Online Ensemble Multi-kernel Learning Adaptive to Non-stationary and Adversarial Environments

Yanning Shen* **Tianyi Chen*** **Georgios B. Giannakis**
Dept. of Electrical and Computer Engineering and the Digital Technology Center
University of Minnesota, Minneapolis

Abstract

Kernel-based methods exhibit well-documented performance in various nonlinear learning tasks. Most of them rely on a preselected kernel, whose prudent choice presumes task-specific prior information. To cope with this limitation, multi-kernel learning has gained popularity thanks to its flexibility in choosing kernels from a prescribed kernel dictionary. Leveraging the random feature approximation and its recent orthogonality-promoting variant, the present contribution develops an online multi-kernel learning scheme to infer the intended nonlinear function 'on the fly.' To further boost performance in non-stationary environments, an adaptive multi-kernel learning scheme (termed AdaRaker) is developed with affordable computation and memory complexity. Performance is analyzed in terms of both static and dynamic regret. To our best knowledge, AdaRaker is the *first* algorithm that can optimally track nonlinear functions in non-stationary settings with theoretical guarantees. Numerical tests on real datasets are carried out to showcase the effectiveness of the proposed algorithms.

1 Introduction

Function approximation emerges in various learning tasks such as regression, classification, as well as reinforcement learning [1, 2]. Kernel-based methods are powerful tools for nonlinear function approximation with strong theoretical guarantees. While most kernel methods utilize a pre-selected kernel, multi-kernel learning (MKL) approaches have attracted attention, thanks to their flexibility of selecting the task-specific kernel based on a prescribed kernel dictionary [3, 4].

In addition to the attractive generalization capability that can be achieved by kernel methods, several learning tasks are also expected to be performed in an online fashion. Such need naturally arises when the data arrive sequentially, such as those in online spam detection [5], and time series prediction [6]; or, when the sheer volume of data makes it impossible to carry out data analytics in batch form [7]. This motivates well online kernel-based learning methods that inherit the merits of their batch counterparts, while at the same time allowing efficient online implementation. Taking a step further, the optimal function may itself change over time in *non-stationary* environments. This is the case when the function of interest e.g., represents the state in brain networks, or, captures the temporal processes propagating over time-varying networks. Tackling online kernel-based learning tasks in non-stationary and possibly adversarial environments remains a largely uncharted territory [7, 8].

In accordance with these needs and desiderata, the *primary goal* of this paper is an algorithmic pursuit of online multi-kernel learning in non-stationary environments, along with its performance guarantees. Major challenges come from two aspects: i) the well-known curse of dimensionality in kernel-based learning; and, ii) the defiance of tracking unknown time-varying functions without future information. Regarding i), the representer theorem renders the size of kernel matrix to grow quadratically with the number of data [9], thus the computational complexity to find even the *single* kernel-based predictor is cubic. Furthermore, storage of past data causes memory overflow in large-scale learning tasks such as those emerging in e.g., topology identification of social and brain networks [10, 11], which makes kernel-based methods less scalable relative to their linear counterparts. For ii), most online learning settings presume stationarity, where an

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s). The first two authors contribute equally.

algorithm achieving sub-linear regret incurs on average "no-regret" relative to the *best static* comparator. Clearly, designing online schemes that are comparable to the *best dynamic* solution is appealing though challenging without knowledge of the environment [7, 12].

1.1 Related work

We review prior art in this area from two aspects.

Kernel methods. Significant efforts have been devoted to scaling up kernel methods in batch settings. Specifically, approaches to approximating the kernel matrix using low-rank factorizations were proposed in [13], and their performance was formally established in [14]. More recently, random feature-based methods gain popularity since the seminal work [15], whose performance has been considerably improved thanks to the fresh *orthogonality promoting* technique for variance reduction [16]. These approaches assume that the kernel function is selected a priori, and this selection crucially depends on domain knowledge. Incorporating the kernel selection, several multi-kernel based learning approaches have been proposed in [17, 3, 4], and their performance gain has been observed relative to their single kernel counterparts. However, all the aforementioned methods are designed for batch settings, and thus they are either intractable or become less efficient in online setups. Especially when the optimal functions vary over time, the batch schemes fall short in tracking the optimal predictors.

Online (multi-)kernel learning. Tailored for streaming large-scale datasets, online kernel-based learning methods have gained popularity recently. To deal with the growing complexity of online kernel learning, successful attempts have been made to design *budgeted* kernel learning algorithms, including techniques such as support vector (SV) removal [7, 18], and SV merging [19]. Following the idea of budget maintenance, online multi-kernel learning (OMKL) methods have been studied in applications such as online classification [20, 8, 21], and regression [22]. Devoid the need of budget maintenance, online kernel-based learning algorithms based on random feature approximation [15] have been recently developed in [23, 24], but only learning with a pre-selected kernel is considered. More importantly, existing online kernel-based learning approaches implicitly presume a *stationary* environment, where the benchmark solution is the best static function (a.k.a. static regret) [25]. However, static regret is not a comprehensive metric in the general non-stationary settings considered in this paper.

1.2 Our contributions

The present paper develops an adaptive online multi-kernel learning algorithm, capable of learning a nonlinear function from sequentially arriving data samples. Relative to prior art, our contributions can be summarized as follows.

c1) For the first time, random features are employed for scalable online MKL tackled by a weighted combination of advices from an ensemble of experts - an innovative cross-fertilization of online learning to MKL. Performance of the resultant algorithm (abbreviated as **Raker**) is benchmarked by the best time-invariant function approximant via static regret analysis.

c2) An adaptive algorithm (termed **AdaRaker**) is developed to tackle the multi-kernel learning task in non-stationary setting, and analytically establish that the AdaRaker solver yields sub-linear dynamic regret, so long as the accumulated variation of per-slot minimizers grow sub-linearly with time.

c3) Our novel algorithms are applied to online regression tasks, and tested numerically on several real datasets, which demonstrates the effectiveness of Raker and AdaRaker relative to popular alternatives.

1.3 Preliminaries

This section reviews basics of kernel-based learning, which paves the road for developing our novel schemes.

Notation. Bold uppercase (lowercase) letters will denote matrices (column vectors), while $(\cdot)^\top$ stands for vector and matrix transposition, and $\|\mathbf{x}\|$ denotes the ℓ_2-norm of a vector $\mathbf{x}$. Inequalities for vectors $\mathbf{x} > \mathbf{0}$ are entry-wise. $\langle \cdot, \cdot \rangle$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denote the inner products in Euclidean and Hilbert spaces, respectively.

Setting. Given samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}_{t=1}^T$ with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$, the function approximation task is to find a function $f(\cdot)$ such that $y_n = f(\mathbf{x}_n) + e_n$, where e_n denotes an error term representing noise or un-modeled dynamics. Suppose that $f(\cdot)$ belongs to a reproducing kernel Hilbert space (RKHS), namely, i.e.,

$$\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^{\infty} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\} \tag{1}$$

where $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a basis (so-termed kernel) function, which measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$. Different choices of κ specify various bases. One of the popular kernels is e.g., the Gaussian one that is given by $\kappa(\mathbf{x}, \mathbf{x}_t) := \exp[-(\mathbf{x} - \mathbf{x}_t)^2/(2\sigma^2)]$. A kernel is reproducing if it satisfies $\langle \kappa(\mathbf{x}, \mathbf{x}_1), \kappa(\mathbf{x}, \mathbf{x}_2) \rangle = \kappa(\mathbf{x}_1, \mathbf{x}_2)$, which in turn induces the RKHS norm $\|f\|_{\mathcal{H}}^2 = \sum_t \sum_{t'} \alpha_t \alpha_{t'} \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$. Consider the optimization problem

$$\min_{f \in \mathcal{H}} \mathcal{L}(f) := \frac{1}{T} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{2}$$

where depending on the application, the loss function $\ell(\cdot,\cdot)$ can be selected to be, e.g., the least-squares cost, the logistic or hinge loss, and $\lambda > 0$ is a regularization parameter. Thanks to the representer theorem, the optimal solution of (2) admits the finite-dimensional form, given by [9]

$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t) \tag{3}$$

where $\{\alpha_t \in \mathbb{R}\}_{t=1}^T$ are combination coefficients. While the scalar y_t is used here for notational brevity, coverage can be readily generalized to the vector form.

Note that (2) relies on two facts: i) a properly pre-selected kernel κ is known; and ii) all the data $\{\mathbf{x}_t, y_t\}_{t=1}^T$ are available at hand. In the ensuing sections, an online MKL method will be proposed to select the optimal κ as a convex combination of multiple kernels, when the data become available online.

2 Online MKL with random features

In this section, we develop an online algorithm to simultaneously deal with kernel basis selections and multiple kernel combinations. Our algorithm is based on the recently proposed random feature-based techniques [15, 16], and thus we term it **ra**ndom feature-based multi-**ker**nel (**Raker**) learning approach.

2.1 Kernel learning via random features

Kernel-based methods are challenged by the curse of dimensionality, due to the fact that the optimal kernel function depends on all the previous data samples [cf. (3)]. Unlike online kernel learning schemes that rely on budget maintenance strategies [26], the present section explores an alternative approach for kernel-based learning to make the subsequent online learning task scalable with the sample size. This approach relies on mapping the original data to random features (RFs), and then applying existing linear online learning algorithms in this new feature space [15]. Specifically, given $\mathbf{x}_t$, the RF-based approach constructs a new feature representation $\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) \in \mathbb{R}^{2D}$, where $D \gg d$, $\mathbf{V} \in \mathbb{R}^{D \times d}$ is a random matrix that will be specified later, and $\mathbf{z}_{\mathbf{V}}(\mathbf{x})$ approximates the kernel function by

$$k(\mathbf{x}_i, \mathbf{x}_j) \simeq \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_i) \mathbf{z}_{\mathbf{V}}(\mathbf{x}_j). \tag{4}$$

Hence, the function in the corresponding RKHS can be approximated by (cf. (3))

$$f(\mathbf{x}) \simeq \sum_{t=1}^{T} \alpha_t \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t) \mathbf{z}_{\mathbf{V}}(\mathbf{x}) \tag{5}$$

where $\boldsymbol{\theta} := \sum_{t=1}^T \alpha_t \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)$ denotes the new weight vector, the original kernel-based learning problem can be transformed into a linear problem in the new $2D$-dimensional feature space, namely

$$f(\mathbf{x}) \simeq \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}). \tag{6}$$

To efficiently approximate the kernel function, we will confine our class to kernels that are shift invariant; that is, $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa(\Delta)$ with $\Delta = \mathbf{x}_1 - \mathbf{x}_2$, and $\kappa(\mathbf{0}) = 1$. With the shift-invariant property, viewing the positive definite $\kappa(\Delta)$ as the inverse Fourier transform of $\pi_\kappa(\mathbf{v})$, yields

$$\begin{aligned} \kappa(\mathbf{x}_1, \mathbf{x}_2) &= \int \pi_\kappa(\mathbf{v}) e^{j\mathbf{v}^\top(\mathbf{x}_1 - \mathbf{x}_2)} d\mathbf{v} \\ &= \mathbb{E}_{\mathbf{v}}\left[e^{j\mathbf{v}^\top \mathbf{x}_1} \cdot e^{-j\mathbf{v}^\top \mathbf{x}_2}\right] \end{aligned} \tag{7}$$

where last equality follows by treating $\pi_\kappa(\mathbf{v})$ as the probability density function (pdf) of variable $\mathbf{v}$. Taking the Gaussian kernel as an example, where $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2/(2\sigma^2)\right)$, the corresponding pdf $\pi(\mathbf{v}) = \mathcal{N}(0, \sigma^{-2}\mathbf{I})$ [15]. In this case, plugging $e^{j\mathbf{v}^\top \mathbf{x}_1} = \cos(\mathbf{v}^\top \mathbf{x}_1) + j \sin(\mathbf{v}^\top \mathbf{x}_1)$ into (7) yields

$$\begin{aligned} \kappa(\mathbf{x}_1, \mathbf{x}_2) &= \mathbb{E}_{\mathbf{v}}\left[\cos(\mathbf{v}^\top \mathbf{x}_1)\cos(\mathbf{v}^\top \mathbf{x}_2) + \sin(\mathbf{v}^\top \mathbf{x}_1)\sin(\mathbf{v}^\top \mathbf{x}_2)\right] \\ &:= \mathbb{E}_{\mathbf{v}}\left[\mathbf{z}^\top(\mathbf{x}_1)\mathbf{z}(\mathbf{x}_2)\right] \end{aligned} \tag{8}$$

where $\mathbf{z}(\mathbf{x}) := [\sin(\mathbf{v}^\top \mathbf{x}), \cos(\mathbf{v}^\top \mathbf{x})]^\top$. Clearly, D realizations of RF $\mathbf{z}(\mathbf{x})$ can be obtained by randomly sampling $\{\mathbf{v}_1, \ldots, \mathbf{v}_D\}$ from $\pi_\kappa(\mathbf{v})$, that is

$$\begin{aligned} &\mathbf{z}_{\mathbf{V}}(\mathbf{x}) := \\ &\sqrt{\frac{1}{D}}[\sin(\mathbf{v}_1^\top \mathbf{x}), \cos(\mathbf{v}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{v}_D^\top \mathbf{x}), \cos(\mathbf{v}_D^\top \mathbf{x})] \end{aligned} \tag{9}$$

where the entries of $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_D]^\top \in \mathbb{R}^{D \times d}$ are i.i.d. Gaussian. Thanks to (5), the nonparametric learning task is then approximated as a linear learning task in the Fourier feature space. Specifically, with the loss function [cf. (6)]

$$\ell_t(f(\mathbf{x}_t)) := \ell(f(\mathbf{x}_t), y_t) = \ell(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \tag{10}$$

the online learning task becomes $\min_{\boldsymbol{\theta} \in \mathbb{R}^{2D}} \sum_{t=1}^T \ell(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$. Upon obtaining a new datum $\mathbf{x}_t$, the representations of the data instance $\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)$ can be generated via (9), and online gradient descent can be applied to refine the estimator 'on the fly,' i.e.,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla \ell(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \tag{11}$$

where $\{\eta_t\}$ denotes the sequence of stepsizes, and $\nabla\ell(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$ the gradient with respect to the weight $\boldsymbol{\theta}$ at $\boldsymbol{\theta} = \boldsymbol{\theta}_t$. The update (11) is still *a functional update* in that it is tantamount to updating the function $f_t(\cdot) = \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\cdot)$, but the neat thing here is that this function is in the span of $\{\mathbf{z}_{\mathbf{V}}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}\}$.

Variance-reduced RF. Even if $\mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_1)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_2)$ is an unbiased estimator of κ [cf. (8)], the variance $\mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_1)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_2)$ decays as D increases. This explains why the RF vector dimension is chosen to satisfy $D \gg d$. [15]. Next we will leverage a recent intriguing discovery in [16] to markedly reduce the variance of RF approximation by enforcing orthogonality on the rows of $\mathbf{V}$. For the original RF approach on a Gaussian kernel with bandwidth σ^2, recall that $\mathbf{V} = \sigma^{-1}\mathbf{G}$ in (9), where each entry of $\mathbf{G}$ follows standardized Gaussian pdf. For the variance-reduced RF method, with $D = d$, $\mathbf{V}_{\text{ORF}}$ is formed as

$$\mathbf{V}_{\text{ORF}} = \frac{1}{\sigma}\mathbf{S}\mathbf{Q} \tag{12}$$

where $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is a uniformly distributed random orthonormal matrix, and $\mathbf{S}$ denotes a diagonal matrix with diagonal entries drawn i.i.d. from χ distribution with d degrees of freedom. Matrix $\mathbf{S}$ is introduced to ensure unbiasedness of the kernel approximation [16]. With $D > d$, several weighted orthonormal matrices can be generated independently from (12), and concatenated to form $\mathbf{V}_{\text{ORF}}$. It turns out that $\mathbf{z}_{\mathbf{V}_{\text{ORF}}}^\top \mathbf{z}_{\mathbf{V}_{\text{ORF}}}$ with $\mathbf{z}_{\mathbf{V}_{\text{ORF}}}$ generated as in (9), with $\mathbf{V}_{\text{ORF}}$ replacing $\mathbf{V}$, has variance smaller than σ^2[16]. Through such orthogonal-promoting RF generation, the number of random features needed to achieve certain accuracy can be markedly reduced.

The RF-based online kernel learning scheme in this section presumes that κ is known a priori. Since this is not generally possible, it is more prudent to adaptively select kernels from a dictionary with a set of kernel functions, for which the online ensemble learning scheme in the next section will play a key role.

2.2 Online MKL in stationary settings

Here we preselect a dictionary of possible kernel functions, and then adaptively combine kernels in the dictionary [3]. Specifically, given a dictionary of kernels $\{\kappa_p\}_{p=1}^P$ and the RKHS $\mathcal{H}_p$ induced by κ_p, the solution of (2) is expressible in a separable form as [27]

$$f_t(\mathbf{x}) := \sum_{p=1}^{P} \bar{w}_{p,t} f_{p,t}(\mathbf{x}) \tag{13}$$

where $f_{p,t}(\mathbf{x})$ belongs to RKHS $\mathcal{H}_p$, for $p = 1, \ldots, P$, and $\bar{w}_{p,t} \in [0,1]$ denotes the normalized weight for the pth kernel-based function estimator at time slot t.

To make use of the dictionary, $\{\tilde{w}_{p,t}\}_{p=1}^P$ should be learnt and adjusted in an online fashion. This task fits well the celebrated online learning paradigm, a.k.a., online prediction with expert advice [28, 29]. Specifically, treating $\{\tilde{w}_{p,t}\}$ as an expert, we formulate the multi-kernel based learning problem as an online prediction task with expert advice. Upon obtaining a data sample, the un-normalized weights are updated according to the loss incurred by each learner as

$$w_{p,t+1} = w_{p,t} \exp(-\eta \ell_t(f_{p,t}(\mathbf{x}_t))) \tag{14}$$

where $\eta \in (0,1)$ is a chosen constant that controls the adaptation rate of $\{w_{p,t}\}$. Relative to $\{w_{p,t}\}$, the normalized weights in (13) $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^P w_{p,t}$, $\forall t$. Moreover, it can be observed that when $f_{p,t}$ incurs larger relative to other $f_{p',t}$ with $p' \neq p$ loss at time slot t, the corresponding combination weight decreases in the next time slot. In other words, a more accurate learner tends to play more important role in predicting the upcoming data.

However, relative to the generic expert advice problem [28, 29], the difference here is that the kernel-based function estimator itself performs efficient online learning scheme for self-improvement. Indeed, the random feature approximation in Section 2.1 enables the efficient and scalable self-learning for each kernel-specific expert. Specifically, for the expert associated with kernel p, a feature representation $\mathbf{z}_p(\mathbf{x}_t)$ will be randomly generated from a kernel-specific distribution given datum $\mathbf{x}_t$ (cf. (9)), where we use $\mathbf{z}_p(\mathbf{x}_t) = \mathbf{z}_{\mathbf{V}_p}(\mathbf{x}_t)$ for notational simplicity, and each function estimator at time t can be written as

$$f_{p,t}(\mathbf{x}_t) \simeq \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t) \tag{15}$$

where $\boldsymbol{\theta}_{p,t}$ is the parameter in (6) at time t for kernel p. Similar to (11), the pth kernel, $\boldsymbol{\theta}_{p,t}$ is updated via

$$\boldsymbol{\theta}_{p,t+1} = \boldsymbol{\theta}_{p,t} - \eta\nabla\ell(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) \tag{16}$$

where we use $\ell(f_{p,t}(\mathbf{x}_t), y_t) = \ell(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)$. The Raker algorithm is summarized in Algorithm 1.

Complexity. At the t-th iteration of Algorithm 1, the memory required is fixed and of order $\mathcal{O}(D)$. Regarding computational overhead, the per-iteration computational complexity is of order $\mathcal{O}(D)$ compared with at least $\mathcal{O}(dt)$ for OMKL [22], hence the proposed algorithm is more computationally efficient than conventional MKL [4] or OMKL. This explains the effectiveness of the novel Raker algorithm.

2.3 Static regret analysis

To analyze the performance of Raker, we assume that the following conditions are satisfied.

Algorithm 1 Raker for stationary settings

```
1: Input: Kernels $\kappa_p,\ p=1,\ldots,P$, step size $\eta>0$, and number of random features $D$.
2: Initialization: $\boldsymbol{\theta}_1=\mathbf{0}$.
3: for $t=1,2,\ldots,T$ do
4:     Receive a streaming datum $\mathbf{x}_t$.
5:     Construct $\mathbf{z}_p(\mathbf{x}_t)$ via (9) for $p=1,\ldots,P$.
6:     Predict $f_t(\mathbf{x}_t)$ via (13) with $f_{p,t}(\mathbf{x}_t)$ in (15).
7:     for $p=1,\ldots,P$ do
8:         Obtain loss $\ell(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)$.
9:         Update $w_{p,t+1}$ via (14).
10:        Update $\boldsymbol{\theta}_{p,t+1}$ via (16).
11:    end for
12: end for
```

Assumption 1 *For every slot t, the loss function $\ell(\boldsymbol{\theta}^\top\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t),y_t)$ in (10) is convex w.r.t. $\boldsymbol{\theta}$.*

Assumption 2 *For $\boldsymbol{\theta}$ belonging to a bounded set $\boldsymbol{\Theta}$, the loss is bounded; i.e., $\ell(\boldsymbol{\theta}^\top\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t),y_t)\in[-1,1]$, and has bounded gradient; i.e., $\|\nabla\ell(\boldsymbol{\theta}^\top\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t),y_t)\|\leq L$.*

Assumption 3 *Each κ_p is a shift-invariant kernel with bounded entry, i.e., $\kappa_p(\mathbf{x}_i,\mathbf{x}_j)\leq 1,\ \forall\mathbf{x}_i,\mathbf{x}_j$. Also $\|\mathbf{x}\|\leq 1$ and $\|f^*_{\mathcal{H}_p}\|_1:=\sum_{t=1}^T|\alpha_t^*|\leq C$.*

Assumption 1 enforces convexity of the loss, which is standard in online convex optimization (OCO) [25]. Assumption 2 ensures the losses are bounded, and the gradient of the loss function are bounded, which is also called L-Lipschitz continuity that is also common in OCO [30]. Assumption 3 bounds the norm of the optimal function [23]. When the boundedness of the losses is common given the bounded datum $\mathbf{x}_t$, the Lipschitz continuity is also not restrictive. Considering the kernel-based ridge regression as an example, the gradient w.r.t. $\boldsymbol{\theta}$ is $(\boldsymbol{\theta}^\top\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)-y_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)$. Since the loss function is bounded, e.g., $\|\boldsymbol{\theta}^\top\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)-y_t\|\leq 1$, and the random feature in (9) can be bounded by $\|\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)\|\leq 1$, thus the constant L can be bounded by $L\leq 1$ using the Cauchy-Schwartz inequality. In general, Assumptions 1-3 are common in kernel-based learning tasks [4, 27, 23].

With regard to performance of an online algorithm, static regret is commonly adopted as a metric by most OCO schemes, and measures the difference between the aggregate loss of an OCO algorithm and that of the best fixed solution in hindsight [25, 30]. Specifically, for the sequence of online functions $\{f_t\}$ generated by a kernel learning algorithm $\mathcal{A}$, its static regret is

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(T):=\sum_{t=1}^T\ell_t(f_t(\mathbf{x}_t))-\sum_{t=1}^T\ell_t(f^*(\mathbf{x}_t)) \quad (17)$$

where the best static function estimator $f^*(\cdot)$ is obtained through the following batch optimization

$$f^*(\cdot)\in\arg\min_{f\in\mathcal{F}}\sum_{t=1}^T\ell_t(f(\mathbf{x}_t)) \quad (18)$$

where the function space is $\mathcal{F}:=\bigcup_{p\in\mathcal{P}}\mathcal{H}_p$ by default, with $\mathcal{H}_p$ representing the RKHS induced by κ_p. With this definition in hand, we first establish the static regret of our Raker approach in the following lemma.

Lemma 1 *Consider Assumptions 1 and 2 are satisfied, and $f_p^*(\cdot)$ denotes the best static solution in (18) with $\mathcal{F}_p:=\{f_p|f_p(\mathbf{x})=\boldsymbol{\theta}^\top\mathbf{z}_p(\mathbf{x}),\forall\boldsymbol{\theta}\in\mathbb{R}^{2D}\}$. For the sequences $\{f_{p,t}\}$ and $\{\bar{w}_{p,t}\}$ generated by the Raker algorithm, the following bound holds*

$$\sum_{t=1}^T\ell_t\Bigg(\sum_{p=1}^P\bar{w}_{p,t}f_{p,t}(\mathbf{x}_t)\Bigg)-\sum_{t=1}^T\ell_t(f_p^*(\mathbf{x}_t))\leq\frac{\ln P}{\eta}+\frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta}+\frac{\eta L^2T}{2}+\eta T \quad (19)$$

where $\boldsymbol{\theta}_p^*$ is formed by the parameters of the best function estimator in $\mathcal{F}_p$, i.e., $f_p^*(\mathbf{x})=\left(\boldsymbol{\theta}_p^*\right)^\top\mathbf{z}_p(\mathbf{x})$.

In addition to the static regret bound of the Raker algorithm, the next theorem characterizes the difference between the loss of online MKL algorithm relative to the best functional estimator in the original RKHS.

Theorem 2 *In addition to the conditions of Lemma 1, consider that Assumption 3 is also satisfied. If $f^*_{\mathcal{H}_p}$ is the best static function estimator in (18) belonging to the RKHS $\mathcal{H}_p$, with probability at least $1-2^8\left(\frac{\sigma_p}{\epsilon}\right)^2\exp\left(\frac{-D\epsilon^2}{4d+8}\right)$, the following bound holds*

$$\sum_{t=1}^T\ell_t\left(\sum_{p=1}^P\bar{w}_{p,t}f_{p,t}(\mathbf{x}_t)\right)-\min_{p\in\{1,\ldots,P\}}\sum_{t=1}^T\ell_t\left(f^*_{\mathcal{H}_p}(\mathbf{x}_t)\right)\leq\frac{\ln P}{\eta}+\frac{(1+\epsilon)\|f^*_{\mathcal{H}_p}\|_1^2}{2\eta}+\frac{\eta L^2T}{2}+\eta T+\epsilon LT\|f^*_{\mathcal{H}_p}\|_1 \quad (20)$$

where $\epsilon>0$ is a constant, d is the original feature dimension, and D is the number of random features, while $\sigma_p^2:=\mathbb{E}_{\mathbf{V}}^{\pi_{\kappa_p}}[\mathbf{v}^\top\mathbf{v}]$ is the second moment of random features. Setting $\eta=\epsilon=\mathcal{O}(1/\sqrt{T})$ leads to

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{Raker}}(T)=\mathcal{O}(\sqrt{T}) \quad (21)$$

where the benchmark is from the RKHS $\bigcup_{p\in\mathcal{P}}\mathcal{H}_p$.

Observe that the probability in (20) gets larger as D increases. For a given ϵ, one can always find an appropriate D to ensure a positive probability. Hence, for subsequent discussions, we only use "with high probability" (w.h.p.) to simplify the exposition. Theorem 2 demonstrates that with appropriate choice of parameters, the novel Raker algorithm achieves sub-linear regret relative to the best static functional estimation in the original function space $\bigcup_{p\in\mathcal{P}}\mathcal{H}_p$.

3 Online Ensemble MKL in Non-stationary Environments

The Racker algorithm in Section 2 is capable of combining different kernel learners 'on the fly.' Targeting an optimal scheme in dynamic environments, an **ada**ptive **Raker** approach (termed **AdaRaker**) will be developed in this section.

3.1 An ensemble online MKL approach

To improve performance of the online MKL algorithm, the choice of learning rate η in (14) and (16) can be crucial. Especially in dynamic environments, a large learning rate helps better track the variation of the optimal function estimator, while a smaller one allows the algorithm to tune the unknown parameters in a fine-grained manner. As a result, the optimal choice of η_t explicitly depends on the variability of the optimal function estimator [7, 12]. However, it is often impractical to choose a sequence of optimal stepsizes $\{\eta_t\}$, whenever the variability of underlying environments is unknown a priori. In this context, we develop an adaptive Raker method next.

Akin to combining multiple RF-based kernel predictors in Section 2, the idea here is to hedge between multiple Raker learners with different learning rates. We consider each Raker instance in Algorithm 1 as a black box algorithm $\mathcal{A}_I$, where the subscript I represents the algorithm running on interval $I := [\underline{I}, \bar{I}]$ starting from slot $\underline{I}$ to slot $\bar{I}$. Let a pre-selected set $\mathcal{I}$ collect all these intervals, the design of which will be specified later. At the beginning of each interval $I \in \mathcal{I}$, a new instance of online Raker algorithm $\mathcal{A}_I$ is initialized with an interval-specific learning rate $\eta^{(I)} := \min\{1/2, \eta_0/\sqrt{|I|}\}$ with constant $\eta_0 > 0$. Due to the possible overlapping between intervals, multiple Raker algorithms $\mathcal{A}_I$ will be run in parallel. In this case, the set $\mathcal{I}(t)$ collects all the active intervals at the current slot t, given by

$$\mathcal{I}(t) := \{I \in \mathcal{I} \,|\, t \in [\underline{I}, \bar{I}]\},\ \forall t \in \mathcal{T}. \tag{22}$$

For each Raker instance $\mathcal{A}_I$ with $I \in \mathcal{I}(t)$, let $f_t^{(I)}(\cdot)$ denote its output at time t that is a combination of multiple kernel-based function estimators, and let $\ell_t(f_t^{(I)}(\mathbf{x}_t))$ represent its instantaneous loss. The output of the ensemble learner $\mathcal{A}$ at time t is the weighted combination of all the learners' outputs $\{f_t^{(I)}, \forall I \in \mathcal{I}(t)\}$. With $h_t^{(I)}$ denoting the weight of the Raker instance $\mathcal{A}_I$, we will update it online via

$$h_{t+1}^{(I)} = \begin{cases} 0, & \text{if } t \notin I \\ \eta^{(I)}, & \text{if } t = \underline{I} \\ h_t^{(I)} \exp\left(-\eta^{(I)} r_t^{(I)}\right), & \text{else} \end{cases} \tag{23}$$

where $\underline{I}$ is the first time slot of interval I, and the loss of $\mathcal{A}_I$ *relative* to the overall loss is

$$r_t^{(I)} = \ell_t(f_t(\mathbf{x}_t)) - \ell_t(f_t^{(I)}(\mathbf{x}_t)),\ \forall I \in \mathcal{I}(t). \tag{24}$$

Intuitively, one would wish to decrease (increase) the weights of those instances with small (large) losses in future rounds. Using update (23), and defining the normalized weight as $\bar{h}_t^{(I)} = h_t^{(I)}/\sum_{J\in\mathcal{I}(t)} h_t^{(J)}$, the overall output is given by

$$f_t(\mathbf{x}) = \sum_{I\in\mathcal{I}} \bar{h}_t^{(I)} f_t^{(I)}(\mathbf{x}) = \sum_{I\in\mathcal{I}} \bar{h}_t^{(I)} \sum_{p\in\mathcal{P}} \bar{w}_{p,t}^{(I)} f_{p,t}^{(I)}(\mathbf{x}) \tag{25}$$

where $\{\bar{w}_{p,t}^{(I)}\}$ are the weights of the kernel combination generated by learner $\mathcal{A}_I$ (cf. (13)). The AdaRaker scheme is summarized in Algorithm 2.

Selecting judiciously variable-length intervals in $\mathcal{I}$ can affect performance critically. The criterion for achieving interval regret has been discussed in [31]. Instead, our pursuit is an ensemble MKL method in environments with unknown dynamics using scalable RF-based function approximants. When each interval is long, the Raker algorithm works well on each interval if the loss function is slow-varying, but incurs a higher loss when the objective experiences a rapid change. On the other hand, a short interval can hedge against a possibly rapid change, but its performance on each interval could be degraded. A simple yet efficient interval partitioning scheme follows.

Illustration of interval sets: *One example is to partition the entire horizon into intervals with length $2^0, 2^1, 2^2, \ldots$. Intervals of length 2^j with a given $j \in \mathbb{N}$ are consecutively assigned without overlapping starting from $t = 2^j$. In this case, define a set of intervals $\mathcal{I}_j = [\underline{I}_j, \bar{I}_j]$ such that the length of the intervals $|\mathcal{I}_j| = \bar{I}_j - \underline{I}_j + 1 = 2^j$, $j \in \mathbb{N}$. Therefore, each time slot t is covered by a set of at most $\lceil \log_2 t \rceil$ intervals, which forms the active set of intervals $\mathcal{I}(t)$ at time t.*

3.2 Regret analysis in dynamic environments

Section 2.3 implicitly assumes that the optimal function estimator does not change with time. In non-stationary environments however, the optimal function may change over time. The focus here is to provide theoretical justification of AdaRaker in those settings.

Algorithm 2 AdaRaker for non-stationary settings

```
1: Initialization: learner weights $\{h_1^{(I)}\}$, and their learning rates $\{\eta^{(I)}\}$.
2: for $t = 1, 2, \ldots, T$ do
3:     Obtain $f_t^{(I)}(\mathbf{x}_t)$ from each MKL $\mathcal{A}_I$, $I \in \mathcal{I}(t)$.
4:     Predict $f_t(\mathbf{x}_t)$ via Raker combination (25).
5:     Observe function $\ell_t$, and incur $\ell_t(f_t(\mathbf{x}_t))$.
6:     for $I \in \mathcal{I}(t)$ do
7:         Incur loss $\ell_t(f_t^{(I)}(\mathbf{x}_t))$ and update $f_t^{(I)}(\cdot)$.
8:         Update weights $h_{t+1}^{(I)}$ via (23).
9:     end for
10: end for
```

In response to the quest for improved benchmarks, we consider *dynamic regret* in this context. The notion of dynamic regret (a.k.a. tracking regret) has been introduced in [12, 31, 32] to offer a competitive performance measure of online algorithms. It is defined as

$$\mathrm{Reg}^{\mathrm{d}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \ell_t(f_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \ell_t(f_t^*(\mathbf{x}_t)) \tag{26}$$

where the benchmark is formed via a sequence of best dynamic solutions $\{f_t^*\}$ for the online problem, i.e.,

$$f_t^*(\cdot) \in \arg\min_{f\in\mathcal{F}} \ell_t(f(\mathbf{x}_t)) \tag{27}$$

where $\mathcal{F} := \bigcup_{p\in\mathcal{P}} \mathcal{H}_p$ by default, with $\mathcal{H}_p$ representing the RKHS induced by κ_p. It follows from the definitions (18) and (27) that the dynamic regret is always larger than the static regret in (17). Therefore, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa. Given a sequence of loss functions $\{\ell_t\}$, the goal is to generate a sequence of functions $\{f_t\}$ that minimize the dynamic regret. Before analyzing the dynamic regret of AdaRaker, we first introduce an *intermediate* result regarding the static regret on any sub-interval $I \subseteq \mathcal{T}$.

Lemma 3 *Suppose Assumptions 1-3 are satisfied, and define the static regret on any interval $I \subseteq \mathcal{T}$ as*

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(|I|) := \sum_{t\in I} \ell_t(f_t(\mathbf{x}_t)) - \sum_{t\in I} \ell_t(f_I^*(\mathbf{x}_t)) \tag{28}$$

where $|I|$ denotes the length of interval I, and the fixed solution is $f_I^* \in \arg\min_{f\in\bigcup_{p\in\mathcal{P}}\mathcal{F}_p} \sum_{t\in I} \ell_t(f(\mathbf{x}_t))$, with $\mathcal{F}_p := \{f_p | f_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}), \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$. Then for any interval $I \subseteq \mathcal{T}$, the following bound holds

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) \le C_0\sqrt{|I|} + C_1 \ln T \sqrt{|I|} \tag{29}$$

where C_0 and C_1 are fixed positive constants.

Lemma 3 implies that by combining Raker learners with different learning rates, AdaRaker can achieve sub-linear static regret at *any* interval I with arbitrary interval length. This also holds for intervals overlapping with multiple intervals. Clearly, the best fixed solution in (28) is interval specific, which can vary over different intervals. Therefore, the functions generated by AdaRaker can compete with a *time-varying* comparator. Indeed, this intuition will become concrete in the next theorem, where the dynamic regret is established for our AdaRaker approach.

Theorem 4 *Suppose Assumptions 1-3 are satisfied, and define the accumulated variation of losses as*

$$\mathbb{V}(\{\ell_t\}_{t=1}^T) := \sum_{t=1}^{T} \max_{f\in\mathcal{F}} \left|\ell_{t+1}(f(\mathbf{x}_{t+1})) - \ell_t(f(\mathbf{x}_t))\right| \tag{30}$$

where the function space is $\mathcal{F} := \bigcup_{p\in\mathcal{P}} \mathcal{H}_p$. Then AdaRaker yields a dynamic regret in (26) bounded by

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) \le& (2 + C_0 + C_1 \ln T) T^{\frac{2}{3}} \mathbb{V}^{\frac{1}{3}}(\{\ell_t\}_{t=1}^T)\\ =& \tilde{\mathcal{O}}\left(T^{\frac{2}{3}} \mathbb{V}^{\frac{1}{3}}(\{\ell_t\}_{t=1}^T)\right), \text{ w.h.p.}\end{aligned} \tag{31}$$

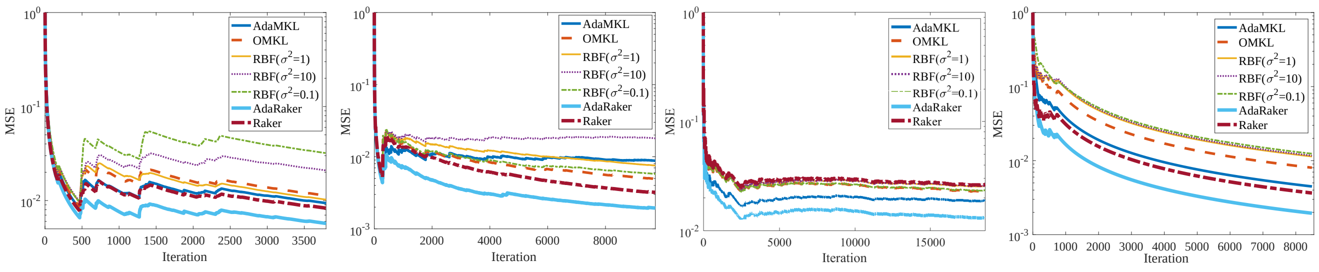
where C_0 and C_1 are universal constants, and $\tilde{\mathcal{O}}$ neglects the terms with a polynomial $\log T$ *rate.*

Theorem 4 asserts that the AdaRaker's dynamic regret depends on the variation of loss functions in (30) and the horizon T. Interesting enough, whenever the loss functions *do not vary on average*, i.e., $\mathbb{V}(\{\ell_t\}_{t=1}^T) = \mathbf{o}(T)$, our AdaRaker approach is able to achieve sub-linear dynamic regret.

4 Experiments

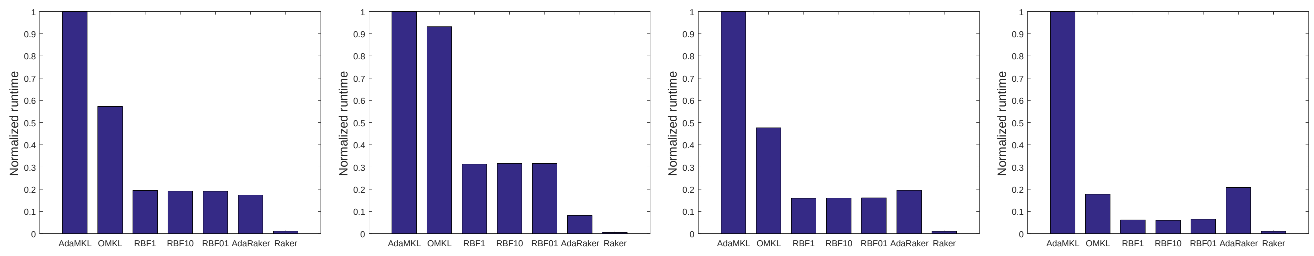
The present section tests the performance of our novel algorithms for online regression tasks. We compared Raker and AdaRaker with online multi-kernel learning algorithm (OMKL) [22] and its adaptive version that we term AdaMKL, and online (single) kernel based learning using Gaussian kernels (RBF) with bandwidth $\sigma^2 = \{0.1, 1, 10\}$. Note that AdaMKL has not been proposed in literature, but we add it only for comparison purposes. All the considered MKL approaches use Gaussian kernels with $\sigma^2 = \{0.1, 1, 10\}$, step-sizes of the single kernel based learning algorithms are set to $\eta = 1/\sqrt{T}$ for all algorithms, and $\eta = 0.5$ and $\lambda = 0.01$ for all MKL approaches. Entries of $\{\mathbf{x}_t\}$ and $\{y_t\}$ are normalized to lie in $[0, 1]$. For RF-based approaches, $D = 50$ *orthogonal random features* were used.

Datasets. Performance is tested on several benchmark datasets [33]. The twitter dataset consists of time series of $T = 6,000$ samples with $\mathbf{x}_t \in \mathbb{R}^{77}$ and



(a) (b) (c) (d)

Figure 1: MSE performance: a) Twitter; b) Tom's hardware; c) energy efficiency; and, d) air quality.



(a) (b) (c) (d)

Figure 2: Normalized CPU time: a) Twitter; b) Tom's hardware; c) energy efficiency; and, d) air quality.

the Tom's hardware dataset contains $T = 10,000$ feature vectors each of size 96, while y_t represents the average number of active discussion about a certain topic on twitter and Tom's hardware [34]. The energy dataset consists of $T = 10,000$ data, with $\mathbf{x}_t \in \mathbb{R}^{27}$ describes the humidity and temperature in and outside the rooms, and y_t denotes the energy use of light fixtures in the house [35]. The air quality dataset collects the sensor responses as $\{\mathbf{x}_t \in \mathbb{R}^{13}\}_{t=1}^{9,358}$, to predict the concentration of polluting chemicals y_t.

Performance. The performance of different algorithms is plotted in Figure 1 in terms of the mean-square error $\text{MSE}(t) := (1/t)\sum_{\tau=1}^{t}(y_\tau - \hat{y}_\tau)^2$. Clearly, Raker achieves competitive performance, and its adaptive variant AdaRaker consistently yields the lowest MSE, especially when abrupt changes occur; e.g., when the underlying models are changing. This observation meets our intuition of designing AdaRaker methods. Aligned with the motivation of using multiple kernels, all MKL methods outperform the algorithms using only a single kernel. The normalized CPU time of all the considered schemes is depicted in Figure 2. A sharp observation is that our RF-based MKL methods including Raker and AdaRaker are computationally more efficient than other online (multi-)kernel methods. Running multiple instances of Raker in parallel, the complexity of AdaRaker is higher than Raker, but its runtime is still only around 20% of that of AdaMKL, and similar to single-kernel alternatives.

5 Conclusions

We considered the multi-kernel learning problem in non-stationary and adversarial environments. Leveraging recent advances in *variance-reduced* random feature approximation, we developed a scalable online *multi-kernel* learning approach that we term Raker. Endowing Raker with capability of tracking time-varying optimal functions, we proposed AdaRaker that is an ensemble version of Raker with variable learning rates. Careful design allows AdaRacker to adapt learning rates to non-stationary and possibly adversarial environments. Rigorous analysis demonstrates that without a-priori knowledge of environments, AdaRaker achieves sub-linear dynamic regret, provided that either the loss function or the optimal function solution does not change on average. To our best knowledge, AdaRaker is the *first* algorithm that optimally tracks nonlinear functions in non-stationary settings with theoretical guarantees. Experiments on real datasets validate the effectiveness of our methods.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant 1500713 and 1711471, and NIH 1R01GM104975-01. Tianyi Chen is also supported by the Doctoral Dissertation Fellowship from the University of Minnesota.

References

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

[2] B. Dai, N. He, Y. Pan, B. Boots, and L. Song, "Learning from conditional distributions via dual embeddings," in *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, Apr. 2017, pp. 1458–1467.

[3] C. Cortes, M. Mohri, and A. Rostamizadeh, "ℓ_2-regularization for learning kernels," in *Proc. Conf. on Uncertainty in Artificial Intelligence*, Montreal, Canada, Jun. 2009, pp. 109–116.

[4] J. A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via sparse kernel-based learning: A unifying view with advances in blind methods," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 112–125, Jul. 2013.

[5] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. Intl. Conf. Mach. Learn.*, Montreal, Canada, Jun. 2009, pp. 681–688.

[6] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Sig. Proc.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[7] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Sig. Proc.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[8] S. C. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Machine Learning*, vol. 90, no. 2, pp. 289–316, Feb. 2013.

[9] G. Wahba, *Spline Models for Observational Data*. Philadelphia, PA: SIAM, 1990.

[10] Y. Shen, B. Baingana, and G. B. Giannakis, "Nonlinear structural vector autoregressive models for inferring effective brain network connectivity," 2016. [Online]. Available: https://arxiv.org/abs/1610.06551

[11] ——, "Kernel-based structural equation models for topology identification of directed networks," *IEEE Trans. Sig. Proc.*, vol. 65, no. 10, pp. 2503–2516, May 2017.

[12] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations Research*, vol. 63, no. 5, pp. 1227–1244, Sep. 2015.

[13] C. K. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Advances in Neural Info. Process. Syst.*, Vancouver, Canada, Dec. 2001, pp. 682–688.

[14] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, Sardinia, Italy, May 2010, pp. 113–120.

[15] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Advances in Neural Info. Process. Syst.*, Vancouver, Canada, Dec. 2007, pp. 1177–1184.

[16] X. Y. Felix, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar, "Orthogonal random features," in *Proc. Advances in Neural Info. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 1975–1983.

[17] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Machine Learning Res.*, vol. 5, pp. 27–72, Jan. 2004.

[18] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a budget," *SIAM J. Computing*, vol. 37, no. 5, pp. 1342–1372, Jan. 2008.

[19] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *J. Machine Learning Res.*, vol. 13, pp. 3103–3131, Oct. 2012.

[20] R. Jin, S. C. Hoi, and T. Yang, "Online multiple kernel learning: Algorithms and mistake bounds," in *Proc. Intl. Conf. on Algorithmic Learning Theory*, Canberra, Australia, Oct. 2010, pp. 390–404.

[21] D. Sahoo, S. C. Hoi, and P. Zhao, "Cost sensitive online multiple kernel classification," in *Proc. Asian Conf. Machine Learning*, Hamilton, New Zealand, Nov. 2016, pp. 65–80.

[22] D. Sahoo, S. C. Hoi, and B. Li, "Online multiple kernel regression," in *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, New York, USA, Aug. 2014, pp. 293–302.

[23] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *J. Machine Learning Res.*, vol. 17, no. 47, pp. 1–43, Apr. 2016.

[24] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random fourier features," *arXiv preprint:1703.08131*, Mar. 2017.

[25] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. and Trends in Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[26] K. Crammer, J. Kandola, and Y. Singer, "Online classification on a budget," in *Proc. Advances in Neural Info. Process. Syst.*, 2004, pp. 225–232.

[27] C. A. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Machine Learning Res.*, vol. 6, pp. 1099–1125, Jul. 2005.

[28] V. G. Vovk, "A game of prediction with expert advice," in *Proc. Annual Conf. Computational Learning Theory*, Santa Cruz, CA, Jul. 1995, pp. 51–60.

[29] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* Cambridge, United Kingdom: Cambridge University Press, 2006.

[30] E. Hazan, "Introduction to online convex optimization," *Found. and Trends in Mach. Learn.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[31] A. Daniely, A. Gonen, and S. Shalev-Shwartz, "Strongly adaptive online learning." in *Proc. Intl. Conf. on Machine Learning*, Lille, France, Jun. 2015, pp. 1405–1411.

[32] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Intl. Conf. Artificial Intell. and Stat.*, San Diego, CA, May 2015.

[33] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[34] F. Kawala, A. Douzal-Chouakria, E. Gaussier, and E. Dimert, "Prédictions d'activité dans les réseaux sociaux en ligne," in *4ième Conférence sur les Modèles et l'Analyse des Réseaux: Approches Mathématiques et Informatiques*, 2013.

[35] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy and Buildings*, vol. 140, pp. 81–97, 2017.

[36] H. Luo, A. Agarwal, and J. Langford, "Efficient contextual bandits in non-stationary worlds," *arXiv preprint:1708.01799*, Aug. 2017.