# FFTRL: A Sparse Online Kernel Classification Algorithm for Large Scale Data

Changzhi Su, Li Zhang(✉), and Lei Zhao

School of Computer Science and Technolgy, Soochow University, Suzhou 215006, China
20205227057@stu.suda.edu.cn, {zhangliml,zhaol}@suda.edu.cn

**Abstract.** Online kernel learning is an efficient way when dealing with nonlinearly large-scale data. The training speed of online kernel learning is improved by Fourier online gradient descent (FOGD). However, FOGD has a high space complexity when the number of features is relatively high because FOGD lacks of sparsity. In this paper, we propose a new sparse online kernel classification algorithm for large-scale data, called Fourier follow-the-regularized-leader (FFTRL). Existing budget (sparse) online kernel learning methods attempt to bound the number of support vectors through some budget maintenance strategies; however, budget maintenance strategies are unsuitable for FOGD. By introducing the proximal algorithm, follow-the-regularized-leader, FFTRL achieves sparsity in a different way. By applying random Fourier features as the kernel approximation schemes, FFTRL finds the optimal sparse solution in a linear manner. The regret bound analysis shows the feasibility of FFTRL in theory. Comprehensive experiments were carried out on public datasets to compare the performance of FFTRL with related online kernel algorithms. Promising results show that our proposed method enjoys both high accuracy and time efficiency and still produces sparse models, opening a window for obtaining sparsity in online kernel learning.

**Keywords:** Online learning · Kernel approximation · Sparsity · Online graident descent

## 1 Introduction

It has been proven that online learning is successful for building accurate and reliable models from a sequence of data elements efficiently. Different from regular batch machine learning algorithms that suffer from massive training time and memory consumption, online learning models often enjoy the properties of fast construction, highly scalable and memory saving. Due to these advantages, online learning algorithms have been successfully used in many real-world applications, such as online advertising [14], weather condition prediction [11], and computational finance [10].

Various algorithms have been developed to tackle online binary classification tasks, which can be simply divided into two types: linear and kernel methods. The linear methods are able to construct linear predictive models at an amazing speed. Some well-known examples include online gradient descent (OGD) [18], forward backward splitting (FOBOS) [7], regularized dual averaging (RDA) [19] and follow-the-regularized-leader (FTRL) [13,14]. However, linear models are not always the right choice. Linear online algorithms may fail to produce effective outcomes when faced with linearly non-separable inputs, which is more common in real-world applications. To overcome this issue, researchers invited kernel functions into online learning methods and came up with field of online kernel learning. Kernel-based estimators avoid the non-separable property in the input space by mapping the instances to a high dimensional feature space implicitly. One key limitation of classical online kernel methods is that the functional representation of the produced estimator will become more complex as the observations grows. To be more specific, the learner is asked to maintain a support vector (SV) set during the online learning process. Whenever a newly arrived instance is misclassified, it will be added to the SV set immediately. Thus the complexity of the estimator and memory resource it demands will increase linearly over time, causing memory overflow for a potentially infinite input data sequence.

Several approaches have been proposed to handle the extension issues of online kernel learning. One interesting aspect, which is usually referred to as "budget online kernel learning" [5], tries to bound the number of SVs within a fixed size during the training process. Two major wildly acknowledged budget maintenance strategies are removal and projection. The former simply evicts a selected SV when the number of SVs overflows. It is adopted by many algorithms, such as Forgetron [6], randomized budget perceptron (RBP) [3], and naive online $R_{reg}$ minimisation algorithm (NORMA) [9]. The latter further projects the selected SV onto the remaining ones, which is explored in algorithms like Projectron [15] and online manifold regularization (OMR) [2], Budget strategies do release the pressure to some extent, but the existing budget online kernel methods are either too simple to achieve promising results or just too slow to perform. The other promising aspect is to use the functional approximation scheme [20]. Unlike the budget maintenance strategy, this kind of scheme tackles the problem in a mathematically elegant way. A certain explicit mapping can be derived by approximating a kernel function, making it possible to project data from the input space to a computable highly dimensional feature space. Combining with linear online learning algorithms like OGD, nonlinear kernel-based algorithms are then trained in an efficient linear manner. As far as we known, Fourier online gradient descent (FOGD) has achieved a success in reducing time cost following this idea [12]. To reduce the required memory, the final model should be stored sparsely, or the number of non-zero coefficients in the final model parameter should be small. However, even employing the $L_1$ penalty, FOGD can hardly produce sparse models. Similarly, it may cause the memory usage problem when the dimension of the feature space becomes too high.

In order to take the advantages of linear online models and produce sparsity simultaneously, we propose a Fourier follow-the-regularized-leader (FFTRL) algorithm in this paper. FFTRL adopts the random Fourier feature technique to approximate shift-invariant kernels and introduces sparsity using the FTRL algorithm. Theoretical analysis and experiments on FFTRL are also provided in this paper.

The rest of the paper is organized as follows. Section 2 details the proposed method. Experimental results and analysis are presented in Sect. 3 and the conclusion is given in Sect. 4.

## 2 Proposed Method

### 2.1 Algorithm Description

The proposed FFTRL is a online kernel learning method for binary classification tasks. The goal of FFTRL is to learn a final mapping or hypothesis $f : \mathbb{R}^n \rightarrow \mathbb{R}$ from a sequence of data elements $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$, where $\mathbf{x}_t \in \mathbb{R}^n$ is the $t$th training instance, and $y_t \in \{+1, -1\}$ is the corresponding class label, $n$ and $T$ are the number of features and samples, respectively. Generally, a convex loss function $l(f(\mathbf{x}), y) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is used to penalize the deviation of the estimation $f(\mathbf{x})$ from the exact class label $y$. Further, we assume $\mathcal{H}_k$ is a reproducing kernel Hilbert space (RKHS). Thus, the function $k(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is defined as the reproducing kernel of $\mathcal{H}_k$ if and only if it implements the inner product $\langle \cdot, \cdot \rangle$ such that

1. $k(\mathbf{x}, \cdot) \in \mathcal{H}_k$ for $\forall \mathbf{x} \in \mathbb{R}^n$;
2. $\langle f, k(\mathbf{x}, \cdot) \rangle = f(\mathbf{x})$ for $\forall \mathbf{x} \in \mathbb{R}^n$ and $\forall f \in \mathcal{H}_k$.

In classical online kernel learning, the computation of kernel functions improves the complexity of algorithms. Inspired by FOGD, FFTRL represents a kernel mapping in a linear manner. Namely,

$$k(\mathbf{x}_j, \mathbf{x}_m) \approx \mathbf{z}(\mathbf{x}_j)^{\mathsf{T}} \mathbf{z}(\mathbf{x}_m), \tag{1}$$

where the superscript $\mathsf{T}$ means the operation of a vector or matrix transpose, $\mathbf{x}_j$ and $\mathbf{x}_m$ are arbitrary instances in the sequence, and $\mathbf{z}(\mathbf{x}_j)$ is an approximate image of $\mathbf{x}_j$ in the feature space.

Let $f(\mathbf{x}) = \mathbf{w}^{\mathsf{T}} \mathbf{z}(\mathbf{x})$, where $\mathbf{w}$ is the weight vector. Then the loss function can be represented as $l(\mathbf{w}, \mathbf{z}(\mathbf{x}), y)$. To find $\mathbf{z}(\mathbf{x})$ related to $k(\cdot, \cdot)$, we introduce random Fourier features [16], which is a kernel functional approximation technique that works for shift-invariant kernels like Gaussian and Laplacian kernels. Such kernels have the form of $k(\mathbf{x}_j, \mathbf{x}_m) = k(\Delta\mathbf{x})$, where $\Delta\mathbf{x} = \mathbf{x}_j - \mathbf{x}_m$ is the divergence between the two instances. Bochner's theorem implies that a positive definite kernel function $k(\varDelta\mathbf{x})$ is the Fourier transform of a proper probability density function $p(\mathbf{u})$ with a random variable $\mathbf{u} \in \mathbb{R}^n$ [17]. Namely,

$$k(\varDelta\mathbf{x}) = \int p(\mathbf{u}) e^{i\mathbf{u}^\top \varDelta\mathbf{x}}\, d\mathbf{u}, \tag{2}$$

where $i$ is the imaginary unit. By contrary, assume we have the right kernel here. By calculating the inverse Fourier transform of the kernel $k(\varDelta\mathbf{x})$, we can obtain

$$p(\mathbf{u}) = \left(\frac{1}{2\pi}\right)^n \int e^{-i\mathbf{u}^\top(\varDelta\mathbf{x})} k(\varDelta\mathbf{x})\, d(\varDelta\mathbf{x}). \tag{3}$$

For example, given a Gaussian kernel $k(\mathbf{x}_j, \mathbf{x}_m) = \exp(-\|\mathbf{x}_j - \mathbf{x}_m\|_2^2/2\sigma^2)$ with the kernel parameter $\sigma > 0$, we have the corresponding distribution $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$ with the identify matrix $\mathbf{I}$. According to (2), we can see that the kernel function can be expressed as the expectation of $\mathbf{u}$ drawn from the distribution $p(\mathbf{u})$. In other words, we have

$$\int p(\mathbf{u}) e^{i\mathbf{u}^\top \varDelta\mathbf{x}}\, d\mathbf{u} = E_{\mathbf{u}}[e^{i\mathbf{u}^\top \mathbf{x}_j} e^{-i\mathbf{u}^\top \mathbf{x}_m}], \tag{4}$$

where the function $E_{\mathbf{u}}[\cdot]$ is to find the expectation of $\mathbf{u}$. Using Euler's formula, we can rewrite (4) as

$$\begin{aligned} &E_{\mathbf{u}}[\cos(\mathbf{u}^\top\mathbf{x}_j)\cos(\mathbf{u}^\top\mathbf{x}_m) + \sin(\mathbf{u}^\top\mathbf{x}_j)\sin(\mathbf{u}^\top\mathbf{x}_m)] \\ =&E_{\mathbf{u}}\left[[\sin(\mathbf{u}^\top\mathbf{x}_j), \cos(\mathbf{u}^\top\mathbf{x}_j)][\sin(\mathbf{u}^\top\mathbf{x}_m), \cos(\mathbf{u}^\top\mathbf{x}_m)]^\top\right]. \end{aligned} \tag{5}$$

According to (5), we can make $\mathbf{z}(\mathbf{x}) = [\sin(\mathbf{u}^\top\mathbf{x}), \cos(\mathbf{u}^\top\mathbf{x})]^\top$ that is a new representation (image) of instance $\mathbf{x}$. Since the kernel function $k(\varDelta\mathbf{x})$ equals the expectation of inner productor of $\mathbf{z}(\mathbf{x}_j)$ and $\mathbf{z}(\mathbf{x}_m)$, we can draw $D$ samples $\mathbf{u}_1, \ldots, \mathbf{u}_D$ independently from the distribution $p$ and construct the image of $\mathbf{x}$ as

$$\mathbf{z}(\mathbf{x}) = \left[\sin(\mathbf{u}_1^\top\mathbf{x}), \cos(\mathbf{u}_1^\top\mathbf{x}), \ldots, \sin(\mathbf{u}_D^\top\mathbf{x}), \cos(\mathbf{u}_D^\top\mathbf{x})\right]^\top \in \mathbb{R}^{2D}. \tag{6}$$

Now, we can ignore the computation of kernel function because we get the explicit images in the high-dimensional feature space that is induced by the corresponding kernel function. If the number of samples $D$ is large enough, the error brought by approximation can be omitted reasonably. Thus, the online kernel learning in the original space is transformed into the linear online learning in a high dimensional feature space.

To produce sparsity in the online process, we introduce FTRL that comprehensively considers the differences between FOBOS and RDA on regularization terms and model parameter $\mathbf{w}$. In the $t$th round, FFTRL performs the update of the weight vector $\mathbf{w}_{t+1}$ as follows:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \mathbf{w}^\top \left(\sum_{s=1}^{t} \mathbf{g}_s\right) + \frac{1}{2}\sum_{s=1}^{t} \|\sqrt{\boldsymbol{\sigma}_s} \odot (\mathbf{w} - \mathbf{w}_s)\|_2^2 + \lambda\|\mathbf{w}\|_1 \right\}, \tag{7}$$

where $\mathbf{g}_s = \nabla_{\mathbf{w}_s} l(\mathbf{w}_s, \mathbf{z}(\mathbf{x}_s), y_s)$ is the gradient in the $s$th iteration, $\odot$ is the element-wise multiplication operator, $\boldsymbol{\sigma}_s = [\sigma_{s,1}, \ldots, \sigma_{s,2D}]^{\mathsf{T}} \in \mathbb{R}^{2D}$ is the parameter related to the current learning rate, and $\lambda$ is a positive regularization parameter. We discuss $\boldsymbol{\sigma}_s$ later.

The basic idea behind FTRL is to minimize the loss cumulated in the online training process, which will get a low-regret solution in the current round. Therefore, FFTRL uses a cumulative gradient to approximately estimate the cumulative loss, or the first term of (7). The second term in (7) works as a stabilization penalty to avoid $\mathbf{w}$ from vibrating extensively in iterations, while the third term is an $L_1$ penalty. With $\lambda > 0$, FFTRL does an excellent job in producing sparsity.

Moreover, we thought that if a feature variable varies more rapidly than the other, then it is reasonable that the learning rate on this feature variable should decline faster. Thus, FFTRL uses the per-coordinate learning rate instead of a global learning rate like setting $\eta_t = \frac{1}{\sqrt{t}}$ $(t > 0)$ for all features. In other words, the learning rate is calculated independently for each feature. Let $\boldsymbol{\eta}_t = [\eta_{t,1}, \ldots, \eta_{t,2D}] \in \mathbb{R}^{2D}$ be the learning rate used in FFTRL. We reflect the rate of change using the gradient component in a certain dimension. Without loss of generality, let $g_{t,h}$ be the $h$th entry in $\mathbf{g}_t$. Then, the corresponding learning rate in the $h$th dimension can be expressed as

$$\eta_{t,h} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,h}^2}} \tag{8}$$

for $t > 0$, where both $\alpha > 0$ and $\beta > 0$ are two parameters needed to be tuned for good performance. When $t = 0$, $g_{s,t} = 0$. Then, $\eta_{0,h} = \alpha/\beta$ for all $h$. For $\boldsymbol{\sigma}_s$, its $h$th component can be defined as

$$\sigma_{s,h} = \frac{1}{\eta_{t,h}} - \frac{1}{\eta_{t-1,h}}. \tag{9}$$

The detail algorithm description of FFTRL is summarized in Algorithm 1. For training data arriving sequentially, we first construct the new representation of an instance using the explicit mapping $\mathbf{z}(\mathbf{x})$ in (6) and then perform a sparse linear online learning using FTRL. The overall time complexity of FTRL in one update round is $O(D)$.

## 2.2 Theoretical Analysis

We further analyze the theoretical property of our proposed method. For the purpose of simplicity, $l_t(f)$ represents $l(f(\mathbf{x}_t), y_t)$, and $l_t(\mathbf{w})$ is $l_t(\mathbf{w}_t, \mathbf{z}(\mathbf{x}_t), y_t)$. In the following, we show that the regret of our algorithm is sub-linear, which indicates the effectiveness of FFTRL .

**Theorem 1.** *Assume that the original data is contained by a ball in $\mathbb{R}^n$ of diameter $\tilde{R}$. Let $k(\mathbf{x}, \mathbf{x}') = k(\Delta\mathbf{x})$ be a positive definite and shift-invariant kernel, and $l(f(\mathbf{x}), y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a convex loss function that is Lipschitz continuous*

**Algorithm 1** Training process of FFTRL.

```
Input: Kernel function $k(\cdot,\cdot)$, parameters $\alpha$, $\beta$, and $\lambda$, and the number of samples $D$.
Initialize: $\mathbf{w}_1 = \mathbf{0}$; $m_h = 0$, $q_h = 0$ $(\forall h \in \{1, 2, \ldots, 2D\})$.
  Calculate $p(\mathbf{u})$ of kernel $k(\cdot,\cdot)$ using (3);
  Draw $D$ independent and identically distributed samples $\mathbf{u}_1, \ldots, \mathbf{u}_D$ from $p(\mathbf{u})$;
  for $t = 1, 2, \ldots, T$ do
    Receive $(\mathbf{x}_t, y_t)$;
    Construct the new representation $\mathbf{z}(\mathbf{x}_t)$ using (6);
    Predict $\widehat{y}_t = \mathrm{sgn}(\mathbf{w}_t^\mathsf{T}\mathbf{z}(\mathbf{x}_t))$;
    Calculate $\mathbf{g}_t = \nabla_{\mathbf{w}_t} l(\mathbf{w}_t, \mathbf{z}(\mathbf{x}_t), y_t)$;
    for $h = 1, 2, \ldots, 2D$ do
      $\sigma_{t,h} = \frac{1}{\alpha}\sqrt{q_h + g_{t,h}^2} - \sqrt{q_h}$; // which is equivalent to (9).
      $q_h \leftarrow q_h + g_{t,h}^2$;
      $m_h \leftarrow m_h + g_{t,h} - \sigma_{t,h}\mathbf{w}_{t,h}$;
      $\mathbf{w}_{t+1,h} = \begin{cases} 0 & \textit{if } |m_h| < \lambda, \\ \frac{\alpha}{\beta+\sqrt{q_h}}(\lambda\,\mathrm{sgn}(m_h) - m_h) & \textit{otherwise.} \end{cases}$
    end for
  end for
```

*with Lipschitz constant $L$. Assume that $\mathbf{w}_1, \ldots, \mathbf{w}_T$ is the sequence of model parameters generated by FFTRL (Algorithm 1) under the mild condition that the learning rate $\eta_{t,h} = \eta_t$ for every dimension in the same iteration, where $\|\mathbf{w}_t\|_2 \leq R$. With probability at least $1 - 2^8(\frac{\varsigma_p \tilde{R}}{\epsilon})^2 \exp(\frac{-D\epsilon^2}{4(n+2)})$, the following inequality*

$$\sum_{t=1}^{T} l_t(\mathbf{w}_t) - \sum_{t=1}^{T} l_t(f^*) \leq \frac{(1+\epsilon)\|f^*\|_1^2}{2\eta_T} + L^2\sum_{t=1}^{T}\eta_t + \frac{3R^2}{2\eta_T} + \sqrt{2D}\lambda R + \epsilon LT\|f^*\|_1$$

*holds true for any $f^*(\mathbf{x}) = \sum_{t=1}^{T}\alpha_t^* k(\mathbf{x}_t, \mathbf{x})$, where $\|f^*\|_1 = \sum_{t=1}^{T}|\alpha_t^*|$, $\varsigma_p^2 = E_p[\mathbf{u}^\mathsf{T}\mathbf{u}]$ is the second moment of the Fourier transform of the kernel function $k(\cdot,\cdot)$ given that $p(\mathbf{u})$ is the probability density function calculated by (3), and $\epsilon$ is a small positive constant.*

*Proof.* Given $f^*(\mathbf{x}) = \sum_{t=1}^{T}\alpha_t^* k(\mathbf{x}_t, \mathbf{x})$ as the optimal solution of FFTRL, we have the corresponding linear model $\mathbf{w}^* = \sum_{t=1}^{T}\alpha_t^*\mathbf{z}(\mathbf{x}_t)$. First of all, we have to bound the regret of the sequence $\mathbf{w}_1, \ldots, \mathbf{w}_T$ learned by FFTRL with respect to the optimal linear model $\mathbf{w}^*$ in the new feature space. According to the regret analysis of the FTRL algorithm with strongly convex regularizers (Lemma 2.3.) [18], we have:

$$\sum_{t=1}^{T}(l_t(\mathbf{w}_t) - l_t(\mathbf{w}^*)) \leq L^2\sum_{t=1}^{T}\eta_t + r_{1:T}(\mathbf{w}^*) + \psi(\mathbf{w}^*), \tag{10}$$

where $r_{1:T}(\mathbf{w}^*) = \sum_{t=1}^{T} r_t(\mathbf{w}^*)$. Let $r_t(\mathbf{w}) = \frac{\sigma_t}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2$ and $\psi(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$. Then, the cumulative sum of regularizers becomes

$$r_{1:T}(\mathbf{w}^*) + \psi(\mathbf{w}^*) = \frac{1}{2}\sum_{t=1}^{T}\sigma_t\|\mathbf{w}^* - \mathbf{w}_t\|_2^2 + \lambda\|\mathbf{w}^*\|_1, \tag{11}$$

which is exactly the same as the regularization term in (7).

For $r_{1:T}(\mathbf{w}^*)$, we can infer that

$$\begin{aligned} r_{1:T}(\mathbf{w}^*) &= \frac{1}{2}\sum_{t=1}^{T}\sigma_t\|\mathbf{w}^* - \mathbf{w}_t\|_2^2 \\ &\le \frac{1}{2}\sum_{t=1}^{T}\sigma_t(\|\mathbf{w}^*\|_2^2 - 2\langle\mathbf{w}^*, \mathbf{w}_t\rangle + \|\mathbf{w}_t\|_2^2) \\ &\le \frac{1}{2}\sum_{t=1}^{T}\sigma_t(\|\mathbf{w}^*\|_2^2 + 3R^2) = \frac{1}{2\eta_T}(\|\mathbf{w}^*\|_2^2 + 3R^2). \end{aligned} \tag{12}$$

For $\psi(\mathbf{w}^*)$, it is upper-bounded by $\sqrt{2D}\lambda R$ according to the arithmetic-geometric mean inequality (AGMI). The regret bound (10) now becomes

$$\sum_{t=1}^{T}(l_t(\mathbf{w}_t) - l_t(\mathbf{w}^*)) \le L^2\sum_{t=1}^{T}\eta_t + \frac{\|\mathbf{w}^*\|_2^2 + 3R^2}{2\eta_T} + \sqrt{2D}\lambda R \tag{13}$$

Next, we examine the difference between $\sum_{t=1}^{T} l_t(\mathbf{w}^*)$ and $\sum_{t=1}^{T} l_t(f^*)$. According to the uniform convergence of random Fourier features (Claim 1 in [16]), with probability at least $1 - 2^8(\frac{\varsigma_p\tilde{R}}{\epsilon})^2\exp(\frac{-D\epsilon^2}{4(n+2)})$, we have

$$\forall j, m, \quad |\mathbf{z}(\mathbf{x}_j)^\mathsf{T}\mathbf{z}(\mathbf{x}_m) - k(\mathbf{x}_j, \mathbf{x}_m)| < \epsilon. \tag{14}$$

In other words, the more we sample, the smaller the probability that the difference between approximated kernel value and real kernel value is greater than the constant $\epsilon$ we will get. We further assume $k(\mathbf{x}_j, \mathbf{x}_m) \le 1$, then we have $\mathbf{z}(\mathbf{x}_j)^\mathsf{T}\mathbf{z}(\mathbf{x}_m) \le 1 + \epsilon$ that leads to

$$\|\mathbf{w}^*\|_2^2 \le (1+\epsilon)\|f^*\|_1^2. \tag{15}$$

With (14), we have:

$$\begin{aligned} \left|\sum_{t=1}^{T} l_t(\mathbf{w}^*) - \sum_{t=1}^{T} l_t(f^*)\right| &\le \sum_{t=1}^{T}|l_t(\mathbf{w}^*) - l_t(f^*)| \\ &\le L\sum_{t=1}^{T}\sum_{j=1}^{T}|\alpha_j^*||\mathbf{z}(\mathbf{x}_j)^\mathsf{T}\mathbf{z}(\mathbf{x}_t) - k(\mathbf{x}_j, \mathbf{x}_t)| \\ &\le \epsilon L\sum_{t=1}^{T}\sum_{j=1}^{T}|\alpha_j^*| = \epsilon LT\|f^*\|_1. \end{aligned} \tag{16}$$

Combing (13), (15) and (16) leads to the completion of the proof.

**Table 1.** Information of eight publicly available datasets used in experiments.

| Datasets | #Instances | #Features |
|---|---|---|
| Titanic | 2201 | 3 |
| Spambase | 4597 | 57 |
| Banana | 5300 | 2 |
| Phoneme | 5404 | 5 |
| Coil2000 | 9822 | 85 |
| W7a | 24,692 | 300 |
| A7a | 32,561 | 123 |
| Ijcnn1 | 141,691 | 22 |

## 3 Experiments

### 3.1 Description of Data and Algorithms Involved

To validate the performance of our proposed algorithm, we conducted extensive experiments on the tasks of online binary classification. We first introduced the datasets used in our experiments and then described the algorithms for comparison.

Table 1 shows the details of eight publicly available datasets where the first five datasets can be downloaded from KEEL dataset repository [1] and the rest three are available at LIBSVM website [4]. We followed the common setting of online binary classification tasks that each dataset should be divided into training and test sets. We adopted the original splits of training and test sets for datasets downloaded from the LIBSVM website. For KEEL datasets, a random split of 4 : 1 training–test was performed.

In experiments, our proposed method was first compared with NORMA and ACCOSVM for regular online kernel classification, which are solved in primal and dual spaces, respectively.

- "NORMA" [9]: Online gradient descent for kernel SVM without budget.
- "ACCOSVM" [8]: An accelerator for online SVM combing quadratic programming and window techniques.

Further, we invited three state-of-the-art budget online kernel learning algorithms to compare with FFTRL. Namely,

- "BNORMA" [9]: The budgeted version of NORMA using removal strategy.
- "Forgetron" [6]: Budget perceptron using the removal strategy.
- "Projectron" [15]: Budget perceptron using the projection strategy.

Finally, we introduced an algorithm sharing the similar idea with our proposed method:

- "FOGD" [12]: Online gradient descent using random Fourier features for kernel approximation.

## 3.2 Experimental Setting

All the experiments were carried out in Python 3.6 on a PC running Windows 10 with a 2.9GHz Intel Core i7 processor and 16 GB RAM. To make a fair comparison, all algorithms adopted the following same setups. The Gaussian kernel was used as the kernel function $k(\cdot, \cdot)$, and the hinge loss was taken as the convex loss function. Since the hinge loss is a non-smooth function, subgradient was adopted instead of gradient, which counts only when $yf(\mathbf{x}) < 1$.

The budget size in budget online learning algorithms and the number of samples in FOGD and our proposed method were set to 100 and 200, respectively, following the same setups in [12]. The learning rate related parameter $\beta$ in our algorithm was set to 1 according to the instruction from [14]. Other hyper-parameters were selected by a standard 5-fold cross validation on the training set, including the kernel bandwidth $\sigma$, the learning rate related parameter $\alpha$ for FFTRL, the regularization parameter $\lambda$ for FFTRL, NORMA and BNORMA, the initial learning rate $\eta_0$ for FGD, NORMA, and BNORMA, and $C$ for ACCOSVM. Then, the training set was refitted using the best model five times, where at each run the instances were shuffled differently. The mean and standard deviation of mistake rate on the training set, training time, accuracy on the test set, and test time were reported as the final results.

## 3.3 Results and Analysis

Table 2 summarizes the evaluation results on the eight datasets, where the best results are in bold. Note that the test process of NORMA on the Ijcnn1 dataset was early stopped after 10,000 s, and the instances being tested at the time of early stopping was reported in italic. From Table 2, we can draw the following conclusions.

First, we found that budget online kernel classification algorithms run much faster than the regular ones (say, NORMA and ACCOSVM) in both training and test process. That means scalable online kernel methods are more practical in terms of time efficiency. However, budget online kernel classification algorithms generally make more mistakes on the training set and then get lower accuracy on the test set. Potentional loss of information is occurred when adopting budget strategies, validating the importance of exploring effective techniques for budget online kernel learning algorithms. The same phenomenon happens inside the family of budget online algorithms too. We notice that Projectron takes more time in training and test but obtains more promising results than both BNORMA and Forgetron in five out of eight datasets since the projection strategy is more complex than just simply remove an SV. The trade-offs between accuracy and time efficiency should be analyzed in specific situations.

Second, we compared the two kernel approximation methods (FOGD and FFTRL) with the budget online kernel classification algorithms. As is listed in Table 2, FOGD takes the least time in training, and our proposed method FFTRL shows competitive results too. Both algorithms achieve amazing speed in training, far exceeding any budget online kernel algorithms. We inferred that

**Table 2.** Comparison of online kernel algorithms on 8 benchmark binary classification datasets.

| Algorithm | Titanic | | | | Spambase | | | |
|---|---|---|---|---|---|---|---|---|
| | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) |
| NORMA | **22.54 ± 0.57** | 10.97 ± 0.02 | 77.65 ± 1.22 | 5.47 ± 0.02 | 12.96 ± 0.51 | 49.44 ± 0.44 | 91.10 ± 0.81 | 24.54 ± 0.29 |
| ACCOSVM | 23.33 ± 0.50 | 137.56 ± 0.54 | 78.67 ± 1.00 | 6.36 ± 0.04 | 12.05 ± 0.88 | 150.47 ± 3.03 | **91.80 ± 0.39** | 17.68 ± 0.27 |
| BNORMA | 22.86 ± 0.40 | 1.34 ± 0.01 | 76.65 ± 1.26 | 0.33 ± 0.00 | 18.76 ± 0.33 | 2.78 ± 0.05 | 83.77 ± 2.98 | 0.67 ± 0.01 |
| Forgetron | 31.13 ± 0.34 | 1.78 ± 0.01 | 76.83 ± 2.77 | 0.35 ± 0.00 | 17.19 ± 0.54 | 3.17 ± 0.04 | 83.47 ± 1.62 | 0.70 ± 0.00 |
| Projectron | 30.14 ± 0.54 | 0.24 ± 0.01 | 77.69 ± 1.62 | 0.05 ± 0.00 | 15.44 ± 0.43 | 10.67 ± 0.44 | 87.21 ± 1.44 | 3.94 ± 0.11 |
| FGD | 25.03 ± 0.55 | **0.02 ± 0.00** | 78.15 ± 1.80 | 0.01 ± 0.00 | 11.75 ± 0.24 | **0.06 ± 0.00** | 90.08 ± 1.20 | 0.01 ± 0.00 |
| FFTRL | 22.56 ± 0.29 | 1.76 ± 0.10 | **79.03 ± 1.05** | **0.01 ± 0.00** | **11.21 ± 0.41** | 5.63 ± 0.03 | 91.56 ± 0.49 | **0.01 ± 0.00** |
| Algorithm | Banana | | | | Phoneme | | | |
| | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) |
| NORMA | 11.62 ± 0.36 | 65.46 ± 0.27 | 89.08 ± 0.58 | 34.52 ± 3.74 | 23.67 ± 0.33 | 67.29 ± 1.06 | 77.28 ± 0.91 | 33.66 ± 0.37 |
| ACCOSVM | **11.11 ± 0.24** | 134.27 ± 4.69 | 89.23 ± 0.62 | 11.86 ± 0.30 | **16.44 ± 0.27** | 412.58 ± 17.45 | **85.75 ± 0.65** | 19.26 ± 0.18 |
| BNORMA | 15.89 ± 0.27 | 3.33 ± 0.11 | 83.06 ± 1.70 | 0.80 ± 0.03 | 23.74 ± 0.39 | 3.32 ± 0.06 | 77.17 ± 0.92 | 0.79 ± 0.01 |
| Forgetron | 18.64 ± 0.49 | 3.67 ± 0.05 | 80.38 ± 2.23 | 0.79 ± 0.01 | 25.80 ± 0.14 | 4.14 ± 0.07 | 73.61 ± 2.34 | 0.82 ± 0.01 |
| Projectron | 14.56 ± 0.33 | 13.14 ± 2.97 | 85.38 ± 1.82 | 4.37 ± 0.59 | 20.38 ± 0.21 | 11.34 ± 0.46 | 80.94 ± 2.05 | 3.46 ± 0.09 |
| FOGD | 15.48 0.34 | **0.06 0.00** | 85.08 1g.13 | 0.01 0.00 | 18.02 0.23 | **0.07 0.01** | 83.62 0.57 | 0.01 0.00 |
| FFTRL | 11.92 0.20 | 5.26 0.15 | **89.81 0.39** | **0.01 0.00** | 17.48 0.23 | 6.68 0.29 | 84.70 0.53 | **0.01 0.00** |
| Algorithm | Coil2000 | | | | W7a | | | |
| | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) |
| NORMA | 7.68 ± 0.17 | 231.33 ± 1.39 | 91.81 ± 1.05 | 114.43 ± 0.20 | 2.95 ± 0.02 | 2021.10 ± 248.08 | 97.14 ± 0.05 | 4003.87 ± 477.64 |
| ACCOSVM | 7.13 ± 0.09 | 2592.71 ± 59.48 | 92.89 ± 0.17 | 118.06 ± 0.74 | **2.03 ± 0.02** | 4927.57 ± 318.12 | **98.48 ± 0.01** | 1770.59 ± 78.54 |
| BNORMA | 7.59 ± 0.11 | 5.94 ± 0.03 | 91.19 ± 1.52 | 1.41 ± 0.01 | 4.27 ± 0.11 | 21.40 ± 1.34 | 96.11 ± 0.69 | 22.63 ± 1.34 |
| Forgetron | 11.19 ± 0.15 | 6.42 ± 0.04 | 92.34 ± 1.73 | 1.52 ± 0.01 | 4.36 ± 0.06 | 13.98 ± 1.04 | 96.69 ± 0.67 | 13.25 ± 0.07 |
| Projectron | 10.75 ± 0.16 | 25.96 ± 0.30 | 91.67 ± 2.14 | 10.53 ± 0.17 | 2.74 ± 0.03 | 45.78 ± 1.08 | 98.33 ± 0.10 | 64.99 ± 1.46 |
| FOGD | 7.27 ± 0.04 | **0.11 ± 0.00** | 92.80 ± 0.12 | 0.02 ± 0.00 | 3.47 ± 0.08 | **0.37 ± 0.01** | 96.67 ± 0.04 | 0.29 ± 0.02 |
| FFTRL | **6.90 ± 0.06** | 10.59 ± 10.4 | **93.55 ± 0.93** | **0.02 ± 0.01** | 3.36 ± 0.05 | 25.86 ± 0.29 | 96.92 ± 0.01 | **0.27 ± 0.01** |
| Algorithm | A7a | | | | Ijcnn1 | | | |
| | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) | Mistake Rate (%) | Train (s) | Accuracy (%) | Test (s) |
| NORMA | 17.99 ± 0.22 | 951.63 ± 18.63 | 82.42 ± 0.02 | 1958.85 ± 48.16 | 9.67 ± 0.16 | 4558.52 ± 104.74 | 90.26 ± 0.01 | *48542.60* |
| ACCOSVM | **16.34 ± 0.15** | 4473.56 ± 192.07 | 83.63 ± 0.05 | 1172.24 ± 24.88 | **6.15 ± 0.01** | 3382.09 ± 338.19 | **94.63 ± 0.05** | 2997.05 ± 50.54 |
| BNORMA | 20.54 ± 0.20 | 12.82 ± 0.40 | 80.91 ± 0.48 | 12.51 ± 0.40 | 13.03 ± 0.05 | 45.91 ± 1.22 | 87.07 ± 2.34 | 16.86 ± 0.85 |
| Forgetron | 24.57 ± 0.22 | 16.97 ± 0.86 | 78.23 ± 0.45 | 13.51 ± 0.27 | 16.60 ± 0.14 | 20.22 ± 0.25 | 86.96 ± 3.68 | 44.39 ± 0.96 |
| Projectron | 20.51 ± 0.37 | 366.03 ± 15.12 | 80.47 ± 0.92 | 372.86 ± 3.45 | 7.64 ± 0.07 | 162.50 ± 1.63 | 94.31 ± 0.90 | 454.11 ± 2.73 |
| FOGD | 18.77 ± 0.14 | **0.23 ± 0.01** | 82.48 ± 1.04 | 0.16 ± 0.00 | 9.33 ± 0.02 | **0.43 ± 0.01** | 91.46 ± 0.40 | 0.77 ± 0.01 |
| FFTRL | 16.84 ± 0.13 | 25.12 ± 0.49 | **84.15 ± 0.19** | **0.15 ± 0.00** | 9.35 ± 0.02 | 49.27 ± 0.44 | 91.90 ± 0.20 | **0.75 ± 0.01** |

the extraordinary time efficiency of kernel approximation methods should be attributed to the linear online learning framework. Moreover, both FOGD and FFTRL also show better mistake rate and accuracy in most cases, which demonstrates that kernel approximation scheme is suitable for large scale online learning.

Finally, we analyzed the performance of FFTRL. It seems surprising to find that FFTRL gets the lowest mistake rate or highest accuracy, even outperforms NORMA in some datasets (such as, Spambase, Coil2000, A7a and Ijcnn1). The reasons may lie in two aspects. The first reason is the appropriate choice of sample number $D$. According to the conclusions from [12], choosing a too large value of $D$ will result in under-fitting for small datasets, and choosing a too small value of $D$ will result in over-fitting. The second reason is the well-designed per-coordinate learning rate. Except from FFTRL, all the gradient-based algorithms adopt the global learning rate schedule. However, we need to use the learning rate to reflect our confidence of each dimension in online setting, which indicates the global learning rate schedule is not the optimal choice. Besides, FFTRL also produces a sparser model than FOGD as expected. Unfortunately, the benefits of sparsity brought to FFTRL are largely obscured by the efficiency of linear learning framework since the test time of FOGD and FFTRL are generally the

**Table 3.** Sparsity promotion of FFTRL against FOGD.

| | Titanic | Spambase | Banana | Phoneme |
|---|---|---|---|---|
| FOGD | Baseline | Baseline | Baseline | Baseline |
| FFTRL | +272.80 | +176.40 | +193.40 | +118.60 |
| | Coil2000 | W7a | A7a | Ijcnn1 |
| FOGD | Baseline | Baseline | Baseline | Baseline |
| FFTRL | +143.00 | +203.40 | +76.40 | +102.20 |

same. To validate the advantage of our proposed method over FOGD, we listed the number of zero components in the weight vector $\mathbf{w}$ in Table 3, where the number of zero coefficients in FOGD is taken as the baseline. From Table 3, we can obviously see that the model generated by FFTRL is much sparser than that of FOGD.

## 4 Conclusion

In this paper, we present a novel sparse algorithm FFTRL for solving large-scale online kernel binary classification tasks. The basic idea of FFTRL is to approximate a kernel function via functional approximation technique, which enables us to transform the original online kernel learning task into an approximate linear online learning task. Random Fourier features are used as the kernel approximation scheme, and then a new high dimensional feature space is induced in this process. We further adopt FTRL to find a sparse solution in the new feature space. In theory, we analyze the regret bound of our proposed algorithm.

We performed extensive experiments to evaluate the performance of FFTRL and other state-of-the-art online kernel learning methods. Our promising results show that FFTRL enjoys both time efficiency and accuracy. Moreover, the sparsity produced by FFTRL fits the need of high dimensional and large-scale data scenarios, making FFTRL suitable for real-world applications. In future work, we plan to extend our work by exploring the field of multi-label online classification tasks.

## References

1. Alcalá-Fdez, J., et al.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple Valued Logic Soft Comput. **17**, 255–287 (2011)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**(11) (2006)
3. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Tracking the best hyperplane with a simple budget perceptron. Mach. Learn. **69**(2), 143–167 (2007)

4. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 1–27 (2011)
5. Crammer, K., Kandola, J., Singer, Y.: Online classification on a budget. In: Advances in Neural Information Processing Systems, vol. 16 (2003)
6. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: A kernel-based perceptron on a budget. SIAM J. Comput. **37**(5), 1342–1372 (2008)
7. Duchi, J., Singer, Y.: Efficient online and batch learning using forward backward splitting. J. Mach. Learn. Res. **10**, 2899–2934 (2009)
8. Guo, H., Zhang, A., Wang, W.: An accelerator for online SVM based on the fixed-size KKT window. Eng. Appl. Artif. Intell. **92**, 103637 (2020)
9. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Trans. Signal Process. **52**(8), 2165–2176 (2004)
10. Li, B., Zhao, P., Hoi, S.C., Gopalkrishnan, V.: PAMR: passive aggressive mean reversion strategy for portfolio selection. Mach. Learn. **87**(2), 221–258 (2012)
11. Li, X., Plale, B., Vijayakumar, N., Ramachandran, R., Graves, S., Conover, H.: Real-time storm detection and weather forecast activation through data mining and events processing. Earth Sci. Inf. **1**(2), 49–57 (2008)
12. Lu, J., Hoi, S.C., Wang, J., Zhao, P., Liu, Z.Y.: Large scale online kernel learning. J. Mach. Learn. Res. **17**(47), 1 (2016)
13. McMahan, B.: Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 525–533. JMLR Workshop and Conference Proceedings (2011). ¡!– Wrong Number: [l]1 –¿
14. McMahan, H.B., et al.: Ad click prediction: a view from the trenches. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1222–1230 (2013)
15. Orabona, F., Keshet, J., Caputo, B.: The projectron: a bounded kernel-based perceptron. In: Proceedings of the 25th International Conference on Machine Learning, pp. 720–727 (2008)
16. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in Neural Information Processing Systems 20 (2007)
17. Rudin, W.: Fourier Analysis on Groups. Courier Dover Publications (2017)
18. Shalev-Shwartz, S., et al.: Online learning and online convex optimization. Found. Trends® Mach. Learn. **4**(2), 107–194 (2012)
19. Xiao, L.: Dual averaging method for regularized stochastic learning and online optimization. In: Advances in Neural Information Processing Systems 22 (2009)
20. Zhang, K., Lan, L., Wang, Z., Moerchen, F.: Scaling up kernel SVM on limited resources: a low-rank linearization approach. In: Artificial Intelligence and Statistics, pp. 1425–1434. PMLR (2012)