Large Scale Online Kernel Classification

Jialei Wang*, Steven C.H. Hoi*, Peilin Zhao*, Jinfeng Zhuang†, Zhi-yong Liu‡
*School of Computer Engineering, Nanyang Technological University, Singapore
†Microsoft Corporation, USA
‡State Key Lab of Management and Control for Complex System,
Chinese Academy of Sciences, China
{jl.wang, chhoi, peilinzhao}@ntu.edu.sg, jeffzh@microsoft.com, zhiyong.liu@ia.ac.cn

Abstract

In this work, we present a new framework for large scale online kernel classification, making kernel methods efficient and scalable for large-scale online learning tasks. Unlike the regular budget kernel online learning scheme that usually uses different strategies to bound the number of support vectors, our framework explores a functional approximation approach to approximating a kernel function/matrix in order to make the subsequent online learning task efficient and scalable. Specifically, we present two different online kernel machine learning algorithms: (i) the Fourier Online Gradient Descent (FOGD) algorithm that applies the random Fourier features for approximating kernel functions; and (ii) the Nyström Online Gradient Descent (NOGD) algorithm that applies the Nyström method to approximate large kernel matrices. We offer theoretical analysis of the proposed algorithms, and conduct experiments for large-scale online classification tasks with some data set of over 1 million instances. Our encouraging results validate the effectiveness and efficiency of the proposed algorithms, making them potentially more practical than the family of existing budget kernel online learning approaches.

1 Introduction

In the fields of machine learning and artificial intelligence, *online learning* refers to a sequential machine learning task where a predictive model is learned incrementally from a sequence of data instances [Rosenblatt, 1958]. Unlike regular batch machine learning methods [Hoi *et al.*, 2006] which usually suffer from a high re-training cost whenever new training data arrive, online learning algorithms are often very efficient and highly scalable, making them more suitable for large-scale online applications where data usually arrive sequentially and evolute dynamically and rapidly. Online learning techniques can be applied to many real-world applications, such as online spam detection, online advertising, multimedia retrieval [Xia *et al.*, 2013], and computational finance [Li *et al.*, 2012]. In this paper, we restrict our discussion of online learning methodology to tackle *online classification* tasks.

In literature, a variety of online learning techniques have been proposed to tackle online classification tasks in recent years. One popular family of online learning algorithms is to learn a linear predictive model on the input feature space, which we refer to as "linear online learning" [Rosenblatt, 1958; Crammer *et al.*, 2006; Dredze *et al.*, 2008]. The key limitation of these algorithms lies in that the linear model could be restricted to make effective classification if training data are linearly non-separable in the input feature space, which can be common for some real-world classification tasks with noisy training data of relatively low dimensionality. This motivates the studies of "kernel based online learning" or referred to as "online kernel classification" [Kivinen *et al.*, 2001; Freund and Schapire, 1999], which aims to learn kernel-based models for improving the classification of non-separable instances in the input feature space.

One of the key challenges of conventional online kernel classification is that an online learner usually has to maintain in memory a set of support vectors for representing the kernel-based predictive model. During the online learning process, whenever a new incoming training instance is wrongly classified, it typically will be added to the support vector set, making the support vector size unbounded and potentially causing memory overflow for a large-scale online learning task. To attack this challenge, an emerging research direction is to explore "budget online kernel classification" [Crammer *et al.*, 2003], which attempts to bound the number of support vectors with a fixed budget size using different budget maintenance strategies whenever the budget overflows. However, the key limitations of the existing budget online kernel methods lie in that some algorithms are too simple to achieve satisfactory approximation accuracy, while some other algorithms are too computationally intensive, making them harm the crucial merit of high efficiency of online learning techniques for large-scale applications.

Unlike the existing budget online kernel methods, in this paper, we present a novel framework of large-scale online kernel classification by exploring a completely different strategy. In particular, the key idea of our framework is to explore functional approximation techniques to approximate a kernel by transforming data from the input space to a new feature space, and then apply existing linear online learning algorithms on the new feature space. Specifically, we propose two different new algorithms: (i) Fourier Online Gradient Descent

(FOGD) algorithm which adopts the random Fourier features for approximating shift-invariant kernels and learns the subsequent model by online gradient descent; and (ii) Nyström Online Gradient Descent (NOGD) algorithm which employs the Nyström method for large kernel matrix approximation followed by online gradient descent learning. We give theoretical analysis of our proposed algorithms, and conduct an extensive set of empirical studies to examine their efficacy.

2 Related Work

Our work is related to two major categories of machine learning research: *online learning* and *kernel methods*.

Our work is closely related to online learning methods for classification tasks [Rosenblatt, 1958; Freund and Schapire, 1999; Crammer *et al.*, 2006; Zhao and Hoi, 2010; Zhao *et al.*, 2011; Wang *et al.*, 2012; Hoi *et al.*, 2013], particularly for budget online kernel machine learning where several algorithms have been proposed in literature [Crammer *et al.*, 2003; Wang and Vucetic, 2010; Zhao *et al.*, 2012]. Some well-known examples include Forgetron [Dekel *et al.*, 2005] which basically discards old support vectors when the budget overflows for adding a new support vector, and Randomized Budget Perceptron(RBP) [Cavallanti *et al.*, 2007] which randomly discards existing support vectors, and Projectron[Orabona *et al.*, 2008; 2009] which uses a sophisticated projection strategy to bound the support vector size with better approximation but falls short in its high computational cost.

Moreover, our work is also related to kernel methods for large-scale classification tasks, especially for some studies on large-scale kernel methods [Williams and Seeger, 2000; Rahimi and Recht, 2007]. In particular, we employ the pioneering technique of random Fourier features, which have been successfully used in speed up batch kernelized SVMs [Rahimi and Recht, 2007], and kernel-based clustering [Chitta *et al.*, 2012], etc. Another technique adopted in our approach is the well-known Nyström method [Williams and Seeger, 2000], which has been widely applied in machine learning tasks, including Gaussian Processes [Williams and Seeger, 2000], Kernelized SVMs [Zhang *et al.*, 2012], Kernel PCA, Spectral Clustering [Zhang and Kwok, 2009], and manifold learning [Talwalkar *et al.*, 2008]. Although these techniques have been applied for batch machine learning tasks, to the best of our knowledge, they have never been applied to online classification tasks as our approach in this paper.

3 Large Scale Online Kernel Classification

3.1 Problem Formulation

Consider an online binary classification task over a sequence of data instances $(\mathbf{x}_t, y_t)$,$t = 1, \ldots, T$, where $\mathbf{x}_t \in \mathbb{R}^d$ is the observed features of the t-th data instance and $y_t \in \{+1, -1\}$ is the true class label which is only revealed from the environment at the end of each online learning round. The goal of a conventional online kernel classification task is to learn a kernel-based predictive model $f(\mathbf{x})$ for classifying a new instance $\mathbf{x} \in \mathbb{R}^d$ as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{B} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) \tag{1}$$

where B is the number of support vectors, α_i denotes the coefficient of the i-th support vector, and $\kappa(\cdot, \cdot)$ denotes the kernel function. The existing budget online kernel classification approach aims to bound the number of support vectors by a constant B using different budget maintenance strategies.

Unlike the existing budget online kernel classification methods using the budget maintenance strategy, we propose to explore the functional approximation strategy to tackle the challenge. In particular, our goal is to construct a new representation $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^D$ such that the inner product is able to approximate the kernel function:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) \approx \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j) \tag{2}$$

By the above approximation, the model can be rewritten:

$$f(\mathbf{x}) = \sum_{i=1}^{B} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) \approx \sum_{i=1}^{B} \alpha_i \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}) = \mathbf{w}^\top \mathbf{z}(\mathbf{x}) \tag{3}$$

where $\mathbf{w}^\top = \sum_{i=1}^{B} \alpha_i \mathbf{z}(\mathbf{x}_i)$ denotes the weight vector to be learned in the new feature space. As a consequence, solving the regular online kernel classification task can be turned into a problem of an linear online classification task on the new feature space derived from the kernel approximation. In the following, we will present two online kernel classification algorithms by applying two different kernel approximation methods: (i) Fourier Online Gradient Descent and (ii) Nyström Online gradient descent methods.

3.2 Fourier Online Gradient Descent

Random fourier features can be used in shift-invariant kernels [Rahimi and Recht, 2007]. A shift-invariant kernel is the kernel that can be written as $\kappa(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1 - \mathbf{x}_2)$, where k is some function. Examples of shift-invariant kernels include some widely used kernels, such as Gaussian and Laplace kernels. By performing an inverse Fourier transform of the shift-invariant kernel function, one can obtain:

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1 - \mathbf{x}_2) = \int p(\mathbf{u}) e^{i\mathbf{u}^\top(\mathbf{x}_1 - \mathbf{x}_2)} d\mathbf{u} \tag{4}$$

where $p(\mathbf{u})$ is a proper probability density function. Given a kernel function is continuous and positive-definite, according to the Bochner's theorem [Rudin, 1990], the kernel function can be expressed as an expectation of function with a random variable $\mathbf{u}$:

$$\int p(\mathbf{u}) e^{i\mathbf{u}^\top(\mathbf{x}_1 - \mathbf{x}_2)} d\mathbf{u} = \mathrm{E}_{\mathbf{u}}[e^{i\mathbf{u}^\top \mathbf{x}_1} \cdot e^{-i\mathbf{u}^\top \mathbf{x}_2}] \tag{5}$$

$$= \mathrm{E}_{\mathbf{u}}[\cos(\mathbf{u}^\top \mathbf{x}_1)\cos(\mathbf{u}^\top \mathbf{x}_2) + \sin(\mathbf{u}^\top \mathbf{x}_1)\sin(\mathbf{u}^\top \mathbf{x}_2)]$$

$$= \mathrm{E}_{\mathbf{u}}[[\cos(\mathbf{u}^\top \mathbf{x}_1), \sin(\mathbf{u}^\top \mathbf{x}_1)] \cdot [\cos(\mathbf{u}^\top \mathbf{x}_2), \sin(\mathbf{u}^\top \mathbf{x}_2)]]$$

The equality (4) can be obtained by only keeping the real part of the complex function. From (5), we can see any shift-invariant kernel function can be expressed by the expectation of the inner product of the new representation of

original data, where the new data representation is $\mathbf{z}(\mathbf{x}) = [\cos(\mathbf{u}^\top \mathbf{x}), \sin(\mathbf{u}^\top \mathbf{x})]$. As a consequence, we can sample many random Fourier components $\mathbf{u}_1, ...\mathbf{u}_D$ independently for constructing the new representation as follows:

$$\mathbf{z}_t(\mathbf{x}) = (\sin(\mathbf{u}_1^\top \mathbf{x}), \cos(\mathbf{u}_1^\top \mathbf{x}), ..., \sin(\mathbf{u}_D^\top \mathbf{x}), \cos(\mathbf{u}_D^\top \mathbf{x})).$$

The online kernel learning task in the original input space can thus be approximated by solving a linear online learning task in the new feature space. More specifically, for a Gaussian kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2}{2\sigma^2}}$, we have the corresponding random Fourier component $\mathbf{u}$ with the distribution $p(\mathbf{u}) = \mathcal{N}(0, \sigma^{-2} I)$. For data arriving sequentially, we can construct the new representation of a data instance on-the-fly, and then perform online learning in the new feature space using the online gradient descent algorithm. We refer to the proposed algorithm as the Fourier Online Gradient Descent (FOGD), as summarized in Algorithm 1.

Algorithm 1 FOGD — Fourier Online Gradient Descent

Input: the number of Fourier components D, stepsize η;
Initialize $\mathbf{w}_1 = 0$.
Generate random Fourier components: $\mathbf{u}_1, ..., \mathbf{u}_D$ sampled from distribution $p(\mathbf{u}) = \mathcal{N}(0, \sigma^{-2} I)$.
for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$;
  Construct new representation:
  $\mathbf{z}_t(\mathbf{x}_t) = (\sin(\mathbf{u}_1^\top \mathbf{x}_t), \cos(\mathbf{u}_1^\top \mathbf{x}_t), ..., \sin(\mathbf{u}_D^\top \mathbf{x}_t), \cos(\mathbf{u}_D^\top \mathbf{x}_t))$
  Predict $\widehat{y}_t = \text{sgn}(\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t)))$;
  Receive y_t and suffer loss $\ell\big(\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t)); y_t\big)$;
  if $\ell\big(\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t)); y_t\big) > 0$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \ell\big(\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t)); y_t\big)$.
  end if
end for

3.3 Nyström Online Gradient Descent

The above random Fourier feature based approach attempts to approximate the kernel function explicitly, which is in general data independent for the given dataset and thus may not fully exploit the potential of data distribution for kernel approximation. To address this, we propose to explore the Nyström method [Williams and Seeger, 2000] to approximate a large kernel matrix by a data-dependent approach.

First we introduce some notations. We denote a kernel matrix by $\mathbf{K} \in \mathbb{R}^{T \times T}$ with rank r. We denote the Singular Value Decomposition (SVD) of $\mathbf{K}$ as $\mathbf{K} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, where the columns of $\mathbf{V}$ are orthogonal and $\mathbf{D} = \text{diag}(\sigma_1, \ldots, \sigma_r,)$ is diagonal. For $k < r$, $\mathbf{K}_k = \sum_{i=1}^k \sigma_i V_i V_i^\top = \mathbf{V}_k \mathbf{D}_k \mathbf{V}_k^\top$ is the best rank-k approximation of $\mathbf{K}$, where V_i is the i-th column of matrix $\mathbf{V}$.

Given a large kernel matrix $\mathbf{K} \in \mathbb{R}^{T \times T}$, the Nyström method randomly samples $B \ll T$ columns to form a matrix $\mathbf{C} \in \mathbb{R}^{T \times B}$, and then derive a much smaller kernel matrix $\mathbf{W} \in \mathbb{R}^{B \times B}$ based on the sampled B instances. We can in turn approximate the original large kernel matrix by

$$\hat{\mathbf{K}} = \mathbf{C}\mathbf{W}_k^+ \mathbf{C}^\top \approx \mathbf{K}, \tag{6}$$

Algorithm 2 NOGD — Nyström Online Gradient Descent

Input: the budget B, stepsize η, rank approximation k.
Initialize support vector set $\mathcal{S}_1 = \emptyset$, and model $f_1 = 0$.
if $|\mathcal{S}_t| < B$ **then**
  for $t = 1, 2, \ldots, T_0$ **do**
    Receive $\mathbf{x}_t$;
    Predict $\widehat{y}_t = \text{sgn}(f_t(\mathbf{x}_t))$;
    Update f_t by regular Online Gradient Descent (OGD)
    Update $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{t\}$ whenever loss is nonzero
  end for
end if
Construct the kernel matrix $\hat{\mathbf{K}}_t$ from $\mathcal{S}_t$.
$[\mathbf{V}_t, \mathbf{D}_t] = \mathsf{eigs}(\hat{\mathbf{K}}_t, k)$, where $\mathbf{V}_t$ and $\mathbf{D}_t$ are Eigenvectors and Eigenvalues of $\hat{\mathbf{K}}_t$, respectively.
Initialize $\mathbf{w}_t = (\alpha_1, ..., \alpha_B)(\mathbf{D}_t^{-0.5}\mathbf{V}_t^\top)^{-1}$.
for $t = T_0 + 1, \ldots, T$ **do**
  Receive $\mathbf{x}_t$;
  Construct the new representation of $\mathbf{x}_t$:
  $\mathbf{z}_t(\mathbf{x}_t) = \mathbf{D}_t^{-0.5}\mathbf{V}_t^\top(\kappa(\mathbf{x}_t, \hat{\mathbf{x}}_1), ..., \kappa(\mathbf{x}_t, \hat{\mathbf{x}}_B))^\top$.
  Predict $\widehat{y}_t = \text{sgn}((\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t))))$;
  Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \ell\big(\mathbf{w}_t^\top(\mathbf{z}_t(\mathbf{x}_t)); y_t\big)$.
end for

where $\mathbf{W}_k$ is the best rank-k approximation of $\mathbf{W}$, $\mathbf{W}^+$ denotes the pseudo inverse of matrix $\mathbf{W}$.

We now apply the above Nyström kernel approximation to large-scale online kernel classification task. Similar to the previous approach, the key idea is how to construct the new representation for every newly arrived data instance based on the kernel approximation principle. In particular, we propose the following scheme: (i) at the very early stage of the online classification task, we simply run any existing online kernel classification methods (e.g., kernel-based online gradient descent in our approach) whenever the number of support vectors is smaller than a predefined budget size B. Once the budget is reached, we then use the stored B instances (support vectors) to approximate the kernel value of any new instances (which is equivalent to using the first B columns to approximate the whole kernel matrix). From the approximated kernel matrix in (6), we could see the kernel value between i-th point $\mathbf{x}_i$ and j-th point $\mathbf{x}_j$ is approximated by

$$\hat{\kappa}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{C}_i \mathbf{V}_k \mathbf{D}_k^{-\frac{1}{2}})(\mathbf{C}_j \mathbf{V}_k \mathbf{D}_k^{-\frac{1}{2}})^\top =$$

$$([\kappa(\mathbf{x}_1, \mathbf{x}_i), ..., \kappa(\mathbf{x}_B, \mathbf{x}_i)]\mathbf{V}_k \mathbf{D}_k^{-\frac{1}{2}})(\kappa(\mathbf{x}_1, \mathbf{x}_j), ..., \kappa(\mathbf{x}_B, \mathbf{x}_j)\mathbf{V}_k \mathbf{D}_k^{-\frac{1}{2}})^\top$$

For a new instance, we construct the new representation as:

$$\mathbf{z}_t(\mathbf{x}) = [\kappa(\mathbf{x}_1, \mathbf{x}), ..., \kappa(\mathbf{x}_B, \mathbf{x})]\mathbf{V}_k \mathbf{D}_k^{-\frac{1}{2}})$$

Similarly, we can then apply the existing online gradient descent algorithm to learn the linear predictive model on the new feature space induced from the kernel. We denote the proposed algorithm the Nyström Online Gradient Descent (NGOD), as summarized in Algorithm 2.

4 Theoretical Analysis

This section aims to analyze the theoretical properties of the two proposed algorithms.

Theorem 1. *Assume we learn with an RBF Kernel of bandwidth σ, i.e., $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2/2\sigma^2)$, and the original data is contained by a ball $\mathbb{R}^d$ of diameter R. Let $\ell(f(\mathbf{x}); y) : \mathbb{R} \to \mathbb{R}$ be a convex loss function that is Lipschitz continuous with Lipschitz constant L. Let $\mathbf{w}_t, t \in [T]$ be the sequence of classifiers generated by FOGD in Algorithm 1. Then, for any $f^* = \sum_{t=1}^T \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)$, we have the following with probability at least $1 - 2^8(\frac{dR}{\sigma^2\epsilon})^2 \exp(\frac{-D\epsilon^2}{4(d+1)})$*

$$\sum_{t=1}^T \ell(\mathbf{w}_t(\mathbf{x}_t)) \leq \frac{\|f^*\|_1^2}{2\eta} + \frac{\eta}{2} L^2 T + \sum_{t=1}^T \ell(f^*(\mathbf{x}_t)) + \epsilon L T \|f^*\|_1$$

where $\|f^*\|_1 = \sum_{t=1}^T |\alpha_t|$.

Proof. Given $f^*(\mathbf{x}) = \sum_{t=1}^T \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)$, according to the Representer Theorem [Schölkopf *et al.*, 2001], we have a corresponding linear model: $\mathbf{w}^* = \sum_{t=1}^T \alpha_t \mathbf{z}(\mathbf{x}_t)$, where

$$\mathbf{z}(\mathbf{x}) = (\sin(\mathbf{u}_1^\top \mathbf{x}), \cos(\mathbf{u}_1^\top \mathbf{x}), ..., \sin(\mathbf{u}_D^\top \mathbf{x}), \cos(\mathbf{u}_D^\top \mathbf{x})).$$

The first step to prove our theorems is to bound the regret of our sequence of linear model $\mathbf{w}_t$ learned by our online learner with respect to the linear model $\mathbf{w}^*$ in the new feature space. First of all, we have

$$\begin{aligned} & \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 \\ = \; & \|\mathbf{w}_t - \eta \nabla \ell_t(\mathbf{w}_t) - \mathbf{w}^*\|^2 \\ = \; & \|\mathbf{w}_t - \mathbf{w}\|^2 + \eta^2 \|\nabla \ell_t(\mathbf{w}_t)\|^2 - 2\eta \nabla \ell_t(\mathbf{w}_t)(\mathbf{w}_t - \mathbf{w}) \end{aligned}$$

Combining the above and the convexity of the loss function:

$$\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}) \leq \nabla \ell_t(\mathbf{w}_t)(\mathbf{w}_t - \mathbf{w}),$$

we have the following

$$\ell_t \mathbf{w}_t - \ell_t(\mathbf{w}) \leq \frac{\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2}{2\eta} + \frac{\eta}{2} \|\nabla \ell_t(\mathbf{w}_t)\|^2$$

Summing the above over $t = 1, ..., T$ leads to:

$$\begin{aligned} & \sum_{t=1}^T (\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*)) \\ & \leq \frac{\|\mathbf{w}_1 - \mathbf{w}\|^2 - \|\mathbf{w}_{T+1} - \mathbf{w}\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\nabla \ell_t(\mathbf{w}_t)\|^2 \\ & \leq \frac{\|\mathbf{w}\|^2}{2\eta} + \frac{\eta}{2} L^2 T \leq \frac{\|f^*\|_1^2}{2\eta} + \frac{\eta}{2} L^2 T \end{aligned} \tag{7}$$

Next we further examine the relationship between $\sum_{t=1}^T \ell_t(\mathbf{w}^*)$ and $\sum_{t=1}^T \ell_t(f^*)$. According to the uniform convergence of Fourier features (Claim 1 in [Rahimi and Recht, 2007]), we have the high probability bound for the difference between the approximated kernel value and the true kernel value, i.e., with probability at least $1 - 2^8(\frac{dR}{\sigma^2\epsilon})^2 \exp(\frac{-D\epsilon^2}{4(d+1)})$, we have $\forall i, j$

$$|\mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j) - \kappa(\mathbf{x}_i, \mathbf{x}_j)| < \epsilon \tag{8}$$

When $|\mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j) - \kappa(\mathbf{x}_i, \mathbf{x}_j)| < \epsilon$, we have

$$\begin{aligned} & |\sum_{t=1}^T \ell_t(\mathbf{w}^*) - \sum_{t=1}^T \ell_t(f^*)| \leq \sum_{t=1}^T |\ell_t(\mathbf{w}^*) - \ell_t(f^*)| \\ & \leq \sum_{t=1}^T L \sum_{i=1}^T \alpha_i |\mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_t) - \kappa(\mathbf{x}_i, \mathbf{x}_t)| \\ & \leq \sum_{t=1}^T L\epsilon \sum_{i=1}^T \alpha_i = \epsilon L T \|f^*\|_1 \end{aligned} \tag{9}$$

Combining (7) and (9) leads to complete the proof. □

Remark. In general, the larger the dimensionality D, the higher the probability of the bound to be achieved. This means that by sampling more random Fourier components, one can approximate the kernel function more accurately and effectively. From the above theorem, it is not difficult to show that, by setting $\eta = \frac{1}{\sqrt{T}}$ and $\epsilon = \frac{1}{\sqrt{T}}$, we have

$$\sum_{t=1}^T \ell(\mathbf{w}_t(\mathbf{x}_t)) - \sum_{t=1}^T \ell(f^*(\mathbf{x}_t)) \quad \leq \quad (\frac{\|f^*\|_1 + L^2}{2} + L)\sqrt{T}$$

which leads to sub-linear regret $O(\sqrt{T})$. However, setting $\epsilon = \frac{1}{\sqrt{T}}$ requires to sample $D = O(T)$ random components in order to achieve a high probability, which seems unsatisfactory since we will have to solve a very high-dimensional linear online learning problem. However, even in this case, for our FOGD algorithm, the learning time cost for each instance is $\mathcal{O}(c_1 T)$, while the time cost for classifying an instance by regular online kernel classification is $\mathcal{O}(c_2 T)$, here c_1 is the time for a scalar product by FOGD, while c_2 is the time for computing the kernel function. Since $c_2 \gg c_1$, our method is still much faster than the regular online kernel classification methods.

Theorem 2. *Assume we learn with kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) \leq 1$. Let $\ell(f(\mathbf{x}); y) : \mathbb{R} \to \mathbb{R}$ be a convex loss function that is Lipschitz continuous with Lipschitz constant L. Let the sequence of T instances $\mathbf{x}_1, ..., \mathbf{x}_T$ form a kernel matrix $\mathbf{K} \in \mathbb{R}^{T \times T}$, and $\mathbf{K}_k$ is the best rank-k approximation of $\mathbf{K}$, $\mathbf{K}_{max} = \max_i \mathbf{K}_{ii}$, $d_{max}^{\mathbf{K}} = \max_{ij} \sqrt{\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}}$. Let $\mathbf{w}_t, t \in [T]$ be the sequence of classifiers generated by NOGD in Algorithm 2 with budget size B. Then, for f^* that minimize $\frac{1}{2}\|f\|_{\mathcal{H}} + \frac{\lambda}{T}\ell_t(\mathbf{x}_t)$, with probability at least $1 - \epsilon$*

$$\begin{aligned} \textstyle\sum_{t=1}^T \ell(\mathbf{w}_t(\mathbf{x}_t); y_t) \leq \; & \tfrac{\|f^*\|_1^2}{2\eta} + \tfrac{\eta}{2} L^2 T + \textstyle\sum_{t=1}^T \ell(f^*(\mathbf{x}_t)) \\ & + \sqrt{2}\lambda L (\sigma_{k+1} + \Delta_B)^{\frac{1}{4}} (1 + \tfrac{\sigma_{k+1} + \Delta_B}{4})^{\frac{1}{4}} \end{aligned}$$

where $\Delta_B = \frac{2T}{\sqrt{B}} \mathbf{K}_{max} (1 + \sqrt{\frac{T-B}{T-0.5} \frac{1}{\beta(B,T)} \log \frac{1}{\epsilon}} \frac{d_{max}^{\mathbf{K}}}{\mathbf{K}_{max}^{\frac{1}{2}}})$, $\beta(B,T) = 1 - \frac{1}{2\max\{B, T-B\}}$, $\|\mathbf{K}\|_2$ is the spectral norm of matrix $\mathbf{K}$.

Proof. We adopt similar procedure that proves the above theorem. We first bound the regret with respect the corresponding $\mathbf{w}^*$ in new feature space, as (7) shows. Then we bound $\sum_{t=1}^T \ell_t(\mathbf{w}^*)$ with respect to $\sum_{t=1}^T \ell_t(f^*)$. As studied in

[Cortes *et al.*, 2010](Corollary 1), we can bound the gap between suffered losses by approximation with respect to spectral norm of kernel approximation gap:

$$|\sum_{t=1}^{T}\ell_t(\mathbf{w}^*)-\sum_{t=1}^{T}\ell_t(f^*)|\leq\sqrt{2}\lambda L\|\hat{\mathbf{K}}-\mathbf{K}\|_2^{\frac{1}{4}}(1+(\frac{\|\hat{\mathbf{K}}-\mathbf{K}\|_2}{4})^{\frac{1}{4}}) \quad (10)$$

The next is to bound the spectral norm of approximated kernel gap $\|\hat{\mathbf{K}}-\mathbf{K}\|_2$. As [Kumar *et al.*, 2012](Theorem 2) shows, this can be bounded by the spectral norm of the best rank-k approximated kernel gap $\|\mathbf{K}-\mathbf{K}_k\|_2=\sigma_{k+1}$:

$$\|\hat{\mathbf{K}}-\mathbf{K}\|_2\leq\sigma_{k+1}+\Delta_B \quad (11)$$

where Δ_B is as follows:

$$\Delta_B=\frac{2T}{\sqrt{B}}\mathbf{K}_{max}(1+\sqrt{\frac{T-B}{T-0.5}\frac{1}{\beta(B,T)}\log\frac{1}{\epsilon}}\frac{d_{max}^{\mathbf{K}}}{\mathbf{K}_{max}^{\frac{1}{2}}}),$$

$\beta(B,T)=1-\frac{1}{2\max\{B,T-B\}}$ and $\|\mathbf{K}\|_2$ is the spectral norm of $\mathbf{K}$. Combining (9), (10), (11) finishes the proof. □

Remark. Clearly, the larger the value of B, the smaller the value of Δ_B, leading to the tighter bound. Basically $\Delta_B=O(T)$ and thus $\sqrt{2}\lambda L(\sigma_{k+1}+\Delta_B)^{\frac{1}{4}}(1+\frac{\sigma_{k+1}+\Delta_B}{4})^{\frac{1}{4}}=O(\sqrt{T})$. By setting $\eta=\frac{1}{\sqrt{T}}$, we have the following:

$$\sum_{t=1}^{T}\ell(\mathbf{w}_t(\mathbf{x}_t))-\sum_{t=1}^{T}\ell(f^*(\mathbf{x}_t))\leq(\frac{\|f^*\|_1+L^2}{2})\sqrt{T}+O(\sqrt{T})$$

The above sub-linear regret $O(\sqrt{T})$ seems better than FOGD since NOGD does not require a very large value of B in order to achieve the $O(\sqrt{T})$ bound.

5 Experimental Results

5.1 Overview and Experimental Testbed

In this section, we evaluate the empirical performance of the proposed algorithms: FOGD and NOGD, by comparing them with the state-of-the-art budget online kernel classification algorithms for large-scale online kernel classification tasks.

Table 1: Details of binary-class datasets in our experiments.

Dataset	# instances	# features
german	1000	24
spambase	4601	57
magic04	19020	10
w8a	24692	300
a9a	48842	123
KDDCUP08	102294	117
ijcnn1	141691	22
codrna	271617	8
KDDCUP99	1131571	127

Table 1 shows the details of 9 publicly available datasets of diverse sizes in our experiments. All of them can be downloaded from LIBSVM website [1], UCI machine learning repository [2] and KDDCUP competition site [3].

[1]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

[2]http://www.ics.uci.edu/~mlearn/

[3]http://www.sigkdd.org/kddcup/

5.2 Compared Algorithms and Setup

As a yardstick for evaluation, we include the following two popular algorithms for regular online kernel classification:

- "Perceptron": the kernelized Perceptron algorithm [Freund and Schapire, 1999] without budget;
- "OGD": the kernelized online gradient descent algorithm [Kivinen *et al.*, 2001] without budget.

Further, we compare our algorithms with the following state-of-the-art budget online kernel learning algorithms:

- "RBP": the random budget perceptron algorithm by random removal strategy [Cavallanti *et al.*, 2007];
- "Forgetron": the Forgetron algorithm by discarding oldest support vectors [Dekel *et al.*, 2005];
- "Projectron": the Projectron algorithm using the projection strategy[Orabona *et al.*, 2009];
- "Projectron++": the aggressive version of Projectron algorithm[Orabona *et al.*, 2008; 2009];
- "BPA-S": the Budget Passive-Aggressive algorithm, we adopt "BPA-S" in [Wang and Vucetic, 2010];
- "BOGD": the Budget Online Gradient Descent algorithm by uniform sampling in [Zhao *et al.*, 2012];
- "BOGD++": the Budget Online Gradient Descent algorithm by nonuniform sampling in [Zhao *et al.*, 2012].

To make fair comparisons, all the algorithms follow the same setups. We adopt the hinge loss as the loss function ℓ, and a Gaussian kernel with bandwidth 8. The stepsize η in the all online gradient descent based algorithms is set to 0.2. We adopt the same parameter B for NOGD and other budget algorithms. The parameter k in NOGD is set to $0.2B$, and the parameter D in FOGD is set to $10B$. For each data set, all the experiments were repeated 20 times using different random permutation of instances in the dataset. All the results were obtained by averaging over these 20 runs. For performance metrics, we evaluate the online classification performance by standard mistake rates and running time (seconds). Finally, all the algorithms were implemented in Matlab, and all the experiments were run in a Linux machine with 2.5GHz CPU.

5.3 Empirical Evaluations

Table 2 summarizes the empirical evaluation results on the 9 diverse data sets. We can draw several observations below.

First of all, in terms of time efficiency, we found that the budget online classification algorithms in general run much faster than the regular online kernel classification algorithms (Perceptron and OGD) especially on the large datasets, validating the importance of studying scalable online kernel methods. By further examining their results of mistake rates, we found that the budget online classification algorithms are generally worse than the two non-budget algorithms, validating the motivation of exploring effective techniques for budget online kernel classification. Among the budget online classification algorithms, no one single algorithm consistently beats all the algorithms. In general, Projectron++, BPA-S and BOGD tend to perform more accurately than RBP and Forgetron, but also take more time cost for most cases.

Table 2: Evaluation of Large-scale online kernel machine via functional approximation.

Algorithm	german, B=100		magic04, B=100		spambase, B=100	
	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)
Perceptron	34.9 %± 1.1	0.268	27.1 %± 0.2	23.820	24.5 %± 0.1	5.599
OGD	30.2 %± 0.5	0.323	20.0 %± 0.1	110.601	22.0 %± 0.1	18.587
RBP	39.1 %± 1.5	0.242	35.1 %± 0.1	4.428	33.3 %± 0.4	1.407
Forgetron	39.7 %± 1.8	0.306	35.1 %± 0.3	5.681	34.6 %± 0.5	1.843
Projectron	35.8 %± 0.5	0.260	30.8 %± 0.3	4.574	30.8 %± 1.2	1.490
Projectron++	35.1 %± 1.5	2.606	30.8 %± 0.3	22.094	30.4 %± 1.0	5.448
BPA-S	35.6 %± 0.8	0.249	32.3 %± 1.8	4.619	30.8 %± 0.8	1.434
BOGD	33.8 %± 1.1	0.274	**30.5 %± 0.1**	5.471	32.2 %± 0.6	1.593
BOGD++	32.9 %± 1.4	0.302	**30.5 %± 0.2**	6.936	33.2 %± 0.4	1.753
FOGD	34.9 %± 1.5	0.517	41.8 %± 0.2	6.741	**27.6 %± 0.4**	1.807
NOGD	**30.8 %± 0.4**	0.378	32.3 %± 3.9	7.111	29.1 %± 0.4	2.185

Algorithm	w8a, B=200		a9a, B=200		ijcnn1, B=200	
	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)
Perceptron	3.4 %± 0.1	1035.545	21.1 %± 0.1	1311.305	12.3 %± 0.1	1454.791
OGD	2.4 %± 0.1	1460.097	16.9 %± 0.2	1882.112	9.0 %± 0.1	1957.517
RBP	4.6 %± 0.1	263.895	25.9 %± 0.2	59.552	15.9 %± 0.1	34.767
Forgetron	4.8 %± 0.1	276.400	26.5 %± 0.1	74.866	17.0 %± 0.2	40.006
Projectron	4.1 %± 0.2	269.659	21.1 %± 0.1	59.643	12.5 %± 0.2	36.643
Projectron++	4.0 %± 1.7	322.500	19.6 %± 2.3	98.083	9.5 %± 0.1	92.653
BPA-S	2.7 %± 0.1	370.653	20.6 %± 0.2	67.891	11.1 %± 0.1	50.088
BOGD	3.8 %± 0.2	367.638	27.2 %± 0.1	70.244	13.3 %± 0.4	51.086
BOGD++	4.3 %± 0.4	363.596	26.7 %± 0.4	71.198	17.4 %± 0.1	54.839
FOGD	3.4 %± 0.1	204.797	21.0 %± 0.1	39.978	12.3 %± 0.1	53.570
NOGD	**2.6 %± 0.1**	354.846	**17.0 %± 0.1**	95.895	**9.1 %± 0.1**	87.420

Algorithm	codrna, B=200		KDDCUP08, B=100		KDDCUP99, B=100	
	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)	Mistake Rate	Time Cost(s)
Perceptron	14.0 %± 0.1	2379.002	0.90 %± 0.0	350.223	0.02 %± 0.00	2648.899
OGD	9.9 %± 0.1	3241.550	0.52 %± 0.0	773.091	0.01 %± 0.00	10732.788
RBP	18.7 %± 0.1	57.352	1.06 %± 0.0	48.907	0.02 %± 0.00	925.876
Forgetron	18.3 %± 0.1	66.244	1.07 %± 0.0	50.587	0.03 %± 0.00	969.976
Projectron	14.6 %± 0.1	62.325	0.94 %± 0.0	49.243	0.02 %± 0.00	934.476
Projectron++	12.5 %± 0.2	374.943	0.84 %± 0.0	143.331	**0.01 %± 0.00**	1475.105
BPA-S	13.3 %± 0.3	266.386	0.62 %± 0.0	57.400	**0.01 %± 0.00**	1326.747
BOGD	14.6 %± 0.1	80.676	0.61 %± 0.0	66.022	0.81 %± 0.06	787.354
BOGD++	16.8 %± 0.1	86.730	0.71 %± 0.0	66.390	0.04 %± 0.01	787.708
FOGD	14.0 %± 0.1	135.094	1.06 %± 0.0	48.935	0.02 %± 0.00	472.141
NOGD	**12.2 %± 0.1**	289.065	**0.59 %± 0.0**	81.811	**0.01 %± 0.00**	511.731

Second, by comparing the proposed algorithms (FOGD and NOGD) with the budget online classification algorithms, we found that they generally achieve the best or close to the best classification performance for most cases using fairly comparable or even lower time cost. Specifically, by comparing with RBP and Forgetron, our algorithms obtain much more accurate results with comparable or slightly more time cost. In comparison to Projectron++, BPA-S and BOGD++, our algorithms achieve better or at least highly comparable accuracy but with smaller or comparable time cost.

Finally, NOGD tends to perform better than FOGD (except for "spambase"). We conjecture that it may because FOGD is data-independent, while NOGD is data-dependent which approximates the kernel by sampling the given data instances. We note that our observation for the empirical advantages of Nyström methods over random Fourier features is consistent to the recent comparison between Fourier random features and Nyström methods [Yang *et al.*, 2012].

6 Conclusion

This paper presented a novel framework of large-scale online kernel classification via functional approximation, going beyond conventional online kernel methods that often adopt the budget maintenance strategy to bound the support vector size. The basic idea of our framework is to approximate a kernel function or kernel matrix by exploring functional approximation techniques. We presented two new algorithms for large-scale online kernel classification: Fourier Online Gradient Descent (FOGD) and Nyström Online Gradient Descent (NOGD). Our promising results on large-scale online kernel classification tasks show the state-of-the-art performance of our algorithms in both classification efficacy and efficiency. As the first attempt of exploring functional approximation for online kernel learning, our framework is generic and can be extended to other settings (e.g., regression) in future work.

Acknowledgements

This work was supported by MOE tier-1 grant (RG33/11).

References

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

[Chitta *et al.*, 2012] Radha Chitta, Rong Jin, and Anil Jain. Efficient kernel clustering using random fourier features. In *ICDM*, 2012.

[Cortes *et al.*, 2010] Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. *Journal of Machine Learning Research - Proceedings Track*, 9:113–120, 2010.

[Crammer *et al.*, 2003] Koby Crammer, Jaz S. Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[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*, 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 *NIPS*, 2005.

[Dredze *et al.*, 2008] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning (ICML2008)*, pages 264–271, 2008.

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

[Hoi *et al.*, 2006] Steven CH Hoi, Michael R Lyu, and Edward Y Chang. Learning the unified kernel machines for classification. In *KDD*, pages 187–196. ACM, 2006.

[Hoi *et al.*, 2013] Steven C. H. Hoi, Rong Jin, Peilin Zhao, and Tianbao Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013.

[Kivinen *et al.*, 2001] Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *NIPS*, pages 785–792, 2001.

[Kumar *et al.*, 2012] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nystr?m method. *Journal of Machine Learning Research*, 13:981–1006, 2012.

[Li *et al.*, 2012] Bin Li, Peilin Zhao, Steven CH Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning*, 87(2):221–258, 2012.

[Orabona *et al.*, 2008] Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *ICML*, pages 720–727, 2008.

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

[Rahimi and Recht, 2007] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[Rudin, 1990] W. Rudin. *Fourier Analysis on Groups*. Wiley-Interscience, 1990.

[Schölkopf *et al.*, 2001] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

[Talwalkar *et al.*, 2008] Ameet Talwalkar, Sanjiv Kumar, and Henry A. Rowley. Large-scale manifold learning. In *CVPR*, 2008.

[Wang and Vucetic, 2010] Zhuang Wang and Slobodan Vucetic. Online passive-aggressive algorithms on a budget. *Journal of Machine Learning Research - Proceedings Track*, 9:908–915, 2010.

[Wang *et al.*, 2012] Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. Cost-sensitive online classification. In *ICDM*, pages 1140–1145, 2012.

[Williams and Seeger, 2000] Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2000.

[Xia *et al.*, 2013] Hao Xia, Pengcheng Wu, and Steven C. H. Hoi. Online multi-modal distance learning for scalable multimedia retrieval. In *WSDM*, pages 455–464, 2013.

[Yang *et al.*, 2012] Tianbao Yang, Yufeng Li, Mehrdad Mahdavi, Rong Jin, and Zhi hua Zhou. Nystrom method vs random fourier features: A theoretical and empirical comparison. In *NIPS*, 2012.

[Zhang and Kwok, 2009] Kai Zhang and James T. Kwok. Density-weighted nyström method for computing large kernel eigensystems. *Neural Computation*, 21(1):121–146, 2009.

[Zhang *et al.*, 2012] Kai Zhang, Liang Lan, Zhuang Wang, and Fabian Moerchen. Scaling up kernel svm on limited resources: A low-rank linearization approach. *Journal of Machine Learning Research - Proceedings Track*, 22:1425–1434, 2012.

[Zhao and Hoi, 2010] Peilin Zhao and Steven CH Hoi. Otl: A framework of online transfer learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[Zhao *et al.*, 2011] Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Double updating online learning. *Journal of Machine Learning Research*, 12:1587–1615, 2011.

[Zhao *et al.*, 2012] Peilin Zhao, Jialei Wang, Pengcheng Wu, Rong Jin, and Steven C. H. Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *ICML*, 2012.