Large-scale Online Kernel Learning with Random Feature Reparameterization

Tu Dinh Nguyen†, Trung Le†, Hung Bui‡, Dinh Phung†
‡Adobe Research, Adobe Systems, Inc.
†Center for Pattern Recognition and Data Analytics, Deakin University, Australia
{tu.nguyen, trung.l, dinh.phung}@deakin.edu.au, hubui@adobe.com

Abstract

A typical online kernel learning method faces two fundamental issues: the complexity in dealing with a huge number of observed data points (a.k.a the curse of kernelization) and the difficulty in learning kernel parameters, which often assumed to be fixed. Random Fourier feature is a recent and effective approach to address the former by approximating the shift-invariant kernel function via Bocher's theorem, and allows the model to be maintained directly in the random feature space with a fixed dimension, hence the model size remains constant w.r.t. data size. We further introduce in this paper the *reparameterized random feature* (RRF), a random feature framework for large-scale online kernel learning to address both aforementioned challenges. Our initial intuition comes from the so-called 'reparameterization trick' [Kingma and Welling, 2014] to lift the source of randomness of Fourier components to another space which can be independently sampled, so that stochastic gradient of the kernel parameters can be analytically derived. We develop a well-founded underlying theory for our method, including a general way to reparameterize the kernel, and a new tighter error bound on the approximation quality. This view further inspires a direct application of stochastic gradient descent for updating our model under an online learning setting. We then conducted extensive experiments on several large-scale datasets where we demonstrate that our work achieves state-of-the-art performance in both learning efficacy and efficiency.

1 Introduction

Massive modern datasets require scalable machine learning methods. Online learning method answers to such call. It exceedingly attracts research and application interest (e.g., [Kivinen *et al.*, 2004; Lu *et al.*, 2015]) due to its seamless capacity in handling large stream of data. Unlike conventional learning algorithms that usually require an expensive procedure to retrain the model on the entire dataset when a new data sample arrives, online learning techniques utilize the 'information' from the new data instance to incrementally update the models without retraining from scratch. An early, seminal line of work is *online linear learning* [Zinkevich, 2003] whose goal is to learn a linear predictive model in the input space. However, it relies on a rather strong assumption that the data are well separated with a hyperplane, thus incapable to model nonlinearity, a nature commonly seen in real-world modern datasets. This motivates the work of *online kernel learning* [Kivinen *et al.*, 2004; Crammer *et al.*, 2006; Le *et al.*, 2016a; Le *et al.*, 2016c] that also uses a linear model, but in the high-dimensional feature space via a kernel transformation, hence offer the capacity to capture the nonlinearity in the original data space.

Nonetheless, online kernel learning methods still suffer from two serious problems when applied to large-scale datasets. First, the model size increases linearly with the data size grown over time, posing the so-called *curse of kernelization* challenge [Wang *et al.*, 2012]. This leads to very high computational complexity and memory demand. Various attempts had been proposed to overcome this problem, which relies on developing effective budget maintenance strategy, such as: removal [Dekel *et al.*, 2005], projection [Orabona *et al.*, 2009] or merging [Wang and Vucetic, 2009], to bound the model size. [Wang *et al.*, 2012] leverage the budgeted approach with stochastic gradient descent (SGD). Although these budget maintenance strategies are proven effective for normal sized datasets, their high computational costs render them inapplicable for large-scale datasets. Alternative ways are to use the random Fourier features [Rahimi and Recht, 2007], or to use Fastfood technique [Le *et al.*, 2013] to approximate the original kernel function [Lu *et al.*, 2015]. These methods first map data into a random-feature space, and then perform the SGD directly on this feature space. However, in order to achieve a good approximation, an excessive number of random features will be required, hence still leading to serious computational issue.

The second drawback is that it is often infeasible to learn kernel parameters (e.g., the width parameter in a Gaussian kernel), since the feature map in kernel function is not defined explicitly and the Fourier components in random feature are randomly sampled from a distribution. A common solution is to search over sets of parameters to choose the best one using cross-validation and leave-one-out [Nguyen *et al.*, 2016]. The grid search, however, has two key limitations: the number of trials grows exponentially with the number of

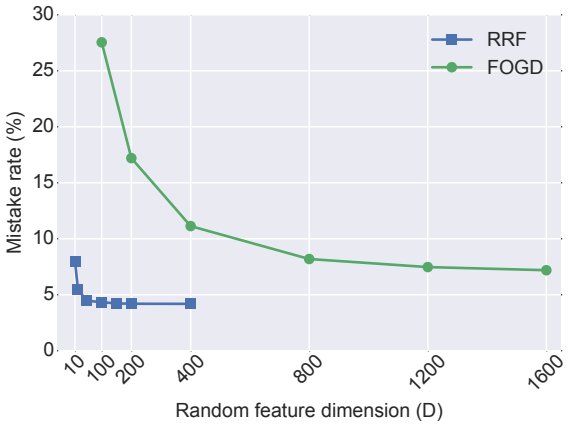

Figure 1: Comparison on the mistake rates of RRF with FOGD on the cod-rna dataset when D is varied.

parameters, hence still computational demanding; it quickly becomes intractable under the online learning setting because the search for parameters will be required after seeing each single data point; and the parameter searching is typically sub-optimal due to the discretizing step of the search space. Another solution, named *à la carte* [Yang *et al.*, 2015], is to use gradient descent to maximize the marginal likelihood of a Gaussian process (GP) that is formalized for kernel learning via Fastfood basis function expansions. This method, however, also suffers from three drawbacks. First, it is expensive to take the derivative of the marginal likelihood containing an inverse matrix. Second, the GP marginal likelihood renders it infeasible for online learning. Lastly, it is not straightforward to extend this work for classification since the GP for classification does not result in a closed-form predictive likelihood, hence requires an approximation such as Laplace or Markov chain Monte Carlo (MCMC).

In this paper, we propose a novel online learning random Fourier feature framework, termed *reparameterized random feature* (RRF). In retrospect, we independently discovered that the idea of reparameterization for kernel learning has been reported in [Băzăvan *et al.*, 2012]; our original intuition, however, comes from the so-called 'reparameterization trick' (recently proposed in the deep learning literature to train a variational Bayes autoencoder [Kingma and Welling, 2014]) to lift the source of randomness of Fourier components to another space which can be independently sampled, leaving the original kernel parameters intact so that gradient of these parameters can analytically be derived. The new kernel representation now becomes a deterministic function of the kernel parameters. Assuming differentiability, we are then able to incrementally update learn these parameters under a suitable optimization formulation. As a motivating example, this allows us to, instead of learning a single kernel shared among the features [Chapelle *et al.*, 2002], efficiently learn a different parameter for each feature, thus achieves better prediction results using a 80-fold smaller random feature dimension (D=20) than that of FOGD (D=1,600) as shown in Fig. 1. The much smaller D, as a result, significantly reduces the computational complexity of RRF which will be reported in the experiment section.

A natural and important question to ask is how good an approximating kernel via random feature representation is. To our knowledge, this question has not been properly addressed in the existing work including [Băzăvan *et al.*, 2012; Moeller *et al.*, 2016]. To fill in this gap, we provide theoretical analysis where we derive the high probability bound for the gap between the approximated kernel value and the true kernel one. We show that with a high probability, the random feature kernel can approximate the original kernel to within arbitrary ε-precision. Furthermore our new bound is considerably tighter than that of the standard random feature [Rahimi and Recht, 2007]. Our work also goes beyond existing literature to suggest a richer class of solutions for kernel learning problem across a wide spectrum of reproducing kernel Hilbert spaces (cf. Section 3.2). This can be performed by applying our framework to transform the optimization problem into a non-convex objective involving both the kernel and model parameters. This view further inspires a direct application of SGD for updating the model under an online learning setting. Moreover, in the light of our proposed framework, a class of reparameterized random features for other kernels (e.g., spectral mixture kernels [Wilson and Adams, 2013] with both parametric and nonparametric [Oliva *et al.*, 2015] versions) can also be derived as long as an appropriate mapping can be specified. These points further differentiate our work from the recent related work [Băzăvan *et al.*, 2012; Moeller *et al.*, 2016] that uses the inverse of Gaussian error function to transform the uniform space to the desirable distribution with no theoretical analysis provided to bound the approximation error. In addition, those studies are limited to batch setting due to their single gradient updates.

To demonstrate the practical applications of RRF, we further conduct extensive experiments on 7 large-scale real-world datasets on classification and regression tasks under online settings. We compare the predictive performance and running time of our proposed models with those of the state-of-the-art online learning methods. The experimental results show that our proposed RRF beats baselines by a large margin, and achieves state-of-the-art mistake rates and regression errors in all cases, whilst its execution time is orders of magnitude faster at than the baselines.

2 Random Fourier Features

Let $\mathbf{x} \in \mathbb{R}^{\mathrm{N}}$ denote the N-dimensional vector in data domain $\mathcal{X}$. The vanilla kernel methods define an implicit lifting $\boldsymbol{\phi}(\mathbf{x})$ from data space to feature space, and the inner product $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ is evaluated through a positive semi-definite kernel $\kappa(\mathbf{x}, \mathbf{x}')$ using the so-called kernel trick. To construct an explicit representation of $\boldsymbol{\phi}(\mathbf{x})$, the key idea is to approximate the original kernel $\kappa(\mathbf{x}, \mathbf{x}')$ using a kernel induced by a random finite-dimensional feature map. The mathematical tool behind this approximation is the Bochner's theorem [Bochner, 1959] which states that every shift-invariant kernel $\kappa(\mathbf{x}, \mathbf{x}')$ can be represented as an inverse Fourier transform of a proper distribution $p(\boldsymbol{\omega})$ as below:

$$\kappa(\mathbf{x}, \mathbf{x}') = k(\mathbf{u}) = \int p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top \mathbf{u}} \mathrm{d}\boldsymbol{\omega} \qquad (1)$$

where $\mathbf{u} = \mathbf{x} - \mathbf{x}'$ and i represents the imaginary unit (i.e., $i^2 = -1$). In addition, the corresponding proper distribution

$p(\boldsymbol{\omega})$ can be recovered through Fourier transform as follows:

$$p(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^{\mathrm{N}} \int k(\mathbf{u})\, e^{-i\mathbf{u}^\top \boldsymbol{\omega}} \mathrm{d}\mathbf{u} \tag{2}$$

Well-known shift-invariant kernels include Gaussian, Laplacian and Cauchy. In this work, we use a single Gaussian kernel: $\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') = \exp[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top \mathrm{diag}(\boldsymbol{\sigma})(\mathbf{x}-\mathbf{x}')]$ parameterized by a vector $\boldsymbol{\sigma} = [\sigma_n]_{n=1}^{\mathrm{N}}$ that denotes the kernel width. From Eq. (2), we obtain the proper distribution:

$$p(\boldsymbol{\omega}) \propto \int \exp\left(-\frac{1}{2}\mathbf{u}^\top \mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{u}\right) e^{-i\mathbf{u}^\top \boldsymbol{\omega}} \mathrm{d}\mathbf{u} \tag{3}$$

From Eqs.(1, 3), we can derive the kernel $\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')$ as:

$$\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') \approx \frac{1}{\mathrm{D}} \sum\nolimits_{d=1}^{\mathrm{D}} \left[\cos\left(\boldsymbol{\omega}_d^\top (\mathbf{x}-\mathbf{x}')\right)\right] \tag{4}$$

where we have sampled $\boldsymbol{\omega}_d \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\omega} \mid \mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}))$ to use Monte Carlo approximation. Eq. (4) sheds light on the construction of a 2D-dimensional random map $\mathbf{z}_{\boldsymbol{\sigma}} : \mathcal{X} \to \mathbb{R}^{2\mathrm{D}}$:

$$\mathbf{z}_{\boldsymbol{\sigma}}^\top(\mathbf{x}) = \left[\cos\left(\boldsymbol{\omega}_d^\top \mathbf{x}\right)/\sqrt{\mathrm{D}}, \sin\left(\boldsymbol{\omega}_d^\top \mathbf{x}\right)/\sqrt{\mathrm{D}}\right]_{d=1}^{\mathrm{D}} \tag{5}$$

resulting in the induced kernel $\tilde{\kappa}_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{x}') = \mathbf{z}_{\boldsymbol{\sigma}}(\mathbf{x})^\top \mathbf{z}_{\boldsymbol{\sigma}}(\mathbf{x}')$ that can accurately and efficiently approximate the original kernel: $\tilde{\kappa}_{\boldsymbol{\omega}}(\mathbf{x}, \mathbf{x}') \approx \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')$.

3 Online learning with RRF

In this section we present our main contribution of a proposed framework for large-scale kernel online learning.

3.1 Reparameterized Random Feature

The advantage of using random feature as in Eq. (5) is twofold. First, the explicit mapping with finite dimension help to avoid the curse of kernelization problem. Second, we can control the discrepancy between the original and approximate kernels by varying D. In principle, a larger D leads to a more precise approximation, but at the cost of computational complexity. In addition, since $\{\boldsymbol{\omega}_d\}_{d=1}^{\mathrm{D}}$ in Eq. (4) are i.i.d drawn from the distribution parameterized by the kernel parameters $\boldsymbol{\sigma}$: $\boldsymbol{\omega}_d \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\omega} \mid \mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}))$, it is infeasible to learn these kernel parameters. To this end, we shift this source of randomness to an auxiliary space of noise variable $\mathbf{e} \in \mathbb{R}^{\mathrm{N}}$ using the reparameterization trick as: $\boldsymbol{\omega}_d = \mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d$ where $\mathbf{e}_d \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{e} \mid \mathbf{0}, \mathbf{I})$. The mapping $\boldsymbol{\omega}_d$ now becomes differentiable function w.r.t $\boldsymbol{\sigma}$, and hence allowing to update $\boldsymbol{\sigma}$. The kernel in Eq. (4) now can be approximated as below:

$$\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') \approx \frac{1}{\mathrm{D}} \sum_{d=1}^{\mathrm{D}} \left\{\cos\left[(\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d)^\top (\mathbf{x}-\mathbf{x}')\right]\right\}$$

Therefore we can construct a new random feature map $\hat{\mathbf{z}}_{\boldsymbol{\sigma}} : \mathcal{X} \to \mathbb{R}^{2\mathrm{D}}$, termed *reparameterized random feature* (RRF), wherein $\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})^\top$ is given as:

$$\left[\cos\left((\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d)^\top \mathbf{x}\right), \sin\left((\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d)^\top \mathbf{x}\right)\right]_{d=1}^{\mathrm{D}} / \sqrt{\mathrm{D}}$$

associated with the induced kernel $\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') = \hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})^\top \hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})$ that approximates original kernel $\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')$. We now analyze the quality of this approximation.

Lemma 1. *Given $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, this statement is guaranteed*

$$\mathbb{P}\left(|\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')| \geq \varepsilon\right) \leq 2\exp\left(-\mathrm{D}\varepsilon^2/2\right)$$

Proof. Let us denote:

$$\mathbf{h}_{\mathbf{e}}(\mathbf{x}) = \left[\cos\left((\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d)^\top \mathbf{x}\right), \sin\left((\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{e}_d)^\top \mathbf{x}\right)\right]^\top$$

It is clear that: $\mathbb{E}_{\mathbf{e}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\mathbf{h}_{\mathbf{e}}(\mathbf{x})^\top \mathbf{h}_{\mathbf{e}}(\mathbf{x}')] = \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')$. We further denote $X_d = \mathrm{D}^{-1}\mathbf{h}_{\mathbf{e}_d}(\mathbf{x})^\top \mathbf{h}_{\mathbf{e}_d}(\mathbf{x}')$, $d = 1, ..., \mathrm{D}$ where $\mathbf{h}_{\mathbf{e}_d}$ is the vector associated with noise vector $\mathbf{e}_d$. It follows that $-\mathrm{D}^{-1} \leq X_d \leq \mathrm{D}^{-1}$. Let us define $S = \sum_{d=1}^{\mathrm{D}} X_d$. We then have:

$$\mathbb{E}[S] = \sum_{d=1}^{\mathrm{D}} \mathbb{E}[X_d] = \sum_{d=1}^{\mathrm{D}} \mathrm{D}^{-1}\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') = \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')$$

Using Chernoff-Hoeffding inequality, we arrive:

$$\mathbb{P}\left(|S - \mathbb{E}[S]| \geq \varepsilon\right) \leq 2\exp\left(-\mathrm{D}\varepsilon^2/2\right)$$
$$\mathbb{P}\left(|\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')| \geq \varepsilon\right) \leq 2\exp\left(-\mathrm{D}\varepsilon^2/2\right)$$

This result guarantees the exponentially fast convergence of the upper bound of approximation error probability w.r.t the number of random features D (cf., Fig. 1 in supplement[1]). □

Theorem 2. *With a probability at least $1 - 2^7\left(\frac{\mathrm{diam}(\mathcal{X})\|\boldsymbol{\sigma}\|}{\varepsilon}\right)^2 \exp\left(\frac{-\mathrm{D}\varepsilon^2}{4(\mathrm{N}+2)}\right)$ where we assume that $0 < \varepsilon \leq \mathrm{diam}(\mathcal{X})\,\|\boldsymbol{\sigma}\|$ and $\mathrm{diam}(\mathcal{X})$ is the diameter of the compact set $\mathcal{X}$, we have the following inequality*

$$\sup_{\mathbf{x},\mathbf{x}'\in\mathcal{X}} |\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')| < \varepsilon$$

Proof. (cf., Section 2 in supplementary material for more details of some derivations) Let us denote $\mathcal{U} = \{\mathbf{u} = \mathbf{x} - \mathbf{x}' : \mathbf{x}, \mathbf{x}' \in \mathcal{X}\}$. It is obvious that $\mathrm{diam}(\mathcal{U}) \leq 2\mathrm{diam}(\mathcal{X})$. It is also clear that $\mathcal{U}$ is compact. Given $r > 0$, let $\mathrm{C} = N(r, \mathcal{U}) \leq \left(\frac{2\mathrm{diam}(\mathcal{U})}{r}\right)^{\mathrm{N}} \leq \left(\frac{4\mathrm{diam}(\mathcal{X})}{r}\right)^{\mathrm{N}}$ be the covering number w.r.t the compact set $\mathcal{U}$ and the radius r. We further denote the corresponding covering set by $U = \{\mathbf{u}_c\}_{c=1}^{\mathrm{C}}$, i.e., $\mathcal{X}' \subset \cup_c \mathbf{S}(\mathbf{u}_c, r)$. Here we note that $\mathbf{S}(\mathbf{u}, r) = \{\mathbf{v} : \|\mathbf{v} - \mathbf{u}\| \leq r\}$.

We define $f(\mathbf{x}, \mathbf{x}') = \tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') = \tilde{k}_{\mathbf{e}}(\mathbf{u}) - k(\mathbf{u}) = g(\mathbf{u})$ where $\mathbf{u} = \mathbf{x} - \mathbf{x}' \in \mathcal{X}'$. We further define the Lipschitz constant of $g(\cdot)$ by L_g. Since $g(\cdot)$ is smooth, $L_g = \max_{\mathbf{u}\in\mathcal{U}} \|\nabla g(\mathbf{u})\|$. Given $\mathbf{x}, \mathbf{x}'$, there exists $\mathbf{u}_c \in \mathcal{U}$ such that $\|\mathbf{u} - \mathbf{u}_c\| \leq r$. We then have:

$$\begin{aligned} |f(\mathbf{x}, \mathbf{x}')| &= |g(\mathbf{u})| \leq |g(\mathbf{u}) - g(\mathbf{u}_c)| + |g(\mathbf{u}_c)| \\ &\leq L_g \|\mathbf{u} - \mathbf{u}_c\| + |g(\mathbf{u}_c)| \leq L_g r + |g(\mathbf{u}_c)| \end{aligned}$$

Therefore, if we ensure that $L_g < \frac{\varepsilon}{2r}$ and $|g(\mathbf{u}_c)| = |\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}_c, \mathbf{x}_c') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}_c, \mathbf{x}_c')| < \frac{\varepsilon}{2}$, $\forall c = 1, ..., \mathrm{C}$, we then have $|f(\mathbf{x}, \mathbf{x}')| < \varepsilon$. Using Markov inequality, we have:

$$\mathbb{P}\left(L_g \geq \frac{\varepsilon}{2r}\right) = \mathbb{P}\left(L_g^2 \geq \frac{\varepsilon^2}{4r^2}\right) \leq \frac{\mathbb{E}\left[L_g^2\right]}{\varepsilon^2/(4r^2)} \leq \frac{4r^2\,\|\boldsymbol{\sigma}\|^2}{\varepsilon^2}$$

[1] https://tund.github.io/papers/tu_etal_ijcai17_rrf_supp.pdf

Using Boole's inequality and Lemma 1, we obtain:

$$\mathbb{P}\left(\cup_{c=1}^{\mathrm{C}} |\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}_c, \mathbf{x}'_c) - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}_c, \mathbf{x}'_c)| \geq \frac{\varepsilon}{2}\right) \leq 2\mathrm{C}\exp\left(\frac{-\mathrm{D}\varepsilon^2}{8}\right)$$

We finally can bound the probability of interest as:

$$\mathbb{P}\left(\sup_{\mathbf{x},\mathbf{x}'\in\mathcal{X}} |\tilde{\kappa}_{\mathbf{e}}(\mathbf{x}, \mathbf{x}') - \kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}')| \geq \frac{\varepsilon}{2}\right)$$
$$\leq 2\left[4\mathrm{diam}(\mathcal{X})/r\right]^{\mathrm{N}} \exp\left(-\mathrm{D}\varepsilon^2/8\right) + 4r^2\|\boldsymbol{\sigma}\|^2/\varepsilon^2$$

The RHS has the form of $\tau_1 r^{-\mathrm{N}} + \tau_2 r^2$. Choosing $r = (\tau_1/\tau_2)^{\frac{1}{\mathrm{N}+2}}$, the RHS becomes:

$$2\sqrt{\tau_1\tau_2}\left(\frac{\tau_1}{\tau_2}\right)^{\frac{-\mathrm{N}+2}{\mathrm{N}+2}} = 2\tau_1^{\frac{2}{\mathrm{N}+2}}\tau_2^{\frac{\mathrm{N}}{\mathrm{N}+2}}$$
$$\leq 2^7\left(\frac{\mathrm{diam}(\mathcal{X})\|\boldsymbol{\sigma}\|}{\varepsilon}\right)^2 \exp\left(-\frac{\mathrm{D}\varepsilon^2}{4(\mathrm{N}+2)}\right) \quad (6)$$

This bound is considerably tighter than what of standard random feature (cf. Claim 1 of [Rahimi and Recht, 2007]). □

3.2 Online Kernel Parameter Learning

We now view the online kernel parameter learning as an optimization problem across the class of parameterized reproducing kernel Hilbert spaces (RKHS). Let $\mathcal{D} = \{(\mathbf{x}_m, \mathrm{y}_m)\}_{m=1}^{\mathrm{M}}$ be a set of M training data samples $\mathbf{x}_m$ and labels y_m. Given a kernel $\kappa_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top \mathrm{diag}(\boldsymbol{\sigma})(\mathbf{x}-\mathbf{x}')\right]$ parameterized by $\boldsymbol{\sigma} \in \mathbb{R}^{\mathrm{N}}$, let us denote the associated feature map and RKHS by $\boldsymbol{\phi}_{\boldsymbol{\sigma}}$ and $\mathbb{H}_{\boldsymbol{\sigma}}$, respectively, that is: $\boldsymbol{\phi}_{\boldsymbol{\sigma}} : \mathcal{X} \to \mathbb{H}_{\boldsymbol{\sigma}}$. To learn the kernel parameters, we propose to solve the following optimization problem:

$$\min_{\boldsymbol{\sigma}} \min_{\mathbf{w}\in\mathbb{H}_{\boldsymbol{\sigma}}} \mathcal{J}(\mathbf{w}) \triangleq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{\mathbb{P}}\left[\ell\left(\mathrm{y}, \mathbf{w}^\top\boldsymbol{\phi}_{\boldsymbol{\sigma}}(\mathbf{x})\right)\right] \quad (7)$$

wherein $\mathbb{P}$ is either the joint distribution $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ in online setting, or the empirical distribution $\mathbb{P}_{\mathcal{D}}$ in batch setting; $\lambda \geq 0$ denotes the regularization parameter; and ℓ is a *convex* loss function parameterized by $\mathbf{w}$. For binary classification, two typical examples are Hinge loss and logistic loss, which are then extended to work for multiclass classification [Crammer and Singer, 2001] with the new objective function: $\mathcal{J}(\mathbf{w}) \triangleq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{\mathbb{P}}\left[\ell\left(\mathbf{w}_{\mathrm{y}}^\top\boldsymbol{\phi}_\sigma(\mathbf{x}) - \mathbf{w}_{\mathrm{z}}^\top\boldsymbol{\phi}_\sigma(\mathbf{x})\right)\right]$ where $\mathrm{z} = \mathrm{argmax}_{k\neq\mathrm{y}}\, \mathbf{w}_k^\top\boldsymbol{\phi}_\sigma(\mathbf{x})$. For regression, three widely-used functions are: ℓ_1, ℓ_2 and ε-insensitive losses.

According to Theorem 2, each $\hat{\mathbf{z}}_{\boldsymbol{\sigma}} : \mathcal{X} \to \mathbb{R}^{2\mathrm{D}}$ is a tight approximation of $\boldsymbol{\phi}_{\boldsymbol{\sigma}}$ whose induced kernels can be arbitrarily close. Therefore we can cast the optimization problem in Eq. (7) into the following objective function:

$$\min_{\boldsymbol{\sigma}\in\mathbb{R}^{\mathrm{N}}, \mathbf{w}\in\mathbb{R}^{2\mathrm{D}}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{\mathbb{P}}\left[\ell\left(\mathrm{y}, \mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right)\right] \quad (8)$$

The advantage of the optimization problem in Eq. (8) is that, together with the model parameter $\mathbf{w}$, the kernel parameter $\boldsymbol{\sigma}$ now becomes a *learnable* variable. More specifically, we first take the gradient w.r.t parameter $\boldsymbol{\sigma}$ as: $\nabla_{\boldsymbol{\sigma}}\ell\left(\mathrm{y}, \mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right) =$

$$\sum_{d=1}^{\mathrm{D}}\left[-\mathbf{x}\sin\left(\boldsymbol{\omega}_d^\top\mathbf{x}\right), \mathbf{x}\cos\left(\boldsymbol{\omega}_d^\top\mathbf{x}\right)\right]^\top \nabla^\top_{[\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})]_d}\ell \odot \mathbf{e}_d \quad (9)$$

Algorithm 1 RRF for online learning.

Input: N, D, learning rate η

$\boldsymbol{\gamma} = \mathbf{0}$, $\mathbf{w} = \mathbf{0}$, $c = 0$, $err = 0$
$\boldsymbol{\Sigma} = [\boldsymbol{\epsilon}_d]_{d=1}^{\mathrm{D}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ with $\boldsymbol{\epsilon}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, 2, ..., \mathrm{T}$ **do**
  Receive $(\mathbf{x}_t, \mathrm{y}_t)$
  $\mathbf{u} = e^{\mathrm{diag}(\boldsymbol{\gamma})}\boldsymbol{\Sigma}$
  $\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x}) = \left[\cos\left(\mathbf{x}_t^\top\mathbf{u}\right), \sin\left(\mathbf{x}_t^\top\mathbf{u}\right)\right]^\top$
  Predict:
    $\hat{\mathrm{y}}_t = \mathrm{sign}\left(\mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right)$ # binary
    $\hat{\mathrm{y}}_t = \mathrm{argmax}\left(\mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right)$ # multiclass
    $\hat{\mathrm{y}}_t = \mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})$ for regression
  $c = c + \mathbb{I}(\mathrm{y}_t \neq \hat{\mathrm{y}}_t)$ for classification
  $err = err + (\mathrm{y}_t - \hat{\mathrm{y}}_t)^2$ for regression
  $\mathbf{w} = \mathbf{w} - \eta\nabla_{\mathbf{w}}\ell\left(\mathrm{y}_t, \mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right)$
  $\boldsymbol{\gamma} = \boldsymbol{\gamma} - \eta\left[\nabla_{\boldsymbol{\sigma}}\ell\left(\mathrm{y}_t, \mathbf{w}^\top\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})\right) \odot e^{\boldsymbol{\gamma}}\right]$
end for

Output: $\frac{c}{\mathrm{T}}$ for classification or $\sqrt{\frac{err}{\mathrm{T}}}$ for regression

The derivative of loss function ℓ w.r.t $[\hat{\mathbf{z}}_{\boldsymbol{\sigma}}(\mathbf{x})]_d$ can be computed similarity to what w.r.t $\mathbf{w}$. We then use SGD to update their values in online setting. The pseudo-code of RRF is presented in Alg. 1. Note that as the variance $\boldsymbol{\sigma}$ is constrained to be positive, we learn its log-variance $\boldsymbol{\gamma}$ instead, hence naturally inducing a positive variance.

4 Experiments

We conduct comprehensive experiments on large-scale real-world datasets to demonstrate the superior performance of our RRF, compared with recent state-of-the-art online learning approaches on classification and regression tasks.

4.1 Data statistics and experimental setup

We use 7 datasets (*casp, slice, ijcnn1*, *cod-rna*, *poker*, *year*, and *airlines*) of varied sizes to clearly expose the differences in scalable capabilities of the models. Three of which are large-scale datasets (*year*: $515,345$; *poker*: $1,025,010$; and *airlines*: $5,929,413$), whilst the rest are medium-sized databases (*casp*: $45,730$; *slice*: $53,500$; *ijcnn1*: $141,691$; and *cod-rna*: $331,152$). These datasets can be downloaded from LIBSVM and UCI websites, except the *airlines* data which was obtained from American Statistical Association (ASA[2]). For the *airlines* dataset, our aim is to predict whether a flight will be delayed or not under binary classification setting, and how long (in minutes) the flight will be delayed in terms of departure time under regression setting. A flight is considered *delayed* if its delay time is above 15 minutes, and *non-delayed* otherwise. Following the procedure in [Hensman *et al.*, 2013], we extract 8 features for flights in the year of 2008, and then normalize them into the range [0,1].

For each dataset, we perform 10 runs on each algorithm with different random permutations of the training data samples. In each run, the model is trained in a single pass through the data. Its prediction result and time spent are then reported by taking the average together with the standard de-

[2]http://stat-computing.org/dataexpo/2009/.

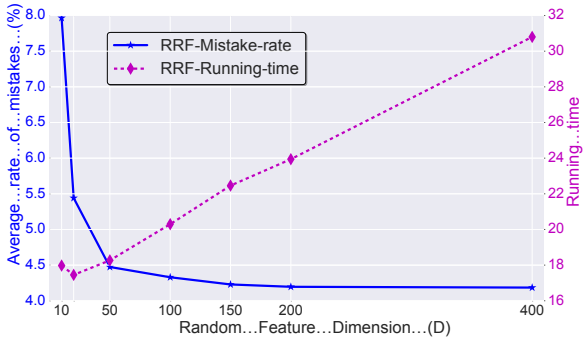

Figure 2: The effect of random feature dimension D on the mistake rate and running time.

viation over all runs. For comparison, we employ 12 state-of-the-art online kernel learning methods: perceptron [Freund and Schapire, 1999], online gradient descent (OGD) [Kivinen *et al.*, 2004], randomized budget perceptron (RBP) [Cavallanti *et al.*, 2007], forgetron [Dekel *et al.*, 2005] projectron, projectron++ [Orabona *et al.*, 2009], budgeted passive-aggressive simple (BPAS) [Wang and Vucetic, 2010], budgeted SGD using merging strategy (BSGD-M) [Wang *et al.*, 2012], bounded OGD (BOGD) [Zhao *et al.*, 2012], Fourier OGD (FOGD), Nystrom OGD (NOGD) [Lu *et al.*, 2015] and Dual Space OGD (DualSGD) [Le *et al.*, 2016b]. Their implementations are published as a part of LIBSVM, BudgetedSVM [Wang *et al.*, 2012] and LSOKL [Lu *et al.*, 2015] toolboxes. We exclude the comparison with [Băzăvan *et al.*, 2012; Moeller *et al.*, 2016; Yang *et al.*, 2015] since they were designed for batch setting.

4.2 Model evaluation on the effect of D

We first investigate the effect of hyperparameters, i.e., random feature dimension D on the performance behavior of RRF. Particularly, we conduct an initial analysis to quantitatively evaluate the sensitivity of this hyperparameter and its impact on the predictive accuracy and wall-clock time. This analysis provides an approach to find the best setting of hyperparameters. Here the RRF with Hinge loss is trained on the *cod-rna* dataset under the online classification setting.

We vary D in the range of {10, 20, 50, 100, 150, 200, 400}. For each setting, we run our models and record the average mistake rates and running time as shown in Fig. 2. There is a pattern that the classification error decreases for larger D whilst the wall-clock time increases. This represents the trade-off between model discriminative performance and model computational complexity via the number of random features. In this analysis, we can choose $\mathrm{D} = 100$ to balance the performance and computational cost.

4.3 Online classification

We now examine the performances of RRFs in the online classification task. We use 4 datasets: *cod-rna*, *ijcnn1*, *poker*, and *airlines* (delayed and non-delayed labels). We create 2 versions of our approach: RRF with Hinge loss (RRF-Hinge) and RRF with Logistic loss (RRF-Logit). It is worth mentioning that the Hinge loss is not a smooth function, i.e., its gradient is undefined at the point that the classification confidence $\mathrm{y}f(\mathbf{x}) = 1$. Following the sub-gradient definition, in our experiment, we compute the gradient if $\mathrm{y}f(\mathbf{x}) < 1$, and set it to 0 otherwise.

Dataset	***cod-rna***		***ijcnn1***	
[B | D | $\hat{\mathrm{D}}$]	[400 | 100 | 1,600]		[1,000 | 100 | 4,000]	
Algorithm	Mistake Rate	Time	Mistake Rate	Time
Perceptron	9.79±0.04	1,393	12.85±0.09	728
OGD	7.81±0.03	2,804	10.39±0.06	960
RBP	26.02±0.39	86	15.54±0.21	54
Forgetron	28.56±2.22	103	16.17±0.26	61
Projectron	11.16±3.61	97	12.98±0.23	59
Projectron++	17.97±15.60	1,800	9.97±0.09	750
BPAS	11.97±0.09	92	10.68±0.05	55
BSGD-M	5.33±0.04	185	9.14±0.18	1,563
BOGD	38.13±0.11	105	10.87±0.18	56
FOGD	7.15±0.03	53	9.41±0.03	26
NOGD	7.83±0.06	105	10.43±0.08	59
DualSGD	4.83±0.21	*32*	8.35±0.20	*12*
RRF-Logit	*4.37±0.04*	37	*4.16±0.28*	23
RRF-Hinge	**4.33±0.05**	**21**	**3.68±0.17**	**10**
Dataset	***poker***		***airlines***	
[B | D | $\hat{\mathrm{D}}$]	[1,000 | 200 | 4,000]		[1,000 | 100 | 4,000]	
Algorithm	Mistake Rate	Time	Mistake Rate	Time
FOGD	52.28±0.04	928.89	20.98±0.01	1,270.75
NOGD	44.90±0.16	4,920.33	25.56±0.01	3,553.50
DualSGD	46.65±0.14	**133.50**	19.28±0.00	*472.21*
RRF-Logit	*43.47±0.12*	186.97	**18.70±0.00**	644.30
RRF-Hinge	**43.39±0.12**	*172.19*	*18.79±0.00*	**430.54**

Table 1: Mistake rate (%) and execution time (seconds). The notation [B; D; $\hat{\mathrm{D}}$] denotes the budget size B, the number of random features D and $\hat{\mathrm{D}}$ of RRF and FOGD, respectively. The best performance is in **bold**, and the runner-up in *italic*.

For each method, we tune its regularization parameter λ or C, learning rate η, and RBF kernel width γ (our RRF can learn γ) using grid search on a subset of data. In particular, we randomly pick 10% of medium-sized datasets, but only 1% of large-scale datasets, so that the searching can finish within an acceptable time budget. The hyperparameters are varied in certain ranges and selected for the best performance (mistake rate) on these subsets. The ranges are given as follows: $\lambda \in \{2^{-4}/\mathrm{M}, 2^{-2}/\mathrm{M}, ..., 2^{16}/\mathrm{M}\}$, $\gamma \in \{2^{-8}, 2^{-4}, 2^{-2}, 2^{0}, 2^{2}, 2^{4}, 2^{8}\}$, and $\eta \in \{10^{-5}, 3 \times 10^{-5}, 10^{-4}, ..., 10^{-2}\}$ where M is the number of data points. The random feature dimension D of RRF is selected following the approach described in Section 4.2. For the budget size B in NOGD and budgeted algorithms such as RBP, BSGD-M, BOGD, and the feature dimension $\hat{\mathrm{D}}$ in FOGD for each dataset, we use identical values to those used in Section 7.1.1 of [Lu *et al.*, 2015]. For DualSGD, we adopt the results from [Le *et al.*, 2016b].

Table 1 reports the average classification results and execution time after the methods see all data samples. Note that for two biggest datasets (*poker*, *airlines*) that consist of millions of data points, we only include the fast algorithms FOGD, NOGD, DualSGD and RRFs. The other methods would exceed the time limit, which we set to two hours, when running on such data as they suffer from serious computation issue. In general, our proposed model beats all recent advanced online methods by a large margin, thus achieves state-of-the-art performance in both learning efficacy and efficiency. In particular, we can draw key observations as follows.

The budgeted online and random feature approaches show their effectiveness with substantially faster computation than

the ones without budgets. More specifically, the execution time of our proposed models is several orders of magnitude (150x) lower than that of regular online algorithms (e.g., 21 and 10 seconds compared with $2,804$ and $1,563$ seconds for *cod-rna* and *ijcnn1* datasets). Moreover, our models are twice as fast as the recent fast algorithm FOGD for *cod-rna* and *ijcnn1* datasets, and approximately 8 and 3 times for vast-sized data *poker* and *airlines*. This is because the FOGD must maintain a very high random feature space, whose dimensionality is 20 or 40 times larger than that of RRFs, to achieve respectable results.

Second, in terms of classification, the RRF-Hinge and RRF-Logit significantly outperform other methods for all datasets. RRF-based methods achieve the best mistake rates 4.33, 3.68 and 18.70 for the *cod-rna*, *ijcnn1* and *airlines* data, that are, respectively, 10%, 56% and 3% lower than the error rates of the runner-up, up-to-date model DualSGD. For *poker* dataset, our methods obtain slightly better results than that of the NOGD, but still surpass DualSGD and FOGD with a large margin. The underlying reason is that the RRF can update the kernel width parameters to reflect the information of incoming data samples. This would help the model adapt to the growth of data, allowing for more accurate predictions.

Finally, two versions of RRFs demonstrate similar discriminative performances and computational complexities wherein the RRF-Logit is slightly slower due to the additional exponential operators. All of these observations validate the state-of-the-art effectiveness and efficiency of our proposed method. Thus, we believe that the RRF is a very competitive technique for building scalable online kernel learning algorithms for large-scale classification tasks.

4.4 Online regression

The last experiment addresses the online regression problem to evaluate the capabilities of our approach with 3 proposed loss functions: ℓ_1, ℓ_2 and ε-insensitive losses. Incorporating these loss functions creates three versions: RRF-ε, RRF-ℓ_1 and RRF-ℓ_2. We use four datasets: *casp*, *slice*, *year* and *airlines* (delay minutes), and 7 baselines: RBP, Forgetron, Projectron, BOGD, FOGD, NOGD and DualSGD. We adopt the same hyperparameter searching procedure as in Section 4.3.

Table 2 reports the average regression errors and computation costs after the methods see all data samples. Our proposed method, once again, outperforms all baselines to achieve state-of-the-art prediction results within much smaller computational time budget. In particular, our models enjoy a significant advantage in computational efficacy whilst achieve better regression results than those of other methods in almost all datasets. Among the baselines, the DualSGD is the fastest, that is, its time costs can be considered to compare with those of our methods, but its regression performances are worse. The remaining algorithms usually obtain better results, but at the cost of scalability. Exceptionally in the *slice* data, the prediction performance of Projectron is surprisingly high, thus we sacrifice the running speed to improve our prediction result using a fivefold increase of D from 100 to 500. Our method now surpasses the Projectron and reduces the prediction error of FOGD by 50%.

Finally, comparing the capability of three RRF's variants, all models demonstrate similar regression capabilities and computational complexities wherein the RRF-ℓ_2 achieves the best prediction results due to the agreement of its objective function and the evaluation criteria (RMSE). The RRF-ε is faster than others since the indicator function $\mathbb{I}\{\cdot\}$ reduces a large number of operations in computing the gradient. These observations, once again, verify the state-of-the-art effectiveness and efficiency of our proposed techniques. Therefore the RRF is also a strong candidate to perform online regression task for large-scale datasets.

Dataset	***casp***		***slice***	
$[\mathrm{B} \mid \mathrm{D} \mid \hat{\mathrm{D}}]$	$[400 \mid 100 \mid 2,000]$		$[1,000 \mid 500 \mid 3,000]$	
Algorithm	RMSE	Time	RMSE	Time
RBP	0.320	7.15	0.115	810.14
Forgetron	0.317	10.14	0.113	1,069.15
Projectron	0.269	8.48	*0.077*	814.37
BOGD	0.286	6.20	0.172	816.16
FOGD	0.378	5.83	0.144	**20.65**
NOGD	0.251	6.99	0.087	812.69
DualSGD	0.320	5.81	0.240	*26.00*
RRF-ε	*0.246*	*3.82*	0.081	94.86
RRF-ℓ_1	0.250	4.13	0.084	377.78
RRF-ℓ_2	**0.241**	**3.78**	**0.076**	319.23
Dataset	***year***		***airlines***	
$[\mathrm{B} \mid \mathrm{D} \mid \hat{\mathrm{D}}]$	$[400 \mid 20 \mid 1,600]$		$[1,000 \mid 50 \mid 2,000]$	
Algorithm	RMSE	Time	RMSE	Time
RBP	0.189	605.42	36.507	3,418.89
Forgetron	0.188	904.09	36.507	5,774.47
Projectron	0.139	605.19	36.137	3,834.19
BOGD	0.201	596.10	35.735	3,058.96
FOGD	0.158	76.70	53.164	646.15
NOGD	0.138	607.37	*34.742*	3,324.38
DualSGD	*0.124*	*47.29*	36.200	*443.39*
RRF-ε	**0.122**	**33.66**	35.390	**438.81**
RRF-ℓ_1	0.127	54.21	35.384	493.91
RRF-ℓ_2	**0.122**	52.29	**33.316**	470.51

Table 2: Root mean squared error (RMSE) and execution time (seconds) of 6 baselines and 3 versions of our RRFs. $[\mathrm{B} \mid \mathrm{D} \mid \hat{\mathrm{D}}]$ denotes the same meanings as those in Table 1. The best performance is in **bold**, and the runner-up in *italic*.

5 Conclusion

We have introduced a novel random Fourier feature framework – reparameterized random feature (RRF) that addresses the two most pressing problems in online kernel learning: curse of kernelization and learning kernel parameters. More specifically, we have reparameterized the Fourier components to lift the source of randomness to another space, enabling to incrementally update the kernel parameters under an online learning setting. We provide a well-founded underlying theory for our method where we offer a principled way to reparameterize the kernel with a newly tight error bound under a different optimization view. Experiments on large-scale datasets demonstrate that our proposed model achieves state-of-the-art results in both learning efficacy and efficiency.

Acknowledgments. This work was partially supported by the Australian Research Council (ARC) and the CoE in Machine Learning and Big Data.

References

[Băzăvan *et al.*, 2012] Eduard Gabriel Băzăvan, Fuxin Li, and Cristian Sminchisescu. Fourier kernel learning. In *European Conference on Computer Vision (ECCV)*, pages 459–473. Springer, 2012.

[Bochner, 1959] Salomon Bochner. *Lectures on Fourier Integrals*, volume 42. Princeton University Press, 1959.

[Cavallanti *et al.*, 2007] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

[Chapelle *et al.*, 2002] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

[Crammer and Singer, 2001] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research (JMLR)*, 2:265–292, December 2001.

[Crammer *et al.*, 2006] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585, 2006.

[Dekel *et al.*, 2005] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems (NIPS)*, pages 259–266, 2005.

[Freund and Schapire, 1999] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[Hensman *et al.*, 2013] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, pages 1–9, 2013.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *The Inter. Conf. on Learning Representations (ICLR)*, 2014.

[Kivinen *et al.*, 2004] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, Aug 2004.

[Le *et al.*, 2013] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood - approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning (ICML)*, pages 1281–1289, 2013.

[Le *et al.*, 2016a] Trung Le, Phuong Duong, Mi Dinh, Tu D Nguyen, Vu Nguyen, and Dinh Phung. Budgeted semi-supervised support vector machine. In *Uncertainty in Artificial Intelligence (UAI)*, pages 377–386, 2016.

[Le *et al.*, 2016b] Trung Le, Tu D Nguyen, Vu Nguyen, and Dinh Phung. Dual space gradient descent for online learning. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4583–4591. 2016.

[Le *et al.*, 2016c] Trung Le, Vu Nguyen, Tu D Nguyen, and Dinh Phung. Nonparametric budgeted stochastic gradient descent. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 654–572, 2016.

[Lu *et al.*, 2015] Jing Lu, Steven CH Hoi, Jialei Wang, Peilin Zhao, and Zhi-Yong Liu. Large scale online kernel learning. *Journal of Machine Learning Research (JMLR)*, 2015.

[Moeller *et al.*, 2016] John Moeller, Vivek Srikumar, Sarathkrishna Swaminathan, Suresh Venkatasubramanian, and Dustin Webb. Continuous kernel learning. In *European Conference on Machine Learning (ECML)*, pages 657–673. Springer, 2016.

[Nguyen *et al.*, 2016] Khanh Nguyen, Trung Le, Vu Nguyen, Tu D Nguyen, and Dinh Phung. Multiple kernel learning with data augmentation. In *the 8th Asian Conference on Machine Learning (ACML)*, volume 63, pages 49–64, 16–18 Nov 2016.

[Oliva *et al.*, 2015] Junier Oliva, Avinava Dubey, Barnabas Poczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. *arXiv preprint arXiv:1506.08776*, 2015.

[Orabona *et al.*, 2009] Francesco Orabona, Joseph Keshet, and Barbara Caputo. Bounded kernel-based online learning. *Journal of Machine Learning Research (JMLR)*, 10:2643–2666, 2009.

[Rahimi and Recht, 2007] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, volume 3, 2007.

[Wang and Vucetic, 2009] Zhuang Wang and Slobodan Vucetic. Twin vector machines for online learning on a budget. In *SDM*, pages 906–917. SIAM, 2009.

[Wang and Vucetic, 2010] Zhuang Wang and Slobodan Vucetic. Online passive-aggressive algorithms on a budget. In *AISTATS*, volume 9, pages 908–915, 2010.

[Wang *et al.*, 2012] Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *JMLR*, 13(1):3103–3131, 2012.

[Wilson and Adams, 2013] Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference in Machine Learning (ICML) 2013*, volume 28, pages 1067–1075, 2013.

[Yang *et al.*, 2015] Zichao Yang, Alexander J Smola, Le Song, and Andrew Gordon Wilson. A la carte-learning fast kernels. In *18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 38, pages 1098–1106, San Diego, USA, 9–12 May 2015.

[Zhao *et al.*, 2012] Peilin Zhao, Jialei Wang, Pengcheng Wu, Rong Jin, and Steven CH Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. *arXiv preprint arXiv:1206.4633*, 2012.

[Zinkevich, 2003] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*. Carnegie Mellon University, 2003.