon \pm \sqrt{\epsilon^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} \quad (59)$$

The quadratic polynomnial defined by (58) opens downward, and $\|f_t - f^*\|_{\mathcal{H}} \geq 0$, so we focus on the positive root, substituting the approximation budget selection $\epsilon = K\eta^{3/2}$ to define the radius of convergence as

$$\Delta := \frac{\epsilon + \sqrt{\epsilon^2 + \lambda\eta^3\sigma^2}}{\lambda\eta} = \frac{\sqrt{\eta}}{\lambda}\Big(K + \sqrt{K^2 + \lambda\sigma^2}\Big) \quad (60)$$

The definition (60) allows us to construct a stopping process. In particular, define the stochastic process δ_t as

$$\delta_t = \|f_t - f^*\|_{\mathcal{H}} \mathbb{1}\Big\{ \min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta \Big\} \quad (61)$$

where $\mathbb{1}\{E\}$ denotes the indicator process of event $E \in \mathcal{F}_t$. Note that $\delta_t \geq 0$ for all t, since both $\|f_t - f^*\|_{\mathcal{H}}$ and the indicator function are nonnegative. Observe that, given the definition (61), either $\min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta$ holds, in which case we may compute the square root of the condition in (57) to write

$$\mathbb{E}[\delta_{t+1} \,|\, \mathcal{F}_t] \leq \delta_t \quad (62)$$

Alternatively, $\min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 \leq \Delta$, in which case the indicator function is null for all subsequent times, due to the use of the minimum inside the indicator in the definition of (61). Thus in either case, (62) holds, which implies that δ_t converges almost surely to null, which, as a consequence we obtain the fact that either $\lim_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} - \Delta = 0$ or the indicator function is null for large t, i.e. $\lim_{t\to\infty} \mathbb{1}\{\min_{u \leq t} -\eta\lambda\|f_u - f^*\|_{\mathcal{H}}^2 + 2\epsilon\|f_u - f^*\|_{\mathcal{H}} + \eta^2\sigma^2 > \Delta\} = 0$ almost surely. Therefore, we obtain that

$$\liminf_{t\to\infty} \|f_t - f^*\|_{\mathcal{H}} \leq \Delta = \frac{\sqrt{\eta}}{\lambda}\Big(K + \sqrt{K^2 + \lambda\sigma^2}\Big) \qquad \text{a.s.} \quad (63)$$

which is as stated in Theorem 3. ■

Appendix D. Proofs Leading to Theorem 4

Before proving Theorem D, we present a lemma which allows us to relate the stopping criterion of our sparsification procedure to a Hilbert subspace distance.

Lemma 10 *Define the distance of an arbitrary feature vector* $\mathbf{x}$ *evaluated by the feature transformation* $\phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot)$ *to* $\mathcal{H}_{\mathbf{D}} = span\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$*, the subspace of the Hilbert space spanned by a dictionary* $\mathbf{D}$ *of size* M*, as*

$$dist(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}}) = \min_{f \in \mathcal{H}_{\mathbf{D}}} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} \, . \tag{64}$$

This set distance simplifies to following least-squares projection when $\mathbf{D} \in \mathbb{R}^{p \times M}$ *is fixed*

$$dist(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}}) = \left\| \kappa(\mathbf{x}, \cdot) - [\mathbf{K}_{\mathbf{D},\mathbf{D}}^{-1} \boldsymbol{\kappa}_{\mathbf{D}}(\mathbf{x})]^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot) \right\|_{\mathcal{H}} . \tag{65}$$

Proof: The distance to the subspace $\mathcal{H}_{\mathbf{D}}$ is defined as

$$\text{dist}(\kappa(\mathbf{x}, \cdot), \mathcal{H}_{\mathbf{D}_t}) = \min_{f \in \mathcal{H}_{\mathbf{D}}} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} = \min_{\mathbf{v} \in \mathbb{R}^M} \|\kappa(\mathbf{x}, \cdot) - \mathbf{v}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}} \, , \tag{66}$$

where the first equality comes from the fact that the dictionary $\mathbf{D}$ is fixed, so $\mathbf{v} \in \mathbb{R}^M$ is the only free parameter. Now plug in the minimizing weight vector $\tilde{\mathbf{v}}^* = \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)$ into (66) which is obtained in an analogous manner to the logic which yields (19) - (20). Doing so simplifies (66) to the following

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) = \left\| \kappa(\mathbf{x}_t, \cdot) - [\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \right\|_{\mathcal{H}} . \tag{67}$$

■

D.1 Proof of Theorem 4

Proof: The proof proceeds by the following logic. We begin by considering the model order at two arbitrary subsequent iterates of Algorithm 2, and reduce model order growth at a given time to a criterion involving the approximation error γ_{M_t+1} associated with removing the most recent feature vector $\mathbf{x}_t$, and then analyze the conditions under which this simplified criterion is not satisfied for all subsequent times, meaning that the model order does not grow beyond a certain point. To do so, we prove that this quantity reduces to a weighted set distance to the Hilbert subspace $\mathcal{H}_{\mathbf{D}_t}$ defined by dictionary $\mathbf{D}_t$, and thus we are able to invoke point-set topological properties of the compact feature space $\mathcal{X}$, specifically, its packing number, which guarantee that the number of dictionary elements remains finite, in a manner similar to the proof of Theorem 3.1 in Engel et al. (2004).

Consider the model order of the function iterates f_t and f_{t+1} generated by Algorithm 2 denoted by M_t and M_{t+1}, respectively, at two arbitrary subsequent times t and $t+1$. Assume a constant algorithm step-size η has been chosen such that $\eta < 1/\lambda$ and the approximation budget ϵ satisfies $\epsilon = K\eta^{3/2}$ for some positive scalar $K > 0$. Suppose the model order of the function f_{t+1} is less than or equal to that of f_t, i.e. $M_{t+1} \leq M_t$. This relation holds when the stopping criterion of KOMP (Algorithm 1), stated as $\min_{j=1,\ldots,M_t+1} \gamma_j > \epsilon$, *is*

not satisfied for the kernel dictionary matrix with the newest sample point $\mathbf{x}_t$ appended: $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t; \mathbf{x}_t]$ [cf. (18)], which is of size $M_t + 1$. Thus, the negation of the termination condition of Algorithm 1 must hold for this case, stated as

$$\min_{j=1,\dots,M_t+1} \gamma_j \leq \epsilon \; . \tag{68}$$

Observe that the left-hand side of (68) lower bounds the approximation error γ_{M_t+1} of removing the most recent feature vector $\mathbf{x}_t$ due to the minimization over j, that is, $\min_{j=1,\dots,M_t+1} \gamma_j \leq \gamma_{M_t+1}$. Consequently, if $\gamma_{M_t+1} \leq \epsilon$, then (68) holds and the model order does not grow. Thus it suffices to consider γ_{M_t+1}.

The definition of γ_{M_t+1} with the substitution of $\tilde{f}_{t+1}$ in (17) allows us to write

$$\begin{aligned} \gamma_{M_t+1} &= \min_{\mathbf{u}\in\mathbb{R}^{M_t}} \Big\| (1-\eta\lambda) f_t - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k \kappa(\mathbf{d}_k, \cdot) \Big\|_{\mathcal{H}} \\ &= \min_{\mathbf{u}\in\mathbb{R}^{M_t}} \Big\| (1-\eta\lambda) \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} w_k \kappa(\mathbf{d}_k, \cdot) - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k \kappa(\mathbf{d}_k, \cdot) \Big\|_{\mathcal{H}} , \end{aligned} \tag{69}$$

where we denote the k^{th} column of $\mathbf{D}_t$ as $\mathbf{d}_k$. The minimal error is achieved by considering the square of the expression inside the minimization and expanding terms to obtain

$$\begin{aligned} &\Big\| (1-\eta\lambda) \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} w_k \kappa(\mathbf{d}_k, \cdot) - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) - \sum_{k\in\mathcal{I}\setminus\{M_t+1\}} u_k \kappa(\mathbf{d}_k, \cdot) \Big\|_{\mathcal{H}}^2 \\ &\quad = (1-\eta\lambda)^2 \mathbf{w}^T \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t} \mathbf{w} + \eta^2 \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)^2 \kappa(\mathbf{x}_t, \mathbf{x}_t) + \mathbf{u}^T \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t} \mathbf{u} \\ &\quad\quad -2(1-\eta\lambda)\eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)^2 \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t) + 2\eta^2 \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t) \mathbf{u}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t) - 2(1-\eta\lambda) \mathbf{w}^T \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t} \mathbf{u}. \end{aligned} \tag{70}$$

To obtain the minimum, we compute the stationary solution of (70) with respect to $\mathbf{u} \in \mathbb{R}^{M_t}$ and solve for the minimizing $\tilde{\mathbf{u}}^*$, which in a manner similar to the logic in (19) - (20), is given as

$$\tilde{\mathbf{u}}^* = (1-\eta\lambda)\mathbf{w} - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t) \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t) \; . \tag{71}$$

Plug $\tilde{\mathbf{u}}^*$ in (71) into the expression in (69) and using the short-hand notation $f_t(\cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$ and $\sum_k u_k \kappa(\mathbf{d}_k, \cdot) = \mathbf{u}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$. Doing so simplifies (69) to

$$\begin{aligned} &\Big\| (1-\eta\lambda) \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) - \mathbf{u}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \Big\|_{\mathcal{H}} \\ &\quad = \Big\| (1-\eta\lambda) \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) \\ &\quad\quad - [(1-\eta\lambda)\mathbf{w} - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t) \mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \Big\|_{\mathcal{H}}. \end{aligned} \tag{72}$$

The above expression may be simplified by cancelling like terms $(1-\eta\lambda)\mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)$ and pulling out a common factor of $\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|$ outside the norm as

$$\begin{aligned} &\Big\| - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)\kappa(\mathbf{x}_t, \cdot) - \eta \ell'(f_t(\mathbf{x}_t), \mathbf{y}_t) [\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \Big\|_{\mathcal{H}} \\ &\qquad = \eta |\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)| \Big\| \kappa(\mathbf{x}_t, \cdot) - [\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t}^{-1} \boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T \boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot) \Big\|_{\mathcal{H}} \; . \end{aligned} \tag{73}$$

Notice that the right-hand side of (73) may be identified as the distance to the subspace $\mathcal{H}_{\mathbf{D}_t}$ in (67) defined in Lemma 10 scaled by a factor of $\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|$. We may upper-bound the right-hand side of (73) as

$$\eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|\left\|\kappa(\mathbf{x}_t, \cdot) - [\mathbf{K}^{-1}_{\mathbf{D}_t,\mathbf{D}_t}\boldsymbol{\kappa}_{\mathbf{D}_t}(\mathbf{x}_t)]^T\boldsymbol{\kappa}_{\mathbf{D}_t}(\cdot)\right\|_{\mathcal{H}} \leq \eta|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) \tag{74}$$

where we have applied (65) regarding the definition of the subspace distance on the right-hand side of (74) to replace the Hilbert-norm term. Now, when the KOMP stopping criterion is violated, i.e., (68) holds, which implies $\gamma_{M_t+1} \leq \epsilon$. Therefore, the right-hand side of (74) is upper-bounded by ϵ, which we select as $\epsilon = K\eta^{3/2}$.

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) \leq \frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|}\,, \tag{75}$$

where we have divided both sides by $|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|$ and cancelled out a common factor of η. Observe that if (75) holds, then $\gamma_{M_{t+1}} \leq \epsilon$ holds, but since $\gamma_{M_{t+1}} \geq \min_j \gamma_j$, we may conclude that (68) is satisfied. Consequently the model order at the subsequent step does not grow $M_{t+1} \leq M_t$ whenever (75) is valid.

Now, let's take the contrapositive of the preceding expressions to observe that growth in the model order ($M_{t+1} = M_t + 1$) implies that the condition

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) > \frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|} \tag{76}$$

holds. Therefore, each time a new point is added to the model, the corresponding kernel function is guaranteed to be at least a distance of $\frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|}$ from every other kernel function in the current model.

By the C-Lipschitz continuity of the instantaneous loss (Assumption 2): specifically $1/|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)| \geq 1/C$, we can lower-bound the threshold condition in (76) as

$$\frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|} \geq \frac{K\sqrt{\eta}}{C} \tag{77}$$

Therefore, the KOMP stopping criterion is violated for the newest point whenever distinct dictionary points $\mathbf{d}_k$ and $\mathbf{d}_j$ for $j, k \in \{1, \ldots, M_t\}$, satisfy the condition $\|\phi(\mathbf{d}_j) - \phi(\mathbf{d}_k)\|_2 > \frac{K\sqrt{\eta}}{C}$. We shall now proceed in a manner similar to the proof of Theorem 3.1 in Engel et al. (2004). Since $\mathcal{X}$ is compact and κ is continuous, the range $\phi(\mathcal{X})$ (where $\phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot)$ for $\mathbf{x} \in \mathcal{X}$) of the kernel transformation of feature space $\mathcal{X}$ is compact. Therefore, the number of balls of radius δ (here, $\delta = \frac{K\sqrt{\eta}}{C}$) needed to cover the set $\phi(\mathcal{X})$ is finite (see, e.g., Anthony and Bartlett (2009)). Therefore, for some finite M^∞, if $M_t = M^\infty$, the left-hand side of (**??**) holds, which implies (68) is true for all t. We conclude that $M_t \leq M^\infty$ for all t. ■

References

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005.

Antoine Bordes, Léon Bottou, and Patrick Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

S. Boyd and L. Vanderberghe. *Convex Programming*. Wiley, New York, NY, 2004.

P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, 1966.

Emmanuel J Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

Aymeric Dieuleveut and Francis Bach. Non-parametric stochastic approximation with large step sizes. *arXiv preprint arXiv:1408.0361*, 2014.

Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, Aug 2004. ISSN 1053-587X. doi: 10.1109/TSP.2004.830985.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50, 2000.

W. J. Fu. Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

Simon Haykin. Neural networks: A comprehensive foundation. *Macmillan College Publishing Company*, 1994.

P. Honeine. Online kernel principal component analysis: A reduced-order model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1814–1826, 2012.

Thorsten Joachims and Chun-Nam John Yu. Sparse kernel svms via cutting-plane training. *Machine Learning*, 76(2-3):179–193, 2009.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

S Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7(Jul):1493–1515, 2006.

George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52:2165–2176, August 2004. doi: 10.1109/TSP.2004.830991.

A. Koppel, J. Fink, G. Warnell, E. Stump, and A. Ribeiro. Online learning for characterizing unknown environments in ground robotic vehicle models. In *Proc. Int. Conf. Intelligent Robots and Systems*, volume (to appear). Daejeon, Korea, Oct 9 - 14 2016.

Alec Koppel, Garrett Warnell, Ethan Stump, and A. Ribeiro. Parsimonious online kernel learning via sparse projections in function space. In *2017 IEEE Proc. Int. Conf. Acoust. Speech Signal Process.*, March 5-9 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Trung Le, Tu Nguyen, Vu Nguyen, and Dinh Phung. Dual space gradient descent for online learning. In *Advances in Neural Information Processing Systems*, pages 4583–4591, 2016a.

Trung Le, Vu Nguyen, Tu Dinh Nguyen, and Dinh Phung. Nonparametric budgeted stochastic gradient descent. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 654–662, 2016b.

Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. *Database*, 2009. URL `http://yann.lecun.com/exdb/mnist/`.

T. Leung and J. Malik. Representing and Recognizing the Visual Appearence of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, 1999.

Jun-Bao Li, Shu-Chuan Chu, and Jeng-Shyang Pan. *Kernel Learning Algorithms for Face Recognition*. Springer, 2014.

Weifeng Liu, Puskal P Pokharel, and Jose C Principe. The kernel least-mean-square algorithm. *Signal Processing, IEEE Transactions on*, 56(2):543–554, 2008.

Jing Lu, Steven CH Hoi, Jialei Wang, Peilin Zhao, and Zhi-Yong Liu. Large scale online kernel learning. *The Journal of Machine Learning Research*, 17(1):1613–1655, 2016.

Julien Mairal, Julien Mairal, Michael Elad, Michael Elad, Guillermo Sapiro, and Guillermo Sapiro. Sparse representation for color image restoration. In *the IEEE Trans. on Image Processing*, pages 53–69. ITIP, 2007.

Aryan Mokhtari and Alejandro Ribeiro. Res: Regularized stochastic bfgs algorithm. *Signal Processing, IEEE Transactions on*, 62(23):6089–6104, 2014.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research*, 16:3151–3181, 2015. URL `http://jmlr.org/papers/v16/mokhtari15a.html`.

Shayan Mukherjee and Shree K Nayar. Automatic generation of rbf networks using wavelets. *Pattern Recognition*, 29(8):1369–1383, 1996.

Klaus-Robert Müller, Tülay Adali, Kenji Fukumizu, Jose C. Principe, and Sergios Theodoridis. Special issue on advances in kernel-based learning for signal processing [from the guest editors]. *IEEE Signal Process. Mag.*, 30(4):14–15, 2013. doi: 10.1109/MSP.2013.2253031. URL `http://dx.doi.org/10.1109/MSP.2013.2253031`.

K. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

Deanna Needell, Joel Tropp, and Roman Vershynin. Greedy signal recovery review. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1048–1050. IEEE, 2008.

Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–773, 1980. ISSN 0025-5718.

Vladimir Norkin and Michiel Keyzer. On stochastic optimization and statistical learning in reproducing kernel hilbert spaces by support vector machines (svm). *Informatica*, 20(2): 273–292, 2009.

Y. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, 1993.

Massimiliano Pontil. A note on different covering numbers in learning theory. *Journal of Complexity*, 19(5):665–671, 2003.

Massimiliano Pontil, Yiming Ying, and Ding xuan Zhou. Error analysis for online gradient descent algorithms in reproducing kernel hilbert spaces. Technical report, University College London, 2005.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

Cédric Richard, José Carlos M Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57(3):1058–1067, 2009.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586.

Geoffrey Sampson, Robin Haigh, and Eric Atwell. Natural language analysis by stochastic optimization: A progress report on project april. *J. Exp. Theor. Artif. Intell.*, 1(4): 271–287, October 1990. ISSN 0952-813X. doi: 10.1080/09528138908953710. URL `http://dx.doi.org/10.1080/09528138908953710`.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. *Subseries of Lecture Notes in Computer Science Edited by JG Carbonell and J. Siekmann*, page 416, 2001.

Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 436–443, 2007.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11(Oct):2635–2670, 2010.

Konstantinos Slavakis, Pantelis Bouboulis, and Sergios Theodoridis. Online learning in reproducing kernel hilbert spaces. *Signal Processing Theory and Machine Learning*, pages 883–987, 2013.

Victor Solo and X. Kong. *Adaptive Signal Processing Algorithms: Stability and Performance*. Prentice-Hall information and system sciences series. Prentice Hall, 1995. ISBN 9780135012635. URL `https://books.google.com/books?id=3AkfAQAAIAAJ`.

Murat Taşan, Gabriel Musso, Tong Hao, Marc Vidal, Calum A MacRae, and Frederick P Roth. selecting causal genes from genome-wide association studies via functionally coherent subnetworks. *Nature methods*, 2014.

Ambuj Tewari and Peter L. Bartlett. Learning theory. In Paulo S.R. Diniz, Johan A.K. Suykens, Rama Chellappa, and Sergios Theodoridis, editors, *Academic Press Library in Signal Processing: Volume 1 Signal Processing Theory and Machine Learning*, volume 1 of *Academic Press Library in Signal Processing*, chapter 14, pages 775–816. Elsevier, 2014. URL `http://dx.doi.org/10.1016/B978-0-12-396502-8.00014-0`.

Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2015.

P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48(1):165–187, 2002.

Zhuang Wang and Slobodan Vucetic. Online passive-aggressive algorithms on a budget. In *International Conference on Artificial Intelligence and Statistics*, 2010a.

Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *The Journal of Machine Learning Research*, 13(1):3103–3131, 2012.

R. Wheeden, R.L. Wheeden, and A. Zygmund. *Measure and Integral: An Introduction to Real Analysis*. Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1977. ISBN 9780824764999. URL `https://books.google.com/books?id=YDkDmQ_hdmcC`.

Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

Y. Ying and D. X. Zhou. Online regularized classification algorithms. *IEEE Transactions on Information Theory*, 52(11):4775–4788, Nov 2006. ISSN 0018-9448. doi: 10.1109/TIT.2006.883632.

Kai Zhang, Ivor W Tsang, and James T Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.

Lijun Zhang, Jinfeng Yi, Rong Jin, Ming Lin, and Xiaofei He. Online kernel learning with a near optimal sparsity bound. In *ICML (3)*, pages 621–629, 2013.

Peilin Zhao and Steven Hoi. Bduol: Double updating online learning on a fixed budget. *Machine Learning and Knowledge Discovery in Databases*, pages 810–826, 2012.

Shilei Zhao, Peng Wu, and Yupeng Liu. An online kernel learning algorithm based on orthogonal matching pursuit. *Journal of Software*, 7(Sep):2076–2082, 2012.

Ding-Xuan Zhou. The covering number in learning theory. *Journal of Complexity*, 18(3): 739–767, 2002.

J. Zhu and T. Hastie. Kernel Logistic Regression and the Import Vector Machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205, 2005.

Zeyuan Allen Zhu, Weizhu Chen, Gang Wang, Chenguang Zhu, and Zheng Chen. P-packsvm: Parallel primal gradient descent kernel svm. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 677–686. IEEE, 2009.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.