Decentralized Online Learning With Kernels

Alec Koppel, Santiago Paternain, Cédric Richard, Alejandro Ribeiro

Decentralized Online Learning with Kernels

Alec Koppel[§], Santiago Paternain[⋆], Cédric Richard[†] and Alejandro Ribeiro[⋆]

***Abstract*—We consider multi-agent stochastic optimization problems over reproducing kernel Hilbert spaces (RKHS). In this setting, a network of interconnected agents aims to learn decision functions, i.e., nonlinear statistical models, that are optimal in terms of a global convex functional that aggregates data across the network, with only access to locally and sequentially observed samples. We propose solving this problem by allowing each agent to learn a local regression function while enforcing consensus constraints. We use a penalized variant of functional stochastic gradient descent operating simultaneously with low-dimensional subspace projections. These subspaces are constructed greedily by applying orthogonal matching pursuit to the sequence of kernel dictionaries and weights. By tuning the projection-induced bias, we propose an algorithm that allows for each individual agent to learn, based upon its locally observed data stream and message passing with its neighbors only, a regression function that is close to the globally optimal regression function. That is, we establish that with constant step-size selections agents' functions converge to a neighborhood of the globally optimal one while satisfying the consensus constraints as the penalty parameter is increased. Moreover, the complexity of the learned regression functions is guaranteed to remain finite. On both multi-class kernel logistic regression and multi-class kernel support vector classification with data generated from class-dependent Gaussian mixture models, we observe stable function estimation and state of the art performance for distributed online multi-class classification. Experiments on the Brodatz textures further substantiate the empirical validity of this approach.**

I. Introduction

We consider decentralized online optimization problems: a network $\mathcal{G} = (V, \mathcal{E})$ of agents aims to minimize a global objective that is a sum of local convex objectives available only to each node. The problem is online and distributed because data samples upon which the local objectives depend are sequentially and locally observed by each agent. In this setting, agents aim to make inferences as well as one which has access to all data at a centralized location in advance. Instead of assuming agents seek a common parameter vector $\mathbf{w} \in \mathbb{R}^p$, we focus on the case where agents seek to learn a common *decision function* $f(\mathbf{x})$ that belong to a reproducing kernel Hilbert space (RKHS). Such functions represent, e.g., nonlinear statistical models [2] or trajectories in a continuous space [3]. Learning in multi-agent settings arises predominately in two technological settings: industrial-scale machine learning, where optimizing statistical model parameters is decentralized across a parallel processing architecture to attain computational speedup; and networked intelligent systems such as sensor networks [4], multi-robot teams [5], [6], and Internet of Things [7], [8]. In the later setting, decentralized processing justified as opposed to using a fusion center when the communication cost of centralization exceeds the cost of distributed information protocols. This is true of multi-agent systems with streaming data considered here.

Efforts to develop optimization tools for multi-agent online learning have thus far been restricted to the case where each agent learns a linear statistical model [9] or a task-driven dictionary [10] that is as good as one with data aggregated across the network. However, these efforts exclude the state of the art tools for statistical learning based on nonlinear interpolators: namely, kernel methods [11], [12] and neural networks [13], [14]. We note that instabilities associated with non-convexity which are only a minor issue in centralized settings [15] become both theoretically and empirically difficult to overcome in settings with consensus constraints [10], and therefore efforts to extend neural network learning to multi-agent online learning likely suffer the same drawbacks.[1] Therefore, we focus on extending kernel methods to decentralized online settings, motivated both by its advantageous empirical performance, as well as the theoretical and practical benefits of the fact that the optimization problem defined by their training is convex. This stochastic convex problem, however, is defined over an infinite dimensional space, and therefore it is not enough to solve the optimization problem, but one must also solve it in an optimally sparse way. Doing so in multi-agent settings is the goal of this work.

To contextualize our solution methodology, consider centralized vector-valued stochastic convex programming, which has classically been solved with stochastic gradient descent (SGD) [16]. SGD involves descending along the negative of the stochastic gradient rather than the true gradient to avoid the fact that computing the gradient of the average objective has complexity comparable to the training sample size, which could be infinite. In contrast, the setting considered in this work is a stochastic program defined over a function space, which is in general an intractable variational inference problem. However, when the function space is a RKHS [17], the Representer Theorem allows us to transform a search over an infinite space into one over a set of weights and data samples [18]. Unfortunately, the feasible set of the resulting problem has complexity comparable to the sample size N, and thus is intractable for $N \to \infty$ [19]. Compounding this problem is that the storage required to construct the functional

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 [1].

[§]Computational and Information Sciences Directorate, U.S. Army Research Laboratory, Adelphi, MD, 20783. Email: alec.e.koppel.civ@mail.mil

[⋆]Department of ESE, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Email: {spater, aribeiro}@seas.upenn.edu

[†]Laboratory Lagrange - UMR CNRS 7293, Observatory of the French Riviera University of Nice Sophia-Antipolis, Nice, France, 06108

[1]In general, globally convergent decentralized online training of neural networks is an open problem, whose solution requires fundamentally new approaches to stochastic global optimization.

generalization of SGD is comparable to the iteration index of the algorithm, which is untenable for online settings.

Efforts to mitigate the complexity of the function representation ("the curse of kernelization") have been previously developed. These combine functional extensions of stochastic gradient method with compressions of the function parameterization independently of the optimization problem to which they are applied [20]–[24] or approximate the kernel during training [25]–[29], and at best converge on average. In contrast, a method was recently proposed that combines greedily constructed [30] sparse subspace projections with functional stochastic gradient method and guarantees exact convergence to the minimizer of the average risk functional. This technique, called parsimonious online learning with kernels (POLK), tailors the parameterization compression to preserve the descent properties of the underlying RKHS-valued stochastic process [31], and inspires the approach considered here.

In this work, we extend the ideas in [31] to multi-agent settings. Multiple tools from distributed optimization may be used to do so; however, we note that the Representer Theorem [18] has not been established for general stochastic saddle point problems in RKHSs. Therefore, we adopt an approximate primal-only approach based on penalty methods [32], [33], which in decentralized optimization is known as distributed gradient descent (DGD). Using functional stochastic extensions of DGD, together with the greedy Hilbert subspace projections designed in POLK, we develop a method such that each agent, through its local data stream and message passing with only its neighbors, learns a memory-efficient approximation to the globally optimal regression function with probability 1. Moreover, the average rate at which the algorithm settles to this neighborhood is linear. Such global stability guarantees are in contrast to specialized results for multi-agent kernel learning [34], [35] and alternative distributed online nonlinear function estimation methods such as dictionary learning [10], [15], [36] or neural networks [14], which suffer from instability due to the non-convexity of the optimization problem their training defines.

The result of the paper is organized as follows. In Section II we clarify the problem setting of stochastic programming in RKHSs in the centralized and decentralized case. In Section III, we propose a new penalty functional that permits deriving a decentralized online method for kernel regression without any complexity bottleneck by making use of functional stochastic gradient method (Section III-A) combined with greedy subspace projections (Section III-B). In Section IV we present our main theoretical results, which establishes that the function sequence of each agent generated by the proposed technique converges to a neighborhood of the globally optimal function with probability 1. We further establish that the mean convergence rate is linear to a neighborhood when used with constant step-size and compression budget. In Section V, we present numerical examples of decentralized online multi-class kernel logistic regression and kernel support vector machines with data generated from Gaussian mixtures, and observe a state of the art trade-off between Lyapunov stability and statistical accuracy. We then apply the resulting method to the benchmark Brodatz texture dataset [37] and observe state of the art decentralized online multi-class classification performance.

II. Problem Formulation

A. Decentralized Functional Stochastic Programming

Consider the problem of expected risk minimization, 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 very large or infinite. In this work, input-output examples, $(\mathbf{x}_n, \mathbf{y}_n)$, are i.i.d. realizations drawn from a stationary joint distribution over the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^p$ and $\mathcal{Y} \subset \mathbb{R}$. Here, we consider finding regressors that are not vector valued parameters, but rather functions $\tilde{f} \in \mathcal{H}$ in a hypothesized function class $\mathcal{H}$, which allows for learning nonlinear statistical models rather than generalized linear models that rarely achieve satisfactory statistical error rates in practice [12], [38]. The merit of the function $\tilde{f}$ 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 $\tilde{f}(\tilde{\mathbf{x}})$ evaluated at feature vector $\tilde{\mathbf{x}}$. This loss is averaged over all possible training examples to define the statistical loss $\tilde{L}(\tilde{f}) := \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\tilde{f}(\mathbf{x}), y)]$, which we combine with a Tikhonov regularizer to construct the regularized loss $\tilde{R}(\tilde{f}) := \operatorname{argmin}_{\tilde{f}\in\mathcal{H}} \tilde{L}(\tilde{f}) + (\lambda/2)\|\tilde{f}\|^2_{\mathcal{H}}$ [39], [40]. We then define the optimal function as

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

In this work, we focus on extensions of the formulation in (1) to the case where data is scattered across an interconnected network that represents, for instance, robotic teams [10], communication systems [41], or sensor networks [4]. To do so, we define a symmetric, connected, and directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = V$ nodes and $|\mathcal{E}| = E$ edges and denote as $n_i := \{j : (i, j) \in \mathcal{E}\}$ the neighborhood of agent i. For simplicity we assume that the number of edges E is even. Each agent $i \in \mathcal{V}$ observes a local data sequence as realizations $(\mathbf{x}_{i,n}, y_{i,n})$ from random pair $(\mathbf{x}_i, y_i) \in \mathcal{X}\times\mathcal{Y}$ and seeks to learn a common globally optimal regression function f. This setting may be mathematically captured by associating to each node i a convex loss functional $\ell_i : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the merit of the estimator $f_i(\mathbf{x}_i)$ evaluated at feature vector $\mathbf{x}_i$, and defining the goal for each node as the minimization of the common global loss

$$f^* = \operatorname*{argmin}_{f\in\mathcal{H}} \sum_{i\in\mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i,y_i}\Big[\ell_i(f\big(\mathbf{x}_i\big), y_i)\Big] + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} \right) \quad (2)$$

Observe that this global loss is a network-wide average (scaled by V) of all local losses, and therefore the minimizers of (1) and (2) coincide when $(\mathbf{x}_i, y_i)$ have a common joint distribution for each i. However, in multi-agent optimization, this is not generally the case, thus when selecting a regression function f with only local data, different agents will learn a different decision function f_i^* that it is not optimal as compared to one selected in a centralized manner, i.e., with the data gathered by all agents. To overcome this limitation

we allow message passing between agents and we impose a consensus constraint on the regression function among neighbors $f_i = f_j$, $(i,j) \in \mathcal{E}$. Thus we consider the nonparametric decentralized stochastic program:

$$f^* = \underset{\{f_i\}\subset\mathcal{H}}{\operatorname{argmin}} \quad \sum_{i\in\mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i,y_i}\Big[\ell_i(f_i(\mathbf{x}), y_i)\Big] + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} \right) \quad \text{such that} \quad f_i = f_j \ , (i,j)\in\mathcal{E} \tag{3}$$

For further define the product Hilbert space $\mathcal{H}^V$ of functions aggregated over the network whose elements are stacked functions $f(\cdot) = [f_1(\cdot); \cdots; f_V(\cdot)]$ that yield vectors of length V when evaluated at local random vectors $f(\mathbf{x}) = [f_1(\mathbf{x}_1); \cdots; f_V(\mathbf{x}_V)] \in \mathbb{R}^V$. Moreover, define the stacked random vectors $\mathbf{x} = [\mathbf{x}_1; \cdots; \mathbf{x}_V] \in \mathcal{X}^V \subset \mathbb{R}^{Vp}$ and $\mathbf{y} = [y_1; \cdots y_V] \in \mathbb{R}^V$ that represents V labels or physical measurements, for instance.

The goal of this paper is to develop an algorithm to solve (3) in distributed online settings where nodes do not know the distribution of the random pair $(\mathbf{x}_i, y_i)$ but observe local independent training examples $(\mathbf{x}_{i,n}, y_{i,n})$ sequentially.

B. Function Estimation in Reproducing Kernel Hilbert Spaces

The optimization problem in (1), and hence (3), is intractable in general, since it defines a variational inference problem integrated over the unknown joint distribution $\mathbb{P}(\mathbf{x}, y)$. However, when $\mathcal{H}$ is equipped with a *reproducing kernel* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (see [12], [42]), a function estimation problem of the form (1) may be reduced to a parametric form via the Representer Theorem [19], [43]. Thus, we restrict the Hilbert space in Section II-A to be one equipped with a kernel κ that satisfies for all functions $\tilde{f} : \mathcal{X} \to \mathbb{R}$ in $\mathcal{H}$:

$$(i)\ \langle \tilde{f}, \kappa(\mathbf{x}_i, \cdot)\rangle_{\mathcal{H}} = \tilde{f}(\mathbf{x}_i), \quad (ii)\ \mathcal{H} = \overline{\operatorname{span}\{\kappa(\mathbf{x}_i, \cdot)\}} \tag{4}$$

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

In (4), property (i) is the reproducing property (via Riesz Representation Theorem [43]). Replacing $\tilde{f}$ by $\kappa(\mathbf{x}'_i, \cdot)$ in (4) (i) yields $\langle \kappa(\mathbf{x}'_i, \cdot), \kappa(\mathbf{x}_i, \cdot)\rangle_{\mathcal{H}} = \kappa(\mathbf{x}_i, \mathbf{x}'_i)$ which is the origin of the term "reproducing kernel." This property induces a nonlinear transformation of the input space $\mathcal{X}$: denote by $\phi(\cdot)$ a nonlinear map of the feature space that assigns to each $\mathbf{x}_i$ the kernel function $\kappa(\cdot, \mathbf{x}_i)$. The reproducing property yields that the inner product of the image of distinct feature vectors $\mathbf{x}_i$ and $\mathbf{x}'_i$ under the map ϕ requires only kernel evaluations: $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}'_i)\rangle_{\mathcal{H}} = \kappa(\mathbf{x}_i, \mathbf{x}'_i)$ (the 'kernel trick').

Moreover, property (4) (ii) states that functions $\tilde{f} \in \mathcal{H}$ may be written as a linear combination of kernel evaluations. For kernelized and regularized empirical risk minimization (ERM), the Representer Theorem [17], [18] establishes that the optimal $\tilde{f}$ in hypothesized function class $\mathcal{H}$ admit an expansion in terms of kernel evaluations *only* over training examples

$$\tilde{f}(\mathbf{x}_i) = \sum_{n=1}^{N} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}_i) \ , \tag{5}$$

where $\mathbf{w}_i = [w_{i,1}, \cdots, w_{i,N}]^T \in \mathbb{R}^N$ denotes a set of weights. The upper index N in (5) is referred to as the model order, and for ERM the model order and training sample size are equal. Common choices κ include the polynomial and radial basis kernels, i.e., $\kappa(\mathbf{x}_i, \mathbf{x}'_i) = \big(\mathbf{x}_i^T\mathbf{x}'_i + b\big)^d$ and $\kappa(\mathbf{x}_i, \mathbf{x}'_i) = \exp\{-\|\mathbf{x}_i - \mathbf{x}'_i\|_2^2/2d^2\}$, respectively, where $\mathbf{x}_i, \mathbf{x}'_i \in \mathcal{X}$.

Suppose, for the moment, that we have access to N i.i.d. realizations of the random pairs $(\mathbf{x}_i, y_i)$ for each agent i such that the expectation in (3) is computable, and we further ignore the consensus constraint. Then the objective in (3) becomes:

$$f^* = \underset{f\in\mathcal{H}^V}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\sum_{i\in\mathcal{V}} \ell(f_i(\mathbf{x}_{i,n}), y_{i,n}) + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} \tag{6}$$

Then, by substituting the Representer Theorem [cf. (5)] into (3), we obtain that optimizing in $\mathcal{H}^V$ reduces to optimizing over the set of NV weights:

$$f^* = \underset{\{\mathbf{w}_i\}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\sum_{i\in\mathcal{V}} \ell_i(\mathbf{w}_i^T\boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,n}), y_{i,n}) + \frac{\lambda}{2}\mathbf{w}_i^T\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i}\mathbf{w}_i, \tag{7}$$

where we have defined the Gram (or kernel) matrix $\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i} \in \mathbb{R}^{N\times N}$, with entries given by the kernel evaluations between $\mathbf{x}_{i,m}$ and $\mathbf{x}_{i,n}$ as $[\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i}]_{m,n} = \kappa(\mathbf{x}_{i,m}, \mathbf{x}_{i,n})$. We further define the vector of kernel evaluations $\boldsymbol{\kappa}_{\mathbf{X}_i}(\cdot) = [\kappa(\mathbf{x}_{i,1}, \cdot) \dots \kappa(\mathbf{x}_{i,N}, \cdot)]^T$, which are related to the kernel matrix as $\mathbf{K}_{\mathbf{X}_i,\mathbf{X}_i} = [\boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,1}) \dots \boldsymbol{\kappa}_{\mathbf{X}_i}(\mathbf{x}_{i,N})]$. The dictionary of training points associated with the kernel matrix is defined as $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ \dots \ , \mathbf{x}_{i,N}]$.

By exploiting the Representer Theorem, we transform a nonparametric infinite dimensional optimization problem in $\mathcal{H}^V$ (6) into a finite NV-dimensional parametric problem (7). Thus, for empirical risk minimization, the RKHS provides a principled framework to solve nonparametric regression problems as a search over $\mathbb{R}^{VN}$ for an optimal set of coefficients.

However, to solve problems of the form (6) when training examples $(\mathbf{x}_{i,n}, y_{i,n})$ become sequentially available or their total number N is not finite, the objective in (6) becomes an expectation over random pairs $(\mathbf{x}_i, y_i)$ as [11]

$$f^* = \underset{\mathbf{w}_i\in\mathbb{R}^{\mathcal{I}}, \{\mathbf{x}_{i,n}\}_{n\in\mathcal{I}}}{\operatorname{argmin}} \sum_{i\in\mathcal{V}} \mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(\sum_{n\in\mathcal{I}} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}_i), y_i)] + \frac{\lambda}{2}\|\sum_{n,m\in\mathcal{I}} w_{i,n}w_{i,m}\kappa(\mathbf{x}_{i,m}, \mathbf{x}_{i,n})\|^2_{\mathcal{H}} \ , \tag{8}$$

where we substitute the Representer Theorem generalized to the infinite sample-size case established in [19] into the objective (3) with $\mathcal{I}$ as some countably infinite indexing set. That is, as the data sample size $N \to \infty$, the representation of f_i becomes infinite as well. Thus, our goal is to solve (8) in an approximate manner such that each f_i admits a finite representation near f_i^*, while satisfying the consensus constraints $f_i = f_j$ for $(i,j) \in \mathcal{E}$ (which were omitted for the sake of discussion between (6) - (8)).

III. Algorithm Development

We turn to developing an online iterative and decentralized solution to solving (3) when the functions $\{f_i\}_{i\in\mathcal{V}}$ are elements of a RKHS, as detailed in Section II-B. To exploit the

Algorithm 1 Greedy Projected Penalty Method (GPPM)

Require: $\{\mathbf{x}_t, \mathbf{y}_t, \eta_t, \epsilon_t\}_{t=0,1,2,\ldots}$
initialize $f_{i,0}(\cdot) = 0, \mathbf{D}_{i,0} = [], \mathbf{w}_0 = []$, i.e. initial dictionary, coefficients are empty for each $i \in \mathcal{V}$
for $t = 0, 1, 2, \ldots$ **do**
loop in parallel for agent $i \in \mathcal{V}$
Observe local training example realization $(\mathbf{x}_{i,t}, y_{i,t})$
Send obs. $\mathbf{x}_{i,t}$ to nodes $j \in n_i$, receive scalar $f_{j,t}(\mathbf{x}_{i,t})$
Receive obs. $\mathbf{x}_{j,t}$ from nodes $j \in n_i$, send $f_{i,t}(\mathbf{x}_{j,t})$
Compute unconstrained stochastic grad. step [cf. (22)]

$$\tilde{f}_{i,t+1}(\cdot) = (1 - \eta_t\lambda) f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), \mathbf{y}_{i,t}) \; .$$

Update params: $\tilde{\mathbf{D}}_{i,t+1} = [\mathbf{D}_{i,t}, \;\; \mathbf{x}_{i,t}]$, $\tilde{\mathbf{w}}_{i,t+1}$ [cf. (23)]
Greedily compress function using matching pursuit

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

end loop
end for

properties of this function space, we require the applicability of the Representer Theorem [cf. (5)], but this result holds for any regularized minimization problem with a convex functional. Thus, we may address the consensus constraint $f_i = f_j \; , (i,j) \in \mathcal{E}$ in (3) by enforcing approximate consensus on estimates $f_i(\mathbf{x}_i) = f_j(\mathbf{x}_i)$ in expectation. This specification may be met by introducing the penalty functional

$$\psi_c(f) = \sum_{i \in \mathcal{V}} \Big(\mathbb{E}_{\mathbf{x}_i, \mathbf{y}_i} \Big[\ell_i(f_i(\mathbf{x}_i), y_i) \Big] + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 + \frac{c}{4} \sum_{j \in n_i} \mathbb{E}_{\mathbf{x}_i} \left\{ [f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2 \right\} \Big) \quad (9)$$

The reasoning for the definition (9) rather than one that directly addresses the consensus constraint deterministically is given in Remark 1, motivated by following the algorithm derivation. For future reference, we also define the local penalty as

$$\psi_{i,c}(f_i) = \mathbb{E}_{\mathbf{x}_i, \mathbf{y}_i} \Big[\ell_i(f_i(\mathbf{x}_i), y_i) \Big] + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 + \frac{c}{4} \sum_{j \in n_i} \mathbb{E}_{\mathbf{x}_i} \left\{ [f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2 \right\} \quad (10)$$

and we observe from (9) - (10) that $\psi_c(f) = \sum_i \psi_{i,c}(f_i)$. Further define $f_c^* = \operatorname{argmin}_{f \in \mathcal{H}^V} \psi_c(f)$. We note that in the vector-valued decision variable case, other techniques to address the constraint in (3) are possible such as primal-dual methods [9] or dual methods [44], but the Representer Theorem has not been established for RKHS-valued stochastic saddle point problems. It is an open question whether expressions of the form (5) apply to problems with general functional constraints, but this matter is beyond the scope of this work. Therefore, these other approaches which make use of Lagrange duality do not readily extend to the nonparametric setting considered here.

A. Functional Stochastic Gradient Method

Given that the data distribution $\mathbb{P}(\mathbf{x}, \mathbf{y})$ is unknown, minimizing $\psi_c(f)$ directly via variational inference is not possible. Rather than postulate a specific distribution for $(\mathbf{x}, \mathbf{y})$, we only assume access to sequentially available (streaming) independent and identically distributed samples $(\mathbf{x}_t, \mathbf{y}_t)$ from their joint density. Then, we may wield tools from stochastic approximation to minimize (9), which in turn yields a solution to (3). Begin by defining, $\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)$, the stochastic approximation of the penalty function $\psi_c(f)$, evaluated at a realization $(\mathbf{x}_t, \mathbf{y}_t)$ of the stacked random pair $(\mathbf{x}, \mathbf{y})$:

$$\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) = \sum_{i \in \mathcal{V}} \Big(\ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t}) + \frac{\lambda}{2} \|f_i\|_{\mathcal{H}}^2 + \frac{c}{4} \sum_{j \in n_j} (f_i(\mathbf{x}_{i,t}) - f_j(\mathbf{x}_{i,t}))^2 \Big) \quad (11)$$

and the local instantaneous penalty function $\hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}), \mathbf{y}_{i,t})$ similarly. To compute the functional stochastic gradient of $\psi_c(f)$ evaluated at a sample point $(\mathbf{x}_t, \mathbf{y}_t)$, we first address the local loss $\ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})$ in (11) as [22], [31]:

$$\nabla_{f_i} \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})(\cdot) = \frac{\partial \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})}{\partial f_i(\mathbf{x}_{i,t})} \frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i}(\cdot) \quad (12)$$

where we have applied the chain rule. Now, define the shorthand notation $\ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) := \partial \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t}) / \partial f_i(\mathbf{x}_{i,t})$ for the derivative of $\ell_i(f(\mathbf{x}_{i,t}), y_{i,t})$ with respect to its first scalar argument $f_i(\mathbf{x}_{i,t})$ evaluated at $\mathbf{x}_{i,t}$. To evaluate the second term on the right-hand side of (12), differentiate both sides of the expression defining the reproducing property of the kernel [cf. (4)(i)] with respect to f_i to obtain

$$\frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i} = \frac{\partial \langle f_i, \kappa(\mathbf{x}_{i,t}, \cdot) \rangle_{\mathcal{H}}}{\partial f_i} = \kappa(\mathbf{x}_{i,t}, \cdot) \quad (13)$$

Then, given (12) - (13), we may compute the overall gradient of the instantaneous penalty function $\hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)$ in (11) as

$$\nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) = \text{vec} \Big[\ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) \kappa(\mathbf{x}_{i,t}, \cdot) + \lambda f_i + c \sum_{j \in n_i} (f_i(\mathbf{x}_{i,t}) - f_j(\mathbf{x}_{i,t})) \kappa(\mathbf{x}_{i,t}, \cdot) \Big] \quad (14)$$

where on the right-hand side of (14), we have defined the vector stacking notation vec$[\cdot]$ to denote the stacking of V component-wise functional gradients, each associated with function f_i, $i \in \mathcal{V}$, and used the fact that the variation of the instantaneous approximate of the cross-node term, $[f_i(\mathbf{x}_i) - f_j(\mathbf{x}_i)]^2$, by the same reasoning as (12) - (13), is $2[f_i(\mathbf{x}_{i,t}) - f_j(\mathbf{x}_{i,t})] \kappa(\mathbf{x}_{i,t}, \cdot)$. With this computation in hand, we present the stochastic gradient method for the λ-regularized multi-agent expected risk minimization problem in (3) as

$$f_{t+1} = (1 - \eta_t \lambda) f_t - \eta_t \text{vec} \Big[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) \kappa(\mathbf{x}_{i,t}, \cdot) + c \sum_{j \in n_i} (f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t})) \kappa(\mathbf{x}_{i,t}, \cdot) \Big] \; , \quad (15)$$

where $\eta_t > 0$ is an algorithm step-size either chosen as diminishing with $\mathcal{O}(1/t)$ or a small constant – see Section IV. We may glean from (15) that the update for the network-wide

function f_t decouples into ones for each agent $i \in \mathcal{V}$, using the node-separability of the penalty $\psi_c(f) = \sum_i \psi_{i,c}(f_i)$, i.e.,

$$f_{i,t+1} = (1-\eta_t\lambda)f_{i,t} - \eta_t\Big[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\kappa(\mathbf{x}_{i,t},\cdot) + c\sum_{j\in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\kappa(\mathbf{x}_{i,t},\cdot)\Big] . \quad (16)$$

We further require that, given $\lambda > 0$, the step-size satisfies $\eta_t < 1/\lambda$ and the global sequence is initialized as $f_0 = 0 \in \mathcal{H}^V$. With this initialization, the Representer Theorem (5) implies that, at time t, the function $f_{i,t}$ admits an expansion in terms of feature vectors $\mathbf{x}_{i,t}$ observed thus far as

$$f_{i,t}(\mathbf{x}) = \sum_{n=1}^{t-1} w_{i,n}\kappa(\mathbf{x}_{i,n},\mathbf{x}) = \mathbf{w}_{i,t}^T\boldsymbol{\kappa}_{\mathbf{X}_{i,t}}(\mathbf{x}) . \quad (17)$$

On the right-hand side of (17) we have introduced the notation $\mathbf{X}_{i,t} = [\mathbf{x}_{i,1},\ldots,\mathbf{x}_{i,t-1}] \in \mathbb{R}^{p\times(t-1)}$, $\boldsymbol{\kappa}_{\mathbf{X}_{i,t}}(\cdot) = [\kappa(\mathbf{x}_{i,1},\cdot),\ \ldots\ ,\kappa(\mathbf{x}_{i,t-1},\cdot)]^T$, and $\mathbf{w}_{i,t} = [w_{i,1},\ \ldots\ w_{i,t-1}] \in \mathbb{R}^{t-1}$. Moreover, observe that the kernel expansion in (17), taken together with the functional update (15), yields the fact that performing the stochastic gradient method in $\mathcal{H}^V$ amounts to the following V parallel parametric updates on the kernel dictionaries $\mathbf{X}_i$ and coefficients $\mathbf{w}_i$:

$$\mathbf{X}_{i,t+1} = [\mathbf{X}_{i,t},\ \ \mathbf{x}_{i,t}] , \quad (18)$$

$$[\mathbf{w}_{i,t+1}]_u = \begin{cases}(1-\eta_t\lambda)[\mathbf{w}_{i,t}]_u & \text{for } 0\le u\le t-1\\ -\eta_t\Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t}) + c\sum_{j\in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\Big),\end{cases}$$

where the second case on the last line of (18) is for $u = t$. Observe that this update causes $\mathbf{X}_{i,t+1}$ to have one more column than $\mathbf{X}_{i,t}$. We define the *model order* as number of data points $M_{i,t}$ in the dictionary of agent i at time t (the number of columns of $\mathbf{X}_t$). FSGD is such that $M_{i,t} = t-1$, and hence grows unbounded with iteration index t. Next we address this intractable memory growth such that we may execute stochastic descent through low-dimensional projections of the stochastic gradient, inspired by [31]. First, we clarify the motivation for the choice of the penalty function (9).

Remark 1 In principle, it is possible to address the RKHS-valued consensus constraint in (3) directly, through primal-only stochastic methods, by introducing the penalty function

$$\tilde{\psi}_c(f) = \sum_{i\in\mathcal{V}}\Big(\mathbb{E}_{\mathbf{x}_i,\mathbf{y}_i}\Big[\ell_i(f_i(\mathbf{x}_i),y_i)\Big] + \frac{\lambda}{2}\|f_i\|_{\mathcal{H}}^2 + \frac{c}{2}\sum_{j\in n_i}\|f_i - f_j\|_{\mathcal{H}}^2\Big) \quad (19)$$

Observe, however, that FSGD applied to (19), using comparable reasoning to that which leads to (16) from (9), yields

$$f_{i,t+1} = (1-\eta_t\lambda)f_{i,t} - \eta_t\Big[\nabla_{f_i}\ell_i'(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\kappa(\mathbf{x}_{i,t},\cdot) + c\sum_{j\in n_i}(f_{i,t} - f_{j,t})\Big] . \quad (20)$$

Unfortunately, we cannot inductively define a parametric representation of (20) for node i in terms of its own kernel dictionaries and weights independently of the *entire function* associated to node j, since the last term in (20) lives directly in the Hilbert space. Thus, to implement (20) each agent would need to store the entire kernel dictionary and weights of all its neighbors at each step, which is impractically costly. The use of (9) rather than (19) is further justified that under a hypothesis regarding the mean transformation of the local data spaces, $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i,\cdot)]$, consensus with respect to the Hilbert norm, in addition to the mean square sense, is achieved when the penalty coefficient is $c\to\infty$ (see Section IV for details).

B. Sparse Subspace Projections

To mitigate the complexity growth noted in Section III-A, we approximate the function sequence (15) by one that is orthogonally projected 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., $\mathcal{H}_{\mathbf{D}} = \{f\ :\ f(\cdot) = \sum_{n=1}^M w_n\kappa(\mathbf{d}_n,\cdot) = \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_n,\cdot)\}_{n=1}^M$, and $\{\mathbf{d}_n\} \subset \{\mathbf{x}_u\}_{u\le t}$. For convenience we 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 function parsimony by selecting dictionaries $\mathbf{D}_i$ with $M_{i,t} << \mathcal{O}(t)$ for each i [31].

To be specific, we propose replacing the local update (16) in which the dictionary grows at each iteration by its projection onto subspace $\mathcal{H}_{\mathbf{D}_{i,t+1}} = \text{span}\{\kappa(\mathbf{d}_{i,n},\cdot)\}_{n=1}^{M_{t+1}}$ as

$$f_{i,t+1} = \operatorname*{argmin}_{f\in\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big\|f - \Big(f_{i,t} - \eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}),y_{i,t})\Big)\Big\|_{\mathcal{H}}^2 := \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[(1-\eta_t\lambda)f_{i,t} - \eta_t\Big(\nabla_{f_i}\ell_i(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t}) + c\sum_{j\in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\kappa(\mathbf{x}_{i,t},\cdot)\Big)\Big] . \quad (21)$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{i,t+1}} \subset \mathcal{H}$ by the update (21).

Coefficient update The update (21), for a fixed dictionary $\mathbf{D}_{i,t+1} \in \mathbb{R}^{p\times M_{t+1}}$, yields one in the coefficient space only. This fact may be observed by defining the un-projected stochastic gradient step starting at function $f_{i,t}$ parameterized by dictionary $\mathbf{D}_{i,t}$ and coefficients $\mathbf{w}_{i,t}$:

$$\tilde{f}_{i,t+1} = f_{i,t} - \eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_i(\mathbf{x}_{i,t}),y_{i,t}) . \quad (22)$$

This update may be represented using dictionary and weights

$$\tilde{\mathbf{D}}_{i,t+1} = [\mathbf{D}_{i,t},\ \ \mathbf{x}_{i,t}] , \quad (23)$$

$$[\tilde{\mathbf{w}}_{i,t+1}]_u = \begin{cases}(1-\eta_t\lambda)[\mathbf{w}_{i,t}]_u & \text{for } 0\le u\le t-1\\ -\eta_t\Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t}) + c\sum_{j\in n_i}(f_{i,t}(\mathbf{x}_{i,t}) - f_{j,t}(\mathbf{x}_{i,t}))\Big),\end{cases}$$

where the last coefficient is for $u = t$. Note that $\tilde{\mathbf{D}}_{i,t+1}$ has $\tilde{M} = M_{i,t}+1$ columns, which is also the length of $\tilde{\mathbf{w}}_{i,t+1}$. For a fixed $\mathbf{D}_{i,t+1}$, the stochastic projection (21) is a least-squares update on the coefficient vector: the Representer Theorem allows us to rewrite (21) in terms of kernel expansions as in Section 3.2 of [31], which yields

$$\mathbf{w}_{i,t+1} = \mathbf{K}_{\mathbf{D}_{i,t+1}\mathbf{D}_{i,t+1}}^{-1}\mathbf{K}_{\mathbf{D}_{i,t+1}\tilde{\mathbf{D}}_{i,t+1}}\tilde{\mathbf{w}}_{i,t+1} , \quad (24)$$

where we define the cross-kernel matrix $\mathbf{K}_{\mathbf{D}_{i,t+1},\tilde{\mathbf{D}}_{i,t+1}}$ whose $(n,m)^{\text{th}}$ entry is given by $\kappa(\mathbf{d}_{i,n},\tilde{\mathbf{d}}_{i,m})$. The other kernel matrices $\mathbf{K}_{\tilde{\mathbf{D}}_{i,t+1},\tilde{\mathbf{D}}_{i,t+1}}$ and $\mathbf{K}_{\mathbf{D}_{i,t+1},\mathbf{D}_{i,t+1}}$ are defined similarly. Observe that $M_{i,t+1}$ is the number of columns in $\mathbf{D}_{i,t+1}$, while

$\tilde{M}_i = M_{i,t} + 1$ is the number of columns in $\tilde{\mathbf{D}}_{t+1}$ [cf. (23)]. Given that the local projections of $\tilde{f}_{i,t+1}$ onto stochastic subspaces $\mathcal{H}_{\mathbf{D}_{i,t+1}}$, for a fixed node-specific dictionaries $\mathbf{D}_{i,t+1}$, is a least-squares problem, we now detail the kernel dictionary $\mathbf{D}_{i,t+1}$ selection from past data $\{\mathbf{x}_{i,u}, y_{i,u}\}_{u \leq t}$.

Dictionary Update The selection procedure for the kernel dictionary $\mathbf{D}_{i,t+1}$ is based upon greedy compression [45]: function $\tilde{f}_{i,t+1}$ defined by the stochastic gradient method without projection is parameterized by dictionary $\tilde{\mathbf{D}}_{i,t+1}$ [cf. (23)] of model order $\tilde{M}_i = M_{i,t} + 1$. We form $\mathbf{D}_{i,t+1}$ by selecting a subset of $M_{i,t+1}$ columns from $\tilde{\mathbf{D}}_{i,t+1}$ that best approximate $\tilde{f}_{i,t+1}$ in terms of Hilbert norm error, which may be done by executing *kernel orthogonal matching pursuit* (KOMP) [30], [46] with error tolerance ϵ_t to find a kernel dictionary matrix $\mathbf{D}_{i,t+1}$ based on the one which adds the latest sample point $\tilde{\mathbf{D}}_{i,t+1}$. 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 IV for details.

We now describe the variant of KOMP we propose using, called Destructive KOMP with Pre-Fitting (see [46], Section 2.3). Begin with an input a candidate function $\tilde{f}$ of model order $\tilde{M}$ parameterized by kernel dictionary $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times \tilde{M}}$ and coefficients $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$. The method then approximates $\tilde{f}$ by a 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. See Appendix A for further details.

We summarize the key steps of the proposed method in Algorithm 1 for solving (3) while maintaining a finite model order, thus allowing for the memory-efficient learning of nonparametric regression functions online in multi-agent systems. The method, Greedy Projected Penalty Method, executes the stochastic projection of the functional stochastic gradient iterates onto sparse subspaces $\mathcal{H}_{\mathbf{D}_{i,t+1}}$ stated in (21). Initial functions are set to null $f_{i,0} = 0$, i.e., it has empty dictionary $\mathbf{D}_{i,0} = []$ and coefficient vector $\mathbf{w}_{i,0} = []$. 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 $(\mathbf{x}_{i,t}, y_{i,t})$ and step-size η_t, we compute the *unconstrained* functional stochastic gradient iterate (22) with respect to the instantaneous penalty function (11) which admits the parameterization $\tilde{\mathbf{D}}_{i,t+1}$ and $\tilde{\mathbf{w}}_{i,t+1}$ as stated in (23). These parameters are then fed into KOMP with approximation budget ϵ_t, such that $(f_{i,t+1}, \mathbf{D}_{i,t+1}, \mathbf{w}_{i,t+1}) = \text{KOMP}(\tilde{f}_{i,t+1}, \tilde{\mathbf{D}}_{i,t+1}, \tilde{\mathbf{w}}_{i,t+1}, \epsilon_t)$.

Communication Complexity Before shifting focus to a discussion of the analytical properties of Algorithm 1 for solving nonparametric learning problems in multi-agent systems, we discuss the communication requirements and contrast them with the complexity that is required for online training of linear statistical models or neural networks. For the Greedy Projected Penalty Method to work, we require Steps 2 and 3. Specifically, at time t, each agent i sends its observation $(\mathbf{x}_{i,t}, y_{i,t})$ to its neighbors, which is of complexity $p|n_i|$, where $|n_i|$ denotes the size of the neighborhood of agent i. Then, node j replies with the scalar $f_j(\mathbf{x}_{i,t})$ back to node i, which requires $\mathcal{O}(M_{j,t}p)$ local computations at node j. Here $M_{i,t}$ is the model order of agent i's regression function. Then this protocol is mirrored: node j sends its observation $(\mathbf{x}_{j,t}, y_{j,t})$ to node i so that it may reply with $f_i(\mathbf{x}_{j,t})$ to send back to node j, which requires $\mathcal{O}(M_{i,t}p)$. Thus, compared to the centralized complexity of POLK [31], at each time we require each node to execute $\mathcal{O}(M_{i,t}p)$ local computations and send its observations to its neighbors (one communication round). If we run the algorithm for T iterations, this means that T communication rounds are required, and $\mathcal{O}(\sum_{i,t} M_{i,t}p)$ additional computations. Compared to other methods for nonlinear interpolation such as neural networks, there exists no method for *synchronized* decentralized training, since their non-convexity means that consensus-based constraints may not be imposed without duality gap [2]. On the hand, decentralized online training of linear statistical models requires as many communication rounds (with local per-round of complexity $p|n_i|$) [33], [47], but without additional local computations. Thus, we may interpret the additional local computations as the price for learning a function in a much richer hypothesis class $\mathcal{H}$.

IV. Convergence Analysis

We turn to establishing that the method presented in Algorithm 1 converges with probability 1 to the minimizer of the penalty function $\psi_c(f)$ [cf. (9)] when attenuating algorithm step-sizes are used, and to a neighborhood of the minimizer along a subsequence when constant step-sizes are used. Moreover, for the later case, the kernel dictionary that parameterizes the regression function f_i for each agent i remains finite in the worst case. This analysis is an application of Section IV of [31], but these results, together with the properties of the penalty function $\psi_c(f)$ allow us to establish bounds on the deviation for each individual in the network from the common globally optimal regression function.

Before analyzing the proposed method developed in Section III, we define key quantities to simplify the analysis and introduce standard assumptions which are necessary to establish convergence. Define the local projected stochastic functional gradient associated with the update in (21) as

$$\tilde{\nabla}_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) = \Big(f_{i,t} - \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t} - \eta_t \nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\Big]\Big)/\eta_t \tag{25}$$

such that the local update of Algorithm 1 [cf. (21)] may be expressed as a stochastic descent using projected functional gradients $f_{i,t+1} = f_{i,t} - \eta_t \tilde{\nabla}_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})$. The definitions of (25) and the local stochastic gradient $\nabla_{f_i} \hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})$ may be stacked to analyze the global convergence behavior of the algorithm. For further reference, we define the stacked projected functional stochastic gradient of the penalty function as $\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) = [\tilde{\nabla}_{f_1} \hat{\psi}_{1,c}(f_{1,t}(\mathbf{x}_{1,t}), y_{1,t}); \cdots; \tilde{\nabla}_{f_V} \hat{\psi}_{V,c}(f_{V,t}(\mathbf{x}_{V,t}), y_{V,t})]$. Then the stacked global update of the algorithm is

$$f_{t+1} = f_t - \eta_t \tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) \; . \tag{26}$$

Moreover, observe that the stochastic functional gradient in (14), 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 penalty function $\psi_c(f)$ in (9), i.e.

$$\mathbb{E}[\nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t) \,\big|\, \mathcal{F}_t] = \nabla_f \psi_c(f) \tag{27}$$

for all t. In (27), we denote as $\mathcal{F}_t$ the sigma algebra which measures the algorithm history for times $u < t$, i.e. $\mathcal{F}_t = \{\mathbf{x}_u, y_u, u_u\}_{u=1}^{t-1}$. 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 kernel map may be bounded as*

$$\sup_{\mathbf{x} \in \mathcal{X}} \sqrt{\kappa(\mathbf{x}, \mathbf{x})} = X < \infty \tag{28}$$

Assumption 2 *The local losses $\ell_i(f_i(\mathbf{x}), y)$ are convex and differentiable with respect to the first (scalar) argument $f_i(\mathbf{x})$ on $\mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. Moreover, the instantaneous losses $\ell_i : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are C_i-Lipschitz continuous for all $z \in \mathbb{R}$ for a fixed $y \in \mathcal{Y}$*

$$|\ell_i(z, y) - \ell_i(z', y)| \leq C_i |z - z'| \tag{29}$$

with $C := \max_i C_i$ as the largest modulus of continuity.

Assumption 3 *The projected functional gradient of the instantaneous penalty function defined by stacking* (25) *has finite conditional second moments:*

$$\mathbb{E}[\|\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}}^2 \mid \mathcal{F}_t] \leq \sigma^2 \tag{30}$$

Assumption 1 holds in most settings by the data domain itself, and justifies the bounding of the loss. Taken together, these conditions permit bounding the optimal function f_c^* 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 [48], [49], and 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 standard in the analysis of descent methods (see [50]). Assumption 3 is common in stochastic methods, and ensures that the stochastic approximation error has finite variance.

Next we establish a few auxiliary results needed in the proof of the main results. Specifically, we introduce a proposition which quantifies the error due to sparse projections in terms of the ratio of the compression budget to the learning rate.

Proposition 1 *Given independent realizations $(\mathbf{x}_t, \mathbf{y}_t)$ of the random pair $(\mathbf{x}, \mathbf{y})$, the difference between the stacked projected stochastic functional gradient and the its un-projected variant defined by* (25) *and* (14)*, respectively, is bounded as*

$$\|\tilde{\nabla}_f \hat{\psi}_c(f_t(\mathbf{x}_t), \mathbf{y}_t) - \nabla_f \hat{\psi}_c(f(\mathbf{x}_t), \mathbf{y}_t)\|_{\mathcal{H}} \leq \frac{\epsilon_t V}{\eta_t} \tag{31}$$

where $\eta_t > 0$ denotes the algorithm step-size and $\epsilon_t > 0$ is the approximation budget parameter of Algorithm 2.

Proof: See Appendix B. ■

With the error induced by sparse projections quantified, we may now shift focus to analyzing the Hilbert-norm suboptimality of the stacked iterates generated by Algorithm 1. Specifically, we have a descent property of the sequence $\{f_t\}$.

Lemma 1 *(Stochastic Descent) Consider the sequence generated $\{f_t\}$ by Algorithm 1 with $f_0 = 0$. Under Assumptions 1-3, the following expected descent relation holds.*

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2 \,\big|\, \mathcal{F}_t\right] \leq \|f_t - f_c^*\|_{\mathcal{H}}^2 - 2\eta_t[\psi_c(f_t) - \psi_c(f_c^*)] + 2\epsilon_t V \|f_t - f_c^*\|_{\mathcal{H}} + \eta_t^2 \sigma^2 \tag{32}$$

Proof: See Appendix B. ■

Now that Lemma 1 establishes a descent-like property, we may apply the proof of Theorem 1 in [31] to $\|f_t - f_c^*\|_{\mathcal{H}}$ with diminishing step-sizes. Thus we have the following corollary.

Corollary 1 *Consider the sequence $\{f_t\}$ generated by Algorithm 1 with $f_0 = 0$ and regularizer $\lambda > 0$. Under Assumptions 1-3 and the hypothesis that the projection sets $\mathcal{H}_{\mathbf{D}_{i,t}}$ in* (21) *are intersected with some finite Hilbert-norm ball $\|f\|_{\mathcal{H}} \leq D$ for all t, with diminishing step-sizes and compression budget, i.e.,*

$$\sum_{t=0}^{\infty} \eta_t = \infty \,, \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty \,, \quad \epsilon_t = \eta_t^2 \,, \tag{33}$$

such that $\eta_t < 1/\lambda$, the sequence converges exactly to the minimizer of the penalty [cf. (9)*]: $f_t \to f_c^*$ with probability* 1*.*

To attain exact convergence to the minimizer of the penalty, f_c^*, we require the compression budget determining the error ϵ_t incurred by sparse projections to approach null. This means that to have exact convergence, we require the function representation to require an increasing amount of memory which is, in the limit, of infinite complexity. In contrast, when constant step-size and compression budget are used, then the algorithm settles to a neighborhood, as we state next.

Theorem 1 *The sequence $\{f_t\}$ generated by Algorithm 1 with $f_0 = 0$ and regularizer $\lambda > 0$, under Assumptions 1-3, with constant step-size selection $\eta_t = \eta < 1/\lambda$ and constant compression budget $\epsilon_t = \epsilon = K\eta^{3/2}$ for a positive constant K, converges to a neighborhood of f_c^* with probability* 1*:*

$$\liminf_t \|f_t - f_c^*\|_{\mathcal{H}} \leq \frac{\sqrt{\eta}}{\lambda}\left[KV + \sqrt{K^2V^2 + \lambda\sigma^2}\right] = \mathcal{O}(\sqrt{\eta}) \text{ a.s.} \tag{34}$$

Proof: See Appendix D. ■

Empirically, the use of constant step-sizes has the effect of maintaining consistent algorithm adaptivity in the face of new data, at the cost of losing exact convergence. Moreover, the rate at which Algorithm 1 settles to a neighborhood of the minimizer of the penalty function is linear. This is the first finite sample analysis of multi-agent online learning with kernels.

Theorem 2 *The sequence $\{f_t\}$ generated by Algorithm 1 with $f_0 = 0$ and regularizer $\lambda > 0$, under Assumptions 1-3, with constant step-size selection $\eta_t = \eta < 1/\lambda$ and constant compression budget $\epsilon_t = \epsilon = K\eta^2$ for a positive constant K, linearly converges in mean to a neighborhood of f_c^**

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)^t \mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + \frac{4\eta(KVD + \sigma^2)}{\lambda}. \tag{35}$$

where D is the bound on $\|f_t\|$ as in Corollary 1.

Proof: See Appendix E. ■

The interpretation of Theorem 2 is as follows: the algorithm converges linearly to a bounded error neighborhood of the optimal on average. The rate at which this occurs is $\rho = 1-\eta\lambda$. Thus, to make the learning rate faster, one should make the product $\eta\lambda$ smaller. Given that λ is fixed at time of selecting η, this means larger learning rates yield a smaller contraction on average per iteration. However, selecting larger η translates to converging to a looser neighborhood of f_c^*, the goal limiting solution. This tradeoff between rate and limiting solution accuracy is conventional of stochastic optimization algorithms when run with constant step-sizes – see [51], for instance.

The cost of losing exact convergence is that we are allowed to ensure that the algorithm compression budget does not go to null. This means that the resulting complexity of the regression functions' parameterization does not grow out of control. In particular, with constant step-size and compression budget, we may apply Theorem 3 of [31], which guarantees the model order of the function sequence remains finite, and in the worst case, is related to the covering number of the data domain.

Corollary 2 *Denote $f_t \in \mathcal{H}^V$ as the stacked function sequence defined by Algorithm 1 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 the stacked function 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 $M_t \leq M^\infty$.*

Thus, only constant step-sizes attain a reasonable tradeoff between performance relative to f_c^* and the complexity of storing the function sequence $\{f_t\}$: in this setting, we obtain approximate convergence to f_c^* while ensuring the memory requirements are always finite, as stated in Corollary 2.

We are left to analyze the goodness of the solution f_c^* as an approximation of the solution of the original problem (3). In particular, we establish consensus in the mean square sense. Let us start by establishing that the penalty term is bounded by a p^*/c, where p^* is the primal value of the optimization problem (3) and c is the barrier parameter introduced in (9).

Proposition 2 *Let Assumptions 1 - 3 hold. Let f_c^* be the minimizer of the penalty function* (9) *and let p^* be the primal optimal value of* (3)*. Then, it holds that*

$$\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in n_i}\mathbb{E}_{\mathbf{x}_i}\left\{[f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2\right\} \leq \frac{p^*}{c}. \tag{36}$$

Proof: See Appendix F. ■

Proposition 2 establishes a relationship between the choice of penalty parameter c and constraint satisfaction. This result may be used to attain convergence in mean square of each individual agent's regression function to ones which coincide with one another. Under an additional hypothesis, we obtain exact consensus, as we state next.

Theorem 3 *Let Assumptions 1 - 3 hold. Let f_c^* be the minimizer of the penalty function* (9)*. Then, suppose the penalty parameter c in* (9) *approaches infinity $c \to \infty$, and that the node-pair differences $f_{i,c}^* - f_{j,c}^*$ are not orthogonal to mean transformation $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i, \cdot)]$ of the local input spaces $\mathbf{x}_i$ for all $(i,j) \in \mathcal{E}$. Then $f_{i,c}^* = f_{j,c}^*$ for all $(i,j) \in \mathcal{E}$.*

Proof: As a consequence, the limit of (36) when c tends to infinity yields consensus in L^2 sense, i.e.,

$$\lim_{c\to\infty} \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in n_i}\mathbb{E}_{\mathbf{x}_i}\left\{[f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2\right\} = 0, \tag{37}$$

which, by pulling the limit outside the sum in (37), yields

$$\lim_{c\to\infty} \mathbb{E}_{\mathbf{x}_i}\left\{[f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2\right\} = 0\,, \tag{38}$$

for all $(i,j) \in \mathcal{E}$. Consensus in the mean square sense is a less stringent constraint that equality in the Hilbert norm as desired in (3). In particular, for any $(i,j) \in \mathcal{E}$, if $f_i = f_j$, then consensus in the mean square sense is satisfied as well. Then, apply the reproducing property of the kernel (4)(i), to write

$$\begin{aligned} 0 &= \lim_{c\to\infty} \mathbb{E}_{\mathbf{x}_i}\left\{\left|< f_{c,i}^* - f_{c,j}^*, k(\mathbf{x}_i, \cdot) >\right|\right\} \\ &\geq \lim_{c\to\infty} \left|\mathbb{E}_{\mathbf{x}_i}\left\{< f_{c,i}^* - f_{c,j}^*, k(\mathbf{x}_i, \cdot) >\right\}\right| \\ &= \lim_{c\to\infty} \left|< f_{c,i}^* - f_{c,j}^*, \mathbb{E}_{\mathbf{x}_i} k(\mathbf{x}_i, \cdot) >\right| \end{aligned} \tag{39}$$

where in the previous expression we pull the absolute value outside the expectation, and in the later we apply linearity of the expectation. Thus, (39) implies consensus is achieved with respect to the Hilbert norm, whenever the function differences $f_{c,i}^* - f_{c,j}^*$ are not orthogonal to $\mathbb{E}_{\mathbf{x}_i}[\kappa(\mathbf{x}_i, \cdot)]$, the mean of the transformation of the local input data $\mathbf{x}_i$. ■

V. Numerical Experiments

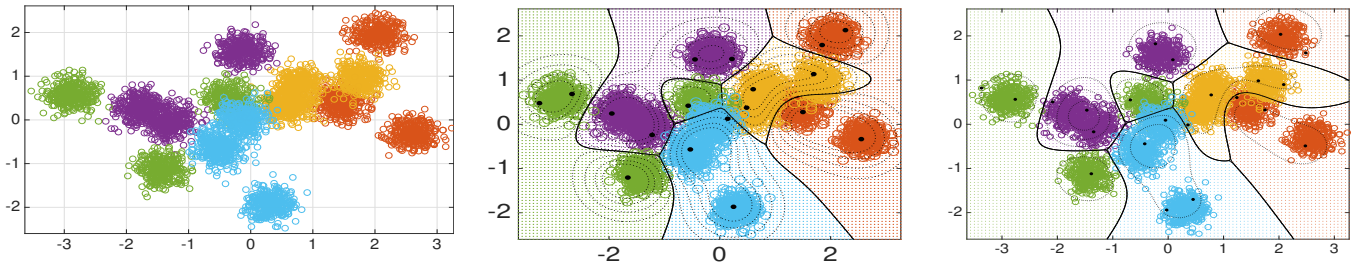
We consider the task of kernel logistic regression (KLR) (Section V-A) from multi-class training data scattered across a multi-agent system in two settings: classification of data from a Gaussian mixture model and texture classification. In Section V-B, we consider kernel support vector machines (KSVM).[2]

A. Kernel Logistic Regression

For KLR, the merit of a particular regressor for agent i is quantified by its contribution to the class-conditional probability. We define a set of class-specific functions $f_{i,k} : \mathcal{X} \to \mathbb{R}$, and denote them jointly as $\mathbf{f}_i \in \mathcal{H}^D$, where $\{1, \ldots, D\}$ denotes the set of classes. Then, define the probabilistic model

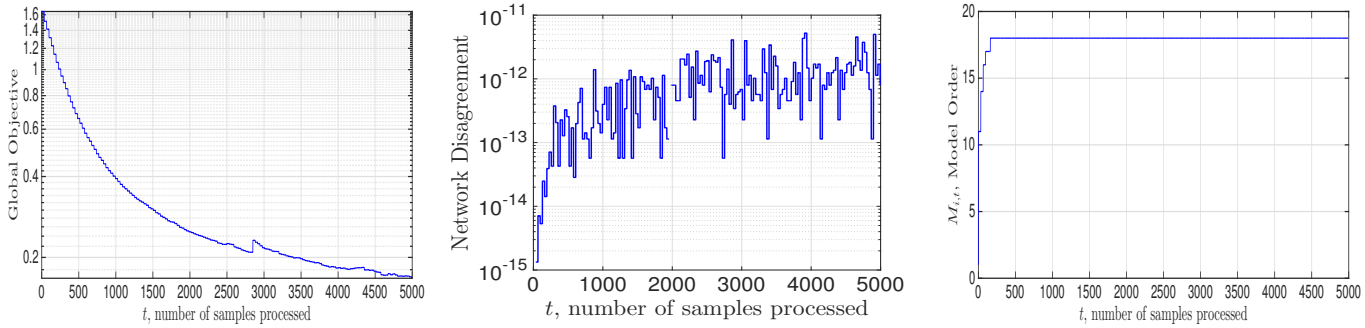
$$P(y_i = d \,|\, \mathbf{x}_i) := \frac{\exp(f_{i,d}(\mathbf{x}_i))}{\sum_{d'} \exp(f_{i,d'}(\mathbf{x}_i))}. \tag{40}$$

[2]We thank Garrett Warnell and Ethan Stump of the U.S. Army Research Laboratory for invaluable assistance in the algorithm implementation.



(a) Gaussian Mixtures data. (b) Logistic Decision surface. (c) Hinge Decision surface.

Fig. 1: Visualizations of the Gaussian mixture data set (Figure 1a) as in [24] and the learned low-memory multi-class kernel logistic regressor of a randomly chosen agent in the network (Figure 1b), which attains 95.2% classification accuracy on a hold-out test set. Curved black lines denote decision boundaries between classes; dotted lines denote confidence intervals; bold black dots denote kernel dictionary elements associated to an arbitrary $i \in \mathcal{V}$. Kernel dictionary elements concentrate at peaks of the Gaussian clusters and near points of overlap between classes. In Figure 1c we plot the resulting decision surface learned by kernel SVM which attains 95.7% accuracy – the state of the art.



(a) Global objective vs. samples processed (b) Disagreement vs. samples processed (c) Model Order $M_{i,t}$ vs. samples processed

Fig. 2: In Fig. 2a, we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ versus the number of samples processed, and observe convergence. In Fig. 2b we display the Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|^2_{\mathcal{H}}$ with a penalty parameter c that doubles every 200 samples. As c increases, agents attain consensus. In Fig. 2c, we plot the model order of a randomly chosen agent's regression function, which stabilizes to 18 after 162 samples.

which models the odds ratio of a sample being in class d versus all others. The negative log likelihood defined by (40) is the instantaneous loss (see, e.g., [52]) at sample $(\mathbf{x}_{i,n}, y_{i,n})$:

$$\ell_i(\mathbf{f}_i, \mathbf{x}_{i,n}, y_{i,n}) = -\log P(y_i = y_{i,n}|\mathbf{x}_{i,n}). \tag{41}$$

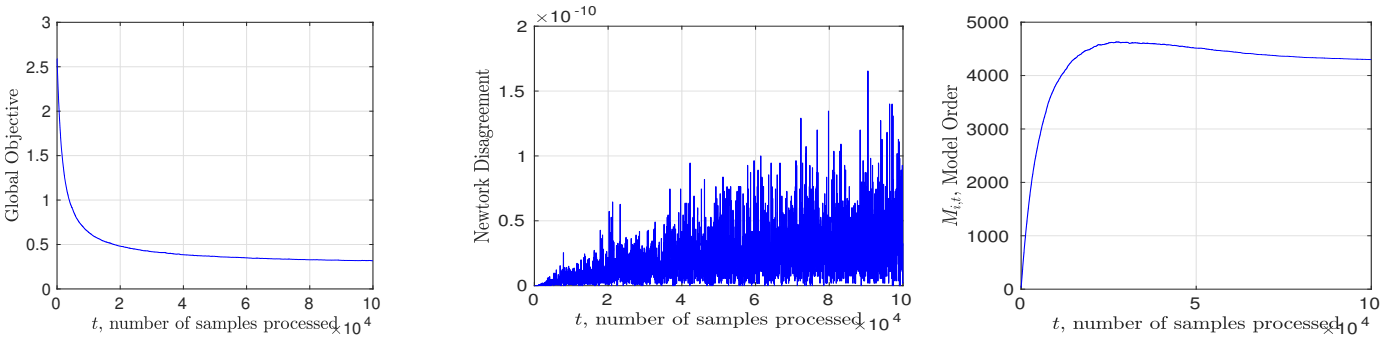
For a given set of activation functions, classification decisions $\tilde{d}$ for $\mathbf{x}_i$ is given by the maximum likelihood estimate, i.e., $\tilde{d} = \text{argmax}_{d\in\{1,\dots,D\}} f_{i,d}(\mathbf{x})$.

Gaussian Mixture Model Following [24], [31], we generate a data set from Gaussian mixture models, which 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., $\mathbf{x}\,|\,y \sim (1/3)\sum_{j=1}^{3}\mathcal{N}(\boldsymbol{\mu}_{y,j}, \sigma^2_{y,j}\mathbf{I})$ where $\sigma^2_{y,j} = 0.2$ 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., $\boldsymbol{\mu}_{y,j} \sim \mathcal{N}(\boldsymbol{\theta}_y, \sigma^2_y\mathbf{I})$, where $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_D\}$ are equitably spaced around the unit circle, one for each class label, and $\sigma^2_y = 1.0$. We fix the number of classes $D = 5$, meaning that the feature distribution has, in total, 15 distinct modes. The data is plotted in Figure 1a.

Each agent in a $V = 20$ network observes a unique stream of training examples from this common data set. Here the communications graph is a random network with edges generated randomly between nodes with probability $1/5$ repeatedly until we obtain one that is connected, and then symmetrize it. We run Algorithm 1 when the entire training set is fed to each agent in a streaming fashion, a Gaussian kernel is used with bandwidth $d = 0.6$, with constant learning rate $\eta = 3$, compression budget chosen as $\epsilon = \eta^{3/2}$ with parsimony constant $K = 0.04$, mini-batch size 32, and regularizer $\lambda = 10^{-6}$. The penalty coefficient is initialized as $c = 0.01$ and doubled after every 200 training examples.

We plot the results of this implementation in Figures 1b and 2. In Figure 2a, we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ relative to the number of training examples processed, and observe stable convergence to a global minimum. In Figure 2b we display Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|^2_{\mathcal{H}}$ versus observed sample points. Since each regression function is initialized as null, initially the disagreement is trivially null, but it remains small over the function sample path as model training occurs. Moreover, the model order of an arbitrarily chosen agent $i = 15$ versus samples processed is given in Figure 2c: observe that the model order stabilizes after only a couple hundred training examples to 18, which is only a couple more than 15, the number of modes of the joint data density function. The resulting decision surface of node 15 is given in Figure 1b.

In the second column of Table I, we compare how Al-



(a) Global Objective vs. samples processed (b) Disagreement vs. samples processed (c) Model Order $M_{i,t}$ vs. samples processed

Fig. 3: Here we run the algorithm on the Brodatz dataset [37]. In Fig. 3a, we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ versus the number of samples processed, and observe convergence. In Fig. 3b we display the Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|^2_{\mathcal{H}}$ with a penalty parameter $c=0.02$. In Fig. 3c, we plot the model order of a randomly chosen agent's regression function, which stabilizes to 4299.

gorithm 1 compares to several alternatives for this data domain and task. Specifically, we compare with both a first and second-order large-scale batch solver (Batch Gradient, or BatchGrad for short, and an L-BFGS solver [53]), the centralized sparse solver POLK [31], as well as when Algorithm 1 is run with null penalty coefficient, which we call Local Only learning. We observe that Algorithm 1 attains competitive performance with the centralized online and batch solvers and outperforms local-only learning. Specifically, GPPM achieves 95.2% classification accuracy on the test set which not far from batch approaches to kernel logistic regression.

Texture Classification We generated the *brodatz* data set using a subset of the images provided in [37]. Specifically, we used 13 texture images (i.e. D=13), and from them generated a set of 256 textons [54]. 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. We then randomly selected $N=10000$ feature-label pairs for training and 5000 for testing. Each agent in network with $V=5$ observes a unique stream of training examples from this common data set. Here the communication graph is a random network with edges generated randomly between nodes with probability $1/5$ repeatedly until we obtain one that is connected, and then symmetrize it. To train the classifier we run Algorithm 1 ten epoches: in each epoch we fed the entire training set to each agent in a streaming fashion. A Gaussian kernel is used with bandwith $\sigma^2=0.1$, with constant learning rate $\eta=4$, compression budget $\epsilon=\eta^{3/2}$ with parsimony constant $K=0.04$, mini-batch size 32 and regularizer $\lambda=10^{-5}$. The penalty coefficient is set to $c=0.02$.

We plot the results of this experiment in Figure 3. In Figure 3a we display the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ relative to the number of observed examples, and observe convergence to a global minimum. In Figure 3b we plot the Hilbert norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|^2_{\mathcal{H}}$. Since the initial regression function is null for all agents the disagreement is zero and as observed in Figure 3b it remains small over the training. Moreover, the model order of an agent chosen at random versus samples processed is given in Figure 3c. The resulting decision function achieves 93.5% classification accuracy over the test set which is comparable with the accuracy of the centralized version (95.6%) [31]. However the model order requiered is more than twice the model order in the centralized case (4358 in average v.s. 1833 [31]). Compared to other distributed classification algorithms the current algorithm outperforms them. For instance D4L achieves around 75% classification accuracy [10].

In the third column of Table I, we evaluate Algorithm 1 relative to the aforementioned alternatives for on the *brodatz* data for kernel logistic regression. We again observe that Algorithm 1 attains competitive performance with the centralized online and batch solvers and outperforms local-only learning. Specifically, GPPM achieves 93.5% classification accuracy on the test set which is not far from centralized batch approaches to kernel logistic regression.

B. Kernel Support Vector Machines

Now we address the problem of training a multi-class kernel support vector machine online in a multi-agent systems. The merit of a particular regressor is defined by its ability to maximize its classification margin, which may be formulated by first defining a set of class-specific activation functions $f_{i,d}:\mathcal{X}\to\mathbb{R}$, and denote them jointly as $\mathbf{f}_i\in\mathcal{H}^D$. 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.,

$$\ell_i(\mathbf{f}_i,\mathbf{x}_n,y_n)=\max(0,1+f_{i,r}(\mathbf{x}_n)-f_{i,y_n}(\mathbf{x}_n)), \quad (42)$$

where $r=\operatorname{argmax}_{d'\neq y}f_{i,d'}(\mathbf{x})$. See [52] for further details.

We consider an implementation where each agent in a $V=20$ network observes a unique stream of training examples from the Gaussian mixtures data set (see Figure 1a). Moreover, the communications graph is fixed as a random network with edges generated randomly between nodes with probability $1/5$

	Gaussian Mixtures		brodatz
Algorithm	Multi-KSVM (risk/error/model order)	Multi-Logistic (risk/error/model order)	Multi-Logistic (risk/error/model order)
LIBSVM	$-/3.92/656$	$-/-/-$	$-/-/-$
BatchGrad	$0.0993/3.80/5000$	$.131/3.84/5000$	$0.0613/4.22/10000$
L-BFGS	$0.0854/4.08/5000$	$0.0854/4.04/5000$	$0.0572/4.00/10000$
POLK	$0.0919/3.98/16$	$0.120/4.36/16$	$0.0871/4.41/1833$
GPPM ($c = 0$, Local Only)	$0.218/4.82/23$	$.221/5.16/17$	$.427/7.13/3103$
GPPM ($c > 0$)	$0.101/4.32/22$	$.173/4.82/18$	$.374/6.52/4299$

TABLE I: Comparison of LIBSVM, BatchGrad, L-BFGS, POLK, and GPPM with null penalty coefficient c as well as positive penalty on the Gaussian Mixtures and Brodatz [37] data sets. Reported risk, error, and model complexity values are reported for POLK, Local Only (GPPM with $c = 0$), and GPPM 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. LIBSVM is used as a baseline but only for SVM classification. Since it uses a fundamentally different model for multi-class problems (1v1 + majority vote), its objective value is omitted. We may observe the benefits of coordination in that agents are able to learn a regression function that is as good as a centralized online learning agent with access to all information, and outperform local-only learning. The online methods do not perform as well as the batch methods LIBSVM, BatchGrad, L-BFGS, although the batch methods take several hours to compute, whereas the online methods complete training in minutes.

repeatedly until we obtain one that is connected, and then symmetrize it. We run Algorithm 1 when the entire training set is fed to each agent in a streaming fashion, a Gaussian kernel is used with bandwidth $\tilde{\sigma}^2 = 0.6$, with constant learning rate $\eta = 3$, compression budget chosen as $\epsilon = \eta^{3/2}$ with parsimony constant $K = 0.04$, mini-batch size 32, and regularizer $\lambda = 10^{-6}$. The penalty coefficient is initialized as $c = 0.01$ and doubled after every 200 training examples.

We plot the results of this implementation in Figures 1c and 4. In Figure 4a, we observe that the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}), y_i)])$ converges stably to a global minimum as the number of samples processed increases. In Figure 4b we display Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}} \|f_{i,t} - f_{j,t}\|^2_{\mathcal{H}}$ versus observed sample points. Since each regression function is initialized as null, initially the disagreement is trivially null, but it remains small over the function sample path as model training occurs, and periodically spikes when the penalty parameter is increased. Moreover, the model order of an arbitrarily chosen agent $i = 6$ versus samples processed is given in Figure 4c: the model order stabilizes after only a couple hundred training examples to 22, which is only a couple more than 15, the number of modes of the joint data density function. The resulting decision surface of node 6 is given in Figure 1c, which achieves 95.7% classification accuracy, which is approximately state of the art.

In the first column of Table I, we evaluate Algorithm 1 relative to centralized online and batch solvers fvor this kernel SVM classification task. The result of this comparison corroborates the trends we previously observed for kernel logistic regression: Algorithm 1 attains competitive performance with the centralized online and batch solvers and outperforms local-only learning. Specifically, GPPM achieves 95.7% classification accuracy on the test set which is comparable to existing centralized batch approaches.

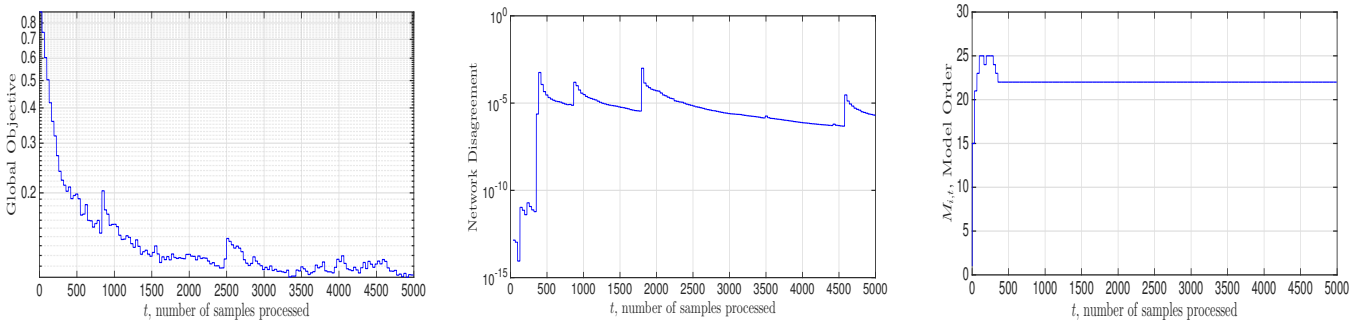
VI. Conclusion

In this paper, we extended the ideas in [31] to multi-agent settings with the intent of developing a method such that a network of autonomous agents, based on their local data stream, may learn a kernelized statistical model which is optimal with respect to information aggregated across the entire network. To do so, we proposed an unusual penalty function whose structure is amenable to efficient parameterizations when developing stochastic approximation-based updates. By applying functional stochastic gradient method to this node-separable penalty combined with greedily constructed subspace projections, we obtain a decentralized online algorithm for memory-efficient nonparametric function approximation that is globally convergent. We obtain a controllable trade-off between optimality and memory requirements through the design of the greedy subspace projections. Moreover, for large penalty parameter selections, agents achieve consensus.

The empirical performance of this protocol, the Greedy Projected Penalty Method, yields state of the art statistical accuracy for a team of interconnected agents learning from streaming data for both multi-class kernel logistic regression and multi-class kernel support vector machines problems. These results provide a mathematical and empirical foundation for accurate and stable multi-agent statistical inference in online settings while preserving memory-efficiency.

Appendix A: Details of Matching Pursuit

The removal procedure is as follows: at each step, a single dictionary element j of $\mathbf{D}$ is selected to be removed which contributes the least to the Hilbert-norm 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 incurred by removing this kernel dictionary element exceeds the given compression 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 $\mathbf{w} = \operatorname{argmin}_{\mathbf{w}\in\mathbb{R}^M} \|\tilde{f}(\cdot) - \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$, and the process repeats as long as the current function approximation is defined by a nonempty dictionary. This procedure is summarized in Algorithm 2.



(a) Global objective vs. samples (b) Disagreement vs. samples (c) Model Order $M_{i,t}$ vs. samples

Fig. 4: In Fig. 4a, we plot the global objective $\sum_{i\in\mathcal{V}}(\mathbb{E}_{\mathbf{x}_i,y_i}[\ell_i(f_{i,t}(\mathbf{x}),y_i)])$ versus the number of samples processed, and observe convergence, albeit more noisily than for the differentiable logistic loss. In Fig. 4b we display the Hilbert-norm network disagreement $\sum_{(i,j)\in\mathcal{E}}\|f_{i,t}-f_{j,t}\|_{\mathcal{H}}^2$ with a penalty parameter c that doubles every 200 samples. As c increases, agents attain consensus with respect to the Hilbert norm. In Fig. 4c, we plot the model order of a randomly chosen agent's regression function, which stabilizes to 22 after 354 samples. Here we obtain a slightly higher complexity classifier that achieves slightly better accuracy.

Algorithm 2 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 $j=1,\ldots,\tilde{M}$ **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 index minimizing approx. error: $j^*=\operatorname{argmin}_{j\in\mathcal{I}}\gamma_j$
  if minimal approx. error exceeds threshold $\gamma_{j^*}>\epsilon_t$
    stop
  else
    Prune dictionary $\mathbf{D}\leftarrow\mathbf{D}_{\mathcal{I}\setminus\{j^*\}}$
    Revise set $\mathcal{I}\leftarrow\mathcal{I}\setminus\{j^*\}$, model order $M\leftarrow M-1$.
    Update 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 $\|f-\tilde{f}\|_{\mathcal{H}}\leq\epsilon_t$

Appendix B: Proof of Proposition 1

Consider the square-Hilbert-norm difference of the stacked projected stochastic gradient $\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t),y_t)$ and its unprojected variant $\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)$ defined in (25) and (14), respectively,

$$\begin{aligned}
&\|\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t),y_t)-\nabla_f\hat{\psi}_c(f(\mathbf{x}_t),\mathbf{y}_t)\|_{\mathcal{H}}^2 \qquad (43)\\
&=\Big\|\operatorname{vec}\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t\\
&\quad-\operatorname{vec}\Big(\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big)\Big\|_{\mathcal{H}}^2\\
&\leq V^2\max_{i\in\mathcal{V}}\Big\|\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t\\
&\quad-\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big\|_{\mathcal{H}}^2
\end{aligned}$$

where we apply the fact that the functional gradient is a concatenation of functional gradients associated with each agent in (43) for the first equality, and for the second inequality we consider the worst-case estimate across the network. Now, let's focus on the term inside the Hilbert-norm on the right-hand side. Multiply and divide $\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})$, the last term, by η_t, and reorder terms to write

$$\begin{aligned}
&\Big\|\Big(f_{i,t}-\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big)/\eta_t\\
&\quad-\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big\|_{\mathcal{H}}^2\\
&=\Big\|\frac{1}{\eta_t}\Big(f_{i,t}-\eta_t\nabla_{f_i}\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big)\\
&\qquad-\frac{1}{\eta_t}\mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\Big[f_{i,t}-\eta_t\hat{\psi}_{i,c}(f_{i,t}(\mathbf{x}_{i,t}),y_{i,t})\Big]\Big\|_{\mathcal{H}}^2\\
&\qquad=\frac{1}{\eta_t^2}\|\tilde{f}_{i,t+1}-f_{i,t+1}\|_{\mathcal{H}}^2 \qquad (44)
\end{aligned}$$

where we have substituted the definition of $\tilde{f}_{i,t+1}$ and $f_{i,t+1}$ in (22) and (21), respectively, and pulled the nonnegative scalar η_t outside the norm. Now, observe that the KOMP residual stopping criterion in Algorithm 2 is $\|\tilde{f}_{i,t+1}-f_{i,t+1}\|_{\mathcal{H}}\leq\epsilon_t$, which we may apply to the last term on the right-hand side of (44). This result with the inequality (43) yields (31). ■

Appendix C: Proof of Lemma 1

Begin by considering the square of the Hilbert-norm difference between f_{t+1} and $f_c^*=\operatorname{argmin}\psi_c(f)$ which minimizes

(9), and expand the square to write

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

Add and subtract the functional stochastic gradient of the penalty function $\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)$ defined in (14) to the second term on the right-hand side of (45) to obtain

$$\begin{aligned}\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2 =&\|f_t-f_c^*\|_{\mathcal{H}}^2-2\eta_t\langle f_t-f_c^*,\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)\rangle_{\mathcal{H}}\\ &-2\eta_t\langle f_t-f_c^*,\tilde{\nabla}_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)-\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)\rangle_{\mathcal{H}}\\ &+\eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t),\mathbf{y}_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{46}$$

We deal with the third term on the right-hand side of (46), which represents the directional error associated with the sparse stochastic projections, by applying the Cauchy-Schwartz inequality together with Proposition 1 to obtain

$$\begin{aligned}\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2=&\|f_t-f_c^*\|_{\mathcal{H}}^2-2\eta_t\langle f_t-f_c^*,\nabla_f\hat{\psi}_c(f_t(\mathbf{x}_t),\mathbf{y}_t)\rangle_{\mathcal{H}}\\ &+2\epsilon_t V\|f_t-f_c^*\|_{\mathcal{H}}+\eta_t^2\|\tilde{\nabla}_f\ell(f_t(\mathbf{x}_t),y_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{47}$$

Now compute the expectation of (47) conditional on the algorithm history $\mathcal{F}_t$

$$\begin{aligned}\mathbb{E}\left[\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2\,\middle|\,\mathcal{F}_t\right] =& \|f_t-f_c^*\|_{\mathcal{H}}^2+2\epsilon_t V\|f_t-f_c^*\|_{\mathcal{H}}+\eta_t^2\sigma^2\\ &-2\eta_t\langle f_t-f_c^*,\nabla_f\psi_c(f_t)\rangle_{\mathcal{H}} \end{aligned} \tag{48}$$

where we have applied the fact that the stochastic functional gradient in (14) is an unbiased estimator [cf. (27)] for the functional gradient of the penalty function in (9), as well as the fact that the variance of the functional projected stochastic gradient is finite stated in (30) (Assumption 3). Observe that since $\psi_c(f)$ is an expectation of a convex function, it is also convex, which allows us to write

$$\psi_c(f_t)-\psi_c(f_c^*)\leq\langle f_t-f_c^*,\nabla_f\psi_c(f_t)\rangle_{\mathcal{H}}\,, \tag{49}$$

which we substitute into the second term on the right-hand side of the relation given in (48) to obtain

$$\begin{aligned}\mathbb{E}\left[\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2\,\middle|\,\mathcal{F}_t\right]\leq&\|f_t-f_c^*\|_{\mathcal{H}}^2-2\eta_t[\psi_c(f_t)-\psi_c(f_c^*)]\\ &+2\epsilon_t V\|f_t-f_c^*\|_{\mathcal{H}}+\eta_t^2\sigma^2\,. \end{aligned} \tag{50}$$

Thus the claim in Lemma 1 is valid. ■

Appendix D: Proof of Theorem 1

The use of the regularizer $(\lambda/2)\|f\|_{\mathcal{H}}^2$ in (9) implies that the penalty is λ-strongly convex in $f\in\mathcal{H}$, yielding

$$\frac{\lambda}{2}\|f_t-f_c^*\|_{\mathcal{H}}^2\leq\psi_c(f_t)-\psi_c(f_c^*) \tag{51}$$

Substituting the relation (51) into the second term on the right-hand side of the expected descent relation stated in Lemma 1, with constant step-size $\eta_t=\eta$ and budget $\epsilon_t=\epsilon$, yields

$$\begin{aligned}&\mathbb{E}[\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2\,|\,\mathcal{F}_t]\\ &\quad\leq(1-\eta\lambda)\|f_t-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_t-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2\,. \end{aligned} \tag{52}$$

The expression in (52) may be used to construct a stopping stochastic process , which tracks the suboptimality of $\|f_t-f_c^*\|_{\mathcal{H}}^2$ until it reaches a specific threshold, as in the proof of Theorem 2 of [31]. In doing so, we obtain convergence to a neighborhood. We may define a stochastic process δ_t that qualifies as a supermartingale, i.e. $\mathbb{E}\left[\delta_{t+1}\,\middle|\,\mathcal{F}_t\right]\leq\delta_t$ by considering (52) and solving for the appropriate threshold by analyzing when the following holds true

$$\begin{aligned}&\mathbb{E}[\|f_{t+1}-f_c^*\|_{\mathcal{H}}^2\,|\,\mathcal{F}_t]\\ &\quad\leq(1-\eta\lambda)\|f_t-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_t-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2\\ &\quad\leq\|f_t-f_c^*\|_{\mathcal{H}}^2\,. \end{aligned} \tag{53}$$

which may be rearranged to obtain the sufficient condition

$$-\eta\lambda\|f_t-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_t-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2\leq 0\,. \tag{54}$$

Note that (54) defines a quadratic polynomial in $\|f_t-f_c^*\|_{\mathcal{H}}$, which, using the quadratic formula, has roots

$$\|f_t-f_c^*\|_{\mathcal{H}}=\frac{\epsilon V\pm\sqrt{\epsilon^2V^2+\lambda\eta^3\sigma^2}}{\lambda\eta} \tag{55}$$

Observe (54) is a downward-opening polynomial in $\|f_t-f_c^*\|_{\mathcal{H}}$ which is nonnegative. Thus, focus on the positive root, substituting the approximation budget selection $\epsilon=K\eta^{3/2}$ to define the radius of convergence as

$$\Delta:=\frac{\epsilon V+\sqrt{\epsilon^2V^2+\lambda\eta^3\sigma^2}}{\lambda\eta}=\frac{\sqrt{\eta}}{\lambda}\Big(KV+\sqrt{K^2V^2+\lambda\sigma^2}\Big) \tag{56}$$

(56) allows us to construct a stopping process: define δ_t as

$$\begin{aligned}&\delta_t=\|f_t-f_c^*\|_{\mathcal{H}}\\ &\times\mathbb{1}\Big\{\min_{u\leq t}-\eta\lambda\|f_u-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_u-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2>\Delta\Big\}\end{aligned} \tag{57}$$

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. The rest of the proof applies the same reasoning as that of Theorem 2 in [31]: in particular, given the definition (57), either $\min_{u\leq t}-\eta\lambda\|f_u-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_u-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2>\Delta$ holds, in which case we may compute the square root of the condition in (53) to write

$$\mathbb{E}[\delta_{t+1}\,|\,\mathcal{F}_t]\leq\delta_t \tag{58}$$

Alternatively, $\min_{u\leq t}-\eta\lambda\|f_u-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_u-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2\leq\Delta$, in which case the indicator function is null for all $s\geq t$ from the use of the minimum inside the indicator in (57). Thus in either case, (58) is valid, implying δ_t converges almost surely to null, which, as a consequence we obtain the fact that either $\lim_{t\to\infty}\|f_t-f_c^*\|_{\mathcal{H}}-\Delta=0$ or the indicator function is null for large t, i.e. $\lim_{t\to\infty}\mathbb{1}\{\min_{u\leq t}-\eta\lambda\|f_u-f_c^*\|_{\mathcal{H}}^2+2\epsilon V\|f_u-f_c^*\|_{\mathcal{H}}+\eta^2\sigma^2>\Delta\}=0$ almost surely. Therefore, we obtain that

$$\liminf_{t\to\infty}\|f_t-f_c^*\|_{\mathcal{H}}\leq\Delta=\frac{\sqrt{\eta}}{\lambda}\Big(KV+\sqrt{K^2+\lambda\sigma^2}\Big)\ \text{a.s.}\,, \tag{59}$$

as stated in Theorem 1. ■

APPENDIX E: PROOF OF THEOREM 2

Begin with the expression in Lemma 1 and compute the total expectation, i.e., fix the filtration as $\mathcal{F}_0$, and set $\eta_t = \eta$ and $\epsilon_t = \epsilon = K\eta^2$ to write

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq \mathbb{E}\left[\|f_t - f_c^*\|_{\mathcal{H}}^2\right] - 2\eta\mathbb{E}\left[[\psi_c(f_t) - \psi_c(f_c^*)]\right] + 2K\eta^2 V\mathbb{E}\left[\|f_t - f_c^*\|_{\mathcal{H}}\right] + \eta^2\sigma^2 \ . \tag{60}$$

Now, apply the strong convexity of the penalty function (51) to the second-to-last term on the right-hand side of (60) and the fact that our function estimates have bounded Hilbert norm $\|f\| \leq D$ owing to the fact that the projection sets are intersected with some finite Hilbert-norm ball to obtain

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)\mathbb{E}\left[\|f_t - f_c^*\|_{\mathcal{H}}^2\right] + \eta^2 4(KVD + \sigma^2) \ . \tag{61}$$

Observe that the sub-optimality of the function sequence with respect to the minimizer of the penalty function at time $t+1$ is upper bounded by its sub-optimality at the previous time multiplied by a contractive factor $\rho = (1-\eta\lambda) < 1$ plus an error term. We can use this relationship to establish linear convergence to a neighborhood by rewriting (61) at time t instead of time $t+1$:

$$\mathbb{E}\left[\|f_t - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)\mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + \eta^2 4(KVD + \sigma^2) \ . \tag{62}$$

and then substitute the right-hand side of (62) into (61) to obtain

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)^2\mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + [1 + (1-\eta\lambda)]\eta^2 4(KVD + \sigma^2)] \ . \tag{63}$$

We can repeatedly apply steps (62) - (63) backwards in time to $t = 0$ to obtain a bound in terms of the initialization-based sub-optimality $\|f_0 - f_c^*\|_{\mathcal{H}}$, which, owing to our choice of $f_0 =$, reduces to

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)^{t+1}\mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + 4\eta^2(KVD + \sigma^2)\sum_{u=0}^{t}(1-\eta\lambda)^u \ . \tag{64}$$

Now, let's substitute t for $t-1$ in the above expression, and use the expression for a finite geometric sum $\sum_{u=0}^{t}(1-\eta\lambda)^u = 1 - (1-\eta\lambda)^t/(\eta\lambda)$ to simplify the last term on the right side of (64):

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)^t\mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + \frac{4\eta(KVD + \sigma^2)}{\lambda}[1 - (1-\eta\lambda)^t] \ . \tag{65}$$

The last multiplicative factor $1 - (1-\eta\lambda)^t]$ is strictly smaller than 1 when $\eta < 1/\lambda$, which is a criterion for the choice of the step-size of Algorithm 1 in Section III. Thus, we may simplify the above expression to

$$\mathbb{E}\left[\|f_{t+1} - f_c^*\|_{\mathcal{H}}^2\right] \leq (1-\eta\lambda)^t\mathbb{E}\left[\|f_{t-1} - f_c^*\|_{\mathcal{H}}^2\right] + \frac{4\eta(KVD + \sigma^2)}{\lambda} \ . \tag{66}$$

as stated in Theorem 2. ■

APPENDIX F: PROOF OF PROPOSITION 2

Let f_c^* be the minimizer of $\psi_c(f)$ defined in (9) and f^* be the solution of the problem (3). Since the former is the minimizer of $\psi_c(f)$ it holds that

$$\psi_c(f_c^*) \leq \psi_c(f^*) = \sum_{i\in\mathcal{V}}\Big(\mathbb{E}_{\mathbf{x}_i,\mathbf{y}_i}\Big[\ell_i(f_i^*(\mathbf{x}_i), y_i)\Big] + \frac{\lambda}{2}\|f_i^*\|_{\mathcal{H}}^2 + \frac{c}{2}\sum_{j\in n_i}\mathbb{E}_{\mathbf{x}_i}\left\{[f_i^*(\mathbf{x}_i) - f_j^*(\mathbf{x}_i)]^2\right\}\Big) \ . \tag{67}$$

Where the equality follows from the definition of $\psi_c(f)$ in (9). Since f^* is solution to the problem (3) it satisfies that $f_i = f_j$ for all $(i,j) \in \mathcal{E}$, thus

$$\mathbb{E}_{\mathbf{x}_i}\left\{[f_i^*(\mathbf{x}_i) - f_j^*(\mathbf{x}_i)]^2\right\} = 0 \ , \tag{68}$$

for all $(i,j) \in \mathcal{E}$. As a consequence, replacing $\psi_c(f_c^*)$ by its expression in the first equality in (67) and rearranging terms yields a bound the constraint violation of f_c^* as

$$\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in n_i}\mathbb{E}_{\mathbf{x}_i}\left\{[f_{c,i}^*(\mathbf{x}_i) - f_{c,j}^*(\mathbf{x}_i)]^2\right\} \leq \frac{1}{c}\left(R(f^*) - R(f_c^*)\right) \ , \tag{69}$$

where $R(f)$ is the global regularized objective in (2), i.e.,

$$R(f) = \sum_{i\in\mathcal{V}}\Big(\mathbb{E}_{\mathbf{x}_i,\mathbf{y}_i}\Big[\ell_i(f_i(\mathbf{x}_i), y_i)\Big] + \frac{\lambda}{2}\|f_i\|_{\mathcal{H}}^2\Big). \tag{70}$$

The fact that by definition $p^* = R(f^*)$ yields (36).

REFERENCES

[1] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "'decentralized efficient nonparametric stochastic optimization'," in *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on (to appear)*. IEEE, 2017.

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

[3] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces," in *Proceedings of Robotics: Science and Systems*, Ann Arbor, MI, July 2016.

[4] R. J. Kozick and B. M. Sadler, "Source localization with distributed sensor arrays and partial spatial coherence," *IEEE Transactions on Signal Processing*, vol. 52, no. 3, pp. 601–616, 2004.

[5] 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*.

[6] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Robotics Research*. Springer, 2017, pp. 455–472.

[7] J. Liu, Q. Chen, and H. D. Sherali, "Algorithm design for femtocell base station placement in commercial building environments," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2951–2955.

[8] A. Ghosh and S. Sarkar, "Pricing for profit in internet of things," in *Information Theory (ISIT), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 2211–2215.

[9] A. Koppel, F. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Trans. Signal Process.*, p. 15, Oct 2015.

[10] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "D4l: Decentralized dynamic discriminative dictionary learning," *IEEE Trans. Signal and Info. Process. over Networks*, vol. (submitted), June 2017, available at http://www.seas.upenn.edu/ aribeiro/wiki.

[11] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel hilbert spaces," *Signal Processing Theory and Machine Learning*, pp. 883–987, 2013.

[12] J.-B. Li, S.-C. Chu, and J.-S. Pan, *Kernel Learning Algorithms for Face Recognition*. Springer, 2014.

[13] S. Haykin, "Neural networks: A comprehensive foundation," 1994.

[14] 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.

[15] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 791–804, 2012.

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

[17] 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.

[18] 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.

[19] V. Norkin and M. Keyzer, "On stochastic optimization and statistical learning in reproducing kernel hilbert spaces by support vector machines (svm)," *Informatica*, vol. 20, no. 2, pp. 273–292, 2009.

[20] 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.

[21] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 543–554, 2008.

[22] 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.

[23] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a fixed budget," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, p. 259266. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=78226

[24] J. Zhu and T. Hastie, "Kernel Logistic Regression and the Import Vector Machine," *Journal of Computational and Graphical Statistics*, vol. 14, no. 1, pp. 185–205, 2005.

[25] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Advances in Neural Information Processing Systems*, 2014, pp. 3041–3049.

[26] T. Le, V. Nguyen, T. D. Nguyen, and D. Phung, "Nonparametric budgeted stochastic gradient descent," in *Artificial Intelligence and Statistics*, 2016, pp. 654–572.

[27] T. Le, T. Nguyen, V. Nguyen, and D. Phung, "Dual space gradient descent for online learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4583–4591.

[28] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *Journal of Machine Learning Research*, vol. 17, no. 47, p. 1, 2016.

[29] D. Calandriello, A. Lazaric, and M. Valko, "Second-order kernel online convex optimization with adaptive sketching," in *International Conference on Machine Learning*, 2017.

[30] 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.

[31] 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.

[32] B. Johansson, T. Keviczky, M. Johansson, and K. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4185–4190.

[33] S. Ram, A. Nedic, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J Optimiz. Theory App.*, vol. 147, no. 3, pp. 516–545, Sep. 2010.

[34] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Nonparametric decentralized detection using kernel methods," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4053–4066, 2005.

[35] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[36] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*. IEEE, 2013, pp. 133–136.

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

[38] S. Mukherjee and S. K. Nayar, "Automatic generation of rbf networks using wavelets," *Pattern Recognition*, vol. 29, no. 8, pp. 1369–1383, 1996.

[39] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan, "Learnability, stability and uniform convergence," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2635–2670, 2010.

[40] 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.

[41] A. Ribeiro, "Ergodic stochastic optimization algorithms for wireless communication and networking," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6369–6386, 2010.

[42] K. Müller, T. Adali, K. Fukumizu, J. C. Principe, and S. Theodoridis, "Special issue on advances in kernel-based learning for signal processing [from the guest editors]," *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 14–15, 2013. [Online]. Available: http://dx.doi.org/10.1109/MSP.2013.2253031

[43] 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

[44] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *Proc. 30th Int. Conf. Machine Learning*, vol. 28, no. 1, Atlanta, GA, USA, Jun. 16-21 2013, pp. 392–400.

[45] 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.

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

[47] J. A. Nelder and R. J. Baker, "Generalized linear models," *Encyclopedia of statistical sciences*.

[48] M. Pontil, Y. Ying, and D. xuan Zhou, "Error analysis for online gradient descent algorithms in reproducing kernel hilbert spaces," Tech. Rep., 2005.

[49] Y. Ying and D. X. Zhou, "Online regularized classification algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4775–4788, Nov 2006.

[50] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," 1998.

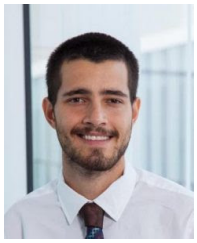
[51] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

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

[53] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[54] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearence of Materials using Three-dimensional Textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 1999.

Alec Koppel began as a Research Scientist at the U.S. Army Research Laboratory in the Computational and Information Sciences Directorate in September of 2017. He completed his Master's degree in Statistics and Doctorate in Electrical and Systems Engineering, both at the University of Pennsylvania (Penn) in August of 2017. He is also a participant in the Science, Mathematics, and Research for Transformation (SMART) Scholarship Program sponsored by the American Society of Engineering Education. Before coming to Penn, he completed his Master's degree in Systems Science and Mathematics and Bachelor's Degree in Mathematics, both at Washington University in St. Louis (WashU), Missouri. His research interests are in the areas of signal processing, optimization and learning theory. His current work focuses on optimization and learning methods for streaming data applications, with an emphasis on problems arising in autonomous systems. He co-authored a paper selected as a Best Paper Finalist at the 2017 IEEE Asilomar Conference on Signals, Systems, and Computers.



Santiago Paternain received the B.Sc. degree in electrical engineering from Universidad de la Republica Oriental del Uruguay, Montevideo, Uruguay in 2012. Since August 2013, he has been working toward the Ph.D. degree in the Department of Electrical and Systems Engineering, University of Pennsylvania. He was the recipient of the 2017 CDC Best Student Paper Award. His research interests include optimization and control of dynamical systems.

Cédric Richard (S'98-M'01-SM'07) received the Dipl.-Ing. and the M.S. degrees in 1994, and the Ph.D. degree in 1998, from Compiégne University of Technology, France. He is a Full Professor at the University of Nice Sophia Antipolis, France. He was a junior member of the Institut Universitaire de France in 2010-2015. His current research interests include statistical signal processing and machine learning. Prof. Richard is the author of over 250 papers. He was the General Co-Chair of the IEEE SSP'11 Workshop that was held in Nice, France. He was the Technical Co-Chair of EUSIPCO '15 that was held in Nice, France, and of the IEEE CAMSAP'15 Workshop that was held in Cancun, Mexico. Since 2015, he serves as a Senior Area Editor of the IEEE Transactions on Signal Processing, and as an Associate Editor of the IEEE Transactions on Signal and Information Processing over Networks. He is an Associate Editor of Signal Processing Elsevier since 2009. Prof. Richard is member of the IEEE Machine Learning for Signal Processing Technical Committee, and served as member of the IEEE Signal Processing Theory and Methods Technical Committee in 2009-2014.

Alejandro Ribeiro received the B.Sc. degree in electrical engineering from the Universidad de la Republica Oriental del Uruguay, Montevideo, in 1998 and the M.Sc. and Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, the University of Minnesota, Minneapolis in 2005 and 2007. From 1998 to 2003, he was a member of the technical staff at Bellsouth Montevideo. After his M.Sc. and Ph.D studies, in 2008 he joined the University of Pennsylvania (Penn), Philadelphia, where he is currently the Rosenbluth Associate Professor at the Department of Electrical and Systems Engineering. His research interests are in the applications of statistical signal processing to the study of networks and networked phenomena. His current research focuses on wireless networks, network optimization, learning in networks, networked control, robot teams, and structured representations of networked data structures. Dr. Ribeiro received the 2012 S. Reid Warren, Jr. Award presented by Penn?s undergraduate student body for outstanding teaching, the NSF CAREER Award in 2010, and student paper awards at the 2013 American Control Conference (as adviser), as well as the 2005 and 2006 International Conferences on Acoustics, Speech and Signal Processing. Dr. Ribeiro is a Fulbright scholar and a Penn Fellow.