Journal of Machine Learning Research 19 (2018) 1-36 Submitted 1/18; Revised 11/18; Published 12/18

Random Feature-based Online Multi-kernel Learning in Environments with Unknown Dynamics

Yanning Shen SHENX513@UMN.EDU
Tianyi Chen CHEN3827@UMN.EDU
Georgios B. Giannakis GEORGIOS@UMN.EDU
Department of Electrical and Computer Engineering, University of Minnesota
Minneapolis, MN, 55455, USA

Editor: Karsten Borgwardt

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. Especially when the latter is not available, 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 a scalable multi-kernel learning scheme (termed Raker) to obtain the sought nonlinear learning function 'on the fly,' first for static environments. To further boost performance in dynamic environments, an adaptive multi-kernel learning scheme (termed AdaRaker) is developed. AdaRaker accounts not only for data-driven learning of kernel combination, but also for the unknown dynamics. Performance is analyzed in terms of both static and dynamic regrets. AdaRaker is uniquely capable of tracking nonlinear learning functions in environments with unknown dynamics, and with with analytic performance guarantees. Tests with synthetic and real datasets are carried out to showcase the effectiveness of the novel algorithms.[1]

Keywords: Online learning, reproducing kernel Hilbert space, multi-kernel learning, random features, dynamic and adversarial environments.

1. Introduction

Function approximation emerges in various learning tasks such as regression, classification, clustering, dimensionality reduction, as well as reinforcement learning (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Dai et al., 2017). Among them, the emphasis here is placed on supervised functional learning tasks: 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 goal is to find a function $f(\cdot)$ such that the discrepancy between each pair of y_t and $f(\mathbf{x}_t)$ is minimized. Typically, such discrepancy is measured by a cost function $\mathcal{C}(f(\mathbf{x}_t), y_t)$, which requires to find $f(\cdot)$ minimizing $\sum_{t=1}^T \mathcal{C}(f(\mathbf{x}_t), y_t)$. While this goal is too ambitious to achieve in general, the problem becomes tractable when $f(\cdot)$ is assumed to belong to a reproducing kernel Hilbert space (RKHS) induced by a kernel (Schölkopf and Smola, 2002). Comparable to deep neural networks, functions defined

1. Preliminary results in this paper were presented in part at the 2018 International Conference on Artificial Intelligence and Statistics (Shen et al., 2018).

in RKHS can model highly nonlinear relationship, and thus kernel-based methods have well-documented merits for principled function approximation. Despite their popularity, most kernel methods rely on a single pre-selected kernel. Yet, multi-kernel learning (MKL) is more powerful, thanks to its data-driven kernel selection from a given dictionary; see e.g., (Shawe-Taylor and Cristianini, 2004; Rakotomamonjy et al., 2008; Cortes et al., 2009; Gönen and Alpaydın, 2011), and (Bazerque and Giannakis, 2013).

In addition to the attractive representation power that can be afforded by kernel methods, several learning tasks are also expected to be performed in an online fashion. Such a need naturally arises when the data arrive sequentially, such as those in online spam detection (Ma et al., 2009), and time series prediction (Richard et al., 2009); or, when the sheer volume of data makes it impossible to carry out data analytics in batch form (Kivinen et al., 2004). 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 environments with *unknown dynamics*. This is the case when the function of interest e.g., represents the state in brain graphs, or, captures the temporal processes propagating over time-varying networks. Especially when variations are due to adversarial interventions, the underlying dynamics are unknown. Online kernel-based learning in such environments remains a largely uncharted territory (Kivinen et al., 2004; Hoi et al., 2013).

In accordance with these needs and desiderata, the *goal* of this paper is an algorithmic pursuit of scalable online MKL in environments with unknown dynamics, along with their associated performance guarantees. Major challenges come from two sources: 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 matrices to grow quadratically with the number of data (Wahba, 1990), thus the computational complexity to find even a *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 (Shen et al., 2016, 2017; Shen and Giannakis, 2018), which makes kernel-based methods less scalable relative to their linear counterparts. For ii), most online learning settings presume time invariance or slow dynamics, where an algorithm achieving sub-linear regret incurs on average "no-regret" relative to the *best static* benchmark. Clearly, designing online schemes that are comparable to the *best dynamic* solution is appealing though formidably challenging without knowledge of the dynamics (Kivinen et al., 2004).

1.1 Related works

To put our work in context, we review prior art from the following two aspects.

Batch kernel methods. Kernel methods are known to suffer from the growing dimensionality in large-scale learning tasks (Shawe-Taylor and Cristianini, 2004). Major efforts have been devoted to scaling up kernel methods in batch settings. Those include approaches to approximating the kernel matrix using low-rank factorizations (Williams and Seeger, 2001; Sheikholeslami et al., 2018), whose performance was analyzed in (Cortes et al., 2010). Recently, random feature (RF) based function estimators have gained popularity since the work of (Rahimi and Recht, 2007) and (Dai et al., 2014), whose variance has been con-

siderably reduced through an *orthogonality promoting* RF modification (Yu et al., 2016). These approaches assume that the kernel is known, a choice crucially dependent on domain knowledge. Enabling kernel selection, several MKL-based approaches have emerged, see e.g., (Lanckriet et al., 2004; Rakotomamonjy et al., 2008; Bach, 2008; Cortes et al., 2009; Gönen and Alpaydın, 2011), and their performance gain has been documented relative to their single kernel counterparts. However, the aforementioned methods are designed for batch settings, and are either intractable or become less efficient in online setups. When the sought functions vary over time and especially when the dynamics are unknown (as in adversarial settings), batch schemes fall short in tracking the optimal function estimators.

Online (multi-)kernel learning. Tailored for streaming large-scale datasets, online kernel-based learning methods have gained due popularity. 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 removal (Kivinen et al., 2004; Dekel et al., 2008), and support vector merging (Wang et al., 2012). Maintaining an affordable budget, online multi-kernel learning (OMKL) methods have been reported for online classification (Jin et al., 2010; Hoi et al., 2013; Sahoo et al., 2016), and regression (Sahoo et al., 2014; Lu et al., 2018). Devoid of the need for budget maintenance, online kernel-based learning algorithms based on RF approximation (Rahimi and Recht, 2007) have been developed in (Lu et al., 2016; Bouboulis et al., 2018; Ding et al., 2017), but only with a single pre-selected kernel. More importantly, existing kernel-based learning approaches implicitly presume a *static* environment, where the benchmark is provided through the best static function (a.k.a. static regret) (Shalev-Shwartz, 2011). However, static regret is not a comprehensive metric for dynamic settings, where the optimal kernel also varies over time and the dynamics are generally unknown as with adversarial settings.

1.2 Our contributions

The present paper develops an adaptive online MKL 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, RFs 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) A novel adaptive approach (termed **AdaRaker**) is introduced for scalable online MKL in environments with unknown dynamics. AdaRaker is a hierarchical ensemble learner with scalable RF-based modules that provably yields sub-linear dynamic regret, so long as the accumulated variation grows sub-linearly with time.

c3) The novel algorithms are compared with competing alternatives for online nonlinear regression on both synthetic and real datasets. The tests corroborate that Raker and AdaRaker exhibit attractive performance in both accuracy and scalability.

Outline. Section 2 presents preliminaries, and states the problem. Section 3 develops the Raker for online MKL in static environments, and Section 4 develops its adaptive version for online MKL in environments with unknown dynamics. Section 5 reports numerical tests with both synthetic and real datasets, while conclusions are drawn in Section 6.

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 $\mathbf{x} > \mathbf{0}$, and the projection operator $[\mathbf{a}]^+ := \max\{\mathbf{a}, \mathbf{0}\}$ are defined entrywise. Symbol † represents the Hermitian operator, while the indicator function $\mathbb{1}_{\{A\}}$ takes value 1 when the event A happens, and 0 otherwise. $\mathbb{E}$ denotes the expectation, while $\langle \cdot, \cdot \rangle$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ the vector inner product in Euclidian and Hilbert space respectively.

2. Preliminaries and Problem Statement

This section reviews briefly basics of kernel-based learning, to introduce notation and the needed background for our novel online MKL schemes.

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 $y_t = f(\mathbf{x}_t) + e_t$, where e_t denotes an error term representing noise or un-modeled dynamics. It is supposed that $f(\cdot)$ belongs to a reproducing kernel Hilbert space (RKHS), namely $\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^\infty \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\}$, where $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a symmetric positive semidefinite basis (so-termed kernel) function, which measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$. Among the choices of κ specifying different bases, a popular one is the Gaussian 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 $\langle \kappa(\mathbf{x}, \mathbf{x}_t), \kappa(\mathbf{x}, \mathbf{x}_{t'}) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$, 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

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

where depending on the application, the cost function $\mathcal{C}(\cdot, \cdot)$ can be selected to be, e.g., the least-squares (LS), the logistic or the hinge loss; $\Omega(\cdot)$ is an increasing function; and, $\lambda > 0$ is a regularization parameter that controls overfitting. According to the representer theorem, the optimal solution of (1) admits the finite-dimensional form, given by (Wahba, 1990)

$$\hat{f}(\mathbf{x}) = \sum_{t=1}^T \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t) := \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}) \tag{2}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_T]^\top \in \mathbb{R}^T$ collects the combination coefficients, and the $T \times 1$ kernel vector is $\mathbf{k}(\mathbf{x}) := [\kappa(\mathbf{x}, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}, \mathbf{x}_T)]^\top$. Substituting (2) into the RKHS norm, we find $\|f\|_{\mathcal{H}}^2 := \sum_t \sum_{t'} \alpha_t \alpha_{t'} \kappa(\mathbf{x}_t, \mathbf{x}_{t'}) = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$, where the $T \times T$ kernel matrix $\mathbf{K}$ has entries $[\mathbf{K}]_{t,t'} := \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$; thus, the functional problem (1) boils down to a T-dimensional optimization over $\boldsymbol{\alpha}$, namely

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^T} \ \frac{1}{T} \sum_{t=1}^T \mathcal{C}(\boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}_t), y_t) + \lambda \Omega\left(\boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}\right) \tag{3}$$

where $\mathbf{k}^\top(\mathbf{x}_t)$ is the tth row of the matrix $\mathbf{K}$. While a scalar y_t is used here for brevity, coverage extends readily to vectors $\{\mathbf{y}_t\}$.

Note that (1) relies on: i) a known pre-selected kernel κ; and ii) having $\{\mathbf{x}_t, y_t\}_{t=1}^T$ available in batch form. A key observation here is that the dimension of the variable $\boldsymbol{\alpha}$ in

(3) grows with time T (or, the number of samples in the batch form), making it less scalable in online implementation. In the ensuing section, an online MKL method will be proposed to select κ as a superposition of multiple kernels, when the data become available online.

3. Online MKL in static environments

In this section, we develop an online learning approach that builds on the notion of **ra**ndom features (Rahimi and Recht, 2007; Yu et al., 2016), and leverages in a unique way multi-**ker**nel approximation – two tools justifying our acronym **Raker** used henceforth.

3.1 RF-based single kernel learning

To cope with the curse of dimensionality in optimizing (3), we will reformulate the functional optimization problem (1) as a parametric one with the dimension of optimization variables not growing with time. In this way, powerful toolboxes from convex optimization and online learning in vector spaces can be leveraged. We achieve this goal by judiciously using RFs. Although generalizations will follow, this subsection is devoted to RF-based single kernel learning, where basics of kernels, RFs, and online learning will be revisited.

As in (Rahimi and Recht, 2007), we will approximate κ in (2) using shift-invariant kernels that satisfy $\kappa(\mathbf{x}_t, \mathbf{x}_{t'}) = \kappa(\mathbf{x}_t - \mathbf{x}_{t'})$. For $\kappa(\mathbf{x}_t - \mathbf{x}_{t'})$ absolutely integrable, its Fourier transform $\pi_\kappa(\mathbf{v})$ exists and represents the power spectral density, which upon normalizing to ensure $\kappa(\mathbf{0}) = 1$, can also be viewed as a probability density function (pdf); hence,

$$\kappa(\mathbf{x}_t - \mathbf{x}_{t'}) = \int \pi_\kappa(\mathbf{v}) e^{j\mathbf{v}^\top(\mathbf{x}_t - \mathbf{x}_{t'})} d\mathbf{v} := \mathbb{E}_{\mathbf{v}}\left[e^{j\mathbf{v}^\top(\mathbf{x}_t - \mathbf{x}_{t'})}\right] \tag{4}$$

where the last equality is just the definition of the expected value. Drawing a sufficient number of D independent and identically distributed (i.i.d.) samples $\{\mathbf{v}_i\}_{i=1}^D$ from $\pi_\kappa(\mathbf{v})$, the ensemble mean in (4) can be approximated by the sample average

$$\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'}) := \frac{1}{D}\sum_{i=1}^{D} e^{j\mathbf{v}_i^\top(\mathbf{x}_t - \mathbf{x}_{t'})} := \boldsymbol{\zeta}_{\mathbf{V}}^\dagger(\mathbf{x}_t)\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x}_{t'}) \tag{5}$$

where $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_D]^\top \in \mathbb{R}^{D\times d}$, symbol $\dagger$ represents the Hermitian (conjugate-transpose) operator, and $\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x})$ the complex RF vector

$$\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x}) := \frac{1}{\sqrt{D}}\left[e^{j\mathbf{v}_1^\top\mathbf{x}}, \ldots, e^{j\mathbf{v}_D^\top\mathbf{x}}\right]^\top . \tag{6}$$

Taking expected values on both sides of (5) and using (4) yields $\mathbb{E}_{\mathbf{v}}[\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})] = \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$, which means $\hat{\kappa}_c$ is unbiased. Likewise, $\hat{\kappa}_c$ can be shown consistent since $\mathbb{V}\text{ar}[\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})] \propto D^{-1}$ vanishes as $D \to \infty$. Finding $\pi_\kappa(\mathbf{v})$ requires d-dimensional Fourier transform of κ, generally through numerical integration. For a number of popular kernels however, $\pi_\kappa(\mathbf{v})$ is available in closed form. Taking the Gaussian kernel as an example, where $\kappa_G(\mathbf{x}_t, \mathbf{x}_{t'}) = \exp\left(\|\mathbf{x}_t - \mathbf{x}_{t'}\|_2^2/(2\sigma^2)\right)$, has Fourier transform corresponding to the pdf $\pi_G(\mathbf{v}) = \mathcal{N}(0, \sigma^{-2}\mathbf{I})$.

Instead of the *complex* RFs $\{\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x}_t)\}$ in (6) forming the linear kernel estimator $\hat{\kappa}_c$ in (5), one can consider its real part $\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}) := \Re\{\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})\}$ that is also an unbiased estimator

of κ. Defining the real RF vector $\mathbf{z}_{\mathbf{V}}(\mathbf{x}) := [\Re^\top\{\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x}_t)\}, \Im^\top\{\boldsymbol{\zeta}_{\mathbf{V}}(\mathbf{x}_t)\}]^\top$, this real kernel estimator becomes (cf. (5))

$$\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}) = \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_{t'}) \tag{7}$$

where the $2D \times 1$ *real* RF vector can be written as

$$\mathbf{z}_{\mathbf{V}}(\mathbf{x}) = \frac{1}{\sqrt{D}} \left[\sin(\mathbf{v}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{v}_D^\top \mathbf{x}), \cos(\mathbf{v}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{v}_D^\top \mathbf{x})\right]^\top . \tag{8}$$

Hence, the nonlinear function that is optimal in the sense of (1) can be approximated by a linear one in the new $2D$-dimensional RF space, namely (cf. (2) and (7))

$$\hat{f}^{\mathrm{RF}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}) := \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}) \tag{9}$$

where $\boldsymbol{\theta}^\top := \sum_{\tau=1}^{T} \alpha_\tau \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_\tau)$ is the new weight vector of size $2D$ whose dimension does not increase with number of data samples T.

While the solution $\hat{f}$ in (2) is the superposition of nonlinear functions κ, its RF approximant $\hat{f}^{\mathrm{RF}}$ in (9) is a linear function of $\mathbf{z}_{\mathbf{V}}(\mathbf{x})$. As a result, the loss becomes

$$\mathcal{L}_t\big(f(\mathbf{x}_t)\big) := \mathcal{C}(f(\mathbf{x}_t), y_t) + \lambda\Omega\left(\|f\|_{\mathcal{H}}^2\right) = \mathcal{C}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) + \lambda\Omega\left(\|\boldsymbol{\theta}\|^2\right) \tag{10}$$

where $\|\boldsymbol{\theta}\|^2 := \sum_t \sum_{t'} \alpha_t \alpha_{t'} \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_{t'}) := \|f\|_{\mathcal{H}}^2$; and the online learning task is

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{2D}} \sum_{t=1}^{T} \mathcal{L}\left(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\right), \text{ with } \mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) := \mathcal{C}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) + \lambda\Omega\big(\|\boldsymbol{\theta}\|^2\big). \tag{11}$$

Compared with the functional optimization in (1), the reformulated problem (11) is parametric, and more importantly it involves only optimization variables of fixed size $2D$. We can thus solve (11) using the online gradient descent iteration, e.g., (Hazan, 2016). Acquiring $\mathbf{x}_t$ per slot t, its RF $\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)$ is formed as in (8), and $\boldsymbol{\theta}_{t+1}$ is updated online as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla\mathcal{L}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \tag{12}$$

where $\{\eta_t\}$ is the sequence of stepsizes that can tune learning rates, and $\nabla\mathcal{L}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$ the gradient at $\boldsymbol{\theta} = \boldsymbol{\theta}_t$. Iteration (12) provides *a functional update* since $\hat{f}_t^{\mathrm{RF}}(\mathbf{x}) = \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x})$, but the upshot of involving RFs is that this approximant is in the span of $\{\mathbf{z}_{\mathbf{V}}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}\}$. Since $\mathbb{E}[\hat{\kappa}] = \kappa$, we find readily that $\mathbb{E}[\hat{f}^{\mathrm{RF}}] = \hat{f}$; in words, unbiasedness of the kernel approximation ensures that the RF-based function approximant is also unbiased.

Variance-reduced RF. Besides unbiasedness, performance of the RF approximation is also influenced by the variance of RFs. Note that the variance of $\hat{\kappa}$ in (7) is of order $\mathcal{O}(D^{-1})$, but its scale can be reduced if $\mathbf{V}$ is formed to have orthogonal rows (Yu et al., 2016). Specifically for a Gaussian kernel with bandwidth σ^2, recall that $\mathbf{V} = \sigma^{-1}\mathbf{G}$ in (8), where each entry of $\mathbf{G}$ is drawn from $\mathcal{N}(0, 1)$. For the variance-reduced orthogonal (O)RF

with $D = d$, one starts with Q-R factorization of $\mathbf{V} = \mathbf{QR}$, and uses the $d \times d$ factor $\mathbf{Q}$ along with a diagonal matrix $\mathbf{\Lambda}$, to form (Yu et al., 2016)

$$\mathbf{V}_{\text{ORF}} = \sigma^{-1}\, \mathbf{\Lambda Q} \tag{13}$$

where the diagonal entries of $\mathbf{\Lambda}$ are drawn i.i.d. from the χ distribution with d degrees of freedom, to ensure unbiasedness of the kernel approximant. For $D > d$, one selects $D = \nu d$ with $\nu > 1$ integer, and generates independently ν matrices each of size $d \times d$ as in (13). The final $\mathbf{V}_{\text{ORF}}$ is formed by concatenating these $d \times d$ sub-matrices. The upshot of ORF is that (Yu et al., 2016) $\mathbb{V}\text{ar}(\hat{\kappa}_{\text{ORF}}(\mathbf{x}_t, \mathbf{x}_{t'})) \leq \mathbb{V}\text{ar}(\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}))$. As we have also confirmed via simulated tests, ORF-based function approximation can attain a prescribed accuracy with considerably less ORFs than what required by its RF-based counterpart.

The RF-based online single kernel learning scheme in this section presumes that κ is known a priori. Since this is not generally possible, it is prudent to adaptively select kernels by superimposing multiple kernel functions from a prescribed dictionary. This superposition will play a key role in the RF-based online MKL approach presented next.

3.2 Raker for online MKL

Specifying the kernel that "shapes" $\mathcal{H}$ is a critical choice for single kernel learning, since different kernels yield function estimates of variable accuracy. To deal with this, combinations of kernels from a prescribed and sufficiently rich dictionary $\{\kappa_p\}_{p=1}^P$ can be employed in (1). Each combination belongs to the convex hull $\bar{\mathcal{K}} := \{\bar{\kappa} = \sum_{p=1}^P \bar{\alpha}_p \kappa_p,\ \bar{\alpha}_p \geq 0,\ \sum_{p=1}^P \bar{\alpha}_p = 1\}$, and is itself a kernel (Schölkopf and Smola, 2002). With $\bar{\mathcal{H}}$ denoting the RKHS induced by $\bar{\kappa} \in \bar{\mathcal{K}}$, one then solves (1) with $\mathcal{H}$ replaced by $\bar{\mathcal{H}} := \mathcal{H}_1 \bigoplus \cdots \bigoplus \mathcal{H}_P$, where $\{\mathcal{H}_p\}_{p=1}^P$ represent the RKHSs corresponding to $\{\kappa_p\}_{p=1}^P$ (Micchelli and Pontil, 2005).

The candidate function $\bar{f} \in \bar{\mathcal{H}}$ is expressible in a separable form as $\bar{f}(\mathbf{x}) := \sum_{p=1}^P \bar{f}_p(\mathbf{x})$, where $\bar{f}_p(\mathbf{x})$ belongs to $\mathcal{H}_p$, for $p \in \mathcal{P} := \{1, \ldots, P\}$. To add flexibility per kernel in our ensuing online MKL scheme, we let wlog $\{\bar{f}_p = w_p f_p\}_{p=1}^P$, and seek functions of the form

$$f(\mathbf{x}) := \sum_{p=1}^P \bar{w}_p f_p(\mathbf{x}) \in \bar{\mathcal{H}} \tag{14}$$

where $f := \bar{f} / \sum_{p=1}^P w_p$, and the normalized weights $\{\bar{w}_p := w_p / \sum_{p=1}^P w_p\}_{p=1}^P$ satisfy $\bar{w}_p \geq 0$, and $\sum_{p=1}^P \bar{w}_p = 1$. Plugging (14) into (1), MKL solves the nonconvex problem

$$\min_{\{\bar{w}_p\},\{f_p\}} \ \frac{1}{T}\sum_{t=1}^T \mathcal{C}\left(\sum_{p=1}^P \bar{w}_p f_p(\mathbf{x}_t), y_t\right) + \lambda\Omega\left(\left\|\sum_{p=1}^P \bar{w}_p f_p\right\|_{\bar{\mathcal{H}}}^2\right) \tag{15a}$$

$$\text{s. to} \ \ \sum_{p=1}^P \bar{w}_p = 1,\ \bar{w}_p \geq 0,\ p \in \mathcal{P} \tag{15b}$$

$$f_p \in \mathcal{H}_p,\ p \in \mathcal{P}. \tag{15c}$$

If Ω is convex over f, then (15a) is biconvex, meaning it is convex wrt $\{f_p\}$ ($\{\bar{w}_p\}$) when $\{\bar{w}_p\}$ ($\{f_p\}$) is given. Leveraging biconvexity, existing batch MKL schemes solve (15) via

alternating minimization that is known not to scale well with P and T (Micchelli and Pontil, 2005; Cortes et al., 2009; Gönen and Alpaydın, 2011).

To deal with scalability, our novel approach will leverage for the first time (O)RFs in a uniquely principled MKL formulation to end up with an efficient online learning approach. To this end, we will minimize a cost that upper bounds that in (15a), namely

$$\min_{\{\bar{w}_p\},\{f_p\}} \frac{1}{T}\sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_p\, \mathcal{C}\left(f_p(\mathbf{x}_t), y_t\right) + \lambda \sum_{p=1}^{P} \bar{w}_p\, \Omega\left(\|f_p\|_{\mathcal{H}_p}^2\right) \quad \text{s. to (15b) and (15c)} \tag{16}$$

where Jensen's inequality confirms that under (15b) the cost in (16) upper bounds that of (15a). A key advantage of (16) is that its objective is separable across kernel 'atoms.'

We will exploit this separability jointly with the RF-based function approximation per kernel, to formulate our scalable online MKL task as

$$\min_{\{\bar{w}_p\},\{\hat{f}_p^{\mathrm{RF}}\}} \sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_p\, \mathcal{L}_t\left(\hat{f}_p^{\mathrm{RF}}(\mathbf{x}_t)\right) \quad \text{s. to (15b) and } \hat{f}_p^{\mathrm{RF}} \in \left\{\hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}_p}(\mathbf{x})\right\} \tag{17}$$

where we interchangeably use $\mathcal{L}_t(\hat{f}(\mathbf{x}_t))$ as defined in (10) and $\mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big)$ as in (11). We will efficiently solve (17) 'on-the-fly' using our Raker algorithm, and what more, we will provide analytical performance guarantees. Our iterative solution will update separately each $\hat{f}_p^{\mathrm{RF}}$ as in Section 3.1 using the scalable (O)RF-based function approximation scheme. Given $\mathbf{x}_t$, an RF vector $\mathbf{z}_p(\mathbf{x}_t)$ will be generated per p from pdf $\pi_{\kappa_p}(\mathbf{v})$ (cf. (8)), where we let $\mathbf{z}_p(\mathbf{x}_t) := \mathbf{z}_{\mathbf{V}_p}(\mathbf{x}_t)$ for notational brevity. Hence, for each p and slot t, we have

$$\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t) = \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t) \tag{18}$$

and as in (12), $\boldsymbol{\theta}_{p,t}$ is updated via

$$\boldsymbol{\theta}_{p,t+1} = \boldsymbol{\theta}_{p,t} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t). \tag{19}$$

As far as solving for $\bar{w}_{p,t}$, since it resides on a probability simplex (15b), our idea is to employ a multiplicative update (a.k.a. exponentiated gradient descent), e.g., (Hazan, 2016). Specifically, the un-normalized weights are found first as

$$w_{p,t+1} = \arg\min_{w_p} \eta\, \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)\right)(w_p - w_{p,t}) + \mathcal{D}_{\mathrm{KL}}(w_p \| w_{p,t}) \tag{20}$$

where $\mathcal{D}_{\mathrm{KL}}(w_p\|w_{p,t}) := w_p \log(w_p/w_{p,t})$ is the KL-divergence. It can be readily verified that (20) admits the following closed-form update

$$w_{p,t+1} = w_{p,t} \exp\left(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)\right)\right) \tag{21}$$

where $\eta \in (0,1)$ is a chosen constant that controls the adaptation rate of $\{w_{p,t}\}$. Having found $\{w_{p,t}\}$ as in (21), the normalized weights in (14) are obtained as $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^{P} w_{p,t}$. Update (21) is intuitively pleasing because when $\hat{f}_{p,t}^{\mathrm{RF}}$ contributes a larger loss relative to other $\hat{f}_{p',t}^{\mathrm{RF}}$ with $p' \neq p$ at slot t, the corresponding $w_{p,t+1}$ decreases more than the other

Algorithm 1 Raker for online MKL in static environments

Input: Kernels $\kappa_p,\ p=1,\ldots,P$, step size $\eta>0$, and number of random features D.
Initialization: $\boldsymbol{\theta}_1=\mathbf{0}$.
for $t=1,2,\ldots,T$ **do**
  Receive a streaming datum $\mathbf{x}_t$.
  Construct $\mathbf{z}_p(\mathbf{x}_t)$ via (8) using κ_p for $p=1,\ldots,P$.
  Predict $\hat{f}_t^{\mathrm{RF}}(\mathbf{x}_t):=\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)$ with $\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)$ in (18).
  Observe loss function $\mathcal{L}_t$, incur $\mathcal{L}_t(\hat{f}_t^{\mathrm{RF}}(\mathbf{x}_t))$.
  for $p=1,\ldots,P$ **do**
    Obtain loss $\mathcal{L}(\boldsymbol{\theta}_{p,t}^{\top}\mathbf{z}_p(\mathbf{x}_t),y_t)$ or $\mathcal{L}_t(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t))$.
    Update $\boldsymbol{\theta}_{p,t+1}$ via (19).
    Update $w_{p,t+1}$ via (21).
  end for
end for

weights in the next time slot. In other words, a more accurate RF-based approximant tends to play more important role in predicting the upcoming data.

Remark 1. The update (21) resembles the online learning paradigm, a.k.a. *online prediction with* (weighted) *expert advices* (Vovk, 1995; Cesa-Bianchi and Lugosi, 2006). Building on but going beyond OMKL in (Sahoo et al., 2014), the idea here is to view MKL with *RF-based function approximants* as a weighted combination of advices from an ensemble of P function approximants (experts). Besides permeating benefits from online learning to MKL, what is distinct here relative to (Vovk, 1995; Cesa-Bianchi and Lugosi, 2006) is that each function approximant also performs online learning for self improvement (cf. (19)).

In summary, our Raker for static (or slow-varying) dynamics is listed as Algorithm 1.

Memory requirement and computational complexity. At the t-th iteration, our Raker in Algorithm 1 needs to store a real $2D$ RF vector, and its corresponding weight vector per κ_p. Hence, the memory required is of order $\mathcal{O}(dDP)$. Regarding computational overhead, the per-iteration complexity (e.g., calculating inner products) is again of order $\mathcal{O}(dDP)$. Compared with the complexity of $\mathcal{O}(tdP)$ for OMKL by (Sahoo et al., 2014), or, $\mathcal{O}(t^3P)$ when matrix inversion required for the batch MKL, e.g., (Bazerque and Giannakis, 2013), the Raker is clearly more scalable, as t grows. Even when OMKL is confined to a budget of B past samples, the corresponding complexity of $\mathcal{O}(dBP)$ is comparable to that of Raker. This speaks for Raker's merits, whose performance guarantees will be proved analytically, and also demonstrated by numerical tests to outperform budgeted schemes.

Application examples: Online MKL regression and classification. To appreciate the usefulness of RF-based online MKL, consider first nonlinear regression, where given samples $\{\mathbf{x}_t\in\mathbb{R}^d, y_t\in\mathbb{R}\}_{t=1}^{T}$, the goal is to find a nonlinear function $f\in\mathcal{H}$, such that $y_t=f(\mathbf{x}_t)+e_t$. The criterion is to minimize the regularized prediction error of y_t, typically using the LS loss $\mathcal{L}(f(\mathbf{x}_t),y_t):=[y_t-f(\mathbf{x}_t)]^2+\lambda\|f\|_{\mathcal{H}}^2$, whose gradient is (cf. (19))

$$\nabla\mathcal{L}\left(\boldsymbol{\theta}_{p,t}^{\top}\mathbf{z}_p(\mathbf{x}_t),y_t\right)=2(\boldsymbol{\theta}_{p,t}^{\top}\mathbf{z}_p(\mathbf{x}_t)-y_t)\mathbf{z}_p(\mathbf{x}_t)+2\lambda\boldsymbol{\theta}_{p,t}. \tag{22}$$

It is clear that the per iteration complexity of Raker is only related to the dimension of $\mathbf{z}_p(\mathbf{x}_t)$, and does not increase over time.

For nonlinear classification, consider kernel-based perceptron and kernel-based logistic regression, which aim at learning a nonlinear classifier that best approximates either y_t or the pdf of y_t conditioned on $\mathbf{x}_t$. With binary labels $\{\pm 1\}$, the perceptron solves (1) with $\mathcal{L}(f(\mathbf{x}_t), y_t) = \max(0, 1 - y_t f(\mathbf{x}_t)) + \lambda \|f\|^2_{\mathcal{H}}$, which equals zero if $y_t = f(\mathbf{x}_t)$, otherwise it equals 1. Raker's gradient in this case is (cf. (19))

$$\nabla \mathcal{L}\left(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t\right) = -2y_t \mathcal{C}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)\mathbf{z}_p(\mathbf{x}_t) + 2\lambda \boldsymbol{\theta}_{p,t}. \tag{23}$$

Accordingly, given $\mathbf{x}_t$, logistic regression postulates that $\Pr(y_t = 1|\mathbf{x}_t) = 1/(1+\exp(f(\mathbf{x}_t)))$. Here the gradient of Raker takes the form (cf. (19))

$$\nabla \mathcal{L}\left(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t\right) = \frac{2y_t \exp(y_t \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t))}{1 + \exp(y_t \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t))} \mathbf{z}_p(\mathbf{x}_t) + 2\lambda \boldsymbol{\theta}_{p,t}. \tag{24}$$

To compare alternatives on equal footing, the numerical tests in Section 5 will deal with kernel-based regression and classification.

3.3 Static regret analysis of Raker

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

(as1) *Per slot t, the loss function $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$ in (11) is convex w.r.t. $\boldsymbol{\theta}$.*

(as2) *For $\boldsymbol{\theta}$ belonging to a bounded set $\boldsymbol{\Theta}$ with $\|\boldsymbol{\theta}\| \leq C_\theta$, the loss is bounded; that is, $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \in [-1, 1]$, and has bounded gradient, meaning, $\|\nabla \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)\| \leq L$.*

(as3) *Kernels $\{\kappa_p\}_{p=1}^P$ are shift-invariant, standardized, and bounded, that is, $\kappa_p(\mathbf{x}_i, \mathbf{x}_j) \leq 1$, $\forall \mathbf{x}_i, \mathbf{x}_j$; and w.l.o.g. they also have bounded entries, meaning $\|\mathbf{x}\| \leq 1$.*

Convexity of the loss under (as1) is satisfied by the popular loss functions including the square loss and the hinge loss. As far as (as2), it ensures that the losses, and their gradients are bounded, meaning they are L-Lipschitz continuous. While boundedness of the losses commonly holds since $\|\boldsymbol{\theta}\|$ is bounded, Lipschitz continuity is also not restrictive. Considering kernel-based regression as an example, the gradient is $(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) - y_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) + \lambda \boldsymbol{\theta}$. Since the loss is bounded, e.g., $\|\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) - y_t\| \leq 1$, and the RF vector in (8) can be bounded as $\|\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)\| \leq 1$, the constant is $L := 1 + \lambda C_\theta$ using the Cauchy-Schwartz inequality. Kernels satisfying conditions in (as3) include Gaussian, Laplacian, and Cauchy (Rahimi and Recht, 2007). In general, (as1)-(as3) are standard in online convex optimization (OCO) (Shalev-Shwartz, 2011; Hazan, 2016), and in kernel-based learning (Micchelli and Pontil, 2005; Rahimi and Recht, 2007; Lu et al., 2016).

With regard to the performance of an online algorithm, static regret is commonly adopted as a metric by most OCO schemes to measure the difference between the aggregate loss of an OCO algorithm, and that of the best fixed function approximant in hindsight, e.g., (Shalev-Shwartz, 2011; Hazan, 2016). Specifically, for a generic sequence $\{\hat{f}_t\}$ generated by an RF-based kernel learning algorithm $\mathcal{A}$, its static regret is

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{x}_t)) \tag{25}$$

where $\hat{f}_t$ will henceforth represent $\hat{f}_t^{\text{RF}}$ without the superscript for notational brevity; and, $f^*(\cdot)$ is obtained as the batch solution

$$f^*(\cdot) \in \arg \min_{\{f_p^*,\, p\in\mathcal{P}\}} \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \quad \text{with} \quad f_p^*(\cdot) \in \arg \min_{f\in\mathcal{F}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{x}_t)) \tag{26}$$

with $\mathcal{F}_p := \mathcal{H}_p$, and $\mathcal{H}_p$ representing the RKHS induced by κ_p. Using (25) and (26), we first establish the static regret of our Raker approach in the following lemma.

Lemma 1 *Under (as1), (as2), and with $\hat{f}_p^*$ as in* (26) *with $\mathcal{F}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}),\, \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, the sequences $\{\hat{f}_{p,t}\}$ and $\{\bar{w}_{p,t}\}$ generated by Raker satisfy the following bound*

$$\sum_{t=1}^{T} \mathcal{L}_t\Big(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\Big) - \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_p^*(\mathbf{x}_t)\right) \leq \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \tag{27}$$

where $\boldsymbol{\theta}_p^*$ is associated with the best RF function approximant $\hat{f}_p^*(\mathbf{x}) = \left(\boldsymbol{\theta}_p^*\right)^\top \mathbf{z}_p(\mathbf{x})$.

Proof: See Appendix A. ■

Besides Raker's static regret bound, the next theorem compares the Raker loss relative to that of the best functional estimator in the original RKHS.

Theorem 1 *Under (as1)-(as3) and with f_p^* in* (26) *belonging to the RKHS $\mathcal{H}_p$, for a fixed $\epsilon > 0$, the following bound holds with probability at least $1 - 2^8\left(\frac{\sigma_p}{\epsilon}\right)^2 \exp\left(\frac{-D\epsilon^2}{4d+8}\right)$*

$$\begin{aligned}\sum_{t=1}^{T} \mathcal{L}_t\left(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\right) - \min_{p\in\{1,\ldots,P\}} \sum_{t=1}^{T} \mathcal{L}_t\left(f_p^*(\mathbf{x}_t)\right) \\ \leq \frac{\ln P}{\eta} + \frac{(1+\epsilon)C^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T + \epsilon LTC\end{aligned} \tag{28}$$

where C is a constant, while $\sigma_p^2 := \mathbb{E}_{\mathbf{V}}^{\pi_{\kappa_p}}[\|\mathbf{v}\|^2]$ is the second-order moment of the RF vector norm. Setting $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$ in (28)*, the static regret in* (25) *leads to*

$$\text{Reg}_{\text{Raker}}^{\text{s}}(T) = \mathcal{O}(\sqrt{T}). \tag{29}$$

Proof: See Appendix B. ■

Observe that the probability of (28) to hold grows as D increases, and one can always find a D to ensure a positive probability for a given ϵ. Bearing this in mind, we will henceforth use "with high probability" (w.h.p.) to summarize the sense (28) and (29) hold. Theorem 1 establishes that with proper choice of parameters, the Raker achieves sub-linear regret relative to the best static function approximant in (26).

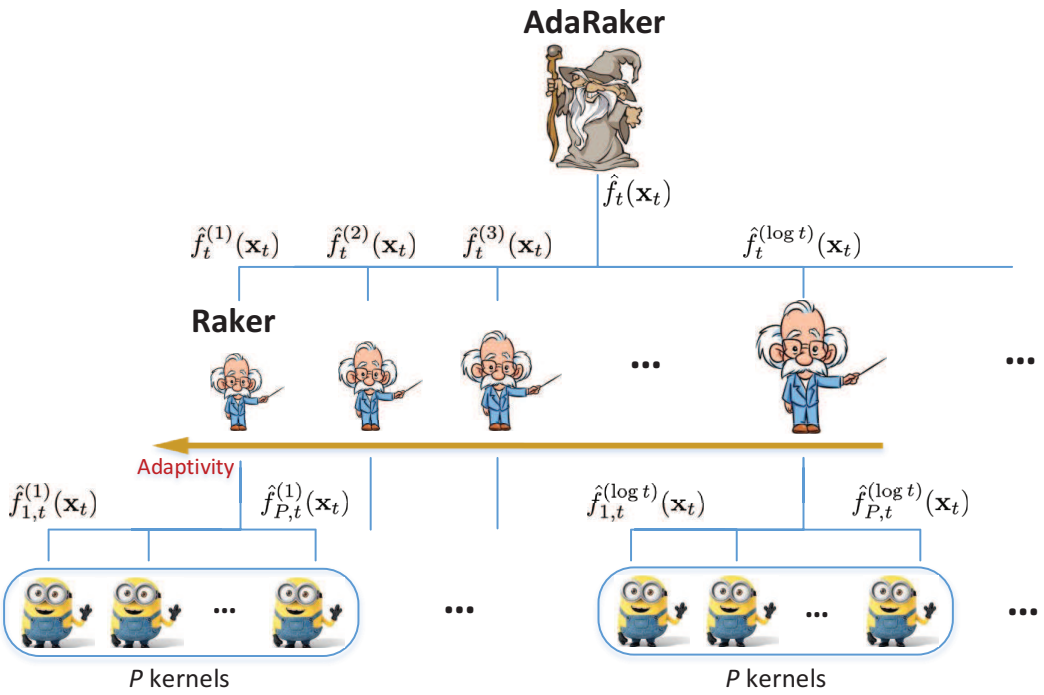


Figure 1: Hierarchical AdaRaker structure. Experienced experts in the middle layer present a Raker instance, where the size of expert cartoons is proportional to the interval length.

4. Online MKL in Environments with Unknown Dynamics

Our Raker in Section 3 combines an ensemble of kernel learners 'on the fly,' and performs on average as the "best" fixed function, thus fulfilling the learning objective in environments with zero (or slow) dynamics. To broaden its scope to environments with unknown dynamics, this section introduces an **ada**ptive **Raker** approach (termed **AdaRaker**).

4.1 AdaRaker with hierarchical ensembles

As with any online learning algorithm, the choice of η in (19) and (21) affects the performance critically. Especially in environments with unknown dynamics, a large η improves the tracking ability of fast-varying functions, while a smaller one allows improved estimation of slow-varying parameters $\{\boldsymbol{\theta}_t, w_{p,t}\}$. The optimal choice of η_t clearly depends on the variability of the optimal function estimator (Kivinen et al., 2004; Besbes et al., 2015). Selecting $\{\eta_t\}$ however, is formidably challenging if the environment dynamics are unknown.

Toward addressing this challenge, our idea here is to hedge between multiple Raker learners with different learning rates. Specifically, we view 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 the 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 $\eta_0 > 0$. Allowing for overlap between intervals, multiple Raker instances $\{\mathcal{A}_I\}$ will be run in parallel. Consider now

Algorithm 2 AdaRaker for online MKL in dynamic environments

Initialization: learner weights $\{h_1^{(I)}\}$, and their learning rates $\{\eta^{(I)}\}$.
for $t = 1, 2, \ldots, T$ **do**
  Obtain $\hat{f}_t^{(I)}(\mathbf{x}_t)$ from each Raker instance $\mathcal{A}_I$, $I \in \mathcal{I}(t)$.
  Predict $\hat{f}_t(\mathbf{x}_t)$ via a weighted combination (33).
  Observe loss function $\mathcal{L}_t$, and incur $\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t))$.
  for $I \in \mathcal{I}(t)$ **do**
    Incur loss $\mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t))$.
    Update $\hat{f}_t^{(I)}$ via Raker in Algorithm 1.
    Update weights $h_{t+1}^{(I)}$ via (31).
  end for
end for

collecting all active intervals at the current slot t in the set

$$\mathcal{I}(t) := \{I \in \mathcal{I} \,|\, t \in [\underline{I}, \bar{I}]\}, \ \forall t \in \mathcal{T}. \tag{30}$$

For each Raker instance $\mathcal{A}_I$ with $I \in \mathcal{I}(t)$, let $\hat{f}_t^{(I)}(\cdot)$ denote its output at time t that combines multiple kernel-based function estimators, and $\mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t))$ represent the associated instantaneous loss. The output of the ensemble learner $\mathcal{A}$ at time t is the weighted combination of outputs from all learners, namely $\{\hat{f}_t^{(I)}, \forall I \in \mathcal{I}(t)\}$. With $h_t^{(I)}$ denoting the weight of the Raker instance $\mathcal{A}_I$, we will update it online via

$$h_{t+1}^{(I)} = \begin{cases} 0, & \text{if } t \notin I \\ \eta^{(I)}, & \text{if } t = \underline{I} \\ h_t^{(I)} \exp\big(-\eta^{(I)} r_t^{(I)}\big), & \text{else} \end{cases} \tag{31}$$

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

$$r_t^{(I)} = \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t)), \ \forall I \in \mathcal{I}(t). \tag{32}$$

Intuitively thinking, one would wish to decrease (increase) the weights of those instances with small (large) losses in future rounds. Using update (31), 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

$$\hat{f}_t(\mathbf{x}) := \sum_{I \in \mathcal{I}} \bar{h}_t^{(I)} \hat{f}_t^{(I)}(\mathbf{x}) \quad \text{with} \quad \hat{f}_t^{(I)}(\mathbf{x}) := \sum_{p \in \mathcal{P}} \bar{w}_{p,t}^{(I)} \hat{f}_{p,t}^{(I)}(\mathbf{x}) \tag{33}$$

where $\{\bar{w}_{p,t}^{(I)}\}$ are the kernel combination weights generated by Raker $\mathcal{A}_I$ (cf. (21)).

The AdaRaker scheme is summarized in Algorithm 2, and depicted in Figure 1.

Selecting judiciously variable-length intervals in $\mathcal{I}$ can affect performance critically. Such a selection criterion for achieving interval regret has been reported in (Daniely et al., 2015). Instead, our pursuit is a hierarchical ensemble design for online MKL in environments with unknown dynamics using scalable RF-based function approximants. This hierarchical

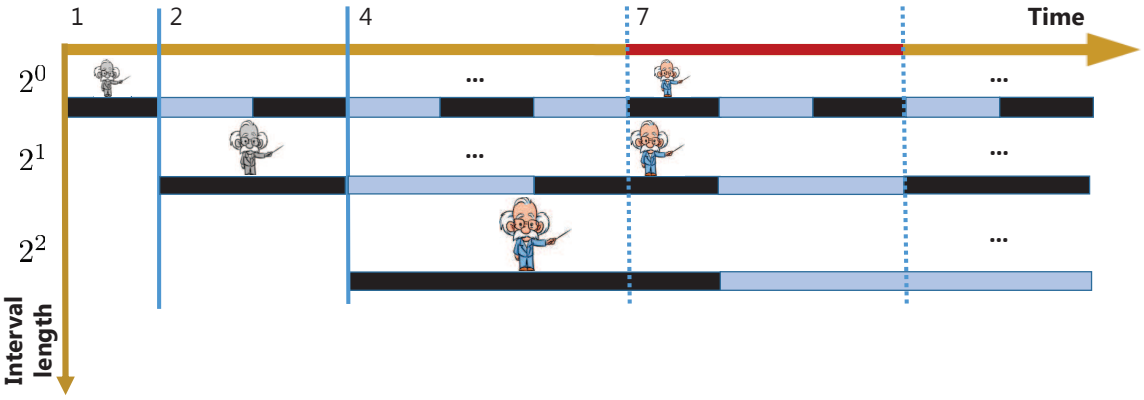


Figure 2: AdaRaker as an ensemble of Rakers with different learning rates: Each light/dark black interval initiates a Raker learner. At slot 7, colored experts are active, and gray ones are inactive.

design is well motivated because with long intervals, the Raker loss per interval is relatively low in slow-varying settings, but higher as the dynamics become more pronounced. On the other hand, a short interval can hedge against a possibly rapid change, but its performance on each interval could suffer if the objective stays nearly static. Bearing these tradeoffs in mind, we present next a simple yet efficient interval partitioning scheme.

Illustration of interval sets: *Consider partitioning the entire horizon into intervals of length* $2^0, 2^1, 2^2, \ldots$. *Intervals of length* 2^j *with a given* $j \in \mathbb{N}$ *are consecutively assigned without overlap starting from* $t = 2^j$. *In the non-overlapping case, define a set of intervals* $\mathcal{I}_j = [\underline{I}_j, \bar{I}_j]$ *such that each interval's length is* $|\mathcal{I}_j| = \bar{I}_j - \underline{I}_j + 1 = 2^j$, $j \in \mathbb{N}$. *For this selection of intervals, 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. *See the diagram in Fig. 2.*

4.2 Dynamic regret analysis of AdaRaker

The static regret in Theorem 1 is with respect to a time-invariant optimal function estimator benchmark. In dynamic environments however, this optimal function benchmark may change over time - what justifies this subsection's performance analysis of AdaRaker.

Our analysis will rely on the *dynamic regret* that is related to tracking regret, and has been introduced in (Besbes et al., 2015; Jadbabaie et al., 2015) to quantify the performance of online algorithms. The dynamic regret is defined as (cf. (25))

$$\mathrm{Reg}^{\mathrm{d}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f_t^*(\mathbf{x}_t)) \tag{34}$$

where the benchmark is the aggregate loss incurred by a sequence of the best dynamic functions$\{f_t^*\}$ from $\mathcal{F}$ formed by the union of function spaces $\mathcal{H}_p$ induced by $\{\kappa_p\}$, given by

$$f_t^*(\cdot) \in \arg \min_{\{f_{p,t}^*, p \in \mathcal{P}\}} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \quad \text{with} \quad f_{p,t}^*(\cdot) \in \arg \min_{f \in \mathcal{H}_p} \mathcal{L}_t(f(\mathbf{x}_t)) \tag{35}$$

Comparing (26) with (35) we deduce that the dynamic regret is always larger than the static regret in (25). Thus, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa. Given $\{\mathcal{L}_t\}$, AdaRaker generates functions $\{\hat{f}_t\}$ to minimize the dynamic regret.

To assess the AdaRaker performance, we will start with an *intermediate* result on the static regret associated with any sub-interval $I \subseteq \mathcal{T}$.

Lemma 2 *Under (as1)-(as3), the static regret on any interval $I \subseteq \mathcal{T}$ is given by*

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(|I|) := \sum_{t\in I} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \tag{36}$$

where $|I|$ denotes the length of interval I, and the best time-invariant function approximant is $f_I^* \in \arg\min_{f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p} \sum_{t\in I} \mathcal{L}_t(f(\mathbf{x}_t))$, with $\mathcal{H}_p$ denoting the RKHS induced by κ_p. Then for any interval $I \subseteq \mathcal{T}$ and fixed positive constants C_0, C_1, the following bound holds

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) \leq C_0\sqrt{|I|} + C_1 \ln T\sqrt{|I|}, \text{ w.h.p.} \tag{37}$$

Proof: See Appendix C. ■

Lemma 2 establishes that by combining Raker learners with different learning rates, AdaRaker can achieve sub-linear static regret over *any* interval I with arbitrary interval length. This also holds for intervals overlapping with multiple intervals; see e.g., the red interval in Fig. 2. Clearly, the best fixed solution in (36) is interval specific, which can vary over different intervals. This is qualitatively why the function approximants generated by AdaRaker can cope with a *time-varying* benchmark. Such an intuition will in fact become quantitative in the next theorem, which establishes the dynamic regret for AdaRaker.

Theorem 2 *Suppose (as1)-(as3) are satisfied, and define the accumulated variation of online loss functions as*

$$\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T) := \sum_{t=1}^{T} \max_{f\in\mathcal{F}} \left|\mathcal{L}_{t+1}(f(\mathbf{x}_{t+1})) - \mathcal{L}_t(f(\mathbf{x}_t))\right| \tag{38}$$

where $\mathcal{F} := \bigcup_{p\in\mathcal{P}} \mathcal{H}_p$. Then AdaRaker can afford a dynamic regret in (34) *bounded by*

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) \leq& (2 + C_0 + C_1 \ln T) T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T) \\ =& \tilde{\mathcal{O}}\left(T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T)\right), \text{ w.h.p.}\end{aligned} \tag{39}$$

where $\tilde{\mathcal{O}}$ neglects the lower-order terms with a polynomial $\log T$ rate.

Proof: See Appendix D. ■

Theorem 2 asserts that AdaRaker's dynamic regret depends on the variation of loss functions in (38), and on the horizon T. Interesting enough, whenever the loss functions *do not vary on average*, meaning $\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T) = \mathbf{o}(T)$, AdaRaker achieves sub-linear dynamic regret. To this end, it is useful to present an example where this argument holds.

Intermittent switches: With $\mathcal{L}_t \neq \mathcal{L}_{t+1}$ defining a switch, consider that the number of switches is sub-linear over T; that is, $\sum_{t=1}^{T} \mathbb{1}(\mathcal{L}_t \neq \mathcal{L}_{t+1}) = T^{\gamma}$, $\forall \gamma \in [0,1)$. Then it follows that $\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^{T}) = \mathcal{O}(T^{\gamma})$, since the one-slot variation of the loss functions is bounded.

Other setups with sub-linear accumulated variation emerge, e.g., when the per-slot variation decreases as $\mathcal{V}(\mathcal{L}_t) = \mathcal{O}(t^{\gamma-1})$, $\forall \gamma \in [0,1)$. Besides dynamic losses, sub-linear dynamic regrets can be also effected by confining the variability of optimal function estimators.

Theorem 3 *Suppose the conditions of Theorem 2 hold, and define the regret relative to an m-switching dynamic benchmark as* $\mathrm{Reg}_{\mathcal{A}}^{m}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(\check{f}_t^*(\mathbf{x}_t))$, *where* $\{\check{f}_t^*\}$ *is any trajectory from*

$$\left\{ \{\check{f}_t^*\}_{t=1}^{T} \in \bigcup\nolimits_{p\in\mathcal{P}} \mathcal{H}_p \,\middle|\, \sum_{t=1}^{T} \mathbb{1}(\check{f}_t^* \neq \check{f}_{t-1}^*) \leq m \right\}. \tag{40}$$

With C_0 and C_1 denoting some universal constants, it then holds w.h.p. that

$$\mathrm{Reg}_{\mathrm{AdaRaker}}^{m}(T) \leq (C_0 + C_1 \ln T)\sqrt{Tm} = \tilde{\mathcal{O}}\left(\sqrt{Tm}\right). \tag{41}$$

Proof: See Appendix E. ■

Theorem 3 asserts that without prior knowledge of the environment dynamics, the dynamic regret of AdaRaker is sub-linearly growing with time, provided that the number of changes of the optimal function estimators is sub-linear in T; that is, $\mathrm{Reg}_{\mathrm{AdaRaker}}^{m}(T) = \mathbf{o}(T)$ given $m = \mathbf{o}(T)$. Therefore, our AdaRaker can track the optimal dynamic functions, if the optimal function *varies slowly* over time; e.g., it does not change in the long-term average sense. While the conditions to guarantee optimality in dynamic settings may appear restrictive, they are practically relevant, since abrupt changes or adversarial samples will likely not happen at each and every slot in practice.

5. Numerical Tests

This section evaluates the performance of our novel algorithms in online regression tasks using both synthetic and real-world datasets.

In the subsequent tests, we use the following benchmarks.

RBF: the online *single* kernel learning method using Gaussian kernels, a.k.a. radial basis functions (RBFs), with bandwidth $\sigma^2 = \{0.1, 1, 10\}$ (cf. RBF01, RBF1, RBF10);
POLY: the online *single* kernel method using polynomial kernels, with degree $d = \{2, 3\}$ (cf. POLY2, POLY3);
LINEAR: the online *single* kernel learning method using a linear kernel;
AvgMKL: the online *single* kernel learning method using the average of candidate kernels without updating the weights;
OMKL: the popular online (O)MKL algorithm without a budget (Sahoo et al., 2014);
OMKL-B: the OMKL algorithm on a budget for regression modified from its single kernel version (Kivinen et al., 2004), with the kernel combination weights updated as (21);
M-Forgetron: the online multi-kernel based Forgetron modified from its single kernel version (Dekel et al., 2008), with the kernel combination weights updated as in (21);

Time index	$[1, 200]$	$[201, 1000]$	$[1001, 2000]$	$[2001, 2300]$	$[2301, 3000]$
σ^2	0.01	1	10	0.01	1
Time index	$[3001, 3500]$	$[3501, 4300]$	$[4301, 5100]$	$[5101, 5900]$	$5901, 6500$
σ^2	10	0.01	1	0.01	0.1

Table 1: Intervals and $\{\sigma^2\}$ for synthetic dataset.

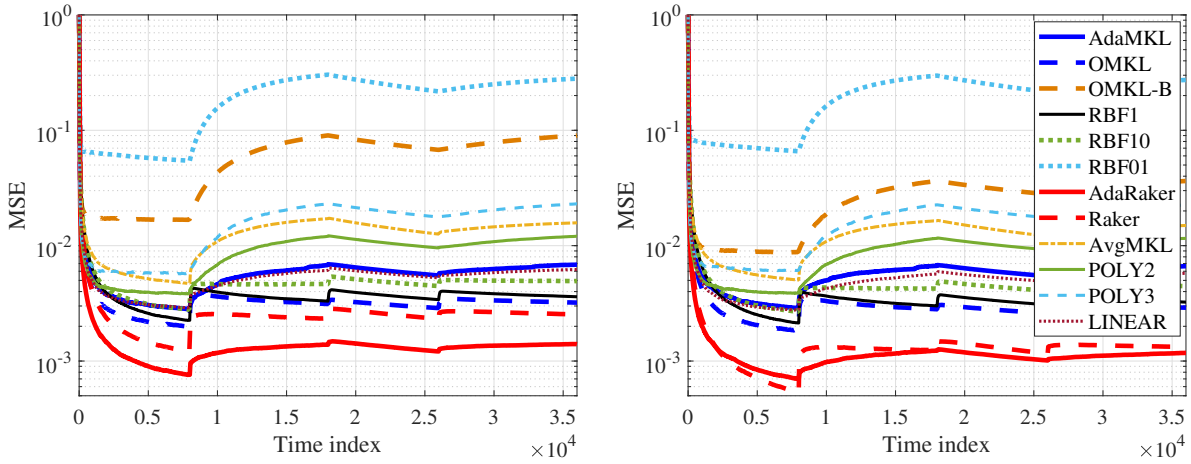
AdaMKL: the adaptive version of OMKL that operates in a similar fashion as Algorithm 2, but instead of using our Raker as an ensemble, it adopts OMKL as an instance $\mathcal{A}_I$.

Note that AdaMKL, OMKL-B, and M-Forgetron have not been formally proposed in existing works, but we introduced them here only for comparison purposes. All the considered MKL approaches use a dictionary of Gaussian kernels with $\sigma^2 = \{0.1, 1, 10\}$, and AvgMKL, OMKL, AdaMKL, OMKL-B, and M-Forgetron also include a linear, and a polynomial kernel with order of 2 into their kernel dictionary. For all MKL approaches, the stepsize for updating kernel combination weights in (21) is chosen as 0.5 uniformly, while the stepsize for updating per-kernel function estimators will be specified later in each test. The regularization parameter is set equal to $\lambda = 0.01$ for all approaches. Entries of $\{\mathbf{x}_t\}$ and $\{y_t\}$ are normalized to lie in $[0, 1]$. Regarding AdaMKL and AdaRaker, multiple instances are initialized on intervals with length $|I| := 2^0, 2^1, 2^2, \ldots$, along with the corresponding learning rate on the interval I as $\eta^{(I)} := \min\{1/2, 10/\sqrt{|I|}\}$; see the example in Figure 2. All the results in the tables were reported using the performance at the last time index.

5.1 Synthetic data tests for regression

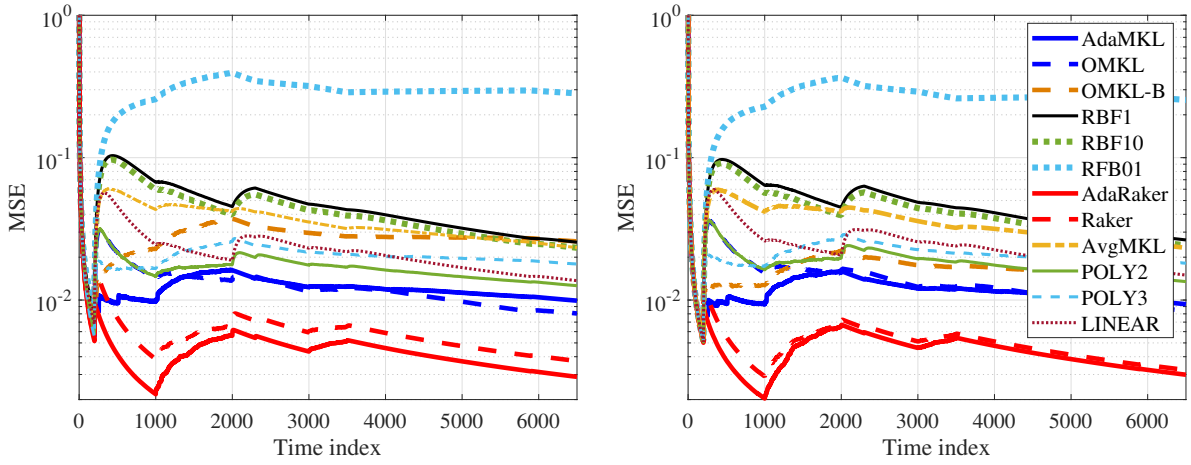
This subsection presents the synthetic data tests for regression.

Data generation. In this test, two synthetic datasets were generated as follows.
For *Dataset 1*, the feature vectors $\{\mathbf{x}_t \in \mathbb{R}^{10}\}_{t=1}^{14,000}$ are generated from the standardized Gaussian distribution, while y_t is generated as $y_t = \sum_{\tau=1}^{t} \alpha_\tau \kappa_\tau(\mathbf{x}_t, \mathbf{x}_\tau)$, where $\{\alpha_t\}$ is generated as $\alpha_t = 1 + e_t$ with $e_t \sim \mathcal{N}(0, \sigma_\alpha^2)$ and $\sigma_\alpha = 0.01$, while $\{\kappa_t\}$ are kernel functions that change overtime: for $t \in [1, 8000] \bigcup [18001, 26000]$, κ_t is a Gaussian kernel with $\sigma^2 = 1$, while for $t \in [8001, 18000] \bigcup [26001, 36000]$ the Gaussian kernel has $\sigma^2 = 10$. Therefore, the underlying nonlinear relationship between $\mathbf{x}_t$ and y_t undergoes intermittent changes, which come from corresponding changes in the optimal kernel combinations.
Dataset 2 is generated with more variance and switching points. Specifically, the feature vectors are generated from the standardized Gaussian distribution, while y_t is generated as $y_t = \sum_{\tau=1}^{t} \alpha_\tau \kappa_\tau(\mathbf{x}_t, \mathbf{x}_\tau)$, where $\{\kappa_t\}$ change over 10 intervals with different σ^2; see Table 1.
Testing performance. The performance of all schemes is tested in terms of the mean-square (prediction) error $\text{MSE}(t) := (1/t) \sum_{\tau=1}^{t} (y_\tau - \hat{y}_\tau)^2$ in Figure 3 and Figure 4, and their CPU time is listed in Table 2. For OMKL-B, $B = 20$ and 50 most recent data samples were kept in the budget; and for RF-based Raker and AdaRaker approaches, $D = 20$ and 50 orthogonal random features were used by default. The default stepsize is chosen as $1/\sqrt{T}$ for RBF, POLY, LINEAR, AvgMKL, OMKL, OMKL-B and Raker. In both tests, AdaRaker outperforms the alternatives in terms of MSE, especially when the true nonlinear relationship between $\mathbf{x}_t$ and y_t changes; e.g., compare the MSE of KL-RBF and Raker with that of AdaRaker at $t = 8000, 18000, 26000$ in Figure 3, and $t = 200, 2000, 3000, 3500$



(a) (b)

Figure 3: MSE performance on synthetic Dataset 1: a) $D = B = 20$; b) $D = B = 50$.



(a) (b)

Figure 4: MSE performance on synthetic Dataset 2: a) $D = B = 20$; b) $D = B = 50$.

in Figure 4. This corroborates the effectiveness of the novel AdaRacker method that can flexibly select learning rates according to the variability of the environments, and adaptively combine multiple kernels when the optimal underlying nonlinear relationship is varying over time. In addition, MKL approaches including our Raker approach enjoy lower MSE than that of the single-kernel approaches as well as the simple AvgMKL approach, which is also aligned with our design principle of developing MKL schemes that broaden generalizability of a kernel-based learner over a larger function space.

Table 2 records the CPU time of all benchmark algorithms running tests on two different datasets. It can be observed that leveraging the RF-based approximation, the proposed AdaRaker and Raker algorithms are much faster than AdaMKL and OMKL; hence, they are preferable especially for large-scale datasets. Although the CPU time of OMKL-B with

	Dataset 1		Dataset 2	
Setting	$D = B = 20$	$D = B = 50$	$D = B = 20$	$D = B = 50$
AdaMKL	318.52		27.29	
OMKL	157.10		5.47	
RBF	47.83		1.06	
POLY2	6.01		0.47	
POLY3	28.27		1.24	
LINEAR	4.80		0.35	
AvgMKL	144.85		5.02	
OMKL-B	3.75	4.05	0.72	0.77
Raker	1.39	1.53	0.18	0.20
AdaRaker	21.94	24.24	3.32	3.54

Table 2: CPU time (in seconds) on synthetic datasets. RBF, POLY represents all single-kernel methods using RBF and polynomial kernels, since they have the same CPU time.

Dataset	**# features** (d)	**# samples** (T)	**feature type**
Twitter	77	$14,000$	real & integer
Twitter (Large)	77	$100,000$	real & integer
Tom's hardware	96	$10,000$	real & integer
Energy	27	$18,600$	real
Air quality	13	$9,358$	real

Table 3: A summary of real datasets used in the tests.

a budget size $B = 20$ or $B = 50$ is relatively low, OMKL-B does not perform as well as AdaRaker and Raker algorithms. Therefore, the AdaRaker and Raker approaches attain a sweet-spot in the performance-complexity tradeoff.

5.2 Real data tests for online regression

To further evaluate our algorithms in real-world scenarios, the present subsection is devoted to testing and comparing on several popular real datasets.

Datasets description. Performance is tested on benchmark datasets from UCI machine learning repository (Lichman, 2013).

- **Twitter** dataset consists of $T = 14,000$ samples from a popular micro-blogging platform Twitter, where $\mathbf{x}_t \in \mathbb{R}^{77}$ include features such as the number of new interactive authors, and the length of discussion on a given topic, while y_t represents the average number of active discussion (popularity) on a certain topic (Kawala et al., 2013). A larger dataset with $T = 100,000$ is also included for testing only (Ada)Raker and OMKL-B, since other methods do not scale to such a large T.

- **Tom's hardware** dataset contains $T = 10,000$ samples from a worldwide new technology forum, where a 96-dimensional feature vector includes the number of discussions involving a certain topic, while y_t represents the average number of display about a certain topic on Tom's hardware (Kawala et al., 2013).

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF ($\sigma^2 = 0.1$)	27.0	14.4	28.9	26.3
RBF ($\sigma^2 = 1$)	13.5	17.0	28.8	12.7
RBF ($\sigma^2 = 10$)	23.3	18.8	28.8	15.5
POLY2	12.7	22.3	28.8	7.34
POLY3	20.4	22.7	28.9	5.91
LINEAR	8.57	19.5	28.8	10.7
AvgMKL	14.4	17.5	28.7	11.9
OMKL	8.55	14.3	28.1	6.4
AdaMKL	16.1	18.4	30.4	10.1
OMKL-B ($B = 50$)	27.0	22.1	73.3	35.9
Raker ($D = 50$)	3.0	3.4	19.3	2.0
AdaRaker ($D = 50$)	**2.6**	**1.9**	**13.8**	**1.3**

Table 4: MSE (10^{-3}) performance of different algorithms with stepsize $1/\sqrt{T}$.

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF ($\sigma^2 = 0.1$)	17.2	3.3	16.6	8.1
RBF ($\sigma^2 = 1$)	3.3	5.1	16.4	2.8
RBF ($\sigma^2 = 10$)	5.6	13.6	16.4	18.9
POLY2	8.1	15.9	16.2	3.3
POLY3	20.4	20.7	16.2	4.6
LINEAR	2.7	4.8	16.3	2.9
AvgMKL	7.1	6.2	16.3	2.8
OMKL	4.2	3.3	16.2	2.4
AdaMKL	16.1	18.4	30.4	10.1
OMKL-B ($B = 50$)	9.9	11.8	19	7.1
Raker ($D = 50$)	2.9	2.6	**13.8**	**1.3**
AdaRaker ($D = 50$)	**2.6**	**1.9**	**13.8**	**1.3**

Table 5: MSE (10^{-3}) performance of different algorithms with optimally chosen stepsizes.

- **energy** dataset consists of $T = 18,600$ samples, with each $\mathbf{x}_t \in \mathbb{R}^{27}$ describing the humidity and temperature indoors and outdoors, while y_t denotes the energy use of light fixtures in the house (Candanedo et al., 2017).
- **air quality** dataset collects $T = 9,358$ instances of hourly averaged responses from five chemical sensors located in a polluted area of Italy. The averaged sensor response $\mathbf{x}_t \in \mathbb{R}^{13}$ contains the hourly concentrations of e.g., CO, Non Metanic Hydrocarbons, and Nitrogen Dioxide (NO2), where the goal is to predict the concentration of polluting chemicals y_t in the air (De Vito et al., 2008).

To highlight the effectiveness of our approaches, the datasets mainly include time series data, where non-stationarity is more likely to happen; see Table 3 for a summary.

MSE performance. The MSE performance of each algorithm on the aforementioned datasets is presented in Table 4. By default, we use the complexity $B = D = 50$ for OMKL-B and (Ada)Raker, and the stepsize $1/\sqrt{T}$ for RBF, POLY, LINEAR, AvgMKL,

MSE	**OMKL-B**				**Raker**				**AdaRaker**
Stepsize	$1/\sqrt{T}$	$0.5/\sqrt{t}$	$0.1/\sqrt{t}$	Tuned	$1/\sqrt{T}$	$0.5/\sqrt{t}$	$0.1/\sqrt{t}$	Tuned	/
Twitter	27.0	27.1	29.6	9.9	3.0	17.9	4.3	2.9	**2.6**
Tom's	22.1	22.1	22.6	11.8	3.4	2.0	7.6	2.6	**1.9**
Energy	73.3	74.1	79.5	19.0	19.3	29.5	25.1	**13.8**	**13.8**
Air	35.9	35.9	40.1	7.1	2.0	29.1	4.0	**1.3**	**1.3**
Twitter (Large)	20.7	27.2	28.0	11.3	3.2	3.1	3.3	3.0	**2.7**

Table 6: MSE (10^{-3}) versus the choice of stepsizes with complexity $B = D = 50$.

OMKL, OMKL-B and Raker. To boost the performance of each algorithm, their MSE when using manually tuned stepsizes is also reported in Table 5, which selects the best stepsize on each dataset among $\{10^{-3}, 10^{-2}, \cdots, 10^{3}\}/\sqrt{T}$. A common observation is that leveraging the flexibility of multiple kernels, MKL methods in most cases outperform the algorithms using only a single kernel. By simply averaging over all the kernels, AvgMKL outperforms most of single kernel methods, but performs worse than the adaptive kernel combination methods. This confirms that relying on a pre-selected kernel function is not sufficient to guarantee low fitting loss, while allowing the MKL approaches to select the best kernel combinations in a data-driven fashion holds the key for improved performance.

In most tested datasets, Raker obtains function approximants with lower MSE relative to MKL alternatives without RF approximation. Furthermore, incorporating multiple Raker instances with variable learning rates, AdaRaker consistently yields the lowest MSE in all the tests. As it has been shown in the synthetic data test, the sizable performance gain of AdaRaker appears when the underlying nonlinear models change in the tested time-series datasets. This observation is aligned with our design principle of AdaRaker; that is, when the optimal function predictor varies slowly (fast), AdaRaker tends to select a Raker instance with small (large) learning rate. Interesting enough, even with adaptive learning rate, AdaMKL does not perform as well as OMKL in some tests. This is partially because unlike AdaRaker with fixed number of RFs, each instance in AdaMKL involves a different *number of support vectors* (samples). The instance operating on the longest interval contains at most $T/2$ support vectors, which may deteriorate performance relative to OMKL with T support vectors.

Table 6 further compares the MSE performance of AdaRaker with OMKL-B and Raker using different stepsizes. Clearly, the performance of OMKL-B and Raker is sensitive to the choice of stepsizes. While the optimal stepsize varies from dataset to dataset, selecting a constant stepsize $1/\sqrt{T}$ generally leads to better performance than a diminishing one of $\mathcal{O}(1/\sqrt{t})$. In the online scenarios however, the choice $1/\sqrt{T}$ may not be feasible if T is unknown ahead of time. In contrast, AdaRaker obtained the best MSE performance without knowing T, and without the need of stepsize selection, which confirms that AdaRaker is capable of adapting its stepsize to variable environments with unknown dynamics.

Computational complexity. The CPU time of all the considered schemes is recorded under all the tests; see Table 7. It is evident that in all tests, our RF-based MKL methods including Raker and AdaRaker are computationally more efficient than other MKL methods except that OMKL-B is faster than AdaRaker. Intuitively speaking, the per-slot complexity of Raker does not grow with time, since it requires computing only one inner product of two

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF	54.7	32.5	26.6	1.71
POLY2	2.25	1.16	2.42	0.58
POLY3	5.62	2.97	7.90	1.51
LINEAR	1.83	0.98	196	0.39
AvgMKL	148.4	81.6	82.4	5.29
OMKL	153.5	81.9	83.3	5.90
AdaMKL	164.1	102.7	117.9	35.3
OMKL-B ($B = 50$)	1.89	1.42	2.02	0.89
Raker ($D = 50$)	**0.51**	**0.38**	**0.65**	**0.28**
AdaRaker ($D = 50$)	8.64	6.03	10.94	5.28

Table 7: A summary of CPU time (second) on real datasets.

MSE	**OMKL-B**			**Raker**			**AdaRaker**		
Complexity	10	50	100	10	50	100	10	50	100
Twitter	28.9	27.0	26.1	5.9	3.0	3.0	3.8	**2.6**	**2.6**
Tom's	22.7	22.1	21.7	8.1	3.4	2.3	7.0	1.9	**1.8**
Energy	79.1	73.3	67.9	25.7	19.3	16.4	18.7	13.8	**13.3**
Air pollution	36.7	35.9	35.8	10.1	2.0	1.7	4.3	1.3	**1.2**
Twitter (Large)	25.0	20.7	19.0	3.9	3.2	3.0	3.3	**2.7**	**2.7**

Table 8: MSE (10^{-3}) versus complexity. For OMKL-B, the complexity measure is the data budget B; and for (Ada)Raker, the complexity measure is the number of RFs D.

$2D$-dimensional vectors per kernel learner, while the computational complexity of AdaMKL, OMKL, POLY, LINEAR, AvgMKL, and RBF increases with time at least linearly. With a fixed budget size, OMKL-B enjoys light-weight updates that leads to a lower CPU time than alternatives, but higher than Raker. However, given such a limited budget of data, OMKL-B exhibits higher MSE than AdaRaker and Raker; see MSE in Tables 4 and 6.

Running multiple instances of Raker in parallel, the complexity of AdaRaker is reasonably higher than Raker (roughly $\log T$ times higher), but its runtime is still only around 10% of that of AdaMKL, and significantly lower than other single-kernel alternatives especially when the actual feature dimension d is higher than the number of random features D. The computational advantage of our MKL algorithms in this test also corroborates the quantitative analysis at the end of Section 3.2. Regarding the tradeoff between learning accuracy and complexity, a delicate comparison among OMKL-B, Raker and AdaRaker follows next.

Accuracy versus complexity. To further understand the tradeoff between complexity and learning accuracy, the performance of three scalable methods AdaRaker, Raker and OMKL-B is tested under different parameter settings, e.g., D, the number of random features, and B, the number of budgeted data. The MSE performance is reported in Table 8 after one pass of all data in each dataset, while the corresponding CPU time is in Table 9.

Not surprising, all three algorithms require longer CPU time as the complexity (in terms of B or D) increases. For given complexity (same B and D), Raker requires the lowest CPU time, and its MSE is also markedly lower than that of OMKL-B in all tests. On the other hand, AdaRaker always attains the lowest MSE, and its performance gain is

Time	**OMKL-B**			**Raker**			**AdaRaker**		
Complexity	10	50	100	10	50	100	10	50	100
Twitter	1.42	1.89	2.84	0.42	0.51	0.80	7.58	8.64	11.65
Tom's	1.00	1.42	2.81	0.41	0.38	0.56	5.09	6.03	8.98
Energy	1.84	2.02	2.32	0.58	0.65	0.76	9.96	10.94	12.47
Air pollution	0.82	0.89	0.97	0.24	0.28	0.32	4.09	5.28	5.29
Twitter (Large)	12.90	16.34	23.6	6.07	6.63	8.42	67.10	78.10	109.40

Table 9: CPU time (second) versus complexity of B for OMKL-B, and D for (Ada)Raker.

	Classification error			**CPU time**		
Algorithms/Datasets	Movement	Devices	Activity	Movement	Devices	Activity
RBF ($\sigma^2 = 0.1$)	43.1	6.67	0.46	2.76	2.13	4.42
RBF ($\sigma^2 = 1$)	41.3	28.1	5.21	2.79	2.04	4.42
RBF ($\sigma^2 = 10$)	40.3	31.8	41.1	2.62	2.09	4.43
POLY2	43.5	14.3	3.13	1.63	0.31	0.81
POLY3	43.6	25.2	2.39	4.75	0.57	1.79
LINEAR	43.8	47.4	4.30	1.26	0.21	0.60
AvgMKL	41.7	23.9	3.16	10.2	6.72	14.67
OMKL	38.2	16.0	0.60	10.27	7.07	14.46
AdaMKL	3.5	0.86	0.98	33.77	10.51	21.46
M-Forgetron ($B = 50$)	1.64	0.53	1.14	0.92	0.27	0.53
Raker ($D = 50$)	9.74	2.54	0.58	**0.40**	**0.12**	**0.21**
AdaRaker ($D = 50$)	**1.10**	**0.36**	**0.34**	6.76	1.73	3.56

Table 10: Classification error (%) and runtime (second) of different algorithms with the default stepsize $1/\sqrt{T}$ for RBF, OMKL and Raker, and with complexity $B = D = 50$.

remarkable especially in the Energy and Air pollution datasets. For Twitter (Large) dataset, the performance of AdaRaker does not improve as RFs increase from $D = 50$ to $D = 100$, which implies that $D = 50$ is enough to provide reliable kernel approximation in this dataset. Considering that AdaRaker is embedded with concurrent $\log t$ Raker instances at time t, its CPU time is relatively higher. However, one would expect a major reduction in the number of concurrent instances and thus markedly lower CPU time, if a larger basic interval size (instead of base number 2 in Figure 2) is incorporated in AdaRaker real implementation.

At this point, one may wonder how many RFs are enough for Raker and AdaRaker to guarantee the same online learning accuracy as that of OMKL-B with B samples. While this intriguing question has been recently studied in the batch setting with an answer of $D = \mathcal{O}(\sqrt{B})$ RFs (Rudi and Rosasco, 2017), its thorough treatment in the online setting constitutes our future research.

5.3 Real data tests for online classification

In this section, the performance of Raker and AdaRaker is tested on real datasets for the online classification task. We use the logistic loss as the learning objective function with the regularization parameter $\lambda = 0.005$ for all considered approaches except for the perceptron-

	Classification error		
Algorithms/Datasets	Movement	Devices	Activity
RBF ($\sigma^2 = 0.1$)	28.9	5.16	0.26
RBF ($\sigma^2 = 1$)	1.27	0.42	0.53
RBF ($\sigma^2 = 10$)	**1.10**	0.36	1.14
POLY2	8.19	1.7	0.56
POLY3	15.2	17.3	0.45
LINEAR	7.46	30.7	0.60
AvgMKL	1.69	2.26	0.48
OMKL	**1.10**	0.36	0.29
AdaMKL	3.50	0.86	1.00
M-Forgetron ($B = 50$)	1.64	0.53	1.14
Raker ($D = 50$)	**1.10**	**0.28**	**0.24**
AdaRaker ($D = 50$)	**1.10**	0.36	0.34

Table 11: Classification error (%) of different algorithms with the dataset-specific optimally chosen stepsizes for RBF, OMKL and Raker, and with complexity $B = D = 50$.

based Forgetron algorithm. Kernels and all other parameters such as the default stepsizes, are chosen as those in the regression task.

Datasets description. We test classification performance on the following datasets.

- **Movement** dataset consists of $T = 13,197$ temporal streams of received signal strength (RSS) measured between the nodes of a wireless sensor network, with each $\mathbf{x}_t \in \mathbb{R}^4$ comprising 4 anchor nodes (Bacciu et al., 2014). Data has been collected during user movements at the frequency of 8 Hz (8 samples per second). The RSS samples in the dataset have been rescaled to lie in $[-1, 1]$. The binary label y_t indicates whether the user's trajectory will lead to a change in the spatial context (here a room change) or not.
- **Electronic Device** dataset consists of $T = 3,600$ samples collected as part of a government sponsored study called 'Powering the Nation,' where the feature vectors $\mathbf{x}_t \in \mathbb{R}^{60}$ represent electricity readings from different households over 15 mins, sampled within a month (Lines et al., 2011). Binary label y_t represents the type of electronic devices used at the certain interval of time time: dishwasher or kettle.
- **Human Activity** dataset consists of $T = 7,352$ samples collected from a group of 30 volunteers wearing a smartphone (Samsung Galaxy S II) on their waist to monitor activities (Anguita et al., 2013). Feature vectors $\{\mathbf{x}_t \in \mathbb{R}^{30}\}$ here measure e.g., triaxial acceleration and angular velocity, while binary label y_t represents the activity during a certain period: walking or not walking.

Classification performance. The classification error $(1/T)\sum_{t=1}^{T} \max\{0, \text{sign}(-y_t\hat{y}_t)\}$ and the CPU time of each algorithm on these datasets are summarized in Table 10 when a default stepsize $1/\sqrt{T}$ is used for POLY, LINEAR, RBF, AvgMKL, OMKL and Raker. The budget of M-Forgetron is set at $B = 50$ samples, while Raker and AdaRaker adopt $D = 50$ RFs. As with the regression tests, it is evident that AdaRaker attains the highest

	OMKL				Raker				AdaRaker
Stepsize	$1/\sqrt{T}$	$1/\sqrt{t}$	$10/\sqrt{t}$	Tuned	$1/\sqrt{T}$	$1/\sqrt{t}$	$10/\sqrt{t}$	Tuned	/
Movement	38.2	39.5	22.3	1.10	12.1	8.60	1.79	1.10	**1.10**
Devices	16.0	13.2	6.06	0.36	2.54	2.04	0.53	**0.28**	0.36
Activity	0.60	0.53	0.50	0.29	0.58	0.52	0.54	**0.24**	0.34

Table 12: Classification error (%) versus different choices of stepsizes with $B = D = 50$.

	Classification error			CPU time		
Algorithms/ Datasets	Movement	Devices	Activity	Movement	Devices	Activity
M-Forgetron ($B = 10$)	1.60	0.53	1.14	0.92	0.26	0.53
M-Forgetron ($B = 50$)	1.64	0.53	1.14	0.92	0.27	0.53
M-Forgetron ($B = 100$)	1.42	0.53	1.14	0.94	0.29	0.53
Raker ($D = 10$)	26.3	8.37	3.26	**0.35**	**0.10**	**0.18**
Raker ($D = 50$)	9.74	2.54	0.58	0.40	0.12	0.21
Raker ($D = 100$)	4.65	1.53	0.43	0.46	0.15	0.26
AdaRaker ($D = 10$)	2.46	0.66	0.68	6.13	0.65	3.22
AdaRaker ($D = 50$)	**1.10**	**0.36**	**0.34**	6.76	1.73	3.56
AdaRaker ($D = 100$)	**1.10**	**0.36**	**0.34**	7.55	2.04	4.22

Table 13: Classification error (%) and CPU time (second) versus complexity.

classification accuracy and the Raker has the lowest CPU time among all competing algorithms. Without having to tune stepsizes, the performance of AdaMKL and M-Forgetron is also competitive in this case. To explore the best performance of each algorithm, the classification performance under manually tuned stepsizes is reported in Table 11, where each algorithm uses the best stepsize among $\{10^{-3}, 10^{-2}, \cdots, 10^{3}\}/\sqrt{T}$ for each dataset. With the optimally chosen stepsizes, the performance of all algorithms improves, and Raker even achieves slightly lower classification error than AdaRaker in some datasets. This is reasonable since compared to Raker with the offline tuned stepsize, AdaRaker will incur some error due to the online adaptation to several (possibly suboptimal) learning rates.

To corroborate the effectiveness of our algorithms in adapting to unknown dynamics (e.g., unknown time horizon T and variability), Table 12 compares the performance of AdaRaker with OMKL and Raker using default, diminishing and optimally tuned stepsizes. Similar to regression tests, the performance of Raker and OMKL is sensitive to the stepsize choice, while AdaRaker achieves the desired performance by combining learners with different learning rates. By simply averaging over all the kernels, AvgMKL outperforms single kernel methods in most cases, but performs much worse than OMKL and (Ada)Raker methods. Note that the Raker also achieves competitive classification accuracy when the constant stepsize $1/\sqrt{T}$ is used. Such a choice is however not always feasible in practice, since it requires knowledge of how many data samples will be available ahead of time.

Accuracy versus complexity. In this experiment, we test classification performance in terms of both classification error and CPU time for different levels of complexity; see Table 13. We use the number of support vectors B for M-Forgetron, and the number of RFs D for Raker and AdaRaker to represent different levels of complexity, and compare

their performance using the default stepsize. It is expected that CPU time increases as the complexity increases, and the classification error decreases as the complexity grows. For all three datasets, the AdaRaker achieves the lowest classification error, and the Raker outperforms the M-Forgetron while at the same time it is more efficient computationally.

6. Concluding Remarks

This paper dealt with kernel-based learning in environments with unknown dynamics that also include static or slow variations. Uniquely combining advances in random feature based function approximation with online learning from an ensemble of experts, a scalable online multi-kernel learning approach termed Raker, was developed for static environments based on a dictionary of kernels. Endowing Raker with capability of tracking time-varying optimal function estimators, AdaRaker was introduced as an ensemble version of Raker with variable learning rates. The key modules of the novel learning approaches are: i) the random features are for scalability, as they reduce the per-iteration complexity; ii) the preselected kernel dictionary is for flexibility, that is to broaden generalizability of a kernel-based learner over a larger function space; iii) the weighted combination of kernels adjusted online accounts for the reliability of learners; and, iv) the adoption of multiple learning rates is for improved adaptivity to changing environments with unknown dynamics.

Complementing the principled algorithmic design, the performance of Raker is rigorously established using static regret analysis. Furthermore, without a-priori knowledge of dynamics, it is proved that AdaRaker achieves sub-linear dynamic regret, provided that either the loss or the optimal learning function does not change on average. Experiments on synthetic and real datasets validate the effectiveness of the novel methods.

Acknowledgments

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

Appendix A. Proof of Lemma 1

To prove Lemma 1, we introduce two intermediate lemmata as follows.

Lemma 3 *Under (as1), (as2), and* $\hat{f}_p^*$ *as in* (26) *with* $\mathcal{F}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}), \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$*, let* $\{\hat{f}_{p,t}(\mathbf{x}_t)\}$ *denote the sequence of estimates generated by Raker with a pre-selected kernel* κ_p*. Then the following bound holds true w.p.1*

$$\sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_p^*(\mathbf{x}_t)) \leq \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} \tag{42}$$

where η *is the learning rate,* L *is the Lipschitz constant in (as2), and* $\boldsymbol{\theta}_p^*$ *is the corresponding parameter (or weight) vector supporting the best estimator* $\hat{f}_p^*(\mathbf{x}) = (\boldsymbol{\theta}_p^*)^\top \mathbf{z}_p(\mathbf{x})$*.*

Proof: Similar to the regret analysis of online gradient descent (Shalev-Shwartz, 2011), using (12) for any fixed $\boldsymbol{\theta}$, we find

$$\begin{aligned}\|\boldsymbol{\theta}_{p,t+1}-\boldsymbol{\theta}\|^2 =&\|\boldsymbol{\theta}_{p,t}-\eta\nabla\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)-\boldsymbol{\theta}\|^2 \\ =&\|\boldsymbol{\theta}_{p,t}-\boldsymbol{\theta}\|^2+\eta^2\|\nabla\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\|^2-2\eta\nabla^\top\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)(\boldsymbol{\theta}_{p,t}-\boldsymbol{\theta}).\end{aligned} \tag{43}$$

Meanwhile, the convexity of the loss under (as1) implies that

$$\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)-\mathcal{L}(\boldsymbol{\theta}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\leq\nabla^\top\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)(\boldsymbol{\theta}_{p,t}-\boldsymbol{\theta}). \tag{44}$$

Plugging (44) into (43) and rearranging terms yields

$$\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)-\mathcal{L}(\boldsymbol{\theta}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\leq\frac{\|\boldsymbol{\theta}_{p,t}-\boldsymbol{\theta}\|^2-\|\boldsymbol{\theta}_{p,t+1}-\boldsymbol{\theta}\|^2}{2\eta}+\frac{\eta}{2}\|\nabla\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\|^2. \tag{45}$$

Summing (45) over $t=1,\ldots,T$, with $\hat{f}_{p,t}(\mathbf{x}_t)=\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t)$, we arrive at

$$\begin{aligned}&\sum_{t=1}^{T}\Big(\mathcal{L}(\hat{f}_{p,t}(\mathbf{x}_t),y_t)-\mathcal{L}(\boldsymbol{\theta}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\Big)\\ \leq&\frac{\|\boldsymbol{\theta}_{p,1}-\boldsymbol{\theta}\|^2-\|\boldsymbol{\theta}_{p,T+1}-\boldsymbol{\theta}\|^2}{2\eta}+\frac{\eta}{2}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top\mathbf{z}_p(\mathbf{x}_t),y_t)\|^2\\ \overset{(a)}{\leq}&\frac{\|\boldsymbol{\theta}\|^2}{2\eta}+\frac{\eta L^2T}{2}\end{aligned} \tag{46}$$

where (a) uses the Lipschitz constant in (as2), the non-negativity of $\|\boldsymbol{\theta}_{p,T+1}-\boldsymbol{\theta}\|^2$, and the initial value $\boldsymbol{\theta}_{p,1}=\mathbf{0}$. The proof of Lemma 3 is then complete by choosing $\boldsymbol{\theta}=\boldsymbol{\theta}_p^*=\sum_{t=1}^{T}\alpha_{p,t}^*\mathbf{z}_p(\mathbf{x}_t)$ such that $\hat{f}_p^*(\mathbf{x}_t)=\boldsymbol{\theta}^\top\mathbf{z}_p(\mathbf{x}_t)$ in (46). ■

Lemma 3 establishes that the static regret of the Raker is upper bounded by some constants, which mainly depend on the stepsize in (19) and the time horizon T.

In addition, we will bound the difference between the loss of the solution obtained from Algorithm 1 and the loss of the best single kernel-based online learning algorithm. Specifically the following lemma holds:

Lemma 4 *Under (as1) and (as2), with $\{\hat{f}_{p,t}\}$ generated from Raker, it holds that*

$$\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t))-\sum_{t=1}^{T}\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t))\leq\eta T+\frac{\ln P}{\eta} \tag{47}$$

where η is the learning rate in (21)*, and P is the number of kernels in the dictionary.*

Proof: Letting $W_t:=\sum_{p=1}^{P}w_{p,t}$, the weight recursion in (21) implies that

$$\begin{aligned}W_{t+1} &= \sum_{p=1}^{P}w_{p,t+1}=\sum_{p=1}^{P}w_{p,t}\exp\left(-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)\right)\\ &\leq \sum_{p=1}^{P}w_{p,t}\left(1-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)+\eta^2\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right)\end{aligned} \tag{48}$$

where the last inequality holds because $\exp(-\eta x) \leq 1-\eta x+\eta^2 x^2$, for $|\eta| \leq 1$. Furthermore, substituting $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^{P} w_{p,t} = w_{p,t}/W_t$ into (48), it follows that

$$\begin{aligned} W_{t+1} &\leq \sum_{p=1}^{P} W_t \bar{w}_{p,t} \left(1 - \eta \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right) \\ &= W_t \left(1 - \eta \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right). \end{aligned} \tag{49}$$

Using $1 + x \leq e^x$, $\forall x$, (49) leads to

$$W_{t+1} \leq W_t \exp\left(- \eta \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right). \tag{50}$$

Telescoping (50) from $t = 1$ to T, we have ($W_1 = 1$)

$$W_{T+1} \leq \exp\left(- \eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right). \tag{51}$$

On the other hand, for any p, the following holds true

$$\begin{aligned} W_{T+1} \geq w_{p,T+1} &= w_{p,1} \prod_{t=1}^{T} \exp(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)) \\ &= w_{p,1} \exp\left(- \eta \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)\right). \end{aligned} \tag{52}$$

Combining (51) with (52), we arrive at

$$\begin{aligned} &\exp\left(-\eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right) \\ &\geq w_{p,1} \exp\left(-\eta \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)\right). \end{aligned} \tag{53}$$

Taking the logarithm on both sides of (53), we find that (cf. $w_{p,1} = 1/P$)

$$-\eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\Big(\hat{f}_{p,t}(\mathbf{x}_t)\Big) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\Big(\hat{f}_{p,t}(\mathbf{x}_t)\Big)^2 \geq -\eta \sum_{t=1}^{T} \mathcal{L}_t\Big(\hat{f}_{p,t}(\mathbf{x}_t)\Big) - \ln P \tag{54}$$

which leads to

$$\sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) \leq \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2 + \frac{\ln P}{\eta} \tag{55}$$

and the proof is complete since $\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2 \leq 1$ and $\sum_{p=1}^{P} \bar{w}_{p,t} = 1$. ■

Moreover, since $\mathcal{L}_t(\cdot)$ is convex under (as1), Jensen's inequality implies that

$$\mathcal{L}_t\Big(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\Big) \leq \sum_{p=1}^{P} \bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right). \tag{56}$$

Plugging (56) into (47) in Lemma 4, we arrive at

$$\begin{aligned}\sum_{t=1}^{T}\mathcal{L}_t\Big(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\Big) &\leq \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta T + \frac{\ln P}{\eta} \\ &\overset{(a)}{\leq} \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p}^{*}(\mathbf{x}_t)\right) + \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \end{aligned} \tag{57}$$

where (a) follows because $\boldsymbol{\theta}_p^*$ is the optimal solution for any given kernel κ_p. This proves the claim in Lemma 1.

Appendix B. Proof of Theorem 1

To derive the performance bound relative to the best function estimator $f^*(\mathbf{x}_t)$ in the RKHS, the key step is to bound the error of approximation. For a given shift-invariant κ_p, the maximum point-wise error of the RF kernel approximant is uniformly bounded with probability at least $1 - 2^8\left(\frac{\sigma_p}{\epsilon}\right)^2 \exp\left(\frac{-D\epsilon^2}{4d+8}\right)$, by (Rahimi and Recht, 2007)

$$\sup_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{X}} \left|\mathbf{z}_p^\top(\mathbf{x}_i)\mathbf{z}_p(\mathbf{x}_j) - \kappa_p(\mathbf{x}_i,\mathbf{x}_j)\right| < \epsilon \tag{58}$$

where $\epsilon > 0$ is a given constant, D the number of features, while d represents the dimension of $\mathbf{x}$, and $\sigma_p^2 := \mathbb{E}_p[\|\mathbf{v}\|^2]$ is the second-order moments of the RF vector norm. Henceforth, for the optimal function estimator (26) in $\mathcal{H}_p$ denoted by $f_p^*(\mathbf{x}) := \sum_{t=1}^{T}\alpha_{p,t}^*\kappa_p(\mathbf{x},\mathbf{x}_t)$, and its RF-based approximant $\check{f}_p^* := \sum_{t=1}^{T}\alpha_{p,t}^*\mathbf{z}_p^\top(\mathbf{x})\mathbf{z}_p(\mathbf{x}_t) \in \mathcal{F}_p$, we have

$$\begin{aligned}\left|\sum_{t=1}^{T}\mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \sum_{t=1}^{T}\mathcal{L}_t\left(f_p^*(\mathbf{x}_t)\right)\right| &\overset{(a)}{\leq} \sum_{t=1}^{T}\left|\mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \mathcal{L}_t(f_p^*(\mathbf{x}_t))\right| \\ &\overset{(b)}{\leq} \sum_{t=1}^{T} L\left|\sum_{t'=1}^{T}\alpha_{p,t'}^*\mathbf{z}_p^\top(\mathbf{x}_{t'})\mathbf{z}_p(\mathbf{x}_t) - \sum_{t'=1}^{T}\alpha_{p,t'}^*\kappa_p(\mathbf{x}_{t'},\mathbf{x}_t)\right| \\ &\overset{(c)}{\leq} \sum_{t=1}^{T} L\sum_{t'=1}^{T}|\alpha_{p,t'}^*|\left|\mathbf{z}_p^\top(\mathbf{x}_{t'})\mathbf{z}_p(\mathbf{x}_t) - \kappa_p(\mathbf{x}_{t'},\mathbf{x}_t)\right| \end{aligned} \tag{59}$$

where (a) follows from the triangle inequality; (b) uses the Lipschitz continuity of the loss, and (c) is due to the Cauchy-Schwarz inequality. Combining with (58), yields

$$\left|\sum_{t=1}^{T}\mathcal{L}_t(\check{f}_p^*(\mathbf{x}_t)) - \sum_{t=1}^{T}\mathcal{L}_t(f_p^*(\mathbf{x}_t))\right| \leq \sum_{t=1}^{T} L\epsilon\sum_{t'=1}^{T}|\alpha_{p,t'}^*| \leq \epsilon LTC, \text{ w.h.p.} \tag{60}$$

where the equality follows from $C := \max_p \sum_{t=1}^T |\alpha_{p,t}^*|$. Under the kernel bounds in (as3), the uniform convergence in (58) implies that $\sup_{\mathbf{x}_t,\mathbf{x}_{t'}\in\mathcal{X}} \mathbf{z}_p^\top(\mathbf{x}_t)\mathbf{z}_p(\mathbf{x}_{t'}) \leq 1+\epsilon$, w.h.p., which in turn leads to

$$\left\|\boldsymbol{\theta}_p^*\right\|^2 := \left\|\sum_{t=1}^T \alpha_{p,t}^* \mathbf{z}_p(\mathbf{x}_t)\right\|^2 = \left|\sum_{t=1}^T\sum_{t'=1}^T \alpha_{p,t}^*\alpha_{p,t'}^* \mathbf{z}_p^\top(\mathbf{x}_t)\mathbf{z}_p(\mathbf{x}_{t'})\right| \leq (1+\epsilon)C^2 \tag{61}$$

where we again used the definition of C.

Lemma 1 together with (60) and (61) leads to the regret of the proposed Raker algorithm relative to the best static function in $\mathcal{H}_p$, that is given by

$$\begin{aligned}
&\sum_{t=1}^T \mathcal{L}_t\bigg(\sum_{p=1}^P w_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\bigg) - \sum_{t=1}^T \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \\
=&\sum_{t=1}^T \mathcal{L}_t\bigg(\sum_{p=1}^P w_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\bigg) - \sum_{t=1}^T \mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) + \sum_{t=1}^T \mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \sum_{t=1}^T \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \\
\leq& \frac{\ln P}{\eta} + \frac{\eta L^2 T}{2} + \eta T + \frac{(1+\epsilon)C^2}{2\eta} + \epsilon LTC, \text{ w.h.p.}
\end{aligned} \tag{62}$$

which completes the proof of Theorem 1.

Appendix C. Proof of Lemma 2

Using Theorem 1 with $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$, it holds w.h.p. that

$$\sum_{t=1}^T \mathcal{L}_t\bigg(\sum_{p=1}^P w_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\bigg) - \sum_{t=1}^T \mathcal{L}_t(f_{p^*}^*(\mathbf{x}_t)) \leq \left(\ln P + \frac{C^2}{2} + \frac{L^2}{2} + LC\right)\sqrt{T} := c_0\sqrt{T} \tag{63}$$

where $p^* := \arg\min_{p\in\mathcal{P}} \sum_{t=1}^T \mathcal{L}_t\left(\hat{f}_p^*(\mathbf{x}_t)\right)$. At the end of interval I, we then deduce that the static regret of the Raker learner $\mathcal{A}_I$ is (cf. (36))

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}_I}(|I|) = \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \leq c_0\sqrt{|I|}, \text{ w.h.p.} \tag{64}$$

where $\hat{f}_t^{(I)}(\mathbf{x}_t)$ is defined in (33), and $f_I^* \in \arg\min_{f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p} \sum_{t\in I} \mathcal{L}_t(f(\mathbf{x}_t))$. To this end, we sketch the main steps leading to Lemma 2 as follows.

For every interval I, the static regret of the AdaRaker can be decomposed as

$$\begin{aligned}
\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) &= \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) + \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \\
&:= \mathcal{R}_1 + \mathcal{R}_2
\end{aligned} \tag{65}$$

where the first two sums in (65) represented by $\mathcal{R}_1$ capture the regret of the Ada-Raker learner $\mathcal{A}$ relative to the Raker learner $\mathcal{A}_I$; while the last two sums in (65) forming $\mathcal{R}_2$ denote the static regret of $\mathcal{A}_I$ on this interval. Notice that $\mathcal{R}_2$ directly follows from (64), while $\mathcal{R}_1$

can be bounded following the same steps in Lemma 4. Different from the kernel selections however, the crux is that the number of Raker learners (experts) $|\mathcal{I}(t)|$ is time-varying.

A tight bound can be derived via the *Sleeping Experts* reformulation of (Luo and Schapire, 2015; Daniely et al., 2015), where an expert that has never appeared is thought of as being *asleep* for all previous rounds. For a looser bound, we assume the experts (instances $\{\mathcal{A}_I\}$) ever appeared until t are all active; that is, the total number of experts is upper bounded by $t \log t$, since at most $\log t$ experts are run during time t. Using (48)-(55), we have that

$$\mathcal{R}_1 \leq \eta^{(I)}|I| + \frac{\ln(t\log t)}{\eta^{(I)}} = \sqrt{|I|}\,(1 + \ln t + \ln(\log t)) \leq \sqrt{|I|}\,(1 + 2\ln t) \tag{66}$$

where $\eta^{(I)} := 1/\sqrt{|I|}$, and $\ln(\log t) \leq \ln(t)$. With (64), for any interval $I \in \mathcal{I}$, we have

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) = \sqrt{|I|}\,(1 + c_0 + 2\ln t) \leq \sqrt{|I|}\,(1 + c_0 + 2\ln T)\,. \tag{67}$$

Since the static regret bound (65) holds only at the end of such interval, the bound (67) only holds for those intervals (collected in $\mathcal{I}$) (re)initializing Raker instance $\mathcal{A}_I$.

The next step is to show that (67) holds for any interval $I \subseteq \mathcal{T}$, possibly $I \notin \mathcal{I}$. This is possible whenever the interval set $\mathcal{I}$ is properly designed, e.g., the interval partition given in Section 4.1. For any interval I, define the set of subintervals covered by I as $\mathcal{I}\|I := \{I' \subseteq I, I' \in \mathcal{I}\}$. As argued in (Daniely et al., 2015, Lemma 5), interval I can be partitioned into two sequences of *non-overlapping* but *consecutive* intervals, given by $\{I_{-m}, \ldots, I_0\} \subseteq \mathcal{I}\|I$ and $\{I_1, \ldots, I_n\} \subseteq \mathcal{I}\|I$, the lengths of which satisfy $|I_{k+1}|/|I_k| \leq 1/2$, $\forall k \in [1, n-1]$ and $|I_k|/|I_{k+1}| \leq 1/2$, $\forall k \in [-m, -1]$. Therefore, we have (using $\sum_{k=1}^{\infty}\sqrt{2^{-k}T_0} \leq 4\sqrt{T_0}$)

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) = \sum_{k=1}^{n-1}\mathrm{Reg}^{\mathrm{s}}_{\mathrm{Ada}}(|I_k|) + \sum_{k=-m}^{-1}\mathrm{Reg}^{\mathrm{s}}_{\mathrm{Ada}}(|I_k|) \leq C_0\sqrt{|I|} + C_1\ln T\sqrt{|I|} \tag{68}$$

where the inequality follows from (67) with $|I|$ replaced by $|I_k|$ ($\leq |I|$), and C_0, C_1 are constants depending on c_0 defined in (63). This completes the proof of Lemma 2.

Appendix D. Proof of Theorem 2

To start, the dynamic regret in (34) can be decomposed as

$$\mathrm{Reg}^{\mathrm{d}}_{\mathcal{A}}(T) := \sum_{t=1}^{T}\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T}\mathcal{L}_t(f^*(\mathbf{x}_t)) + \sum_{t=1}^{T}\mathcal{L}_t(f^*(\mathbf{x}_t)) - \sum_{t=1}^{T}\mathcal{L}_t(f_t^*(\mathbf{x}_t)) \tag{69}$$

where $f^*(\cdot)$ is the best fixed function in (26), and $f_t^*(\cdot)$ is the best dynamic function in (35), both of which belong to the union of spaces $\bigcup_{p\in\mathcal{P}}\mathcal{H}_p$. In (69), the first difference of sums is the static regret of AdaRaker, while the second difference of sums is the relative loss between the best fixed function and the best dynamic solution in the common space.

Intuitively, if the time horizon T is large, then the average static regret will become small, but the gap between the two benchmarks is large. With the insights gained from (Besbes et al., 2015; Luo et al., 2017), T essentially trades off the values of two terms. Thus,

splitting $\mathcal{T}$ into sub-horizons $\{\mathcal{T}_s\}, s = 1, \ldots, \lfloor T/\Delta T \rfloor$, each having length ΔT, the dynamic regret of AdaRaker can be bounded by

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) &= \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \sum_{t\in\mathcal{T}_s} \Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t))\Big) + \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \sum_{t\in\mathcal{T}_s} \big(\mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t)) - \mathcal{L}_t(f^*_t(\mathbf{x}_t))\big) \\ &:= \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \mathcal{R}_1 + \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \mathcal{R}_2 \end{aligned} \tag{70}$$

where the first sum over $\mathcal{T}_s$ we define as $\mathcal{R}_1$ can be bounded under AdaRaker from Lemma 2, while the second sum over $\mathcal{T}_s$ that we define as $\mathcal{R}_2$ depends on the variability of the environments $\mathcal{V}(\{\mathcal{L}_t\})$, can be bounded by (Besbes et al., 2015, Prop. 2)

$$\mathcal{R}_2 \leq 2\Delta T \mathcal{V}(\{\mathcal{L}_t\}_{t\in\mathcal{T}_s}). \tag{71}$$

Together with Lemma 2, it follows that

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) &\leq \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \Big((C_0 + C_1 \ln T)\sqrt{\Delta T} + 2\Delta T \mathcal{V}(\{\mathcal{L}_t\}_{t\in\mathcal{T}_s})\Big) \\ &= (C_0 + C_1 \ln T)\frac{T}{\sqrt{|\Delta T|}} + 2|\Delta T| \mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T), \text{ w.h.p.} \end{aligned} \tag{72}$$

Since (37) in Lemma 2 holds for any interval $\Delta T \subseteq \mathcal{T}$, after selecting ΔT so that $|\Delta T| = \big(T/\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T)\big)^{\frac{2}{3}}$, we arrive at

$$\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) \leq (C_0 + C_1 \ln T) T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T) + 2T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T), \text{ w.h.p.} \tag{73}$$

which completes the proof of Theorem 2.

Appendix E. Proof of Theorem 3

Suppose that the m-switching dynamic solution $\{\check{f}^*_t\}$ changes at slots $t_1 = 1, \ldots, t_m$, and define the m sub-intervals that partition $\mathcal{T} := \{1, \ldots, T\}$ as $\mathcal{T}_1 := [1, t_2 - 1]$, $\mathcal{T}_2 := [t_2, t_3 - 1], \ldots$, and $\mathcal{T}_m := [t_m, T]$. To use the bound in Lemma 2, we decompose the regret of AdaRaker relative to the m-switching dynamic solution $\{\check{f}^*_t\}$ by

$$\begin{aligned}\mathrm{Reg}^{m}_{\mathrm{AdaRaker}}(T) &\overset{(a)}{=} \sum_{s=1}^{m} \sum_{t\in\mathcal{T}_s} \Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(\check{f}^*_{t_s}(\mathbf{x}_t))\Big) \\ &\overset{(b)}{\leq} \sum_{s=1}^{m} \sum_{t\in\mathcal{T}_s} \Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t))\Big) \end{aligned} \tag{74}$$

where (a) holds because the definition of $\{\check{f}^*_t\}$ in (40) implies that $\check{f}^*_t = \check{f}^*_{t_s}, \forall t \in \mathcal{T}_s$, and (b) because the best fixed function is given by $f^*_{\mathcal{T}_s} \in \arg\min_{f\in f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p} \sum_{t\in\mathcal{T}_s} \mathcal{L}_t(f(\mathbf{x}_t))$. Therefore, using the regret bound of Lemma 2 in (37), we have

$$\mathrm{Reg}^{m}_{\mathrm{AdaRaker}}(T) \leq \sum_{s=1}^{m} \mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(|\mathcal{T}_s|) \leq (C_0 + C_1 \ln T) \sum_{s=1}^{m} \sqrt{|\mathcal{T}_s|}. \tag{75}$$

Holder's inequality further implies that

$$\begin{aligned}\mathrm{Reg}^m_{\mathrm{AdaRaker}}(T) &\leq (C_0 + C_1 \ln T) \left(\sum_{s=1}^{m} (1)^2 \right)^{\frac{1}{2}} \left(\sum_{s=1}^{m} \left(\sqrt{|\mathcal{T}_s|} \right)^2 \right)^{\frac{1}{2}} \\ &\leq (C_0 + C_1 \ln T) \sqrt{Tm}, \text{ w.h.p.} \end{aligned} \tag{76}$$

which completes the proof of Theorem 3.

References

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Euro. Symp. on Artificial Neural Netw., Comp. Intell. and Mach. Learn.*, Bruges, Belgium, Apr. 2013.

Davide Bacciu, Paolo Barsocchi, Stefano Chessa, Claudio Gallicchio, and Alessio Micheli. An experimental characterization of reservoir computing in ambient assisted living applications. *Neural Computing and Applications*, 24(6):1451–1464, May 2014.

Francis R. Bach. Consistency of the group lasso and multiple kernel learning. *J. Machine Learning Res.*, 9:1179–1225, Jun. 2008.

Juan Andres Bazerque and Georgios B. Giannakis. Nonparametric basis pursuit via sparse kernel-based learning: A unifying view with advances in blind methods. *IEEE Signal Processing Magazine*, 30(4):112–125, Jul. 2013.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244, Sep. 2015.

Pantelis Bouboulis, Symeon Chouvardas, and Sergios Theodoridis. Online distributed learning over networks in RKH spaces using random Fourier features. *IEEE Trans. Sig. Proc.*, to appear, 2018.

Luis M. Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, United Kingdom, 2006.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. ℓ_2-regularization for learning kernels. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, pages 109–116, Montreal, Canada, Jun. 2009.

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, pages 113–120, Sardinia, Italy, May 2010.

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F. Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Proc. Advances in Neural Info. Process. Syst.*, pages 3041–3049, Montreal, Canada, Dec. 2014.

Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from conditional distributions via dual embeddings. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, pages 1458–1467, Fort Lauderdale, FL, Apr. 2017.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *Proc. Intl. Conf. on Machine Learning*, pages 1405–1411, Lille, France, Jun. 2015.

Saverio De Vito, Ettore Massera, M Piga, L Martinotto, and G Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, Feb. 2008.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Computing*, 37(5):1342–1372, Jan. 2008.

Yi Ding, Chenghao Liu, Peilin Zhao, and Steven CH Hoi. Large scale kernel methods for online auc maximization. In *Proc. IEEE Intl. Conf. Data Mining*, pages 91–100, New Orleans, LO, November 2017.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *J. Machine Learning Res.*, 12:2211–2268, Jul. 2011.

Elad Hazan. Introduction to online convex optimization. *Found. and Trends in Mach. Learn.*, 2(3-4):157–325, 2016.

Steven CH. Hoi, Rong Jin, Peilin Zhao, and Tianbao Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, Feb. 2013.

Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *Intl. Conf. Artificial Intell. and Stat.*, San Diego, CA, May 2015.

Rong Jin, Steven CH. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Proc. Intl. Conf. on Algorithmic Learning Theory*, pages 390–404, Canberra, Australia, Oct. 2010.

François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache 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.

Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels. *IEEE Trans. Sig. Proc.*, 52(8):2165–2176, Aug. 2004.

Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Machine Learning Res.*, 5:27–72, Jan. 2004.

Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Jason Lines, Anthony Bagnall, Patrick Caiger-Smith, and Simon Anderson. Classification of household devices by electricity usage profiles. In *Intl. Conf. on Intelligent Data Engineering and Automated Learning*, pages 403–412, Norwich, United Kingdom, Sept. 2011.

Jing Lu, Steven CH. Hoi, Jialei Wang, Peilin Zhao, and Zhi-Yong Liu. Large scale online kernel learning. *J. Machine Learning Res.*, 17(47):1–43, Apr. 2016.

Jing Lu, Doyen Sahoo, Peilin Zhao, and Steven CH Hoi. Sparse passive-aggressive learning for bounded online kernel methods. *ACM Trans. Intell. Syst. Tech.*, 9(4):45, February 2018.

Haipeng Luo and Robert E. Schapire. Achieving all with no parameters: Adanormalhedge. In *Proc. Conf. on Learning Theory*, pages 1286–1304, Lille, France, Jul. 2015.

Haipeng Luo, Alekh Agarwal, and John Langford. Efficient contextual bandits in non-stationary worlds. *arXiv preprint:1708.01799*, Aug. 2017.

Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious URLs: An application of large-scale online learning. In *Proc. Intl. Conf. Mach. Learn.*, pages 681–688, Montreal, Canada, Jun. 2009.

Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *J. Machine Learning Res.*, 6:1099–1125, Jul. 2005.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1177–1184, Vancouver, Canada, Dec. 2007.

Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *J. Machine Learning Res.*, 9:2491–2521, Nov. 2008.

Cédric Richard, José Carlos M. Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *IEEE Trans. Sig. Proc.*, 57(3):1058–1067, Mar. 2009.

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Proc. Advances in Neural Info. Process. Syst.*, pages 3218–3228, Long Beach, CA, Dec. 2017.

Doyen Sahoo, Steven CH. Hoi, and Bin Li. Online multiple kernel regression. In *Proc. Intl. Conf. Knowledge Discovery and Data Mining*, pages 293–302, New York, NY, Aug. 2014.

Doyen Sahoo, Steven CH. Hoi, and Peilin Zhao. Cost sensitive online multiple kernel classification. In *Proc. Asian Conf. Machine Learning*, pages 65–80, Hamilton, New Zealand, Nov. 2016.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Found. and Trends in Mach. Learn.*, 4(2):107–194, 2011.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, United Kingdom, 2004.

Fateme Sheikholeslami, Dimitris Berberidis, and Georgios B. Giannakis. Large-scale kernel-based feature extraction via budgeted nonlinear subspace tracking. *IEEE Trans. Sig. Proc.*, 66(8):1967–1981, April 2018.

Yanning Shen and Georgios B Giannakis. Online identification of directional graph topologies capturing dynamic and nonlinear dependencies. In *Proc. of IEEE Data Science Workshop*, pages 195–199, Lausanne, Switzerland, June 2018.

Yanning Shen, Brian Baingana, and Georgios B. Giannakis. Nonlinear structural vector autoregressive models for inferring effective brain network connectivity. Oct. 2016. URL `https://arxiv.org/abs/1610.06551`.

Yanning Shen, Brian Baingana, and Georgios B. Giannakis. Kernel-based structural equation models for topology identification of directed networks. *IEEE Trans. Sig. Proc.*, 65 (10):2503–2516, May 2017.

Yanning Shen, Tianyi Chen, and Georgios B. Giannakis. Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments. In *Proc. of Intl. Conf. on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands, April 2018.

Vladimir G. Vovk. A game of prediction with expert advice. In *Proc. Annual Conf. Computational Learning Theory*, pages 51–60, Santa Cruz, CA, Jul. 1995.

Grace Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, PA, 1990.

Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *J. Machine Learning Res.*, 13:3103–3131, Oct. 2012.

Christopher K.I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Proc. Advances in Neural Info. Process. Syst.*, pages 682–688, Vancouver, Canada, Dec. 2001.

Felix Yu, Ananda Theertha Suresh, Krzysztof Choromanski, Daniel Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1975–1983, Barcelona, Spain, Dec. 2016.