Controlling the Bias-Variance Tradeoff via Coherent Risk for Robust Learning with Kernels

Alec Koppel[†], *Member, IEEE*, Amrit S. Bedi[⋆], *Student Member, IEEE,* and Ketan Rajawat[⋆], *Member, IEEE*

***Abstract*— In supervised learning, we learn a statistical model by minimizing a measure of fitness averaged over data. Doing so, however, ignores the variance, i.e., the gap between the optimal within a hypothesized function class and the Bayes Risk. We propose to account for both the bias and variance by modifying training to incorporate *coherent risk* which quantifies the uncertainty of a given decision. We develop the first online solution to this problem when estimators belong to a reproducing kernel Hilbert space (RKHS), which we call Compositional Online Learning with Kernels (COLK). COLK addresses the fact that (i) minimizing risk functions requires handling objectives which are compositions of expected value problems by generalizing the two time-scale stochastic quasi-gradient method to RKHSs; and (ii) the RKHS-induced parameterization has complexity which is proportional to the iteration index which is mitigated through greedily constructed subspace projections. We establish linear convergence in mean to a neighborhood with constant step-sizes, as well as the fact that its complexity is at-worst finite. Experiments on synthetic and benchmark data demonstrate that COLK exhibits consistent performance across training runs, estimates that are both low bias and low variance, and thus marks a step towards overcoming overfitting.**

I. Introduction

In supervised learning, we devise a statistical model that maps data points to decisions. This task underlies modern technologies such as speech [1] and visual recognition [2], autonomous control [3], and many others. Assuming a fixed data representation, this task may be mathematically formulated by hypothesizing that estimates take the form $f(\mathbf{x})$ and selecting f according to its ability to minimize a loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ averaged over data:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] \,, \tag{1}$$

where $f : \mathcal{X} \to \mathcal{Y}$ maps data points $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ to target variables $\mathbf{y} \in \mathcal{Y}$ (for classification $\mathcal{Y} = \{1, \dots, C\}$; for regression $\mathcal{Y} \subset \mathbb{R}^q$), and the loss $\ell(f(\mathbf{x}), \mathbf{y})$ is small when $f(\mathbf{x})$ and $\mathbf{y}$ are close. Moreover, $\mathcal{H}$ is a hypothesized function class to which the estimator f belongs. Define $L(f)$ as the average of $\ell(f(\mathbf{x}), \mathbf{y})$ over data in (1). This formulation, with data $(\mathbf{x}, \mathbf{y})$ as a pair of random vectors, is called the General Learning problem [4].

As is well known in statistical learning [5], solving (1) is only an approximation of the Bayes optimal estimator $\hat{\mathbf{y}}^\star = \operatorname{argmin}_{\hat{\mathbf{y}} \in \mathcal{Y}^\mathcal{X}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})]$ where $\mathcal{Y}^\mathcal{X}$ denotes the space of *all functions* $\hat{\mathbf{y}} : \mathcal{X} \to \mathcal{Y}$ that map data $\mathbf{x}$ to target variables $\mathbf{y}$. Suppose we try to minimize (1) and obtain estimate $\hat{f}$. Then the performance difference associated with $\hat{f}$ and the Bayes optimal $\hat{\mathbf{y}}^\star$ is

$$\underbrace{\mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{f}(\mathbf{x}), \mathbf{y})] - \min_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})]}_{\text{Bias}}$$
$$+ \underbrace{\min_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] - \min_{\hat{\mathbf{y}} \in \mathcal{Y}^\mathcal{X}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})]}_{\text{Variance}} \tag{2}$$

where we add and subtract the optimal supervised cost in (2) to obtain that this discrepancy decomposes into two terms: the estimation error, or bias – the first line; and approximation error, or variance[1] – the second line. Typically we make the bias small as the number of data points goes to infinity, and resign ourselves to the fact that the variance is an intrinsic penalty we suffer for a particular choice of $\mathcal{H}$.

Various methods have been proposed to mitigate the error variance while minimizing the bias, such as regularization [6] and risk analysis [7]. The former approach is designed only to deal with the discrepancy between the empirical and expected loss, but does not strike at the approximation error in (2). Alternatively, authors in operations research [8] propose to quantify the dispersion of an estimate with respect to its target variable through *coherent risk* [7] as a surrogate for the approximation error. An example is the semivariance:

$$\widetilde{\mathrm{Var}}[\ell(f(\mathbf{x}),\mathbf{y})] = \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big\{\big(\ell(f(\mathbf{x}),\mathbf{y}) - \mathbb{E}_{\mathbf{x}',\mathbf{y}'}[\ell(f(\mathbf{x}'),\mathbf{y}')]\big)_+^2\Big\}. \tag{3}$$

Alternatives include the p-th order semideviation or the conditional value-at-risk (CVaR) [9], which quantifies the loss function at tail-end quantiles of its distribution.

Suppose we fix a dispersion measure $\mathbb{D}[\ell(f(\mathbf{x}), \mathbf{y})]$. Then we can formulate a variant of supervised learning that accounts for approximation error [10]

$$f^\star = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] + \eta \mathbb{D}[\ell(f(\mathbf{x}), \mathbf{y})] \tag{4}$$

where η scales the emphasis on estimation or approximation error in (2). Solutions of (4), as compared with (1), may be better attuned to outliers and higher-order moments of the loss distribution. Thus, for classification, f may be equipped to address situations where training examples possess characteristics of multiple classes, whereas for regression (nonlinear

A. Koppel and A. S. Bedi are with U.S. Army Research Laboratory, Adelphi, MD, USA. (e-mail: alec.e.koppel.civ@mail.mil, amrit0714@gmail.com). K. Rajawat is with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (e-mail:ketan@iitk.ac.in).

[1]The approximation error is a more general quantity than the error variance, but under special cases, i.e., the quadratic loss, the former reduces into the later plus an intrinsic measure of the noise of the data distribution. We slightly conflate these quantities for the purpose of ease of explanation, but they are different.

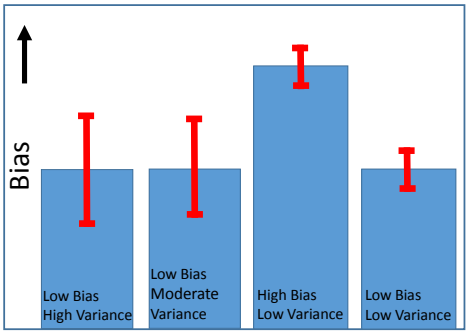


Fig. 1: In supervised learning, we seek a large enough function class $\mathcal{H}$ so as to ensure small approximation error (variance), but small enough that its estimation error (bias) is may be minimized. The Generalized linear models (GLMs) $\hat{\mathbf{y}} = \mathbf{w}^T\mathbf{x}$ ($\mathcal{H} = \mathbb{R}^p$) [15] yield minimal bias through solvable convex programs but large approximation error (first column in Fig.1). GLMs may be robustified through dispersion minimization as in (3) [16] (second column in Fig.1), but their intrinsic approximation error is still large. Neural networks [12] yield state of the art performance in terms of accuracy across many domains [1]–[3] (third column in Fig.1) by having an extremely general $\mathcal{H}$ so their approximation error is small. However, their bias is difficult to control due to the non-convexity defined by their training. On the other hand, selecting $\mathcal{H}$ as a reproducing kernel Hilbert space (RKHS) [13], [17] allows the learning of nonlinear statistical models while preserving convexity, and hence the risk coherence. Motivated by this fact, as well as recent efforts to "deepen" kernel methods that obtain comparable accuracy to neural networks [14], [18], we henceforth assume that f belongs to a RKHS. We develop an online memory-efficient training scheme for kernel methods which yields behavior akin to the fourth column.

filtering [11]), dips in signal to noise ratio may be more gracefully interpolated.

To solve problems of the form (4), one must first choose a function class $\mathcal{H}$ to which the estimator f belongs, and then select the one that minimizes both the supervised loss as well as its dispersion. The preferred choice for supervised learning is a neural network [12], but training neural networks is a non-convex problem, which makes controlling its estimation error challenging, and in the case of (4), makes the dispersion measure no longer a coherent risk [7]. On the other hand, selecting $\mathcal{H}$ as a reproducing kernel Hilbert space (RKHS) [13] allows the learning of nonlinear statistical models while preserving convexity, and hence the risk coherence. Motivated by this fact, as well as recent efforts to "deepen" kernel methods that obtain comparable accuracy to neural networks [14], we henceforth assume that f belongs to an RKHS.

With $\mathcal{H}$ specified, stochastic gradient descent (SGD) algorithms [19] and its improvements [20]–[22] cannot be used to minimize most risk functions such as (3) since they contain expectations of nonlinear functions of $L(f)$ [cf. (1)]. Specifically, (4) is really an instantiation of an optimization problem with *nested expectations* referred to as *compositional stochastic programming*. To conduct stochastic gradient descent on a compositional problem, one requires two samples from the inner random variable and one from the outer random variable to evaluate a single stochastic gradient ("the double sampling problem" [23]).

To ameliorate this issue, we develop a nonparametric extension of stochastic quasi-gradient (SQG) method [24], which uses two-time-scale stochastic approximation: one uses a quasi-stationary estimate of the inner expectation, whereas the other executes stochastic descent [25]. However, the choice of $\mathcal{H}$ as an RKHS, and sequential application of the Representer Theorem [26], means the function parameterization grows with the iteration index [13], and thus becomes untenable for streaming settings. In short, there is no affordable memory method to solve (4) over an RKHS. Thus, we:

- extend SQG to RKHS, which we compress using matching pursuit [27], [28] (Sec. III), which we call Compositional Online Learning with Kernels (COLK). We tie the compression to the step-size to ensure descent [29], [30].
- guarantee convergence to an optimal neighborhood whose radius depends on step-sizes and problem constants under constant learning rates and compression (Theorem **??**), which occurs at a linear rate (Theorem 1). Further, its worst-case complexity is finite (Theorem 2). That is, we achieve a near-optimal approximation to the NP-hard problem of selecting the optimally representative data subset.
- empirically (Sec. V), COLK yields estimators whose bias *and* variance is small, first on a synthetic regression problem with random outliers injected, and then on the `lidar` benchmark data. We observe that COLK yields *consistently* accurate performance across training runs, meaning that it does not overfit, in contrast to other methods, whose performance is high variance. In short, COLK experimentally behaves as the fourth column of Fig. 1.

II. Compositional Optimization with Kernels

In this work, we solve a generalization of the problem (4) stated in Section I with the understanding that it is a special case. We address problems where the objective is a composition of two functions, each of which is an expected value over a set of functions parameterized by a random pair. That is, random variables $\boldsymbol{\xi}_t \in \mathbb{R}^p$ and $\boldsymbol{\theta}_t \in \mathbb{R}^p$. In general both the random variables are allowed to be dependent, but for the ease of analysis and understanding, we assume that $\boldsymbol{\xi} \in \boldsymbol{\Xi} \subset \mathbb{R}^p$, $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^p$ and both $\boldsymbol{\xi}$ and $\boldsymbol{\theta}$ are independent of each other. Considering these random pairs, the cost takes the form $J(f) := (L \circ \mathbf{H})(f)$, where $\mathbf{H}(f) = \mathbb{E}_{\boldsymbol{\xi}}\left[\hbar_{\boldsymbol{\xi}}(f(\boldsymbol{\xi}))\right]$ is a map $\mathbf{H} : \mathcal{H} \to \mathbb{R}^m$ that is an expectation over a set of random functions $\hbar_{\boldsymbol{\xi}}(f(\boldsymbol{\xi}))$. Similarly, $L(\mathbf{u}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\ell_{\boldsymbol{\theta}}(\mathbf{u})\right]$ is a map $L : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ that is an expected value over a random collection variable. Further, $\mathcal{H}$ is a function space to be subsequently specified. In this work, we focus on the functional compositional stochastic program:

$$\min_{f \in \mathcal{H}} \mathbb{E}_{\boldsymbol{\theta}}\left[\ell_{\boldsymbol{\theta}}\left(\mathbb{E}_{\boldsymbol{\xi}}\left[\hbar_{\boldsymbol{\xi}}(f(\boldsymbol{\xi}))\right]\right)\right] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 , \tag{5}$$

where $J(f)$ is assumed to be convex in function f and add a Tikhonov regularizer $\frac{\lambda}{2}\|f\|_{\mathcal{H}}$ to ensure strong convexity [31], defining the regularized loss as $R(f) := J(f) + (\lambda/2)\|f\|_{\mathcal{H}}$. The feasible set $\mathcal{H}$ of (5), the domain of H, and hence

$\boldsymbol{J}$, is not Euclidean space $\mathbb{R}^p$, as in [24], but instead is a Hilbert space equipped with a unique *kernel function* $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ such that: (i) $\langle f, \kappa(\mathbf{u}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{u})$, and (ii) $\mathcal{H} = \overline{\text{span}\{\kappa(\mathbf{u}, \cdot)\}}$ for all $\mathbf{u} \in \mathcal{U}$. where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Hilbert inner product for $\mathcal{H}$ and $\mathcal{U} := \boldsymbol{\Xi} \cup \boldsymbol{\Theta}$ denotes the union of data domains $\boldsymbol{\Xi}$ and $\boldsymbol{\Theta}$, whose elements $\mathbf{u}$ are random variables $\boldsymbol{\xi}$ or $\boldsymbol{\theta}$. We further assume that the kernel is positive semidefinite, i.e. $\kappa(\mathbf{u}, \mathbf{u}') \geq 0$ for all $\mathbf{u}, \mathbf{u}' \in \mathcal{U}$ so that it is a Mercer kernel. This function space, as mentioned in Section I, is called reproducing kernel Hilbert spaces (RKHS) and its elements have bounded Hilbert norm [32].

For regularized empirical risk minimization (i.e., the sample average approximation of (5) for some fixed N realizations of ξ and θ), the Representer Theorem [32], owing to the strong convexity guaranteed by the regularizer, establishes that the optimal f in the RKHS $\mathcal{H}$ admits a basis representation in terms of kernel evaluations of *only* the training set:

$$f(\mathbf{u}) = \sum_{n=1}^{N} w_n \kappa(\boldsymbol{\xi}_n, \mathbf{u}) \,, \tag{6}$$

where $\mathbf{w} = [w_1, \cdots, w_N]^T \in \mathbb{R}^N$ denotes the weight vector. The upper summand index N in (6) is henceforth referred to as the model order. Examples include the polynomial and the radial basis , i.e., $\kappa(\mathbf{u}, \mathbf{u}') = \left(\mathbf{u}^T\mathbf{u}' + b\right)^c$ and $\kappa(\mathbf{u}, \mathbf{u}') = \exp\{-(\|\mathbf{u} - \mathbf{u}'\|_2^2)/2c^2\}$, respectively, where $\mathbf{u}, \mathbf{u}' \in \mathcal{U}$.

III. Compositional Online Learning with Kernels

Our goal is to solve (5) over the RKHS $\mathcal{H}$ in an online manner through sequentially revealed independent and identically distributed realizations of $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$, but no knowledge of the distribution from whence they come. Related ideas are developed for continuous Markov Decision Problems in [30] and vector-valued problems in [24]. The building blocks of the proposed algorithm Compositional Online Learning with Kernels (COLK) are a functional generalization of SQG (Section III-A) and sparse subspace projections that are greedily constructed using matching pursuit (Section III-B).

A. Stochastic Quasi-gradient Descent in Kernel Space

To understand why functional stochastic quasi-gradient is required, consider applying functional SGD to (5). We would require unbiased sampling of the stochastic gradient

$$\langle \nabla_f \hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \nabla \ell_{\boldsymbol{\theta}_t}(\mathbb{E}[\hbar_{\boldsymbol{\xi}}(f(\boldsymbol{\xi}))])\rangle + \lambda f. \tag{7}$$

However, the stochastic gradient (7) at training examples $(\boldsymbol{\xi}_t, \boldsymbol{\theta}_t)$ is not available due to the expectation involved in the argument of $\nabla \ell\,(\cdot)$, which precludes use of SGD for (5). A second realization of $\boldsymbol{\xi}$ is required to estimate the inner-expectation, which has been identified as the "double sampling problem" in reinforcement learning [23] and stochastic optimization [25], [33].

Thus, we propose using a two time-scale stochastic method called stochastic quasi-gradient (SQG) [34], [35]. SQG operates by defining a vector sequence $\mathbf{g}_t$ to track the instantaneous functions $\hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t))$ evaluated at sample pairs $\boldsymbol{\xi}_t$:

$$\mathbf{g}_{t+1} = (1 - \beta_k)\mathbf{g}_t + \beta_k \hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)) \tag{8}$$

with the intent of estimating the expectation $\mathbb{E}\left[\hbar_{\boldsymbol{\xi}}(f(\boldsymbol{\xi}))\right]$. In (8), β_t is a scalar learning rate chosen from the unit interval $(0, 1)$ which may be either diminishing or constant. Then, we define a function sequence $f_t \in \mathcal{H}$ initialized as null $f_0 = 0$, that we sequentially update using stochastic quasi-gradient descent:

$$f_{t+1} = (1 - \lambda\alpha_t) f_t - \alpha_t \langle \nabla_f \hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \nabla \ell_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle \,, \tag{9}$$

where α_t is a step-size parameter chosen as diminishing or constant. Further note that the term $\langle \nabla_f \hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t))$ is a function in $\mathcal{H}$, and thus infinite dimensional. However, by applying the chain rule and the reproducing property of the kernel ("the kernel trick") stated in (i), we obtain

$$\begin{aligned}
&\langle \nabla_f \hbar_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \nabla \ell_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle \\
&\quad = \sum_{i=1}^{m} \nabla_f \hbar^i_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)) \frac{\partial \ell_{\boldsymbol{\theta}_t}(\mathbf{u})}{\partial u_i}\Big|_{\mathbf{u}=\mathbf{g}_{t+1}} \\
&\quad = \sum_{i=1}^{m} \frac{\partial \hbar^i_{\boldsymbol{\xi}_t}(\omega)}{\partial \omega}\Big|_{\omega=f(\boldsymbol{\xi}_t)} \times \frac{\partial \ell_{\boldsymbol{\theta}_t}(\mathbf{u})}{\partial u_i}\Big|_{\mathbf{u}=\mathbf{g}_{t+1}} \kappa(\boldsymbol{\xi}_t, \cdot) \\
&\quad = \langle \hbar'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle \kappa(\boldsymbol{\xi}_t, \cdot)
\end{aligned} \tag{10}$$

In the last line of (10), we have used the vector inner product notation to denote the summation on the previous lines. Note that the kernel function $\kappa(\boldsymbol{\xi}_t, \cdot)$ is common and therefore outside the inner product in (10). Utilizing this notation, the function update equation of (9) may be written as

$$f_{t+1} = (1 - \lambda\alpha_t) f_t - \alpha_t \langle \hbar'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle \kappa(\boldsymbol{\xi}_t, \cdot) \,. \tag{11}$$

Observe that in (11), SQG iteration in RKHS yields the parametric updates on the coefficient vector $\mathbf{w}$ and kernel dictionary $\mathbf{U}$

$$\begin{aligned}
\mathbf{U}_{t+1} &= \left[\mathbf{U}_t, \boldsymbol{\xi}_t\right] \\
\mathbf{w}_{t+1} &= \left[(1 - \alpha_t\lambda)\mathbf{w}_t, -\alpha_t \langle \hbar'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle\right]
\end{aligned} \tag{12}$$

with detailed derivation provided in Appendix **??**. In (12), observe the *kernel dictionary* parameterizing function f_t is a matrix $\mathbf{U}_t \in \mathbb{R}^{p\times(t-1)}$ which stacks past realizations of random variables $\boldsymbol{\xi}$, and the coefficient vector $\mathbf{w}_t \in \mathbb{R}^{t-1}$ as the associated scalars in the kernel expansion (**??**) which are updated according to (12). The function update of (11) implies that the complexity of f_t is $\mathcal{O}(t)$, due to the fact that the number of columns in $\mathbf{U}_t$, or *model order* M_t, is $(t - 1)$, and thus is unsuitable for settings where the total number of data samples is not finite, or are arriving sequentially and repeatedly. This is an inherent challenge of extending [24] to learning nonlinear statistical models that belong to RKHS. To address this, we project (11) onto low-dimensional subspaces, inspired by [29], which we detail in the following subsection.

Algorithm 1 Compositional Online Learning with Kernels

Require: $\{\theta_t, \boldsymbol{\xi}_t, \alpha_t, \beta_t, \epsilon_t\}_{t=0,1,2,\ldots}$
  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**
    Update auxiliary variable $\mathbf{g}_{t+1}$ according to (8)
      $\mathbf{g}_{t+1} = (1-\beta_k)\mathbf{g}_t + \beta_k \hbar_{\xi_t}(f(\xi_t))$
    Compute functional stochastic quasi-gradient step (11)
      $\tilde{f}_{t+1} = (1-\lambda\alpha_t) f_t - \alpha_t \langle \nabla_f \hbar_{\xi_t}(f(\xi_t)), \nabla \ell_{\theta_t}(\mathbf{g}_{t+1}) \rangle$
    Revise function parameters: dictionary & weights (12)
      $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \xi_t, \theta_t]$
      $\tilde{\mathbf{w}}_{t+1} = [(1-\alpha_t\lambda)\mathbf{w}_t, \ -\alpha_t \langle \nabla_f \hbar_{\xi_t}(f(\xi_t)), \nabla \ell_{\theta_t}(\mathbf{g}_{t+1}) \rangle]$
    Compress parameterization via KOMP (Algorithm **??**) $(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \mathbf{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon_t)$
  end for

B. Greedy Subspace Projections

To control the unbounded growth of model order $M_t = (t-1)$ with t, we adopt the idea of bias-inducing projections onto low-dimensional subspaces as in [29]. Specifically, we construct a function sequence by orthogonally projecting functional SQG iterates onto subspaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ that consist only of those that can be represented using some dictionary $\mathbf{D} = [\mathbf{d}_1, \ \ldots, \ \mathbf{d}_M] \in \mathbb{R}^{p \times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \{f : f(\cdot) = \sum_{n=1}^{M} w_n \kappa(\mathbf{d}_n, \cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_n, \cdot)\}_{n=1}^{M}$. Here we define $\mathbf{d}_n \in \mathbb{R}^p$ as a model point which stacks exemplar realizations of $\boldsymbol{\xi}$, i.e., $\mathbf{d}_n = \boldsymbol{\xi}_n$. Further define $\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = [\kappa(\mathbf{d}_1, \cdot) \ldots \kappa(\mathbf{d}_M, \cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We enforce parsimony in function representation by selecting dictionaries $\mathbf{D}$ such that $M_t \ll t$. Specifically, we replace the update (9) by a projected variant onto a subspace spanned by dictionary $\mathbf{D}_t$:

$$f_{t+1} := \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \left[(1-\lambda\alpha_t) f_t - \alpha_t \langle \hbar'_{\xi_t}(f(\xi_t)), \ell'_{\theta_t}(\mathbf{g}_{t+1}) \rangle \right]. \quad (13)$$

The function (13) can be expressed in terms of parameter space of coefficients $\mathbf{w}_{t+1}$ and dictionary updated step $\mathbf{D}_{t+1}$ as detailed in Appendix **??** and Appendix **??** respectively. The function $\tilde{f}_{t+1}$ defined by SQG without projection is parameterized by dictionary $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t; \boldsymbol{\xi}_t]$, 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 best approximate $\tilde{f}_{t+1}$ in terms of Hilbert norm error. We use *kernel orthogonal matching pursuit* (KOMP) [28] with allowed error tolerance ϵ_t to find a kernel dictionary matrix $\mathbf{D}_{t+1}$ based on the one which adds the latest samples $\tilde{\mathbf{D}}_{t+1}$. The variant of KOMP we propose using, called Destructive KOMP with Pre-Fitting (see [28], Section 2.3), is summarized in Algorithm **??** detailed in Appendix **??**. The method, called Compositional Online Learning with Kernels (COLK), is summarized in Algorithm 1. Note that if $\epsilon_t = 0$, no compression happens, and the model order is $M_t = (t-1)$ as defined by (12). For constant step size result, we use $\epsilon_t = \epsilon$ for all t.

IV. Balancing Optimality and Model Complexity

This section establishes the convergence of Algorithm 1 to the minimizer of (5) which is a generalization of (4) under constant step size. To do so, some technicalities are required to be subsequently introduced. Then, we are able to establish exact convergence under attenuating learning rates and compression budget (see Theorem 1 in [37]), which we build upon to establish the following results that adeptly balance optimality with parsimony.

First, denote the filtration $\mathcal{F}_t$ as a time dependent sigma algebra defined as $\mathcal{F}_t \supset (\{f_u, g_u\}_{u=0}^{t} \cup \{\theta_s, \boldsymbol{\xi}_s\}_{s=0}^{t-1})$. This sigma algebra is used for performing the conditional expectation at time time instant t in the analysis. The assumptions required for the analysis of this work are stated next.

AS 1. *At each time instant t, the second moment of the derivative of inner function $\hbar'_{\xi_t}(f(\xi_t))$ and outer function $\ell'_{\theta_t}(\mathbf{g}_{t+1})$ is bounded as*

$$\mathbb{E}\left[|\hbar'_{\xi_t}(f(\xi_t))|^2 \,\middle|\, \theta_t \right] \leq G_{\hbar} \quad \text{and} \quad \mathbb{E}\left\{ |\ell'_{\theta_t}(\mathbf{g}_{t+1})|^2 \,\middle|\, \mathcal{F}_t \right\} \leq G_{\ell}, \quad (14)$$

where $G_{\hbar}$ and G_{ℓ} are finite constants. Moreover, the second moment of the projected stochastic quasi-gradient of the objective function (cf. (5)*) is bounded as*

$$\mathbb{E}\left[\|\tilde{\nabla}_f R(f_t, \mathbf{g}_{t+1}; \boldsymbol{\xi}_t, \theta_t)\|_{\mathcal{H}}^2 \,\middle|\, \mathcal{F}_t \right] \leq \sigma_f^2 . \quad (15)$$

In addition, define $\boldsymbol{\delta}_t := \hbar_{\xi_t}(f_t; \boldsymbol{\xi}_t)$ with $\bar{\boldsymbol{\delta}}_t = \mathbb{E}\left[\boldsymbol{\delta}_t \,\middle|\, \theta_t \right]$. Then $\boldsymbol{\delta}_t$ has finite variance as

$$\mathbb{E}[\|\boldsymbol{\delta}_t - \bar{\boldsymbol{\delta}}_t\|^2 \,\middle|\, \mathcal{F}_t] \leq \sigma_{\delta}^2 . \quad (16)$$

AS 2. *The instantaneous derivative of the outer function $\ell_{\theta}(\cdot)$ is Lipschitz continuous with respect to its first scalar argument so that we may write*

$$|\ell'_{\theta}(\mathbf{u}) - \ell'_{\theta}(\mathbf{v})| \leq L_{\ell} \|\mathbf{u} - \mathbf{v}\| . \quad (17)$$

AS3. *The expected value of the inner function is Lipschitz with respect to its argument:*

$$|\mathbb{E}_{\xi}[\hbar_{\xi}(f)] - \mathbb{E}_{\xi}[\hbar_{\xi}(f')]| \leq L_{\hbar} \|f - f'\|_{\mathcal{H}} . \quad (18)$$

Assumption 1 follows from the compactness of the feature space $\boldsymbol{\Xi} \cup \boldsymbol{\Theta}$. Assumption 1 is regarding the second moments of the derivatives which limits the variance of the stochastic approximation error. This assumption is typical in the literature [36] and usually holds in practice. Assumption 2 and 3 regarding the Lipschitz continuity of the outer and inner function holds for most applications, and holds for most differentiable functions. These assumptions hold for the experiments in the following section.

Theorem 1. *Consider use of a constant step size $\alpha_t = \alpha$ and $\beta_t = \beta$ and compression budget $\epsilon_t = \epsilon = C\alpha^2$, and regularizer $\lambda = L_{\ell} U^2 G_{\hbar}^2 \alpha/\beta + \lambda_0$ such that $\lambda_0 < 1$. The mean sub-optimality $\mathbb{E}\left[\|f_t - f^{\star}\|_{\mathcal{H}}^2 \right]$ of the functions generated by Algorithm 1 converges linearly to an error bound, i.e.*

$$\mathbb{E}\left[\|f_t - f^{\star}\|_{\mathcal{H}}^2 \right] \leq \left(1-\lambda_0\right)^t \mathbb{E}\left[\|f_0 - f^{\star}\|_{\mathcal{H}}^2 \right] + \mathcal{O}(\alpha) . \quad (19)$$

The rate result in (19) (proof in [37]) describes the non-asymptotic behavior of $\mathbb{E}\left[\|f_t - f^{\star}\|_{\mathcal{H}}^2 \right]$, and is comparable

to the learning guarantees of classical stochastic optimization methods. For a given λ_0, Theorem 1 states that f_t converges linearly towards the $\mathcal{O}(\alpha)$ ball around the $f^\star$ in RKHS. This result is important since it explicitly characterizes the performance of the algorithm after t steps. However, if we assume a bound on the initialization term $\|f_0 - f^\star\|_{\mathcal{H}} \leq B$, it is possible to find how may steps t are required to enter a $\mathcal{O}(\alpha)$ ball around the optimal $f^\star$. An additional merit of using constant step-sizes is the model order of the COLK function sequence is at-worst finite, and defined by the metric entropy (covering number) of the feature spaces:

Theorem 2. *Consider use of constant step-sizes $\alpha_t = \alpha$, $\beta_t = \beta$ and constant compression budget $\epsilon = C\alpha^2$, with regularization $\lambda = L_\ell U^2 G_{\hbar}^2 \alpha/\beta + \lambda_0 = \mathcal{O}(\alpha/\beta + 1)$. The model order M_t of function f_t generated by Algorithm 1, i.e. number of columns in its current dictionary D_t, is bounded: $M_t \leq M^\infty < \infty$ for all $t \geq 0$, and the limiting function $f^\infty = \lim_{t\to\infty} f_t$ has finite model order.*

To establish Theorem 2 (proof in [37]), we require an additional assumption on the boundedness of the scalar component of the gradients of the inner and outer functions as defined in Assumption 5.

AS4. *(Gradient boundedness) The instantaneous gradient of both the inner function $\ell'_{\theta}(\mathbf{u})$ and outer function $\hbar'_{\xi}(f(\xi))$ are bounded as*

$$|\ell'_{\theta}(\mathbf{u})| \leq C_\ell \quad \text{and} \quad |\hbar'_{\xi}(f(\xi))| \leq C_{\hbar}. \tag{20}$$

The finite model order claim in Theorem 2 is significant as it ensures that the size of the kernel dictionary $\mathbf{D}_t$ is finite and does not grow unbounded with time. Note further that the result in Theorem 2 sets the present work apart from similar online learning frameworks [13], [38] where the memory size or budget is often selected heuristically. In contrast, given ϵ, the model order M_t obtained from running the algorithm is defined by the problem context and does not grow arbitrarily.

V. Experiments

To show the efficacy of the proposed algorithm, we consider a problem of nonlinear regression (filtering) over a p-dimensional parameter space. We have again have two sets of random variables $(\mathbf{x}, \mathbf{x}') \in \mathcal{X} \subset \mathbb{R}^p$ but now the target variables are real valued $y, y' \in \mathbb{R}$. The merit criterion of model fitness for a given training example $(\mathbf{x}_n, y_n)$ is the humble square loss:

$$\ell(f(\mathbf{x}_n), y_n) = (f(\mathbf{x}_n) - y_n)^2 \tag{21}$$

However, due to the bias-variance tradeoff in Section I, we do not want to only minimize the expectation of (21) plus a regularizer $\lambda\|f\|^2_{\mathcal{H}}$ over all data $(\mathbf{x}, y)$, but also some surrogate [8] for the approximation error over data $(\mathbf{x}', y')$. Due to the fact that many probability distributions may be completely characterized by their moments [39][Chapter 3], a reasonable choice for the risk is to choose the dispersion measure as all p-th order central moments,

$$\mathbb{D}[\ell(f(\mathbf{x}), \mathbf{y})] = \sum_{p=2}^{P} \mathbb{E}_{\mathbf{x},\mathbf{y}}\Big\{\big(\ell(f(\mathbf{x}),\mathbf{y}) - \mathbb{E}_{\mathbf{x}',\mathbf{y}'}[\ell(f(\mathbf{x}'),\mathbf{y}')]\big)^p\Big\} \tag{22}$$

which are just deviations raised to the p-th power [10]. However, since the computational overhead scales with P, we truncate the upper summand index in (22) to $P = 4$. It is remarked that the dispersion measure in (22) is non-convex which is used for the experimental purposes which corresponds to the variance, skewness, and kurtosis of the loss distribution. Note that (22) may be convexified through a positive projection of $(\ell(f(\mathbf{x}),\mathbf{y}) - \mathbb{E}_{\mathbf{x}',\mathbf{y}'}[\ell(f(\mathbf{x}'),\mathbf{y}')])$, in which case the standard deviation becomes a semi-deviation, as do its higher-order analogues. However, for simplicity, we omit the positive projection in experiments. Next, we apply the proposed algorithm to solve the nonlinear regression problem which results in the following updates.

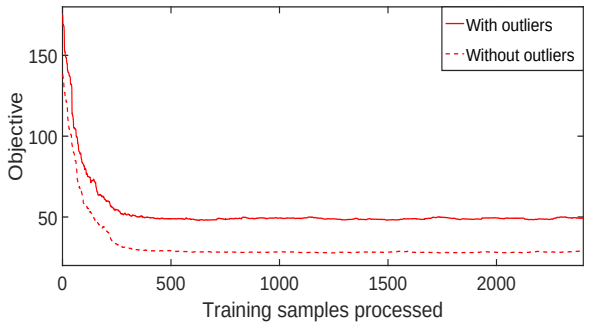
$$\begin{aligned} g_{t+1} &= (1-\beta_t)g_t + \beta_t(f_t(\mathbf{x}'_t) - y'_t)^2 \\ \tilde{f}_{t+1} &= (1-\lambda\alpha_t)f_t - \alpha\Big\{2(f_t(\mathbf{x}_t) - y_t)\kappa(\mathbf{x}_t, \cdot) \\ &\quad + \eta\sum_{p=2}^{4} m(p, g_{t+1})[2(f_t(\mathbf{x}_t) - y_t)\kappa(\mathbf{x}_t,\cdot) - 2(f_t(\mathbf{x}'_t) - y'_t)\kappa(\mathbf{x}'_t,\cdot)]\Big\} \end{aligned} \tag{23}$$

where $m(p, g_{t+1}) := \big[p((f_t(\mathbf{x}_t) - y_t)^2 - g_{t+1})^{p-1}\big]$.

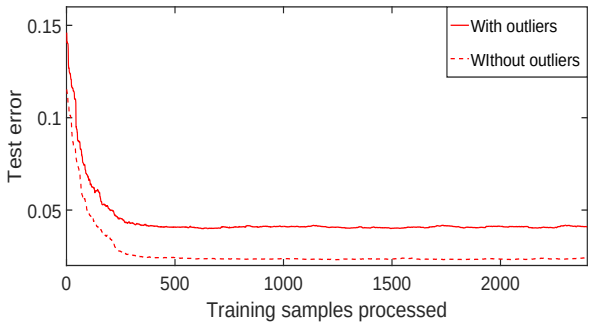
We evaluate COLK on synthetic and real data sets whose distributions are skewed or heavy-tailed, and compare its test accuracy against existing benchmarks that minimize only bias. Firstly, we evaluate performance on synthetic data `regression outliers` which has a heavier tailed distribution, i.e., more outliers are present. We inquire as to which methods overfit versus learn successfully: COLK (Algorithm 1), or methods such as BSGD [40], NPBSG [38], POLK [29].

We generate 20 different sets from the same data distribution and then run both POLK and COLK to learn a regression function. To generate the synthetic dataset `regression outliers`, we used the function $y = 2 * x + 3 * \sin(6x)$ as the original function and target y's observed after adding a zero mean Gaussian noise to $2 * x + 3 * \sin(6x)$. First we generate 60000 samples of the data, and then select 20% as the test data set. From the remaining 4800 samples, we select 50% at random to generate 20 different training sets. We run COLK over these training set with the following parameter selections: a Gaussian kernel with bandwidth $\sigma = .06$, step-size parameters $\alpha = 0.02$, $\beta = 0.01$, $\epsilon = K\alpha^2$ with parsimony constant $K = 5$, variance coefficient $\eta = 0.1$, and mini-batch size of 1. Similarity, for POLK we use $\alpha = 0.5$ and $\epsilon = K\alpha^2$ with parsimony constant $K = 0.09$. We fix the kernel type and bandwidth across the different methods, and the parameters that define comparator algorithms are hand-tuned to optimize performance with the restriction that their model complexity is comparable to each other. We run these algorithms for different realizations of training data and evaluate their test accuracy as well as its standard deviation.

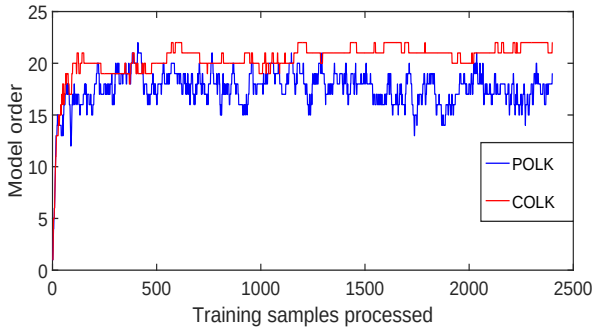
Before summarizing these results, we present an example sample path of Algorithm 1 for this experiment for the



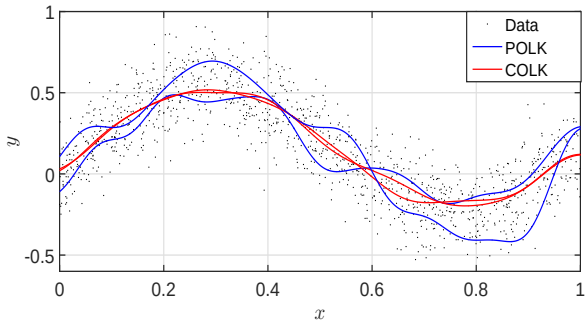
(a) Objective function



(b) Test accuracy



(c) Model order



(d) Visualization of regression function plotted for two training sets

Fig. 2: COLK experimental behavior on a regression on a synthetic data set, without and with training outliers. The presence of outliers does not break the learning stability, and test accuracy remains comparable, at the cost of increased complexity. Here COLK minimizes bias, variance, and third and fourth-order deviations.

`regression outliers` data, with and without outliers, in Fig 2. Specifically, Fig. 2a shows that the mean plus variance of the loss function is minimized as the number of samples processed increases. The time-series of the test-set error of COLK is given in Fig. 2b which converges as the training samples increases. In Fig. 2c we plot the model order of the function sequence defined by COLK, and observe it stabilizes over time regardless of the presence of outliers. These preliminary results validate the convergence results established in Section IV. The advantage of minimizing the bias as well as variance is depicted in Fig. 2d which plots the learned function for POLK and COLK for two training data sets. It can be observed that how POLK learning varies from one training set to other while COLK is robust to this change.

We now discuss our experimental results in terms of how COLK compares to existing techniques that only fit to the mean loss. We run COLK as well as the others for 20 total training runs and report the average test error and the standard deviation in the box and whisker plot given in Fig. 3. The box represents average test error and whisker represents the standard deviation of the corresponding estimate. Observe that COLK yields the lowest error as well as the lowest standard deviation, meaning it yields inferences that are both low bias *and* low variance. To check the proposed algorithm for real data, we consider the performance of filtering laser scans to interpolate range to a target via the `lidar` data [41] (with added outliers) with the results shown in Fig. 5. It is clear from the figure that the proposed algorithm is robust to outliers in the data.

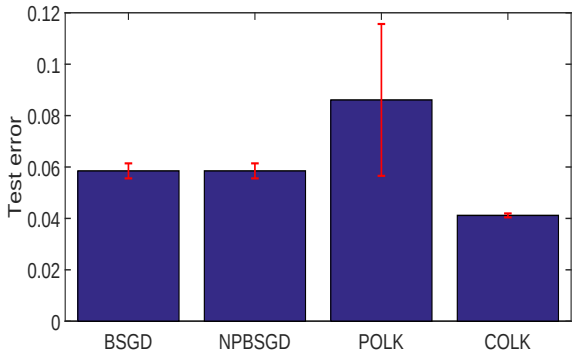


Fig. 3: Statistical Accuracy Comparison

Fig. 4: COLK, with $\alpha = 0.02$, $\epsilon = \alpha^2$, $\beta = 0.01$, $K = 5$, $\eta = 0.1$, bandwidth $c = .06$ as compared to other methods for online learning with kernels that only minimize bias on the `regression outliers` data. This figure reports test error averages over 20 training runs, and we report the standard deviation of test error as error bars. COLK yields both a minimal error rate and variability.

References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

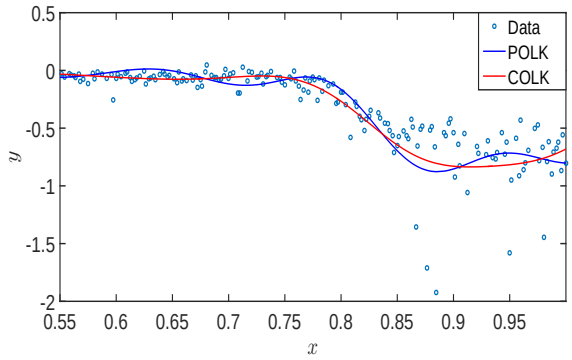


Fig. 5: Visualization of regression function plotted for LIDAR dataset

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[4] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[5] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[6] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[7] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.

[8] A. Ruszczyński and A. Shapiro, "Optimization of risk measures," in *Probabilistic and randomized methods for design under uncertainty*. Springer, 2006, pp. 119–157.

[9] S. Uryasev, "Conditional value-at-risk: Optimization algorithms and applications," in *Computational Intelligence for Financial Engineering, 2000.(CIFEr) Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on*. IEEE, 2000, pp. 49–57.

[10] S. Ahmed, "Convexity and decomposition of mean-risk stochastic programs," *Mathematical Programming*, vol. 106, no. 3, pp. 433–446, 2006.

[11] H. Tanizaki, *Nonlinear filters: estimation and applications*. Springer Science & Business Media, 2013.

[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[13] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2165–2176, August 2004.

[14] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 2627–2635.

[15] J. A. Nelder and R. J. Baker, "Generalized linear models," *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972.

[16] P. Ravikumar, H. Liu, J. Lafferty, and L. Wasserman, "Spam: Sparse additive models," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2007, pp. 1201–1208.

[17] A. Dieuleveut and F. Bach, "Non-parametric stochastic approximation with large step sizes," *arXiv preprint arXiv:1408.0361*, 2014.

[18] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1399–1407.

[19] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 09 1951.

[20] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, 2013, pp. 315–323.

[21] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 436–443.

[22] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Accelerating stochastic gradient descent," *arXiv preprint arXiv:1704.08227*, 2017.

[23] R. S. Sutton, H. R. Maei, and C. Szepesvári, "A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in neural information processing systems*, 2009, pp. 1609–1616.

[24] M. Wang, E. X. Fang, and H. Liu, "Stochastic compositional gradient descent: Algorithms for minimizing compositions of expected-value functions," *Mathematical Programming*, vol. 161, no. 1-2, pp. 419–449, 2017.

[25] V. R. Konda and J. N. Tsitsiklis, "Convergence rate of linear two-time-scale stochastic approximation," *Annals of applied probability*, pp. 796–819, 2004.

[26] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," *Subseries of Lecture Notes in Computer Science Edited by JG Carbonell and J. Siekmann*, p. 416, 2001.

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

[28] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Machine Learning*, vol. 48, no. 1, pp. 165–187, 2002.

[29] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," *arXiv preprint arXiv:1612.04111*, 2016.

[30] A. Koppel, G. Warnell, E. Stump, P. Stone, and A. Ribeiro, "Breaking bellman's curse of dimensionality: Efficient kernel gradient temporal difference," *arXiv preprint arXiv:1709.04221 (Submitted to IEEE TAC, Dec. 2017)*, 2017.

[31] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Advances in computational mathematics*, vol. 13, no. 1, pp. 1–50, 2000.

[32] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[33] V. S. Borkar, "Stochastic approximation with two time scales," *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

[34] A. Korostelev, "Stochastic recurrent procedures: Local properties," *Nauka: Moscow (in Russian)*, 1984.

[35] Y. Ermoliev, "Stochastic quasigradient methods and their application to system optimization," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 9, no. 1-2, pp. 1–36, 1983.

[36] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[37] A. S. Bedi, A. Koppel, and K. Rajawat, "Nonparametric compositional stochastic optimization," *arXiv preprint arXiv:1902.06011*, 2019.

[38] T. Le, V. Nguyen, T. D. Nguyen, and D. Phung, "Nonparametric budgeted stochastic gradient descent," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 654–662.

[39] R. Durrett, *Probability: theory and examples*. Cambridge university press, 2010.

[40] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3103–3131, 2012.

[41] D. Ruppert, M. P. Wand, and R. J. Carroll, "Semiparametric regression during 2003–2007," *Electronic journal of statistics*, vol. 3, p. 1193, 2009.

[42] D. Needell, J. Tropp, and R. Vershynin, "Greedy signal recovery review," in *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*. IEEE, 2008, pp. 1048–1050.

[43] M. Wang and D. P. Bertsekas, "Incremental constraint projection-proximal methods for nonsmooth convex optimization," *SIAM Journal on Optimization (to appear)*, 2014.

[44] R. Wheeden, R. Wheeden, and A. Zygmund, *Measure and Integral: An Introduction to Real Analysis*, ser. Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1977. [Online]. Available: https://books.google.com/books?id=YDkDmQ_hdmcC

[45] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[46] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.