Trading Dynamic Regret for Model Complexity in Nonstationary Nonparametric Optimization

Amrit Singh Bedi, Alec Koppel, Ketan Rajawat, and Brian M. Sadler

***Abstract*—Online convex optimization against dynamic comparators in the literature is limited to linear models. In this work, we relax this requirement and,propose a memory-efficient *online* universal function approximator based on compressed kernel methods. Our approach hinges upon viewing non-stationary learning as online convex optimization with dynamic comparators, for which performance is quantified by dynamic regret. Prior works control dynamic regret growth only for linear models. In contrast, we hypothesize actions belong to reproducing kernel Hilbert spaces (RKHS). We propose a functional variant of online gradient descent (OGD) operating in tandem with greedy subspace projections. Projections are necessary to surmount the fact that RKHS functions have complexity proportional to time. For this scheme, we establish sublinear dynamic regret growth in terms of the functional path length, and that the memory of the function sequence remains moderate. Experiments demonstrate the usefulness of the proposed technique for online nonlinear regression and classification problems with non-stationary data.**

I. Introduction

Many model fitting and control systems design tasks may be formulated as an optimization problem involving a training data set, as in regression [], classification, or driving a linear dynamical system to a goal state []. The samples comprising these training sets are typically assumed independent and identically distributed. However, in problems where the dynamics underlying data in transient phase, i.e., robotic platform in motion or an opinion model with long-range dependences [], the sampling distributions become *non-stationary*. While methods for optimization problems defined by data whose distributions are static is well-explored, dynamic learning tools are less so, and the focus of this work.fill in appropriate references

In pursuit of developing learning tools for dynamic problems, we focus on online convex optimization (OCO) [12], a distribution-free framework in which at each time, a learner selects action $\mathbf{x}_t$ after which an arbitrary convex cost $\ell_t : \mathbb{R}^p \to \mathbb{R}$ is evaluated as well as parameters $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^p$ of the cost ℓ_t, i.e., the learner suffers cost $\ell(\mathbf{x}_t)$. In classic OCO, one compares cost with a single best action in hindsight; however, with non-stationarity, the quintessential quantifier of performance is instead *dynamic regret*, defined as the cost accumulation as compared with a best action at each time:

$$\mathbf{Reg}_T^D = \sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \sum_{t=1}^{T} \ell_t(\mathbf{x}_t^\star) \tag{1}$$

A.S. Bedi and A. Koppel contributed equally to this work. They both are with the U.S. Army Research Laboratory, Adelphi, MD, USA (e-mail: amrit0714@gmail.com, alec.e.koppel.civ@mail.mil). K. Rajawat is with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (e-mail: ketan@iitk.ac.in). B. M. Sadler is a senior scientist with the U.S. Army Research Laboratory, Adelphi, MD, USA (email:brian.m.sadler6.civ@mail.mil).

where $\mathbf{x}_t^\star = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} \ell_t(\mathbf{x}_t)$. OCO concerns the design of methods such that $\mathbf{Reg}_T^D$ grows sublinearly in horizon T for a given sequence $\mathbf{x}_t$, i.e., the average regret goes to null with T (referred to as no-regret [17]).

Classical approaches to time-series suppose the current estimate depends linearly on its past values, as in autoregressive models [1], for which parameter tuning is not difficult [2]. While successful in simple settings, these approaches do not apply to classification, alternate quantifiers of model fitness, or universal statistical models such as deep networks [3] or kernel methods [4]. In the presence of non-stationarity, efforts to train models beyond linear have focused on recurrent networks [6] or compressive sensing [**?**], but such approaches inherently require the temporal patterns of the past and future to be similar. In contrast, transfer learning seeks to adapt a statistical model trained on one domain to another [7], but requires (1) data to be available in advance of training, and (2) a priori knowledge of when domain shifts happen, typically based on hand-crafted features. Meta-learning overcomes the need for hand-crafted statistics of domain shift by collecting experience over disparate domains and discerning decisions that are good with respect to several environments' training objectives [8]. Combining such approaches with deep networks have yielded compelling results recently [9], [10], although they still require (1) offline training. Hence, in domains where a priori data collection is difficult, due to, e.g., lack of cloud access or dynamic world models, transfer and meta-learning do not apply, hence, *online training* is required.

In the standard OCO, Typically, actions $\mathbf{x}_t$ lie in a Euclidean space $\mathbb{R}^p$. In contrast, we hypothesize actions $f_t \in \mathcal{H}$ belong to a *function space* $\mathcal{H}$ motivated by nonparametric regression whose details will be deferred to later sections [16]. Since the actions now belong to the function space $\mathcal{H}$, the definition of dynamic regret in (1) is modified as

$$\mathbf{Reg}_T^D = \sum_{t=1}^{T} L_t(f_t(\mathbf{S}_t)) - \sum_{t=1}^{T} L_t(f_t^\star(\mathbf{S}_t)) \tag{2}$$

where $f_t^\star = \operatorname{argmin}_{f\in\mathcal{H}} L_t(f(\mathbf{S}_t))$ and $\mathbf{S}_t$ denoted the set of data points available for instant t at which f_t is evaluated. We define (2) in terms of an augmented cost-data pair $(L_t, \mathbf{S}_t)$ which arises from several times, either due to new or previously observed pairs $(\ell_t, \mathbf{x}_t)$. Specifications of L_t to time-windowing or batching are discussed in Sec. II. OCO concerns the design of methods such that $\mathbf{Reg}_T^D$ grows sublinearly in horizon T for a given sequence f_t, i.e., the average regret goes to null with T (referred to as no-regret [17]). Observe that $\mathbf{Reg}_T^D$, in general, decouples the problem into T time-invariant optimization problems since the minimizer is inside the sum. However, in practice, temporal dependence is intrinsic, as

in wireless communications [18], autonomous path planning [19], or obstacle detection [20].

A. Related Work and Contributions

OCO seeks to develop algorithms whose regret grows sublinearly in time horizon T. In the static case, the simplest approach is online gradient descent (OGD), which selects the next action to descend along the gradient of the loss at the current time. OGD attains static regret growth $\mathcal{O}(T^{1/2})$ when losses are convex [17] and $\mathcal{O}(\log T)$ strongly convex [27], respectively. See Table I for a summary of related works.

The work in [22] shows that dynamic regret to be an irreducible function of quantifiers of the problem dynamics called the cost function variation V_T and variable variation W_T (definitions in Sec. II). Thus, several works establish sublinear growth of dynamic regret up to factors depending on V_T and W_T, i.e., $\mathcal{O}(T^{1/2}(1+W_T))$ for OGD or mirror descent with convex losses [17], [21], more complicated expressions that depend on D_T, the variation of instantaneous gradients [23], and $\mathcal{O}(1+W_T)$ for strongly convex losses [24].

The aforementioned works entirely focus on the case where decisions define a linear model $\mathbf{w}_t \in \mathcal{W} \subset \mathbb{R}^p$, which, by the estimation-approximation error tradeoff [11], yield small dynamic regret at the cost of large approximation error. Hypothetically, one would like actions to be chosen from a universal function class such as a deep neural network (DNN) [28], [29] or RKHS [30] while attaining no-regret. It's well-understood that no-regret algorithms often prescribe convexity of the loss with respect to actions as a prerequisite [12], thus precluding the majority of DNN parameterizations. While exceptions to this statement exist [31], instead we focus on parameterizations defined in nonparametric statistics [16], namely, RKHS [4], due to the fact they yield universality *and* convexity. Doing so allows us to attain methods that are *both* no-regret and universal in the non-stationary setting. We note that [26] considers a similar setting based on random features [32], but its design cannot be tuned to the learning dynamics; and yields faster regret growth.

Contributions We propose a variant of OGD adapted to RKHS. A challenge for this setting is that the function parameterization stores all observations from the past [33], via the Representer Theorem [34]. To surmount this hurdle, we greedily project the functional OGD iterates onto subspaces constructed from subsets of points observed thus far which are ϵ-close in RKHS norm (Algorithm 1), as in [35], [36], which allows us to explicitly tune the sub-optimality caused by function approximation, in contrast to random feature expansions [32]. Doing so allows us to establish sublinear dynamic regret in terms of the function space path length (Theorem 1). Moreover, the learned functions yield finite memory (Lemma 1). In short, we derive a tunable tradeoff between memory and dynamic regret, establishing for the first time global convergence for a universal function class in the non-stationary regime (up to metrics of non-stationarity [22]). These results translate into experiments in which one may gracefully address online nonlinear regression and classification problems with non-stationary data, contrasting alternative kernel methods and other state of the art online learning methods.

II. Non-Stationary Learning

In this section, we clarify details of the loss, metrics of non-stationarity, and RKHS representations that give rise to the derivation of our algorithms in Sec. III. To begin, we assume Tikhonov regularization, i.e., $\ell_t(f(\mathbf{x})) := \check{\ell}_t(f(\mathbf{x})) + (\lambda'/2)\|f\|_{\mathcal{H}}^2$ for some convex function $\check{\ell}_t : \mathcal{H}\times\mathcal{X} \to \mathbb{R}$, which links these methods to follow the *regularized leader* in [12].

Time-Windowing and Mini-Batching To address when the solutions $f_t^\star$ are correlated across time or allow for multiple samples per time slot, we define several augmentations of loss data-pairs $(\ell_t, \mathbf{x}_t)$.

(i) Classical loss: $L_t = \ell_t$ and $\mathbf{S}_t = \mathbf{x}_t$, and the minimization may be performed over a single datum. In other words, the action taken depends only on the present, as in fading wireless communication channel estimation.

$$
\begin{aligned}
&(ii)\ H\text{-Window} : L_t(f(\mathbf{S}_t)) = \sum_{\tau=t-H+1}^{t} \ell_\tau(f(\mathbf{x}_\tau))\ , \\
&(iii)\ \text{Mini-batch} : L_t(f(\mathbf{S}_t)) = \sum_{i=1}^{B} \ell_t(f(\{\mathbf{x}_t^i\}_{i=1}^B)). \qquad (3)
\end{aligned}
$$

The first cost $L_t(f(\mathbf{S}_t))$ in (3)*(ii)* for each time index t consists $H-1$ previous cost-data pairs $\{\ell_\tau, \mathbf{x}_\tau\}_{\tau=t-P+1}^{t-1}$ and new cost-data pair $(\ell_t, \mathbf{x}_t)$, where we denote samples $\{\mathbf{x}_\tau\}$ in this time window as $\mathbf{S}_t$. $H = 1$ simplifies to dynamic regret as in [26]. (3) is useful for, e.g., obstacle avoidance, where obstacle is correlated with time. Typically, we distinguish between the sampling rate of a system and the rate at which model updates occur. If one takes B samples per update, then mini-batching is appropriate, as in (3)*(iii)* . In this work, we focus windowing in (3)*(ii)*, i.e., $H > 1$. Further, instead of one point at t given by $\mathbf{x}_t$, one may allow B points $\{\mathbf{x}_i\}_{i=1}^B$, yielding a hybrid of (3)*(ii)* - *(iii)*. Our approach naturally extends to mini-batching. For simplicity, we focus on $B = 1$. We denote $\check{L}_t$ as the component of (3) without regularization.

Metric of Non-Stationarity With the loss specified, we shift focus to illuminating the challenges of non-stationarity. As mentioned in Sec. I, [22] establishes that designing no-regret [cf. (2)] algorithms against dynamic comparators when cost functions change arbitrarily is impossible. Moreover, dynamic regret is shown to be an irreducible function of fundamental quantifiers of the problem dynamics called cost function optimal variable variation W_T defined as

$$
W_T := \sum_{t=1}^{T} \|f_{t+1}^\star - f_t^\star\|_{\mathcal{H}} \qquad (4)
$$

which quantifies the drift of the optimal function $f_t^\star$ over time t. One may interpret (4) as the distribution-free analogue of mixing conditions in stochastic approximation with dependent noise in [37] and reinforcement learning [38]. Then, our goal is to design algorithms whose growth in dynamic regret (2) is sub-linear, up to constant factors depending on the fundamental quantity in (4).

III. Algorithm Definition

Reproducing Kernel Hilbert Space With the metrics and motivation clear, we detail the function class $\mathcal{H}$ that defines

Reference	Regret Notion	Loss	Function Class	Regret Bound
[17], [21]	$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}_t)$	Convex	Parametric	$\mathcal{O}\left(\sqrt{T}(1+W_T)\right)$
[22]	$\sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\mathbf{w}_t)\right] - \ell_t(\mathbf{w}_t^*)$	Convex	Parametric	$\mathcal{O}\left(T^{2/3}(1+W_T)^{1/3}\right)$
[22]	$\sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\mathbf{w}_t)\right] - \ell_t(\mathbf{w}_t^*)$	Strongly convex	Parametric	$\mathcal{O}(\sqrt{T(1+W_T)})$
[23]	$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}_t^*)$	Convex	Parametric	$\mathcal{O}\left(\sqrt{D_T+1} + \min\left\{\sqrt{(D_T+1)V_T}, [(D_T+1)W_T T]^{1/3}\right\}\right)$
[24], [25]	$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}_t^*)$	Strongly convex	Parametric	$\mathcal{O}(1+W_T)$
[26]	$\sum_{t=1}^{T} \ell_t(f_t(\mathbf{w}_t)) - \sum_{t=1}^{T} \ell_t(f_t^\star(\mathbf{w}_t))$	Convex	Nonparametric	$\mathcal{O}\left(T^{\frac{2}{3}} V_T^{1/3}\right)$
This Work	$\sum_{t=1}^{T} L_t(f_t(\mathbf{S}_t)) - \sum_{t=1}^{T} L_t(f_t^\star(\mathbf{S}_t))$	Convex	Nonparametric	$\mathcal{O}\left(1 + T\sqrt{\epsilon} + W_T\right)$
This Work	$\sum_{t=1}^{T} L_t(f_t(\mathbf{S}_t)) - \sum_{t=1}^{T} L_t(f_t^\star(\mathbf{S}_t))$	Strongly convex	Nonparametric	$o\left(1 + T\sqrt{\epsilon} + W_T\right)$

TABLE I: Summary of related works on dynamic online learning. In this work, we have derived the dynamic regret both in terms of V_T and W_T with an additional compression parameter ϵ to control complexity of nonparametric functions, which permits sublinear regret growth for dynamic regret in terms of W_T under selection $\epsilon = \mathcal{O}\left(T^{-\alpha}\right)$ with $\alpha \in (0, \frac{1}{p}]$, where p is the parameter dimension. Note that for the strongly convex case with $\epsilon = 0$, we obtain $o(1+W_T)$ which is better than its parametric counterpart obtained in [24]make sure this appears on page 2, not page 3 .

how decisions f_t are made. As mentioned in Sec. I, we would like one that satisfies universal approximation theorems [30], i.e., the hypothesis class containing the Bayes optimal [11], while also permitting the derivation of no-regret algorithms through links to convex analysis. RKHSs [4] meet these specifications, and hence we shift to explaining their properties. A RKHS is a Hilbert space equipped with an inner product-like map called a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies

$$\text{(i) } \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \,, \qquad \text{(ii) } \mathcal{H} = \overline{\text{span}\{\kappa(\mathbf{x}, \cdot)\}} \tag{5}$$

for all $\mathbf{x} \in \mathcal{X}$. Common choices κ include the polynomial kernel and the radial basis kernel, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$ and $\kappa(\mathbf{x}, \mathbf{x}') = e^{-(\|\mathbf{x}-\mathbf{x}'\|_2^2)/2c^2}$, respectively, where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. For such spaces, the function $f^\star(\mathbf{x})$ that minimizes the sum, $R(f; \{\mathbf{x}_t\}_{t=1}^T) = \frac{1}{T}\sum_{t=1}^{T} \ell_t(f; (\mathbf{x}_t))$, over T losses satisfies the Representer Theorem [39], [34]. Specifically, the optimal f may be written as a weighted sum of kernels evaluated *only* at training examples as $f(\mathbf{x}) = \sum_{t=1}^{T} w_t \kappa(\mathbf{x}_t, \mathbf{x})$, where $\mathbf{w} = [w_1, \cdots, w_T]^T \in \mathbb{R}^T$ denotes a set of weights. We define the upper index T as the *model order*.

One may substitute this expression into the minimization of $R(f)$ to glean two observations from the use of RKHS in online learning: the latest action is a weighted combination of kernel evaluations at previous points, e.g., a mixture of Gaussians or polynomials centered at previous data $\{\mathbf{x}_u\}_{u \le T}$; and that the function's complexity becomes unwieldy as time progresses, since its evaluation involves all past points. Hence, in the sequel, we must control both the growth of regret *and* function complexity.

Functional Online Gradient Descent Begin with functional online gradient method, akin to [33]:

$$\begin{aligned} f_{t+1} =& (1 - \eta H \lambda') f_t - \eta \nabla_f \check{L}_t(f_t(\mathbf{S}_t)) \\ =& (1 - \eta H \lambda') f_t - \eta \sum_{\tau=t-P+1}^{t} \check{\ell}_\tau'(f_t(\mathbf{x}_\tau))\kappa(\mathbf{x}_\tau, \cdot) \,, \end{aligned} \tag{6}$$

where the later equality makes use of the definition of $L_t(f_t(\mathbf{S}_t))$ [cf. (3)], the chain rule, and the reproducing property of the kernel (5) – see [33]. We define $\lambda = \lambda' H$. Step-size $\eta > 0$ is chosen as a small constant – see Section IV. We require that, given $\lambda > 0$, the step-size satisfies $\eta < 1/\lambda$ and initialization $f_0 = 0 \in \mathcal{H}$. Given this initialization, one may apply induction and Representer Theorem [34] to write the function f_t at time t as a weighted kernel expansion over past data $\mathbf{x}_t$ as

$$f_t(\mathbf{x}) = \sum_{u=1}^{t-1} w_u \kappa(\mathbf{x}_u, \mathbf{x}) = \mathbf{w}_t^T \boldsymbol{\kappa}_{\mathbf{X}_t}(\mathbf{x}) \,. \tag{7}$$

On the right-hand side of (7) we have introduced the notation $\mathbf{X}_t = [\mathbf{x}_1, \cdots, \mathbf{x}_{t-1}] \in \mathbb{R}^{p \times (t-1)}$, $\boldsymbol{\kappa}_{\mathbf{X}_t}(\cdot) = [\kappa(\mathbf{x}_1^1, \cdot), \cdots, \kappa(\mathbf{x}_{t-1}, \cdot)]^T$, and $\mathbf{w}_t = [w_1; \cdots; w_{t-1}]$. We may glean from (7), that the functional update (6) amounts to updates on the data matrix $\mathbf{X}$ and coefficient w_{t+1}:

$$\mathbf{X}_{t+1} = [\mathbf{X}_t, \ \mathbf{x}_t], \quad w_{t+1} = -\eta \check{\ell}_t'(f_t(\mathbf{x}_t)) \,, \tag{8}$$

In addition, we need to update the last $H-1$ weights over range $\tau = t - H + 1$ to $t-1$:

$$w_\tau = \begin{cases} (1-\eta\lambda) w_\tau - \eta \check{\ell}_\tau'(f_t(\mathbf{x}_\tau)) & \text{for } \tau \in \{t-H+1, \ldots, t-1\} \\ (1-\eta\lambda) w_\tau & \text{for } \tau < t - H + 1. \end{cases} \tag{9}$$

Observe that (8) causes $\mathbf{X}_{t+1}$ to have one more column than $\mathbf{X}_t$. Define the *model order* as number of points (columns) M_t in the data matrix at time t. $M_t = t - 1$ for OGD, growing unbounded.

Model Order Control via Subspace Projection To overcome the aforementioned bottleneck, we propose projecting the OGD sequence (6) onto subspaces $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$ defined by some dictionary $\mathbf{D} = [\mathbf{d}_1, \ \ldots, \ \mathbf{d}_M] \in \mathbb{R}^{p \times M}$, i.e., $\mathcal{H}_{\mathbf{D}} = \{f \ : \ f(\cdot) = \sum_{t=1}^{M} w_t \kappa(\mathbf{d}_t, \cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\} = \text{span}\{\kappa(\mathbf{d}_t, \cdot)\}_{t=1}^{M}$, inspired by [35]. For convenience we have defined $[\boldsymbol{\kappa}_{\mathbf{D}}(\cdot) = \kappa(\mathbf{d}_1, \cdot) \ldots \kappa(\mathbf{d}_M, \cdot)]$, and $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We ensure parsimony by ensuring $M_t \ll t$.

Rather than allowing model order of f to grow in perpetuity [cf. (8)], we project f onto subspaces defined by dictionaries $\mathbf{D} = \mathbf{D}_{t+1}$ extracted from past data. Deferring the selection of $\mathbf{D}_{t+1}$ for now, we note it has dimension $p \times M_{t+1}$, with $M_{t+1} \ll t$. Begin by considering function f_{t+1} is parameterized by dictionary $\mathbf{D}_{t+1}$ and weight vector $\mathbf{w}_{t+1}$. Moreover,

Algorithm 1 Dynamic Parsimonious Online Learning with Kernels (DynaPOLK)

Require: $\{\mathbf{x}_t, \eta, \epsilon\}_{t=0,1,2,\ldots}$
  initialize $f_0(\cdot) = 0, \mathbf{D}_0 = [], \mathbf{w}_0 = []$, i.e. initial dictionary, coefficient vectors are empty
  for $t = 0, 1, 2, \ldots$ **do**
    Obtain independent data realization $(\mathbf{x}_t)$ and loss $\ell_t(\cdot)$
    Compute unconstrained functional online gradient step

$$\tilde{f}_{t+1}(\cdot) = (1 - \eta\lambda) f_t - \eta \nabla_f \check{L}_t(f_t(\mathbf{S}_t))$$

    Revise dict. $\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \mathbf{x}_t]$, weights $\mathbf{w}_{t+1}$ via (11)-(12)
    Compress function via KOMP [40] with budget ϵ
      $(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \textbf{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon)$
  end for

we denote columns of $\mathbf{D}_{t+1}$ as $\mathbf{d}_t$ for $t = 1, \ldots, M_{t+1}$. We propose a projected variant of OGD:

$$f_{t+1} = \operatorname*{argmin}_{f \in \mathcal{H}_{\mathbf{D}_{t+1}}} \left\| f - \Big((1 - \eta\lambda) f_t - \eta \nabla_f \check{L}_t(f_t(\mathbf{S}_t))\Big) \right\|_{\mathcal{H}}^2$$
$$:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}} \Big[(1 - \eta\lambda) f_t - \eta \nabla_f \check{L}_t(f_t(\mathbf{S}_t))\Big] \tag{10}$$

where we define the projection operator $\mathcal{P}$ onto subspace $\mathcal{H}_{\mathbf{D}_{t+1}} \subset \mathcal{H}$ by the update (10).

Coefficient update The update (10), for a fixed dictionary $\mathbf{D}_{t+1} \in \mathbb{R}^{p \times M_{t+1}}$, implies an update only on coefficients. To illustrate this point, define the online gradient update without projection, given function f_t parameterized by dictionary $\mathbf{D}_t$ and coefficients $\mathbf{w}_t$, as $\tilde{f}_{t+1} = (1 - \eta H \lambda) f_t - \eta \nabla_f \check{L}_t(f_t(\mathbf{S}_t))$. This update may be represented using dictionary and weight vector as

$$\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, \ \mathbf{x}_t], \quad w_{t+1} = -\eta \check{\ell}_t'(f_t(\mathbf{x}_t)) \, . \tag{11}$$

and revising last $H - 1$ weights with $\tau = t - H + 1$ to $t - 1$, yielding the update for coefficients as

$$w_\tau = \begin{cases} (1 - \eta\lambda) w_\tau - \eta \check{\ell}_\tau'(f(\mathbf{x}_\tau)) & \text{for } \tau = t - H + 1, \ldots, t - 1 \\ (1 - \eta\lambda) w_\tau & \text{for } \tau < t - H + 1. \end{cases} \tag{12}$$

For fixed dictionary $\mathbf{D}_{t+1}$, the projection (10) is a least-squares problem on coefficients $\mathbf{w}_{t+1}$ [41]:

$$\mathbf{w}_{t+1} = \mathbf{K}_{\mathbf{D}_{t+1}\mathbf{D}_{t+1}}^{-1} \mathbf{K}_{\mathbf{D}_{t+1}\tilde{\mathbf{D}}_{t+1}} \tilde{\mathbf{w}}_{t+1} \, . \tag{13}$$

Given that projection of $\tilde{f}_{t+1}$ onto subspace $\mathcal{H}_{\mathbf{D}_{t+1}}$ for a fixed dictionary $\mathbf{D}_{t+1}$ is a simple least-squares multiplication, we turn to explaining the selection of the kernel dictionary $\mathbf{D}_{t+1}$ from past data $\{\mathbf{x}_u\}_{u \le t}$.

Dictionary Update One way to obtain the dictionary $\mathbf{D}_{t+1}$ from $\tilde{\mathbf{D}}_{t+1}$, as well as the coefficient $\mathbf{w}_{t+1}$, is to apply a destructive variant of *kernel orthogonal matching pursuit* (KOMP) with pre-fitting [40][Sec. 2.3] as in [35]. KOMP operates by beginning with full dictionary $\tilde{\mathbf{D}}_{t+1}$ and sequentially removing columns while the condition $\|\tilde{f}_{t+1} - f_{t+1}\|_{\mathcal{H}} \le \epsilon$ holds. The projected FOGD is defined as:

$$(f_{t+1}, \mathbf{D}_{t+1}, \mathbf{w}_{t+1}) = \textbf{KOMP}(\tilde{f}_{t+1}, \tilde{\mathbf{D}}_{t+1}, \tilde{\mathbf{w}}_{t+1}, \epsilon), \tag{14}$$

where ϵ is the compression budget which dictates how many model points are thrown away during model order reduction. By design, we have $\|f_{t+1} - \tilde{f}_{t+1}\|_{\mathcal{H}} \le \epsilon$, which allows us tune ϵ to only keep dictionary elements critical for online descent directions. These details allow one to implement Dynamic Parsimonious Online Learning with Kernels (DynaPOLK) (Algorithm 1) efficiently. Subsequently, we discuss its theoretical and experimental performance.

IV. Balancing Regret and Model Parsimony

In this section, we establish the sublinear growth of dynamic regret of Algorithm 1 up to factors depending on (4) and the compression budget parameter that parameterizes the algorithm. To do so, some conditions on the loss, its gradient, and the data domain are required which we subsequently state.

Assumption 1. *The feature space $\mathcal{X} \subset \mathbb{R}^p$ is compact, and the reproducing kernel is bounded:*

$$\sup_{\mathbf{x} \in \mathcal{X}} \sqrt{\kappa(\mathbf{x}, \mathbf{x})} = X < \infty. \tag{15}$$

Assumption 2. *The loss $\check{\ell}_t : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ is uniformly C-Lipschitz continuous for all $z \in \mathbb{R}$:*

$$|\check{\ell}_t(z) - \check{\ell}_t(z')| \le C|z - z'|. \tag{16}$$

Assumption 3. *The loss $\check{\ell}_t(f(\mathbf{x}))$ is convex and differentiable w.r.t. $f(\mathbf{x})$ on $\mathbb{R}$ for all $\mathbf{x} \in \mathcal{X}$.*

Assumption 4. *The gradient of the loss $\nabla \ell_t(f(\mathbf{x}))$ is Lipschitz continuous with parameter $\tilde{L} > 0$:*

$$\|\nabla_f \ell_t(f(\mathbf{S}_t)) - \nabla_g \ell_t(g(\mathbf{S}_t))\|_{\mathcal{H}} \le \tilde{L} \|f - g\|_{\mathcal{H}} \tag{17}$$

for all t and $f, g \in \mathcal{H}$.

Assumption 1 and Assumption 3 are standard [33], [42]. Assumptions 2 and 4 ensures the instantaneous loss $\check{\ell}_t(\cdot)$ and its derivative are smooth, which is usual for gradient-based optimization [43], and holds, for instance, for the square, squared-hinge, or logistic losses. Because we are operating under the windowing framework over last P losses (3), we define the Lipschitz constant of $L_t(\cdot)$ as CP and that of its gradient as $L = H\tilde{L}$. Doing so is valid, as the sum of Lipschitz functions is Lipschitz [44].

Before analyzing the regret of Alg. 1, we discern the influence of the learning rate, compression budget, and problem parameters on the model complexity of the function. In particular, we provide a minimax characterization of the number of points in the kernel dictionary in the following lemma, which determines the required complexity for sublinear dynamic regret growth in different contexts.

Lemma 1. *Let f_t be the function sequence of Algorithm 1 with step-size $\eta < \min\{1/\lambda, 1/L\}$ and compression ϵ. Denote M_t as the model order (no. of columns in dictionary $\mathbf{D}_t$) of f_t. For a Lipschitz Mercer kernel κ on compact set $\mathcal{X} \subseteq \mathbb{R}^p$, there exists a constant Y s.t. for data $\{\mathbf{x}_t\}_{t=1}^{\infty}$, M_t satisfies*

$$H \le M_t \le Y (CH)^p \left(\frac{\eta}{\epsilon}\right)^p. \tag{18}$$

Lemma 1 establishes that the model order of the learned function is lower bounded by the time-horizon H and its upper bound depends on the ratio of the step-size to the compression budget, as well as the Lipschitz constant [cf. (16)]. Next, we shift to characterizing the dynamic regret of Algorithm 1.

We note that the path length (4) is unique when losses are strongly convex. On the other hand, when costs are non-strongly convex, then (4) defines a set of optimizers. Thus, these cases must be treated separately. First, we introduce an assumption used in the second part of the upcoming theorem.

Assumption 5. *The instantaneous loss $L_t : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ is strongly convex with parameter μ:*

$$L_t(f) - L_t(\tilde{f}) \geq \mu \|f - \tilde{f}\|_{\mathcal{H}}^2 \tag{19}$$

for all t and any functions $f, \tilde{f} \in \mathcal{H}$.

With the technical setting clarified, we may now present the main theorem regarding dynamic regret in terms of path length (4).

Theorem 1. *Denote $\{f_t\}$ as the function sequence generated by Algorithm 1 run for T iterations. Under Assumptions 1-4, with regularization $\lambda > 0$ the following dynamic regret bounds hold in terms of path length* (4) *and compression budget ϵ:*

(i) when costs ℓ_t are convex, regret is sublinear with $\eta < \min\{\frac{1}{\lambda}, \frac{1}{L}\}$ and for any $\epsilon = \mathcal{O}\left(T^{-\alpha}\right)$ with $\alpha \in (0, \frac{1}{p}]$, we have

$$\begin{aligned} \textbf{\textit{Reg}}_T^D &= \mathcal{O}\left(\frac{1 + T\sqrt{\epsilon} + W_T}{\eta}\right) \\ &= \mathcal{O}\left(1 + T\sqrt{\epsilon} + W_T\right). \end{aligned} \tag{20}$$

(ii) Alternatively, if the cost functions ℓ_t are strongly convex, i.e., Assumption 5 holds, with $\eta < \min\{\frac{1}{\lambda}, \frac{\mu}{L^2}\}$ and for any $\epsilon = \mathcal{O}\left(T^{-\alpha}\right)$ with $\alpha \in (0, \frac{1}{p}]$, we have

$$\begin{aligned} \textbf{\textit{Reg}}_T^D =& \mathcal{O}\left(\frac{1 + T\sqrt{\epsilon} + W_T}{1 - \rho}\right) \\ =& o\left(1 + T\sqrt{\epsilon} + W_T\right), \end{aligned} \tag{21}$$

where $\rho := \sqrt{(1 - 2\eta(\mu - \eta L^2))} \in (0, 1)$ is a contraction constant for a given η.

Theorem 1 generalizes[1] existing dynamic regret bounds of [17], [21], [22], [24] to the case where decisions are defined by functions f_t belonging to RKHS $\mathcal{H}$. To facilitate this generalization, gradient projections are employed to control function complexity, which appears as an additional term depending on compression budget ϵ in the dynamic regret bounds, in particular, the product $T\sqrt{\epsilon}$ in the expressions (20) and (21). For smaller ϵ, the regret is smaller, but the model complexity increases, and vice versa. Overall, this compression induced error in the gradient is a version of inexact functional gradient descent algorithm with a tunable tradeoff between convergence accuracy and memory. Note that for $\epsilon = 0$, these results becomes of the order of $\mathcal{O}(1 + W_T)$ which matches [24] and improves upon existing results [17], [21], [22]. Even for the strongly convex case with $\epsilon = 0$, we obtain $o(1 + W_T)$ which is better than its parametric counterpart obtained in [24].

[1]See URL for the proofs. we should put the url in the references. It's clumsy/unsightly to have it here. Plus it makes the document seem more official if we call it an ARL Technical Report (Preprint)

α	Regret	M	Comments
$\alpha = 0$	$\mathcal{O}(T) + W_T$	$\mathcal{O}(1)$	Linear regret
$\alpha = \frac{1}{p}$	$\mathcal{O}\left(T^{\frac{(2p-1)}{2p}} + W_T\right)$	$\mathcal{O}(T)$	Linear M
$\alpha = \frac{1}{p+1}$	$\mathcal{O}\left(T^{\frac{2p+1}{2p+2}} + W_T\right)$	$\mathcal{O}(T^{p/(1+p)})$	Sublinear M

TABLE II: Summary of dynamic regret rates for convex loss function. Note that the same rates are obtained for the strongly convex loss function but $\mathcal{O}$ is replaced by small o.

Regarding the complexity reduction technique for kernel methods, we note that dynamic regret bounds for random feature approximations have been recently established [26] in terms of the loss function variation not W_T which is considered in this paper. These results hinge upon tuning the random feature incurred error to gradient bias. However, in practice, the number of random features required to ensure a specific directional bias is unknown, which experimentally dictates one using a large enough number of random features to hope the bias is small. However, this error is in the *function representation* itself, not the gradient direction. This issue could be mitigated through double kernel sampling [42], a technique whose use in non-stationary settings remains a direction for future research.

Parameter Selection For step-size $\eta < \min\{\frac{1}{\lambda}, \frac{1}{L}\}$ and compression budget $\epsilon = \mathcal{O}(T^{-\alpha})$, substituted into Lemma 1 yields model complexity $M = \mathcal{O}(T^{\alpha p})$. To obtain sublinear regret (up to factors depending on W_T) *and* model complexity in the non-strongly convex case, we require $\alpha \in (0, \frac{1}{p}]$ and $\alpha p \in (0, 1)$, which holds, for instance, if $\epsilon = T^{-1/(p+1)}$. Note that the dynamic regret result in (20) and the model order, using Lemma 1, becomes

$$\textbf{Reg}_T^D = \mathcal{O}\left(1 + T^{(1-\frac{\alpha}{2})} + W_T\right), \quad M = \mathcal{O}(T^{\alpha p}). \tag{22}$$

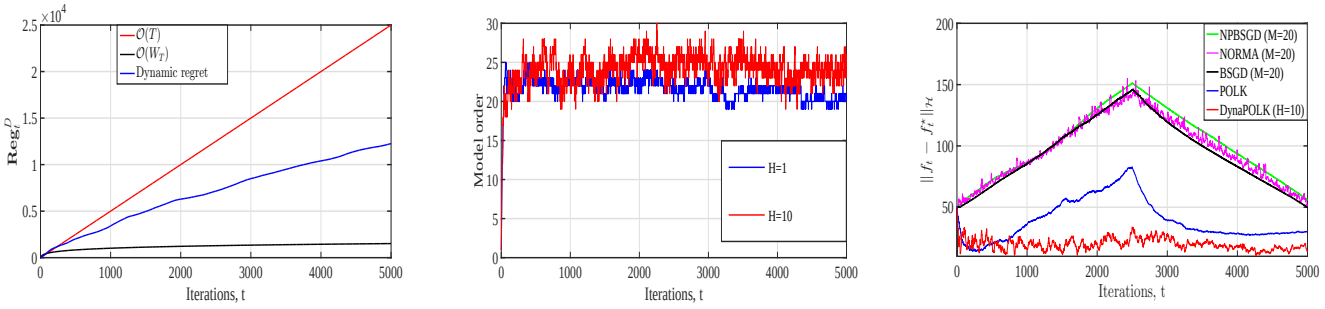
For the regret to be sublinear, we need $\alpha \in (0, \frac{1}{p}]$. As long as the dimension p is not too large, we always have a range for α. This implies that $\alpha p \in (0, 1)$ and hence M is sublinear.

Observe that the rate for the strongly convex case (21) is strictly better the non-strongly convex counterpart (21) whenever η satisfies $(1 - \rho) > \eta$. This holds, provided $\eta < (2(\mu - 1))/(2L^2 - 1)$. Taken together, Theorem 1 establish that Algorithm 1 is effective for non-stationary learning problems. In the next section, we experimentally benchmark these results on representative tasks.

V. Experiments

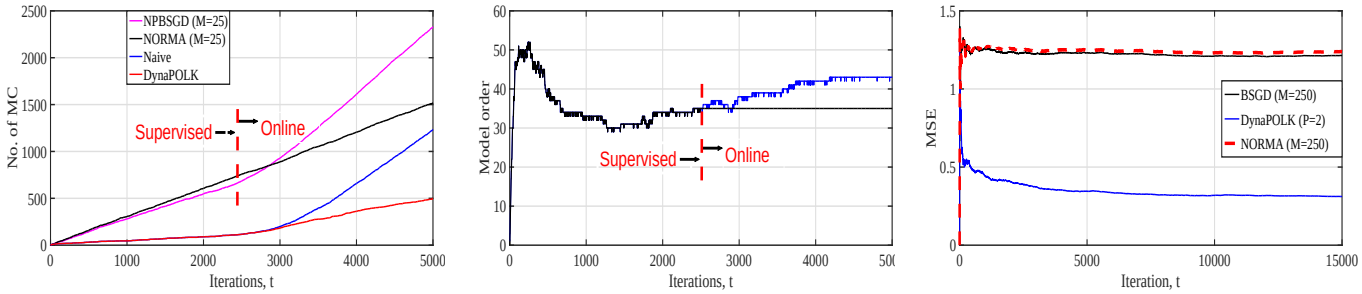
In this section, we evaluate the ability of Algorithm 1 to address online regression in non-stationary regimes and compare it with some alternatives.

Online Regression We first consider a simple online regression to illustrate performance: target variables are of the form $y_t = a_t \sin(b_t \mathbf{x}_t + c_t) + \eta_t$, which one would like to predict upon the basis of sequentially observed values of $\mathbf{x}_t$. Here $\eta \sim \mathcal{N}(\mu_t, \sigma^2)$ is Gaussian noise. Such models arise in phase retrieval, as in medical imaging, acoustics, or communications. Non-stationarity comes from parameters



(a) Dynamic regret (b) Model order M_t (c) Non-Stationary Tracking

Fig. 1: Experiments with non-stationary nonlinear regression common to phase retrieval: scalar targets are $y_t = a_t \sin(b_t \mathbf{x}_t + c_t) + \eta_t$, which one would like to predict via sequentially observed $\mathbf{x}_t$, where η_t is additive Gaussian noise. DynaPOLK attains sublinear regret, and is able to track a shifting nonlinearity with low model complexity. In contrast, alternatives are unable to adapt to drift.



(a) No. of misclassifications (MC) (b) Model order M_t (c) MSE for Classification problem

Fig. 2: Comparison of DynaPOLK to other kernel methods (left) for an online non-stationary classification on Gaussian Mixtures data [45] with dynamic class means. Alternative methods experience nearly linear regret, and their mean-square error on the time-series classification problem defined in [46] is relatively uncontrolled (right).

(a_t, b_t, c_t) changing with t: a_t and c_t increase from 0 to 3 and then decrease to 1, both linearly, while b_t is increased from 0 to 1 linearly. We consider a square loss function given by $\ell_t(f(\mathbf{x})) = (f(\mathbf{x}) - y_t)^2$ and run the simulations for $T = 5000$ iterations. For experiments, we select Gaussian kernels of bandwidth $\sigma = 0.252$, step-size $\eta = T^{-0.4}$, and compression parameter $\epsilon = T^{-0.1}$. The dynamic regret for $H = 1$ is shown in Fig. 1(a) – observe that it grows sublinearly with time. Path length W_T is shown for reference. Fig. 1(b) shows the model order relative to time for window lengths $H = 1$ and $H = 10$, which remains moderate. Observe that Algorithm 1 is able to track shifting data more gracefully with larger H as clear from Fig. 3(a). This figure shows the true function at the first and last time, i.e., $f_1(x)$ at iteration 1 to $f_T(x)$ at iteration T. The red curve shows the learned function via DynaPOLK, which better adheres to the target for $H = 10$. An animation video of online nonlinear regression in the presence of non-stationarity is appended to this submission. We further compare DynaPOLK against the alternative methods, namely, NPBSGD [47], NORMA [33], BSGD [48], and POLK [35]. We plot the distance from the optimal $\|f_t - f_t^\star\|_{\mathcal{H}}$ in Fig. 1(c). Fig. 1(c) we observe DynaPOLK with $H = 10$ is able to track the time-varying nonlinearity, whereas the others experience nearly linear regret during the non-stationary phase. We remark that a recent algorithm AdaRaker is proposed in [26] to solve the nonparametric online learning problems. The

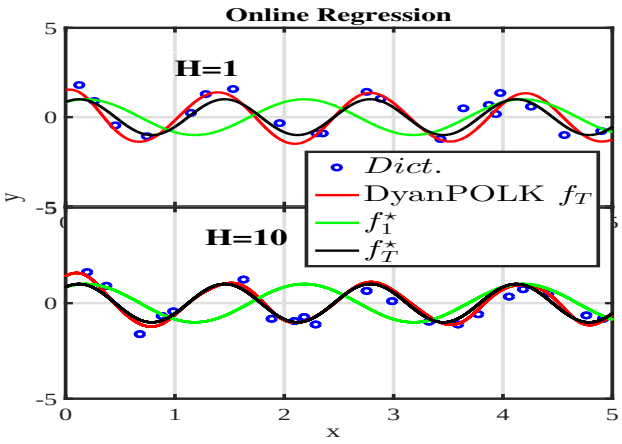
Algorithms/Dataset	**Twitter**	**Tom**	**Energy**	**Air**
AdaRaker	2.6	1.9	13.8	1.3
DyanPOLK	**0.06**	**0.68**	**0.0052**	**0.14**
Model order (DyanPOLK)	50	24	31	33

TABLE III: MSE (10^{-3}) performance of the different algorithms with $B = D = 50$ (as in [26]).

authors in [26] shows that AdaRaker performs better than all the other available techniques in the literature. Hence, in this work, we compare the proposed DynaPOLK algorithm mainly with the algorithms of [26] and show the improvement as provided in Table III (see [26] for the datasets description).

VI. Conclusion

In this work, we focused on non-stationary learning, for which we proposed an online universal function approximator based on compressed kernel methods. We characterized its dynamic regret as well as its model efficiency, and experimentally observed it yields a favorable tradeoffs for learning in the presence of non-stationarity. Future questions involve the development of model order as use for change point detection, improving the learning rates through second-derivative information, variance reduction, or strong convexity, and coupling it to the design of learning control systems.



(a) Initial *and* final nonlinearity

Fig. 3: Left: regression with initial & final target denoted as $f_1^\star$ & $f_T^\star$. DynaPOLK tracks nonlinearity drifting with (a_t, b_t, c_t). Windowing $(H = 10)$ improves performance.

REFERENCES

[1] H. Akaike, "Fitting autoregressive models for prediction," *Ann. Inst. Stat. Math.*, vol. 21, no. 1, pp. 243–247, 1969.

[2] D. R. Brillinger, *Time series: data analysis and theory*. Siam, 1981, vol. 36.

[3] S. Haykin, "Neural networks: A comprehensive foundation," *Macmillan College Publishing Company*, 1994.

[4] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[5] S. Haykin, A. H. Sayed, J. R. Zeidler, P. Yee, and P. C. Wei, "Adaptive tracking of linear time-variant systems by extended rls algorithms," *IEEE Transactions on signal processing*, vol. 45, no. 5, pp. 1118–1128, 1997.

[6] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, vol. 5.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[8] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[9] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[10] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," *arXiv preprint arXiv:1710.11622*, 2017.

[11] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[12] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[13] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009, vol. 48.

[14] M. Mohri and A. Rostamizadeh, "Stability bounds for stationary φ-mixing and β-mixing processes," *J. Mach. Learn. Res.*, vol. 11, no. Feb, pp. 789–814, 2010.

[15] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based rl," *arXiv preprint arXiv:1812.07671*, 2018.

[16] L. Wasserman, *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[17] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th ICML*, vol. 20, no. 2, Washington DC, USA, Aug. 21-24 2003, pp. 928–936.

[18] R. W. Heath and A. Paulraj, "A simple scheme for transmit diversity using partial channel feedback," in *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No. 98CH36284)*, vol. 2. IEEE, 1998, pp. 1073–1078.

[19] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 4312–4320.

[20] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of IEEE ICRA*, vol. 2, 2010.

[21] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 647–662, 2015.

[22] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Ope. Res.*, vol. 63, no. 5, pp. 1227–1244, 2015.

[23] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Artificial Intelligence and Statistics*, 2015, pp. 398–406.

[24] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *IEEE 55th CDC*, 2016, pp. 7195–7201.

[25] A. S. Bedi, P. Sarma, and K. Rajawat, "Tracking moving agents via inexact online gradient descent algorithm," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 202–217, Feb 2018.

[26] Y. Shen, T. Chen, and G. B. Giannakis, "Random feature-based online multi-kernel learning in environments with unknown dynamics," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 773–808, 2019.

[27] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, 2007.

[28] V. Tikhomirov, "On the representation of continuous functions of several variables as superpositions of continuous functions of one variable and addition," in *Selected Works of AN Kolmogorov*. Springer, 1991, pp. 383–387.

[29] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.

[30] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.

[31] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 146–155.

[32] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Adv. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.

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

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

[35] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," *J. Mach. Learn Res.*, vol. 20, no. 3, pp. 1–44, 2019.

[36] A. Koppel, "Consistent online gaussian process regression without the sample complexity bottleneck," in *IEEE ACC*. IEEE, 2019.

[37] V. S. Borkar, "Stochastic approximation with 'controlled markov' noise," *Systems & control letters*, vol. 55, no. 2, pp. 139–145, 2006.

[38] P. Karmakar and S. Bhatnagar, "Two time-scale stochastic approximation with controlled markov noise and off-policy temporal-difference learning," *Mathematics of Operations Research*, vol. 43, no. 1, pp. 130–151, 2017.

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

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

[41] C. K. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Adv. Neural Inf. Process. Syst.*, 2001, pp. 682–688.

[42] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 3041–3049.

[43] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[44] W. Rudin *et al.*, *Principles of mathematical analysis*. McGraw-hill New York, 1964, vol. 3.

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

[46] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proc. Seventh ACM SIGKDD Int. Conf. Knowledge Discovery and Data mining*. ACM, 2001, pp. 377–382.

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

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