Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Online kernel learning with nearly constant support vectors

Ming Lin [a], Lijun Zhang [b], Rong Jin [b], Shifeng Weng [c], Changshui Zhang [a,*]

[a] State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology(TNList), Department of Automation, Tsinghua University, Beijing 10084, China
[b] Computer Science and Engineering, Michigan State University, East Lansing, MI 48823, USA
[c] School of Electronics and Information, Zhejiang Wanli University NO 8 South Qianhu Road, Ningbo City, Zhejiang Province 315100, China

A R T I C L E I N F O

Article history:
Received 30 April 2013
Received in revised form
14 September 2015
Accepted 3 October 2015
Communicated by Steven Hoi
Available online 14 October 2015

Keywords:
Online learning
Kernel machine
Nyström
Sample complexity

A B S T R A C T

Nyström method has been widely used to improve the computational efficiency of batch kernel learning. The key idea of Nyström method is to randomly sample M support vectors from the collection of T training instances, and learn a kernel classifier in the space spanned by the randomly sampled support vectors. In this work, we studied online regularized kernel learning using the Nyström method, with a focus on the sample complexity, i.e. the number of randomly sampled support vectors that are needed to yield the optimal convergence rate $O(1/T)$, where T is the number of training instances received in online learning. We show that, when the loss function is smooth and strongly convex, only $O(\log^2 T)$ randomly sampled support vectors are needed to guarantee an $O(\log T/T)$ convergence rate, which is almost optimal except for the log T factor. We further validate our theory by an extensive empirical study.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Kernel machines are powerful tools to handle non-linear data learning tasks. Kernel function improves the flexibility of learning methods by implicitly mapping data to a high dimensional space [1]. Kernel based methods have been successfully applied to classification, dimensionality reduction, and clustering, including kernel SVM [2], kernel logistic regression [3], kernel PCA [4] and spectral clustering [5].

A main drawback of kernel based methods is their high demand on both storage space and computational cycles. Given T training instances, the storage requirement and computational cost are $O(T^2)$. Online learning improves the efficiency of kernel learning by going through the training data once [6–8]. Although it reduces the storage requirement by retrieving training instances one by one in online settings, its time complexity is still $O(T^2)$ because each received training instance can potentially be a support vector. Budget online learning [9–12] ameliorates this problem by limiting number of support vectors of the intermediate classifiers obtained by online learning. But the final classifier obtained by online-to-batch conversion [13] may still include most of the training examples as support vector, leading to a high computational cost in prediction.

An alternative approach to efficient kernel learning is to generate a compact representation for the target kernel classifier. Random Fourier feature [14] and polynomial feature [15,16] are two examples of this category. Both methods approximate kernel function by an expansion of appropriate basis functions. Since the approximation is made independently from data, both schemes are data independent, and therefore often leads to suboptimal performance, according to the analysis in [17].

In this work, we focus on Nyström method [18], another popular scheme for improving the efficiency of kernel learning. It randomly samples M instances as support vectors from a collection of T training examples, and learns a kernel classifier in the subspace spanned by the randomly sampled support vectors. Nyström method was first introduced to kernel learning in [19], and has found applications in kernel classification [20,21], spectral clustering [22], and eigenmap embedding [23].

The generalization performance of Nyström method was examined recently in [17], in which the authors show that Nyström method is overall more effective for batch kernel learning than random Fourier feature because of the data dependence induced by Nyström method. Unlike [17] where the effect of Nyström method was examined in batch learning, we focus on *online* regularized kernel learning where training examples are received sequentially with one at each time, and every training example will be discarded after it is used to update the prediction model. We show that, in online regularized kernel learning, only $O(\log^2 T)$ randomly sampled support vectors are needed to achieve an $O(\log T/T)$ convergence rate. Compared to the optimal convergence rate $O(1/T)$ for online regularized kernel learning, our result is almost optimal except for the $\log(T)$ factor. We verify our theory by an extensive

* Corresponding author.

http://dx.doi.org/10.1016/j.neucom.2015.10.002
0925-2312/© 2015 Elsevier B.V. All rights reserved.

empirical study. To the best of our knowledge, this is the first work that analyzes the performance of Nyström method in online settings, with nearly optimal guarantee.

The rest of this paper is organized as follows. Section 2 discusses the related work on kernel learning and Nyström method. Section 3 describes online regularized kernel learning with Nyström method in details, and present our theoretical guarantees, where the detailed proof can be found in the Appendix. Section 4 demonstrates our theory by an extensive empirical study. Section 5 encloses our paper with future work.

2. Related work

In this section we briefly review the related works on kernel learning.

2.1. Kernel learning

As mentioned in the introduction section, the main challenge arising from kernel learning is its high demand on computational cycles and storage space. Below we list several major efforts in improving the efficiency of kernel learning.

Explicit kernel feature mapping: Explicit kernel feature mapping approximates a kernel similarity function by a finite feature representation of data. When the kernel is shift-invariant, it can be accomplished by random Fourier sampling [14]. It was shown in [24] that the generalization error caused by random Fourier features is bounded by $O(1/\sqrt{M})$, where M is the number of random Fourier features. When kernel is not shift-invariant, polynomial feature representation is often used to approximate the kernel function by a truncated Taylor expansion [25,15,16]. The key limitation of methods in this category is that the kernel approximations are made independently from the data distribution, leading to suboptimal performance as argued in [17].

Batch sparse kernel learning: Sparse kernel learning aims to compute a compact representation of kernel classifier with a limited number of support vectors. A common idea is to confine the support vectors in a reduced set of training data [26,27]. The reduced set is constructed either by a greedy method [3,28] or by minimizing some criterion as a complementary process [26,29,30]. In [31,32], the authors consider approaches for sparse kernel learning by making appropriate changes to the objective function. In [30], the authors propose to first learn a dense kernel SVM through batch learning, and then approximate the learned SVM by a sparse one. Although the output classifier is sparse in support vectors, most methods in this category are expensive in both storage space and computational cost as they have to deal with the full kernel matrix in the first place.

Budget online learning: Budget Online Learning restricts the number of support vectors to a given budget. Crammer et al. [9] propose the first budget online learning, which was refined later on in [10]. The key of these approaches is to remove the support vectors of least significance to maintain the budget. The Forgetron [33] is the first budget online learning with theoretical guarantees. It decreases the weights of support vectors at each iteration of online learning and removes the support vectors with the smallest weight when the number of support vectors exceeds the budget. Randomized Budget Perceptron [34] achieves similar bounds as Forgetron by replacing one of the randomly selected support vectors with new instances. Projectron [35] improves these ideas by making a new training example to be a support vector only when it is far from the space spanned by the existing support vectors. Peilin et al. [8] developed a stochastic gradient descent based method for budget online learning. Very recently, Wang et al. [36] study the convergence rate of online kernel with random Fourier features and Nyström features. Although their analysis is very similar with ours, they only give a linear sampling complexity for Nyström features, which is significantly inferior than the results presented in this paper. It is important to note that our method needs to sample support vectors beforehand and needs independent assumptions, that are not required in conventional analysis. The method proposed in this paper can be viewed as an extension of Projectron with online-to-batch conversion. In our analysis, the optimal convergence rate is only possible with online-to-batch conversion, thus it is not surprising that Projectron cannot provide such guarantee.

Sparse online kernel learning: Sparse online kernel learning maintains a sparse support vector set at each online step and outputs a compact kernel machine without having to take the online-to-batch conversation. Engel et al. [37] proposed a sparse kernel support vector machine for kernel regression, where a new training instance is added into the set of support vectors only when it cannot be linearly approximated by current support vectors. Their method does not provide any guarantee on generalization bounds. Zhang et al. [11] proposed a stochastic gradient method for sparse online kernel learning that shares the similar idea as [8].

2.2. Nyström method

Nyström method was first proposed by Nyström [18]. It was introduced by Williams and Seeger [19] to accelerate kernel learning, followed by [20]. Various sampling schemes have been proposed to improve the effectiveness of Nyström method [38,39].

Nyström method is often viewed as a low rank matrix approximation method: it approximates the kernel matrix K by a low rank matrix $\widehat{K}$. Several analyses have been developed to bound the difference between K and $\widehat{K}$ [20,40–43]. The most interesting result is given in [40,41], which stated that when the rank of kernel matrix is r, only $O(r \log r)$ samplings is needed by Nyström method to achieve a zero error in approximating the kernel matrix. The impact of the low rank approximation made by Nyström method on the generalization performance of kernel learning was studied in [44]. In [17], the authors proved that the generalization error caused by Nyström method is low bounded by $\mathcal{O}(N/M)$, where N is the number of training instances, which can be improved to $O(N/M^{p-1})$ if the eigenvalues of the kernel matrix follow a p power law. Different from the existing studies, we focus on the generalization performance of Nyström method in the online setting.

3. Online kernel learning with nyström method

3.1. Background and notation

Let $\kappa(\cdot,\cdot)$ be a bounded kernel function, i.e., $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, $|\kappa(\boldsymbol{x},\boldsymbol{x}')| \le 1$. Denote by $\mathcal{H}$ the *Reproducing Kernel Hilbert Space* (RKHS) endowed with $\kappa(\cdot,\cdot)$. Let $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ be the inner product on $\mathcal{H}$ and $\|\cdot\|_{\mathcal{H}}$ be the corresponding norm. Let $\boldsymbol{z}_t = \{\boldsymbol{x}_t, y_t\}, t = 1,\ldots,T$ be the sequence of training examples received in the online setting, where $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$ is a column vector of d dimension and the label $y_t \in \{+1,-1\}$. We assume that all the training examples are i.i.d. samples from an unknown underlying distribution $\mathbb{P}(\boldsymbol{x}, y)$.

Given the sequence of training examples $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_T\}$, we define the kernel matrix $K \in \mathbb{R}^{T\times T}$ by $K_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Through out the paper, we assume that the kernel to be bounded,

$$|\kappa(\boldsymbol{x},\boldsymbol{y})| \le 1.$$

We denote by X the set of training instances, and by V the set of support vectors, i.e.,

$$X \triangleq \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}, \quad V \triangleq \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_M\},$$

where each $\hat{\boldsymbol{x}}_i$ is a support vector used by the kernel classifier and M is the number of support vectors. We define $\mathcal{H}_V$, the subspace spanned by the support vectors in V, i.e.

$$\mathcal{H}_V \triangleq \text{span}\{\kappa(\hat{\boldsymbol{x}}_1,\cdot),\kappa(\hat{\boldsymbol{x}}_2,\cdot),\ldots,\kappa(\hat{\boldsymbol{x}}_M,\cdot)\}.$$

For brevity, we define

$$\{K_{V,V}\}_{i,j} \triangleq \kappa(\hat{\boldsymbol{x}}_i,\hat{\boldsymbol{x}}_j), \quad \{K_{X,V}^\top\}_{i,j} = \{K_{V,X}\}_{i,j} \triangleq \kappa(\hat{\boldsymbol{x}}_i,\boldsymbol{x}_j),$$
$$K_{V,\boldsymbol{x}} \triangleq [\kappa(\hat{\boldsymbol{x}}_1,\boldsymbol{x}),\ldots,\kappa(\hat{\boldsymbol{x}}_M,\boldsymbol{x})]^\top$$

where $K_{V,V}$ is the kernel matrix for all the support vectors in V, $K_{V,X}$ is the kernel matrix between the support vectors in V and all the training instances in X, and $K_{V,\boldsymbol{x}}$ is a column vector representing the kernel similarities between instance $\boldsymbol{x}$ and all the support vectors in V. We will use K^{-1} to represent the Moore–Penrose pseudoinverse of matrix K.

Similar to most studies of online learning, we use $\ell(y,z)$ for loss function, where y is the true label and $z=f(\boldsymbol{x})$ is the prediction made by function $f\in\mathcal{H}:\mathcal{X}\mapsto R$. We assume that $\ell(y,z)$ to be β-smooth and convex in the second argument with bounded gradient, i.e.

$$|\ell'(y,z)-\ell'(y,z')| \le \beta|z-z'|, \quad |\ell'(y,z)| \le G \quad \forall z,z',$$

where the derivative is taken with respect to z. Let $\overline{\ell}(f)=\text{E}_{\boldsymbol{x},y}\ell(y,f(\boldsymbol{x}))$ be the expected loss of f, and f_* be the minimizer of $\overline{\ell}(\cdot)$, i.e.,

$$f_* = \arg\min_f \overline{\ell}(f) \triangleq \text{E}_{\boldsymbol{x},y}\ell(y,f(\boldsymbol{x})).$$

We assume that f_* exists but not necessarily unique.

To ensure the learnability of our problem [45,46], we minimize the regularized loss $\mathcal{L}(f)$ instead of $\overline{\ell}$, which is defined as

$$\mathcal{L}(f) \triangleq \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \overline{\ell}(f).$$

Clearly $\mathcal{L}(f)$ is λ-strongly convex and $(\lambda+\beta)$-smooth. Similarly, we define the regularized loss at step t of online learning as

$$\mathcal{L}_t(f) \triangleq \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \ell(y_t,f(\boldsymbol{x}_t)).$$

To measure the generalization performance of the classifier $\hat{f}$ learned in an online learning process, we will compare $\mathcal{L}(\hat{f})$ with $\mathcal{L}(f_*)$. For online regularized kernel learning using Nyström method, we will show that $\mathcal{L}(\hat{f})-\mathcal{L}(f_*)$ is bounded by $O(\log T/T)$ as long as the number of randomly sampled support vectors is $O(\log^2 T)$.

3.2. The Nyström method

Nyström method is usually viewed as a low rank matrix approximation approach. In order to approximate the full kernel matrix K, it first randomly samples M training instances as support vectors from a collection of T training examples, and approximates K by $\widehat{K}=K_{X,V}K_{V,V}^{-1}K_{X,V}^\top$. This is equivalent to providing, for each instance $\boldsymbol{x}$, a new vector representation $\phi(\boldsymbol{x})=[\phi_1(\boldsymbol{x}),\ldots,\phi_M(\boldsymbol{x})^\top]$,

$$\phi(\boldsymbol{x}) = K_{V,V}^{-1/2}K_{V,\boldsymbol{x}}.$$

As a result, instead of learning a kernel classifier $f(\cdot)\in\mathcal{H}$, we will learn a linear classifier $f(\boldsymbol{x})=\langle\boldsymbol{w},\boldsymbol{x}\rangle$, where the parameter $\boldsymbol{w}\in\mathbb{R}^M$ is a vector of M dimension.

An alternative view is to treat Nyström method as a constrained kernel learning problem [23], in which the solution is restricted to a subspace spanned by the randomly sampled support vectors in V, i.e.

$$\min_{f\in\mathcal{H}_V} \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \frac{1}{T}\sum_{t=1}^{T}\ell(y_t,f(\boldsymbol{x}_t))$$

Since $f\in\mathcal{H}_V$, we can express f as a linear combination of the randomly sampled support vectors, i.e.

$$f(\boldsymbol{x}) = \sum_{i=1}^{M}\alpha_i\kappa(\widehat{\boldsymbol{x}}_i,\boldsymbol{x}) = \boldsymbol{\alpha}^{\mathrm{T}}K_{V,\boldsymbol{x}} \tag{1}$$

where $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_M)^\top$ are the combination coefficients, and therefore, we have

$$\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^T K_{V,V}\boldsymbol{\alpha}.$$

Under this view, in this paper we will show that, the quality of the kernel classifier obtained by Nyström method is determined by the projection errors $\{\delta_t\}_{t=1}^T$ of individual training examples given as

$$\delta_t^2 = \min_{f\in\mathcal{H}_V}\|\kappa(\boldsymbol{x}_t,\cdot)-f(\cdot)\|_{\mathcal{H}}^2 \tag{2}$$

It is easy to verify that (see Appendix B)

$$\delta_t^2 = \kappa(\boldsymbol{x}_t,\boldsymbol{x}_t) - K_{V,\boldsymbol{x}_t}^{\mathrm{T}}K_{V,V}^{-1}K_{V,\boldsymbol{x}_t}. \tag{3}$$

and the optimal solution to the projection problem in (2) is given by

$$Pf(\cdot) = P\kappa(\boldsymbol{x}_t,\cdot) = \phi(\boldsymbol{x}_t)^\top\phi(\cdot), \tag{4}$$

where P is the projection operator that projects a function $f(\cdot)\in\mathcal{H}$ into the subspace $\mathcal{H}_V$.

3.3. Online regularized kernel learning with Nyström method

Algorithm 1 presents the detailed steps for online regularized kernel learning with Nyström method. It first randomly samples M training instances as support vectors from a collection of T training examples. Following the standard stochastic gradient descent approach, it then computes the gradient of the loss function for each training example and updates the solution using the computed gradient. The key difference between Algorithm 1 and standard online kernel learning is at step 7, where the computed gradient $\nabla\mathcal{L}_t$ is projected into the subspace $\mathcal{H}_V$ through the operator P before it is used for updating the solution. This is where the Nyström method plays the role. We set step size $\eta_t\propto 1/[\lambda T]$ because the regularized loss function is strongly convex and according to [47], setting step size $1/[\lambda T]$ yields optimal convergence rate (up to a $\log T$ factor).

Algorithm 1. Online Nyström Kernel Learning (**ONL**).

```
1: Input: Budget of support vectors $M$, regularization parameter $\lambda>0$, and training sequence $\{\boldsymbol{x}_t,y_t\}$, $t=1,2,\cdots T$.
   Choose step size parameter $c\in(1,\infty)$. We assume $c=3$ if not specified.
2: Randomly sampling $M$ training instance as support vectors,
   $V=\{\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2,\ldots,\hat{\boldsymbol{x}}_M\}$
3: $f_1(\cdot)=0$
4: for all $t=1,2,\cdots T$ do
5:   Retrieve training instance $\boldsymbol{x}_t$. Set step size $\eta_t=c/(\lambda t)$.
6:   Compute the gradient:$\nabla\mathcal{L}_t(f_t)=\lambda f_t+\ell'(y_t,f_t(\boldsymbol{x}_t))\kappa(\boldsymbol{x}_t,\cdot)$.
7:   Update the kernel classifier
       $f_{t+1}=f_t-\eta_t P(\nabla\mathcal{L}_t(f_t))$
       $\quad =(1-\eta_t\lambda)f_t-\eta_t\ell'(y_t,f(\boldsymbol{x}_t))P\kappa(\boldsymbol{x}_t,\cdot)$
8:end for
9: Output: $\widehat{f}_T=\frac{1}{T}\sum_{i=1}^T f_i$
```

In the implementation, we need to store the support vector set V, kernel matrix $K_{V,V}^{-1/2}$, and the coefficients $\{\alpha_i\}_{i=1}^M$ for the kernel classifier $f(\cdot)$ learned by Algorithm 1. The space complexity is $d\times M$ for storing V, is $M\times M$ for storing $K_{V,V}$, and is M for storing $\{\alpha_i\}_{i=1}^M$. Thus, the total space complexity is $O(M^2+dM)$. Since the cost of computing $P\kappa(\boldsymbol{x}_t,\cdot)$ at each iteration of online learning is $O(M^2)$, the total time complexity of running Algorithm 1 is $O(M^2\,T)$ where $M=O(\log T)$, which is almost linear in number of training examples, a significant improvement over standard online kernel learning.

One potential issue of Algorithm 1 is that the Nyström method requires to pass data at least once to get a uniform sampling, which is an undesirable feature in online learning. This flaw is amenable if the data is coming in independently and identically (*I. I.D.*). In this I.I.D. case, we only need to simply take the first M instances as support vectors. Other possible solution includes reservoir sampling and adaptive sampling, which we leave for future development.

The following theorem claims that only $O(\log^2 T)$ randomly sampled support vectors (i.e. $M = O(\log^2 T)$) are needed by Algorithm 1 to achieve an almost optimal convergence rate for online regularized kernel learning.

Theorem 1. *Let $(\mathbf{u}_i, \lambda_i), i = 1, \ldots, T$ be the eigenvectors and eigenvalues of K that are ranked in the descending order of eigenvalues. Define $\widehat{\lambda}_r \triangleq \sum_{i=r+1}^{T} \lambda_i$ as the tail sum of the eigenvalues, and μ as the coherence measure of U, i.e.*

$$\mu = \max_{1 \le i \le T, 1 \le j \le T} T u_{i,j}^2$$

Assume that $r \ge \max\{C_{ab} \ln(3T^3), 4\log T/\gamma\}$, where C_{ab} and γ are universal positive constants. Then, with a high probability, we have

$$\mathcal{L}\left(\widehat{f}_T\right) - \mathcal{L}(f_*) \le O(\log T/T).$$

provided the number of randomly sampled support vectors M is sufficiently large, i.e.

$$M \ge \max\left\{\frac{96\beta\mu^2 r\widehat{\lambda}_r}{\lambda}, 16\mu^2\left(\frac{\log T}{\gamma}\right)^2, \mu^2 C_{ab}^2 \log^2(3T^3), 4\mu^2 C_{ab} \log(3T^3)\frac{\log T}{\gamma}\right\},$$

The complete proof for Theorem 1 can be found in Appendix.

The coherence measure μ used by Theorem 1 is used to indicate how well individual training examples are related to the others [40]. A small coherence measure implies that most training examples can be well approximated by the other training examples, and therefore it is possible to accurately learn the kernel classifier using a set of randomly sampled examples as support vectors. Constants C_{ab} and γ were first introduced in [48] for compressive sensing, and were later on utilized for analyzing the generalization performance of Nyström method in [40]. We note that the number of support vectors M depends on $\widehat{\lambda}_r$, the tail sum of eigenvalues. When the kernel matrix is nearly low-rank, we will have a small value for $\widehat{\lambda}_r$, and consequentially a small number of randomly sampled support vectors will be required by Algorithm 1. In the case when $\widehat{\lambda}_r = O(\lambda)$, we have $M = O(\log^2 T)$, an almost constant number of support vectors that are needed to achieve a convergence rate of $O(\log T/T)$. Although $O(\log^2 T)$ is slightly worse than the $O(\log T)$ sample complexity for requested by the Projectron algorithm [7], the focus of our algorithm is different from that of Projectron: Algorithm 1 is designed for online *regularized* kernel learning that achieves an $O(\log T/T)$ convergence rate; in contrast, Projectron is designed for standard online kernel learning that achieves an $O(1/\sqrt{T})$ convergence rate, which is significantly slower than ours. We finally note that according to [47], the optimal convergence rate for online regularized kernel learning is $O(1/T)$, and the convergence rate of Algorithm 1 is almost optimal except for a $\log T$ factor.

4. Experiments

In this section, we conduct experiments to verify our theory, i.e. only a small number of random sampled support vectors is needed for online regularized kernel learning with Nyström method. We compare the proposed algorithm Online Nyström kernel Learning (**ONL**) with four state-of-the-art baselines for large-scale kernel learning:

- Pegasos (**PEG**) algorithm using full kernel matrix [49], a state-of-the-art algorithm for large-scale learning based on the theory of stochastic gradient descent.
- Projectron (**PTR**) [35], a state-of-the-art algorithm for budget online kernel learning. We set the threshold for the projection distance threshold, a key parameter for the Projectron algorithm, to be 0.001, according to the recommendation by the original authors [35].
- Forgetron (**FGT**) [33], which throws support vectors according to their weights after exceeding budget.
- Budget Online Gradient Descend (**BOGD**) [8]. We tune the parameter $\eta = 1/8, \gamma = 2$ in BOGD as suggested in [8].

For the proposed algorithm and baseline methods, we set the number of support vectors M=2000, which appears to yield the best trade off between classification accuracy and computational efficiency.

We test all the algorithms on six public large data sets: *adult*, *COD–RNA*, *covtype*, *ijcnn1*, *mnist*, and *vehicle*. All the features are normalized to unit norm. For the mnist dataset, we convert it into binary classification problem by treating digit '0' to '4' as positive instances and '5' to '9' as negative instances. We use the Gaussian kernel in the experiments for its popularity. The kernel width σ is set to be the fifth percentile of the pairwise distances [50,51]. Table 1 summarized the statistics of data sets. We choose the logit function $\ell(z) = \log(1 + \exp(-z))$ as the loss function because it is a smooth loss function and has been used by many studies. We randomly sample 20% data for testing and use the remaining for training. We apply cross validation to tune the regularization parameter λ in the set of $\{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$. All experiments are repeated five times. The averaged performances over five trials are reported.

4.1. Convergence rate

In order to evaluate the convergence rate of the proposed algorithm ONL, we compare the loss function $\mathcal{L}(f_t)$ of the proposed algorithm over iterations to that of the Pegasos algorithm. Since Pegasos is known to yield a convergence rate of $O(\log T/T)$ for online regularized kernel learning, this comparison will tell if the theoretical convergence result given in Theorem 1 is correct. In this experiment, for the convenience of comparison, we set the regularization parameter $\lambda = 10^{-4}$ for both methods. The value $\lambda = 10^{-4}$ is shown to be optimal on most dataset. We fix λ for both methods because we are interested in the convergence rate of objective value of a fix optimization problem. For different value of λ, the only difference is the declivity of the curve. In Fig. 1, we plot the loss function value on testing set against iteration steps. According to Fig. 1, we observe that ONL converges to almost the same value of loss function as Pegasos with increasing number of iterations, indicating an $O(\log T/T)$ convergence for ONL. The curve of ONL is always above Pegasos, which indicates ONL has a larger constant in the convergence rate, as shown in our upper bound. The peak in early stages of iteration of ONL and Pegasos is because

Table 1
Statistics of data sets.

Data set	#features	#instance	σ
adult	123	182,357	0.78
COD-RNA	8	271,617	0.0057
covtype	54	581,012	0.35
ijcnn1	22	49,990	0.57
mnist	780	60,000	0.86
vehicle	100	78,823	0.86

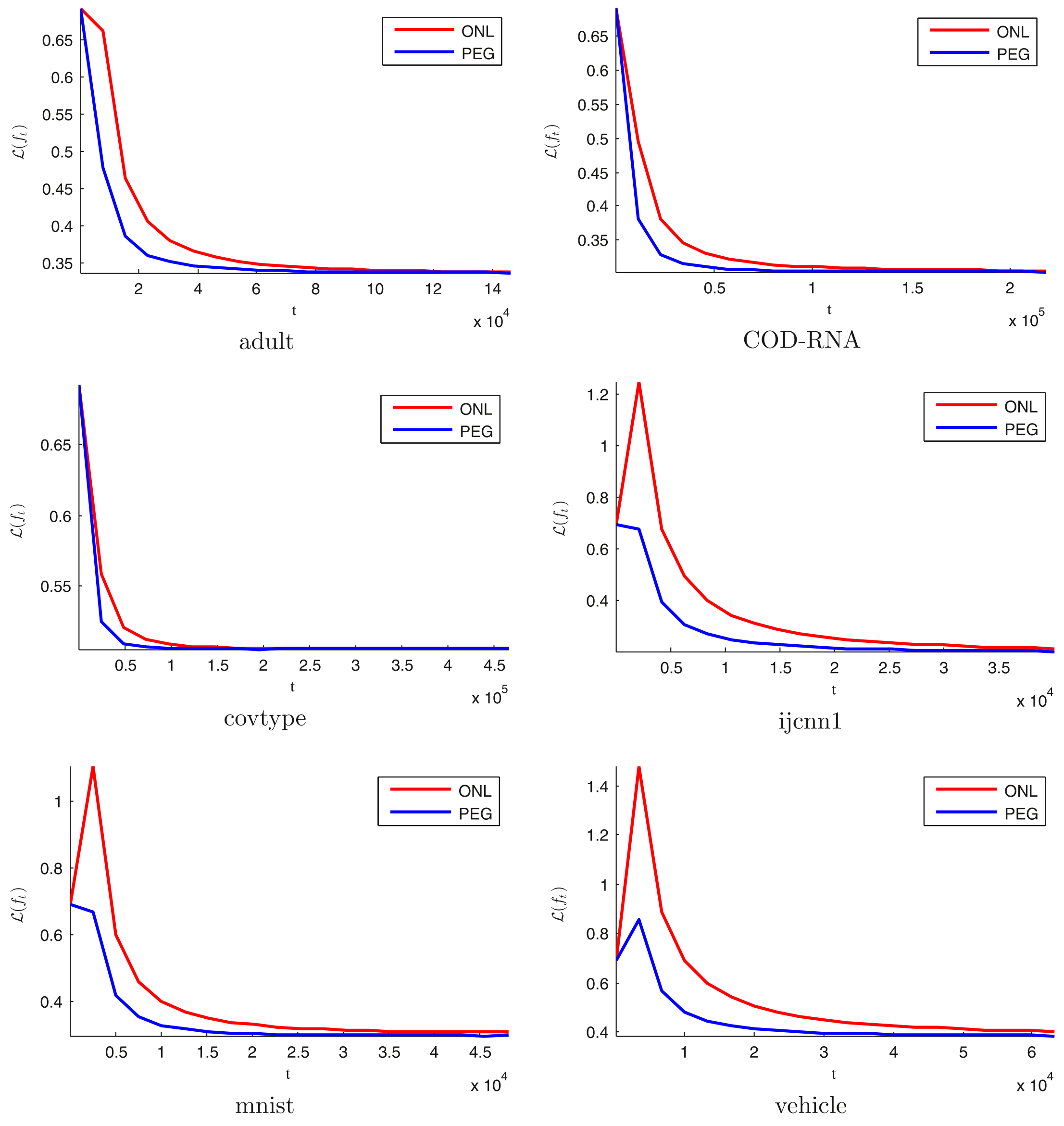


Fig. 1. Evaluation of convergence rates.

we choose step size $1/(\lambda t)$, which is large at the beginning, so the changes in objective value is intensive at the first few steps.

To further verify that the convergence rate is on order of $O(1/t)$, we take logarithm both on loss function value and t. If $f(t)=1/t$, we have $\log(f(t)) = -\log(t)$. Therefore the curve should be a line with slope -1. In Fig. 2, the x-axis is the logarithm of iteration steps. The y-axis is the logarithm of loss function value on testing set. From the figure, the slope of ONL is almost the same as that of Pegasos, except for the convergent part near the end of the iteration. This verifies that the convergence rate of ONL is the same as Pegasos, which is well known to be $O(1/t)$.

4.2. *Classification accuracy*

We evaluate the binary classification accuracies of different methods in Table 2. We observe that with only $M=2000$ support vectors, ONL achieves almost the same accuracy as Pegasos using full kernel for most datasets. Projectron performs significantly worse than the proposed algorithm on datasets vehicle and covtype, and slightly better than the proposed algorithm on the mnist and ijcnn. In addition, compared to the proposed algorithm, Projectron exhibits a significantly larger variance in its classification accuracy, making it a less reliable algorithm for choice. BOGD is reasonable good on adult, COD-RNA, ijcnn1, vehicle, and does not perform well on covtype, mnist. ONL is always better than BOGD on all datasets. Forgetron is the worst method in average and also has large variance, which is also reported by many previous researches [35].

In Fig. 3, we evaluate the performance of the proposed algorithm with the number of support vectors M varied in the set $\{50, 100, 200, 500, 1000, 2000\}$. We also include in Fig. 3 the classification accuracy of Projectron with varied number of support vectors, and the classification accuracy of Pegasos (highlighted by a straight dotted line). We observe that for several datasets (i.e. covtype, adult, and vehicle), even a few hundred of support vectors would be sufficient for the proposed algorithm to yield a classification performance similar to that of Pegasos using full kernel matrix. Similar to the observation from Table 2, we also observe that the Projectron algorithm exhibits a large variance in its classification performance in almost all cases.

4.3. *Running time*

In Table 3, we list the running time of different methods, where the number of support vectors is set as $M=2000$ for the proposed

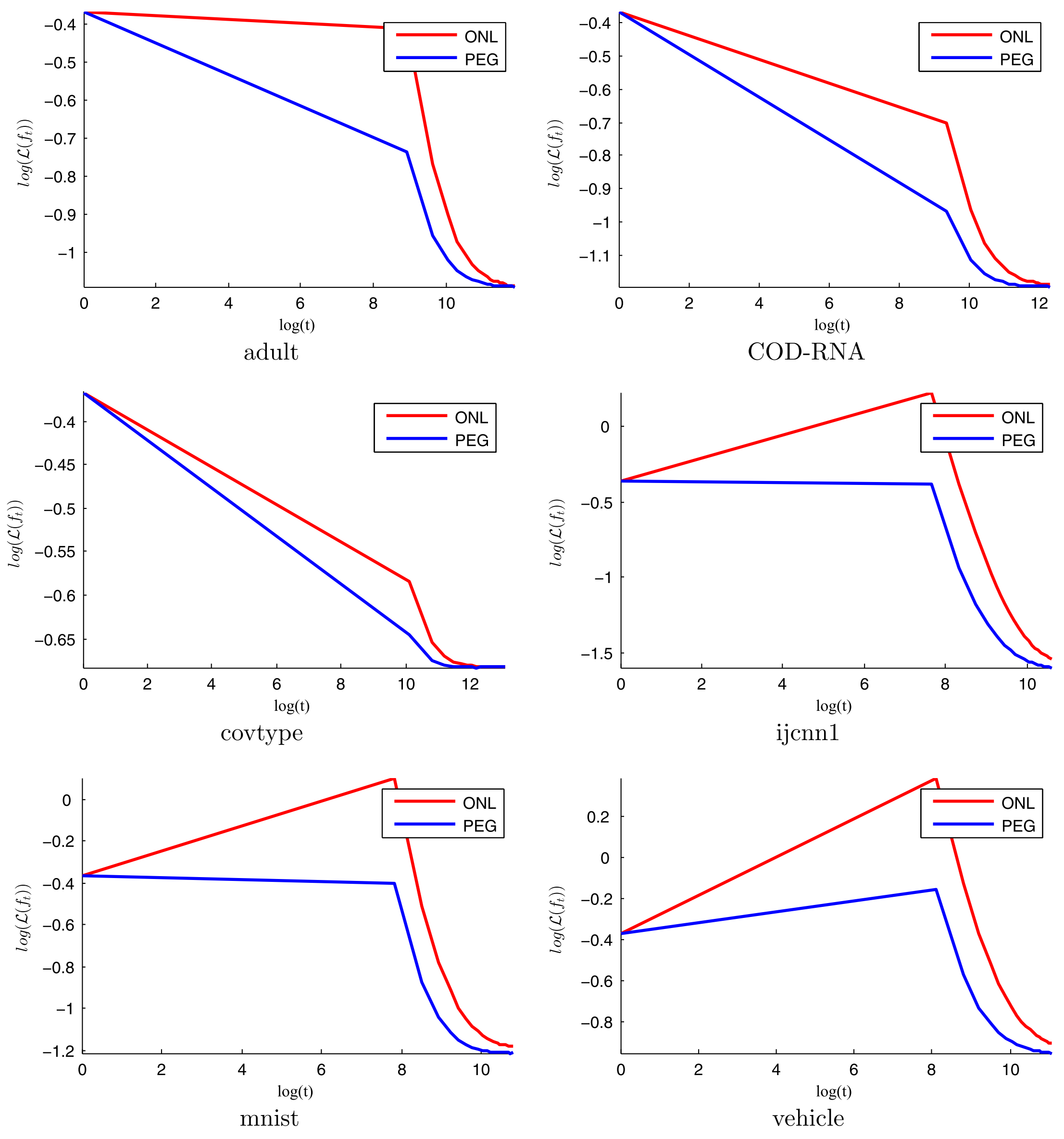


Fig. 2. Evaluation of convergence rates (loglog).

Table 2
Classification accuracy. The number of support vectors for the proposed algorithm ONL is set $M = 2000$.

Dataset	ONL	PTR	PEG	BOGD	FGT
adult	$84.9 \pm 0.2\%$	$83.0 \pm 1.78\%$	$85.08 \pm 0.3\%$	$82.7 \pm 0.9\%$	$70.1 \pm 7.9\%$
COD-RNA	$89.7 \pm 0.2\%$	$90.7 \pm 2.16\%$	$90.4 \pm 0.6\%$	$88.8 \pm 0.2\%$	$82.4 \pm 2.7\%$
covtype	$79.0 \pm 0.3\%$	$73.9 \pm 1.38\%$	$79.5 \pm 0.3\%$	$70.2 \pm 1.1\%$	$59.9 \pm 4.0\%$
ijcnn1	$92.6 \pm 0.3\%$	$95.9 \pm 0.68\%$	$94.0 \pm 0.2\%$	$90.3 \pm 0.3\%$	$86.5 \pm 2.4\%$
mnist	$92.0 \pm 0.7\%$	$93.8 \pm 3.14\%$	$93.7 \pm 0.5\%$	$79.0 \pm 3.2\%$	$69.6 \pm 8.3\%$
vehicle	$84.7 \pm 0.3\%$	$76.0 \pm 5.4\%$	$85.3 \pm 0.2\%$	$81.4 \pm 0.7\%$	$71.0 \pm 7.8\%$

algorithm. We observe that the proposed algorithm is significantly more efficient than Pegasos on all datasets except for the ijcnn dataset where both method share similar running time. The small difference in running time for the ijcnn dataset is mostly due to the fact that it is the smallest dataset used in our study. For some large datasets such as adult and covtype, the improvement made by the proposed algorithm in computational efficiency over Pegasos can be quite dramatic. For example, for the covtype dataset, it takes the proposed algorithm ONL less than 1/30 of the running time of Pegasos to complete the training process. Compared to the Projectron algorithm, the proposed algorithm is significantly more efficient on the datasets adult, covtype, and vehicle, and is less efficient on the other three datasets (i.e. COD-RND, ijcnn1, and mnist). This is because the number of support vectors in Projectron is adaptively increasing during iterations. Although ONL and Projectron share same computational complexity, Projectron will be faster at beginning due to small number of support vectors. However, after large enough iterations, Projectron will select all M support vectors and becomes slower. Thus on large scale dataset such as covtype, ONL is faster than Projectron, while on small dataset such as ijcnn1, Projectron seems to be more efficient. BOGD is very fast on all dataset, because each

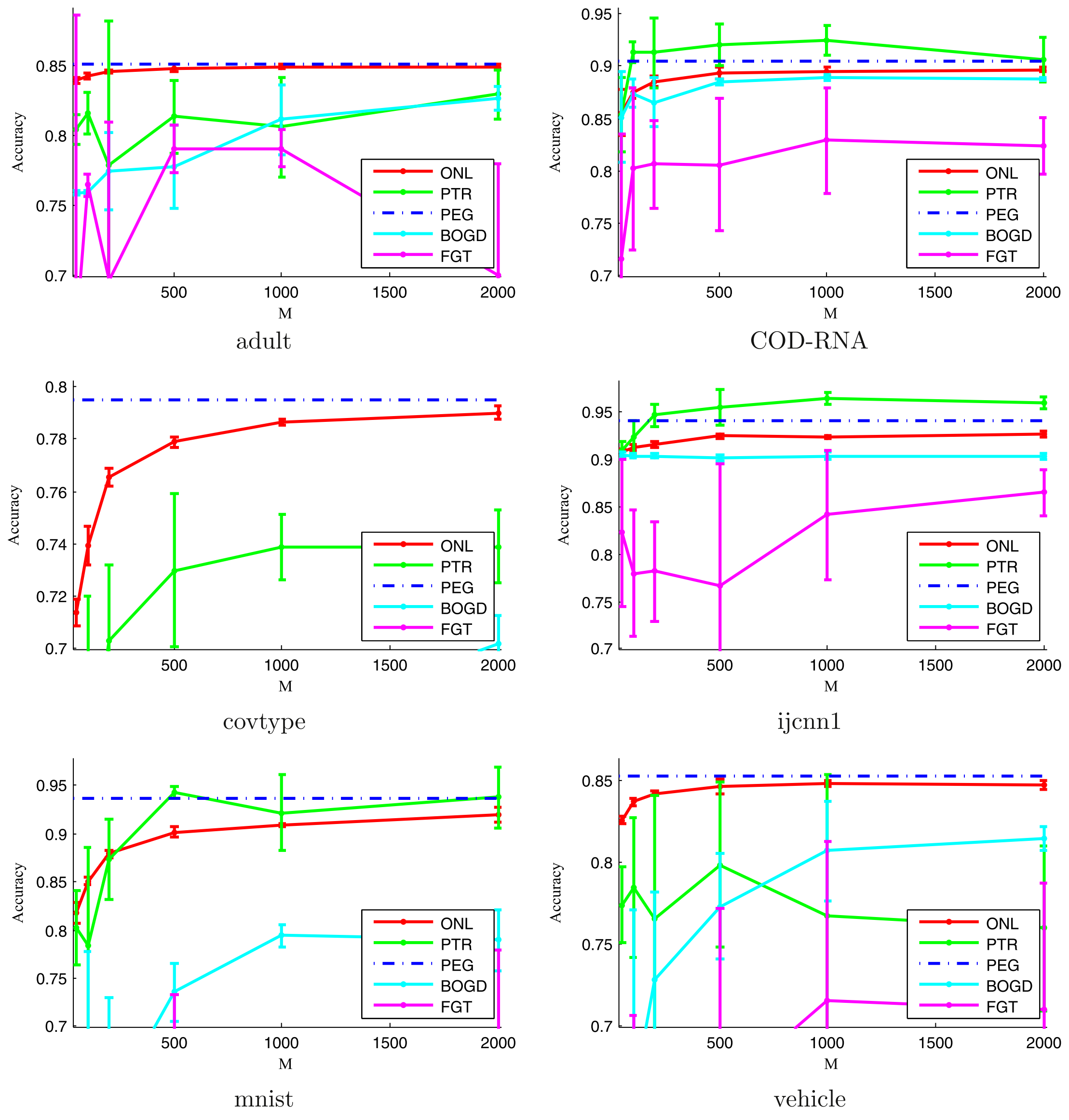


Fig. 3. Classification accuracy with varied number of support vectors *M*. The performance of Pegasos algorithm is highlighted by a straight line because it cannot dynamically adjust the number of support vectors.

Table 3
Running time (second) for different algorithms. The number of support vectors is fixed to be 2000 for the proposed algorithm ONL.

Dataset	ONL	PTR	PEG	BOGD	FGT
adult	590 ± 17	1170 ± 7	$1.5e4 \pm 0.6$	193 ± 14	$1.2e4 \pm 27$
COD-RNA	888 ± 30	88 ± 1	$6.3e3 \pm 0.07$	53 ± 0.2	$2.5e3 \pm 8$
covtype	$1.4e3 \pm 15$	$4.2e3 \pm 23$	$5.2e4 \pm 0.04$	243 ± 7.5	$1.3e3 \pm 26$
ijcnn1	271 ± 9	147 ± 8	222 ± 0.4	12 ± 0.36	417 ± 6
mnist	614 ± 12	407 ± 13	$4.6e3 \pm 1$	299 ± 63	$2.3e3 \pm 154$
vehicle	371 ± 20	611 ± 4	$2.2e3 \pm 0.004$	56 ± 2	$4.1e3 \pm 22$

iteration in BOGD is a simple gradient descend. Forgetron is only faster than full kernel Pegasos and much slower than other methods.

5. Conclusion

In this work, we study the problem of online regularized kernel learning with Nyström method. We show that when the kernel matrix is nearly low rank, with only $M = O(\log^2 T)$ randomly sampled support vectors, the proposed algorithm is able to yield an $O(\log T/T)$ convergence rate. We verify our theory by an extensive empirical study over six benchmark datasets. In summary, our work shows that we can actually train a kernel classifier efficiently without sacrificing the generalization performance with almost constant number of support vectors.

An open question is whether or not $\log^2 T$ sample complexity shown in our theory study can be further improved. We note that the additional $\log T$ factor was introduced in the proof of Theorem 1, where we relax the expectation of projection distance by an union of upper bounds. A more careful analysis may be developed to remove the additional $\log T$ and reduce the sample complexity from $\log^2 T$ to $\log T$. Another open question is if the classification performance of the proposed algorithm can be improved by adaptively selecting support vectors, instead of randomly sampling support vectors before the start of online learning, an approach that was taken by the Projectron algorithm. We will examine both open questions in the future.

Acknowledgments

We would like to thank the anonymous editors and reviewers for their insightful comments. This work is supported by 973 Program (2013CB329503), NSFC (Grant nos. 91120301, 91420203 and 61473167).

Appendix A. Proof of Theorem 1

To commence our proof, we first need the following lemma which bounds the norm of f_t. This lemma is important in our proof because the standard online learning requires the gradient of $\mathcal{L}(f_t)$ to be bounded.

Lemma 1. *Let $f_1, f_2, \ldots, f_T$ be the solutions in Algorithm 1. We have, the norm of f_T is bounded, followed by boundness of the gradient of $\mathcal{L}(f_t)$:*

$$\|f_t\|_{\mathcal{H}} \le \frac{22G}{\lambda}, \quad \|\nabla\mathcal{L}(f_t)\|_{\mathcal{H}} \le 23G.$$

Proof. For $2 \le t \le c+1$, we have

$$\|f_t\| \overset{(1)}{\le} \left|1-\frac{c}{t-1}\right| \|f_{t-1}\| + \frac{c}{\lambda(t-1)}G = \left(\frac{c}{t-1}-1\right)\|f_{t-1}\| + \frac{c}{\lambda(t-1)}G$$
$$\overset{(2)}{\le} (c-1)\|f_{t-1}\| + \frac{c}{\lambda}G \overset{(3)}{\le} \frac{(1+(1+c)^t-c)cG}{(-2+c)(-1+c)\lambda}.$$

(1) comes from gradient descent rule. (2) comes from the fact that $2 \le t$ in the beginning of the proof. (3) comes from expanding the recursive inequality followed by geometrical series.

For $t > c+1$, let $t = c+1+u$, $u \ge 1$,

$$\|f_t\| \le \left|1-\frac{c}{t-1}\right|\|f_{t-1}\| + \frac{cG}{\lambda(t-1)} \overset{(1)}{=} \left(1-\frac{c}{t-1}\right)\|f_{t-1}\| + \frac{cG}{\lambda(t-1)}$$
$$\overset{(2)}{\le} \frac{G}{\lambda} + \frac{(-G+\|f_{c+1}\|\lambda)c!u!}{\lambda(c+u)!} \le \frac{G}{\lambda} + \frac{(-G+\|f_{c+1}\|\lambda)}{\lambda(c+1)}$$
$$= \frac{c}{1+c}\frac{G}{\lambda} + \frac{\|f_{c+1}\|}{c+1}.$$

(2) comes from the recursive inequality about f_t in the above line. Take the equality (1), then we solve the recursive equation to get the expression of f_t at $t = c+1$. Then we get (2). The term f_{c+1} is the intermediate solution at $t = c+1$. Choosing $c=3$, since the gradient of ℓ is bounded, $\|f_{c+1}\|$ must be bounded by a constant. $\|f_{c+1}\|$ Then the proof is completed.

Please note that the constants given in this lemma is calculated at $c=3$. If we take a different value of c, the constants should be modified correspondingly.□

In Algorithm 1, we first compute the online gradient descent intermediate solution f'_t based on current solution f_t, then project f'_t to the subspace $\mathcal{H}_V$ to get f_{t+1}. We introduce an auxiliary variable u_t to transform the projection operation into additive operation. More precisely, we define u_t such that

$$f_{t+1} = f_t - \eta_t \nabla\mathcal{L}_t(f_t) + u_t.$$

Therefore, in Algorithm 1, u_t equals to $f_{t+1} - f'_t$, while in traditional online learning, $u_t = 0$. We have the following basic inequality:

$$\begin{aligned}\|f_{t+1}-f\|^2_{\mathcal{H}} &= \|f_t - \eta_t\nabla\mathcal{L}_t(f_t) - u_t - f\|^2_{\mathcal{H}} = \|f_t - f\|^2_{\mathcal{H}} + \|\eta_t\nabla\mathcal{L}_t(f_t) \\ &\quad + u_t\|^2_{\mathcal{H}} - 2\langle \eta_t\nabla\mathcal{L}_t(f_t), f_t - f\rangle_{\mathcal{H}} - 2\langle u_t, f_t - f\rangle_{\mathcal{H}} \\ &\le \|f_t - f\|^2_{\mathcal{H}} + 2\eta_t^2\|\nabla\mathcal{L}_t(f_t)\|^2_{\mathcal{H}} + 2\|u_t\|^2_{\mathcal{H}} \\ &\quad -2\langle \eta_t\nabla\mathcal{L}_t(f_t), f_t - f\rangle_{\mathcal{H}} - 2\langle u_t, f_t - f\rangle_{\mathcal{H}} \le \|f_t - f\|^2_{\mathcal{H}} \\ &\quad + 1058\eta_t^2 G^2 + 2\|u_t\|^2_{\mathcal{H}} - 2\langle \eta_t\nabla\mathcal{L}_t(f_t), f_t - f\rangle_{\mathcal{H}} \\ &\quad - 2\langle u_t, f_t - f\rangle_{\mathcal{H}}.\end{aligned}$$

The first inequality is because $(a+b)^2 \le 2a^2 + 2b^2$. The second inequality is because the boundness of $\|\nabla\mathcal{L}(f_t)\|_{\mathcal{H}}$. Clear up the inequality, we have

$$\langle \nabla\mathcal{L}_t(f_t), f_t - f\rangle_{\mathcal{H}} \le \frac{\|f_t - f\|^2_{\mathcal{H}} - \|f_{t+1} - f\|^2_{\mathcal{H}}}{2\eta_t} + 1058\eta_t G^2$$
$$+\frac{1}{\eta_t}\|u_t\|^2_{\mathcal{H}} - \frac{1}{\eta_t}\langle u_t, f_t - f\rangle_{\mathcal{H}}. \tag{A.1}$$

The auxiliary variable u_t comes from the projection step, thus is the key part of our proof. Otherwise if $u_t = 0$, the proof is trivially a standard online learning proof. To address the trouble of u_t, we decompose the excess risk into several parts, then we bound each part separately. Finally, we cancel the terms containing u_t by strong convexity of the loss function.

Let the step size $\eta_t = c/(\lambda t)$. From the strong convexity of $\mathcal{L}$,

$$\begin{aligned}&\sum_{t=1}^T \mathcal{L}(f_t) - \mathcal{L}(f_*) \\ &\le \sum_{t=1}^T \langle \nabla\mathcal{L}(f_t), f_t - f_*\rangle_{\mathcal{H}} - \frac{\lambda}{2}\|f_t - f_*\|^2_{\mathcal{H}} \\ &= \sum_{t=1}^T \langle \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}} - \frac{\lambda}{2}\|f_t - f_*\|^2_{\mathcal{H}} + \langle \nabla\mathcal{L}(f_t) \\ &\quad - \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}} \\ &\le \sum_{t=1}^T \frac{\|f_t - f_*\|^2_{\mathcal{H}} - \|f_{t+1} - f_*\|^2_{\mathcal{H}}}{2\eta_t} + 1058\eta_t G^2 + \frac{1}{\eta_t}\|u_t\|^2_{\mathcal{H}} \\ &\quad - \frac{1}{\eta_t}\langle u_t, f_t - f_*\rangle_{\mathcal{H}} - \frac{\lambda}{2}\|f_t - f_*\|^2_{\mathcal{H}} + \langle \nabla\mathcal{L}(f_t) - \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}} \\ &= -T\|f_{T+1} - f_*\|^2_{\mathcal{H}} + \sum_{t=1}^T \left(\frac{\lambda}{2c} - \frac{\lambda}{2}\right)\|f_t - f_*\|^2_{\mathcal{H}} \\ &\quad + 1058\eta_t G^2 + \frac{1}{\eta_t}\|u_t\|^2_{\mathcal{H}} \\ &\quad - \frac{1}{\eta_t}\langle u_t, f_t - f_*\rangle_{\mathcal{H}} + \langle \nabla\mathcal{L}(f_t) - \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}} \\ &\le \sum_{t=1}^T \left(\frac{\lambda}{2c} - \frac{\lambda}{2}\right)\|f_t - f_*\|^2_{\mathcal{H}} + 1058\eta_t G^2 + \frac{1}{\eta_t}\|u_t\|^2_{\mathcal{H}} \\ &\quad - \frac{1}{\eta_t}\langle u_t, f_t - f_*\rangle_{\mathcal{H}} + \langle \nabla\mathcal{L}(f_t) - \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}}\end{aligned} \tag{A.2}$$

$$\begin{aligned}&= \left\{\sum_{t=1}^T \left(\frac{\lambda}{2c} - \frac{\lambda}{2}\right)\|f_t - f_*\|^2_{\mathcal{H}}\right\} + \left\{1058G^2\sum_{t=1}^T \eta_t\right\} \\ &\quad + \left\{\sum_{t=1}^T \langle \nabla\mathcal{L}(f_t) - \nabla\mathcal{L}_t(f_t), f_t - f_*\rangle_{\mathcal{H}}\right\} \\ &\quad + \left\{\sum_{t=1}^T \frac{1}{\eta_t}\|u_t\|^2_{\mathcal{H}}\right\} + \left\{\sum_{t=1}^T -\frac{1}{\eta_t}\langle u_t, f_t - f_*\rangle_{\mathcal{H}}\right\}.\end{aligned} \tag{A.3}$$

The first inequality is because the strong convexity. The second inequality is from Eq. (A.1). The second equality is because $\tau = c/(\lambda t)$ then summation over t. The third inequality is because the first term in the last step is always non-positive. In the last step, we collect the upper bound into five terms. Our strategy is to bound these terms separately. The first term is strictly negative if we choose c large enough. The second term is clearly bounded by $O(\log T)$. The third term is a martingale sequence thus can be bound by $O(\sum_t \|f_t - f_*\|_{\mathcal{H}}^2)$. The last two terms are related to the projection hence are the key parts of our proof. We will show that the last two terms are small enough so they can be canceled out by the first strictly negative term.

The second term is bounded by the upper bound of harmonic series,

$$1058G^2 \sum_{t=1}^{T} \eta_t \le 1058 \frac{cG^2}{\lambda}(1+\log T).$$

The third term can be bounded by Bartlett's Theorem [52]. Define

$$\xi_t = \langle \nabla \mathcal{L}(f_t) - \nabla \mathcal{L}_t(f_t), f_t - f_* \rangle_{\mathcal{H}}$$

as martingale difference sequence,

$$\|\xi_t\| \le 2G(\|f_t\|_{\mathcal{H}} + \|f_*\|_{\mathcal{H}}) = 2G\left(\frac{22G}{\lambda} + \|f_*\|_{\mathcal{H}}\right) \triangleq b,$$

$$\mathrm{Var}\{\xi_t\} \le G^2 \|f_t - f_*\|_{\mathcal{H}}^2.$$

For any auxiliary variable $\tau \ge 0$, with probability at least $1-\delta$,

$$\begin{aligned}
&\sum_{t=1}^{T} \langle \nabla \mathcal{L}(f_t) - \nabla \mathcal{L}_t(f_t), f_t - f_* \rangle_{\mathcal{H}} \\
&\le 2\max\left\{2\sqrt{G^2 \sum_t \|f_t - f_*\|_{\mathcal{H}}^2}, b\sqrt{\log \frac{\log_2 T}{\delta}}\right\}\sqrt{\log \frac{\log_2 T}{\delta}} \\
&\le \sqrt{16G^2 \sum_t \|f_t - f_*\|_{\mathcal{H}}^2}\sqrt{\log \frac{\log_2 T}{\delta}} + 2b \log \frac{\log_2 T}{\delta} \\
&= \sqrt{\frac{16G^2 \tau}{\lambda} \frac{\lambda}{\tau} \sum_t \|f_t - f_*\|_{\mathcal{H}}^2}\sqrt{\log \frac{\log_2 T}{\delta}} + 2b \log \frac{\log_2 T}{\delta} \\
&\le \frac{16G^2 \tau}{\sqrt{2}\lambda} \log \frac{\log_2 T}{\delta} + \frac{\lambda}{\sqrt{2}\tau} \sum_t \|f_t - f_*\|_{\mathcal{H}}^2 + 2b \log \frac{\log_2 T}{\delta}.
\end{aligned}$$

The last step is because $\sqrt{ab} \le (a+b)/\sqrt{2}$.

To bound the last terms, we first notice that

$$u_t = f_{t+1} - \{f_t - \eta_t \nabla \mathcal{L}_t(f_t)\}.$$

Define

$$a_t = \ell'(y_t, f(\boldsymbol{x}_t))$$

which is the gradient of the second argument of $\ell(y,z)$,

$$u_t = -\{f_t - \eta_t \nabla \mathcal{L}_t(f_t)\} + P\{f_t - \eta_t \nabla \mathcal{L}_t(f_t)\} = \eta_t a_t P_\perp \kappa(\boldsymbol{x}_t, \cdot).$$

$P_\perp = I - P$ is the projection operator which projects $\kappa(\boldsymbol{x}_t, \cdot)$ to the subspace which is orthogonal to $\mathcal{H}_V$. From this observation, the fourth term is bounded by

$$\sum_{t=1}^{T} \frac{1}{\eta_t} \|u_t\|_{\mathcal{H}}^2 = \sum_{t=1}^{T} \eta_t a_t^2 \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}^2 \le G^2 \sum_{t=1}^{T} \eta_t \le \frac{G^2 c}{\lambda}(1+\log T).$$

The second inequality is because $|\kappa(\boldsymbol{x}, \boldsymbol{x})| \le 1$. The fifth term is bounded by

$$\begin{aligned}
\sum_{t=1}^{T} -\frac{1}{\eta_t}\langle u_t, f_t - f_* \rangle_{\mathcal{H}} &= \sum_{t=1}^{T} -\langle a_t P_\perp \kappa(\boldsymbol{x}_t, \cdot), f_t - f_* \rangle_{\mathcal{H}} \\
&= \sum_{t=1}^{T} -\langle P_\perp \nabla \ell_t(f_t), f_t - f_* \rangle_{\mathcal{H}} \\
&= \sum_{t=1}^{T} -\langle \nabla \ell_t(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}} \\
= \sum_{t=1}^{T} -\langle \nabla \ell_t(f_t) - \nabla \overline{\ell}(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}} &- \langle \nabla \overline{\ell}(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}}.
\end{aligned}$$

The fourth equality is because $P_\perp$ is a projection operator thus is adjoint. The first term in the last step is a martingale so we can bound it by

$$\begin{aligned}
&\sum_{t=1}^{T} -\langle \nabla \ell_t(f_t) - \nabla \overline{\ell}(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}} \\
&\le \sum_{t=1}^{T} \left| \langle \nabla \mathcal{L}(f_t) - \nabla \mathcal{L}_t(f_t), f_t - f_* \rangle_{\mathcal{H}} \right|.
\end{aligned}$$

The inequality holds because $\|P_\perp(f_t - f_*)\|_{\mathcal{H}} \le \|(f_t - f_*)\|_{\mathcal{H}}$. For the second term, we bound it by

$$\begin{aligned}
&\sum_{t=1}^{T} -\langle \nabla \overline{\ell}(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}} \\
&= \sum_{t=1}^{T} \langle \nabla \overline{\ell}(f_*) - \nabla \overline{\ell}(f_t), P_\perp(f_t - f_*) \rangle_{\mathcal{H}} \\
&= \sum_{t=1}^{T} \langle P_\perp\{\nabla \overline{\ell}(f_*) - \nabla \overline{\ell}(f_t)\}, f_t - f_* \rangle_{\mathcal{H}} \\
&\le \sum_{t=1}^{T} \|P_\perp\{\nabla \overline{\ell}(f_*) - \nabla \overline{\ell}(f_t)\}\|_{\mathcal{H}} \|f_t - f_*\|_{\mathcal{H}} \\
&= \sum_{t=1}^{T} \|\mathrm{E}\{[\ell_t'(f_*) - \ell_t'(f_t)] P_\perp \kappa(\boldsymbol{x}_t, \cdot)\}\|_{\mathcal{H}} \|f_t - f_*\|_{\mathcal{H}} \\
&\le \sum_{t=1}^{T} \mathrm{E}\{\|\ell_t'(f_*) - \ell_t'(f_t)\|_{\mathcal{H}} \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}\} \|f_t - f_*\|_{\mathcal{H}} \\
&\le \sum_{t=1}^{T} \mathrm{E}\{\beta \|f_*(\boldsymbol{x}_t) - f_t(\boldsymbol{x}_t)\|_{\mathcal{H}} \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}\} \|f_t - f_*\|_{\mathcal{H}} \\
&= \sum_{t=1}^{T} \mathrm{E}\left\{\beta \|\langle f_* - f_t, \kappa(\boldsymbol{x}_t, \cdot)\rangle_{\mathcal{H}}\| \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}\right\} \|f_t - f_*\|_{\mathcal{H}} \\
&\le \sum_{t=1}^{T} \mathrm{E}\{\beta \|f_* - f_t\|_{\mathcal{H}} \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}\} \|f_t - f_*\|_{\mathcal{H}} \\
&= \sum_{t=1}^{T} \beta \{\mathrm{E}\|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}\} \|f_t - f_*\|_{\mathcal{H}}^2.
\end{aligned}$$

As we expected, the last term in Eq. (A.3), which comes from sampling, is on order of $O(\sum_{t=1}^{T} \|f_t - f_*\|_{\mathcal{H}}^2)$. The coefficient is determinated by the smoothness parameter β and the expected projection distance $\mathrm{E}\|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}$. The key idea is to show that the projection distance is small enough so we can cancel this term with the first term in Eq. (A.3). To this end, we borrow the proofs of Nyström in [53]. The following lemma reveals the relationship between Projectron and Nyström method. We omitted the proof because it is a direct evaluation.

Lemma 2. *Given training instances X. K is the kernel matrix,* $\{K\}_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. *Let* $\widehat{K} = K_{V,X}^{\mathrm{T}} K_{V,V}^{-1} K_{V,X}$ *is the approximated kernel matrix by Nyström method with i.i.d. sampled support vector set V. We have*

$$\|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}} = \left|\{K - \widehat{K}\}_{t,t}\right|.$$

This lemma tells that the projection distance is in fact the absolute value of the diagonal element of $K - \widehat{K}$. For a tight bound, we need to bound the expectation $\left|\{K - \widehat{K}\}_{t,t}\right|$. However, if we relax a bit by taking the union bound, we can greatly simplify our proofs by borrowing existing results in [53].

Lemma 3. *With probability at least* $1-\delta$,

$$\mathrm{E}\|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}} \le \frac{1}{T}\sum_{t=1}^{T} \left|\{K - \widehat{K}\}_{t,t}\right| + 2\sqrt{\frac{2\log(2/\delta)}{T}} \le \max_t \|P_\perp \kappa(\boldsymbol{x}_t, \cdot)\|_{\mathcal{H}}$$

$$+2\sqrt{\frac{2\log(2/\delta)}{T}}.$$

Proof. The first inequality follows Bernstein's inequality. The second inequality takes the union bound.□

The following theorem in [53] bounds the maximum value of projection distance.

Theorem 2 (*Theorem 15 in Mahdavi et al. [53]*). *Give T training instance. K is $T\times T$ kernel matrix. Let $K=U\Sigma U^{\mathrm{T}}$ be the eigenvalue decomposition with $\Sigma=\mathrm{diag}\{\lambda_1,\lambda_2,\dots\}$, where λ_i is the largest i-th eigenvalue of K. μ is the coherence coefficient of U. Let $r\ge\max\left\{C_{ab}\ln(3T^3),4\log T/\gamma\right\}$, where C_{ab} and γ are some certain positive constant. We have, with the sampling number M of support vectors larger than*

$$M\ge\mu^2\max\left\{16\left(\frac{\log T}{\gamma}\right)^2,C_{ab}^2\log^2(3T^3),4C_{ab}\log(3T^3)\frac{\log T}{\gamma}\right\},$$

with a probability $1-2T^{-3}$, we have

$$\max_t\|P_\perp\kappa(\boldsymbol{x}_t,\cdot)\|_{\mathcal{H}}\le\frac{16\mu^2r}{M}\sum_{i=r+1}^{T}\lambda_i.$$

Theorem 2 tells that if we sample large enough M, the maximum projection distance is very small. The maximum projection distance is also related to the $\sum_{i=r+1}^{T}\lambda_i$, which is the summation over $T-r$ smallest eigen values. We define

$$\widehat{\lambda}_r\triangleq\sum_{i=r+1}^{T}\lambda_i$$

as the accumulation of tail eigenvalues. If the kernel is low rank, then $\widehat{\lambda}_r=0$ and we get a perfect recovery of original kernel space.

Now we are ready to prove Theorem 1.

Proof of Theorem 1. Combine all the above together,

$$\begin{aligned}\sum_{t=1}^{T}\mathcal{L}(f_t)-\mathcal{L}(f_*)\le&\left\{\sum_{t=1}^{T}\left(\frac{\lambda}{2c}-\frac{\lambda}{2}\right)\|f_t-f_*\|_{\mathcal{H}}^2\right\}+\left\{1058G^2\sum_{t=1}^{T}\eta_t\right\}\\&+\left\{\sum_{t=1}^{T}\langle\nabla\mathcal{L}(f_t)-\nabla\mathcal{L}_t(f_t),f_t-f_*\rangle_{\mathcal{H}}\right\}\\&+\left\{\sum_{t=1}^{T}\frac{1}{\eta_t}\|u_t\|_{\mathcal{H}}^2\right\}+\left\{\sum_{t=1}^{T}-\frac{1}{\eta_t}\langle u_t,f_t-f_*\rangle_{\mathcal{H}}\right\}\\\le&\left\{\sum_{t=1}^{T}\left(\frac{\lambda}{2c}-\frac{\lambda}{2}\right)\|f_t-f_*\|_{\mathcal{H}}^2\right\}+1058\frac{cG^2}{\lambda}(1+\log T)\\&+2\left\{\frac{16G^2\tau}{\sqrt{2}\lambda}\log\frac{\log_2T}{\delta}\right.\\&\left.+\frac{\lambda}{\sqrt{2}\tau}\sum_t\|f_t-f_*\|_{\mathcal{H}}^2+2b\log\frac{\log_2T}{\delta}\right\}\\&+\frac{G^2c}{\lambda}(1+\log T)+\sum_{t=1}^{T}\beta\frac{16\mu^2r\widehat{\lambda}_r}{M}\|f_t-f_*\|_{\mathcal{H}}^2\\\le&\left\{\sum_{t=1}^{T}\left(\frac{\lambda}{2c}-\frac{\lambda}{2}+\frac{\sqrt{2}\lambda}{\tau}+\beta\frac{16\mu^2r\widehat{\lambda}_r}{M}\right)\|f_t-f_*\|_{\mathcal{H}}^2\right\}\\&+O(\log T).\end{aligned}$$

If we force the coefficient in the first term is smaller than zero, we already prove $O(\log T/T)$ convergence rate of the excess risk. To this end, we set

$$c=3,\tau=6\sqrt{2},M\ge\frac{96\beta\mu^2r\widehat{\lambda}_r}{\lambda}.$$

It is easy to verify that the first term is strictly negative. Finally, from convexity of $\mathcal{L}$,

$$\mathcal{L}\left(\frac{1}{T}\sum_{t=1}^{T}f_t\right)-\mathcal{L}(f_*)\le\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(f_t)-\mathcal{L}(f_*)\le O(\log T/T).\square$$

Appendix B. The kernel space functional projection and Nyström method

We prove that the optimization problem Eq. (2) will lead to Eq. (4).

For any vector $\boldsymbol{u}$, denote $\Phi(\boldsymbol{u})$ to be the kernel mapping function. Therefore,

$$\kappa(\boldsymbol{u},\boldsymbol{v})=\Phi(\boldsymbol{u})^\top\Phi(\boldsymbol{v}).$$

We denote $\Phi(X_V)=[\Phi(\hat{\boldsymbol{x}}_1),\Phi(\hat{\boldsymbol{x}}_2),\dots,\Phi(\hat{\boldsymbol{x}}_M)]$. Let the singular value decomposition of $\Phi(X_V)$ to be

$$\Phi(X_V)=U\Sigma V^\top\triangleq[U_M,U_M^\perp][\Sigma_M,\mathbf{0}]^\top[V_M,V_M^\perp]^\top=U_M\Sigma_MV_M^\top.$$

Here we ambiguously use V on the right side of the singular value decomposition: the V in X_V denotes support set while the V in $U\Sigma V^\top$ denotes the right singular vector matrix.

The Nyström feature for any instance $\boldsymbol{u}$ is

$$\begin{aligned}\phi(\boldsymbol{u})&=K_{V,V}^{-1/2}K_{V,\boldsymbol{u}}=[\Phi(X_V)^\top\Phi(X_V)]^{-1/2}\Phi(X_V)^\top\Phi(\boldsymbol{u})\\&=[V_M\Sigma_MU_M^\top U_M\Sigma_MV_M^\top]^{-1/2}V_M\Sigma_MU_M^\top\Phi(\boldsymbol{u})=V_MU_M^\top\Phi(\boldsymbol{u}).\end{aligned}$$

On the other hand, denote $f(\cdot)$ to be the projection of $\kappa(\boldsymbol{x}_t,\cdot)$, we have

$$f(\boldsymbol{u})=\sum_{i=1}^{M}\alpha_i\kappa(\hat{\boldsymbol{x}}_i,u)=\boldsymbol{\alpha}^\top K_{V,\boldsymbol{u}}.$$

From Eq. (2), it is easy to see that $\boldsymbol{\alpha}=K_{V,V}^{-1}K_{V,\boldsymbol{x}_t}$. Therefore

$$\begin{aligned}f(\boldsymbol{u})&=\boldsymbol{\alpha}^\top K_{V,\boldsymbol{u}}=[K_{V,V}^{-1}K_{V,\boldsymbol{x}_t}]^\top K_{V,\boldsymbol{u}}=K_{V,\boldsymbol{x}_t}^\top K_{V,V}^{-1}K_{V,\boldsymbol{u}}\\&=\Phi(\boldsymbol{x}_t)^\top\Phi(X_V)[\Phi(X_V)^\top\Phi(X_V)]^{-1}\Phi(X_V)^\top\Phi(\boldsymbol{u})\\&=\Phi(\boldsymbol{x}_t)^\top U_MU_M^\top\Phi(\boldsymbol{u})=\Phi(\boldsymbol{x}_t)^\top U_MV_M^\top V_MU_M^\top\Phi(\boldsymbol{u})\\&=\Phi(\boldsymbol{x}_t)^\top\Phi(\boldsymbol{u}).\end{aligned}$$

The proof is completed.

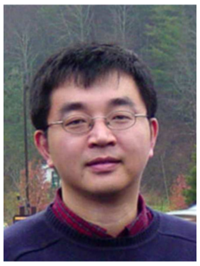
References

[1] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, 2001.
[2] I. Steinwart, A. Christmann, Support Vector Machines, Springer-Verlag, New York, 2008.
[3] J. Zhu, T. Hastie, Kernel logistic regression and the import vector machine, J. Comput. Graph. Stat. 14 (2005) 185–205.
[4] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, G. Rätsch, Kernel pca and de-noising in feature spaces, Adv. Neural Inf. Process. Syst. 11 (1999) 536–542.
[5] P. Netrapalli, Provable matrix completion using alternating minimization, NIPS Workshop, 2012.
[6] J. Kivinen, A. Smola, R. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (2004) 2165–2176.
[7] Y. Sun, F. Gomez, J. Schmidhuber, On the size of the online kernel sparsification dictionary, in: International Conference on Machine Learning, 2012, pp. 816–823.
[8] P. Zhao, J. Wang, P. Wu, R. Jin, S. Hoi, Fast bounded online gradient descent algorithms for scalable kernel-based online learning, in: International Conference on Machine Learning, 2012, pp. 169–176.
[9] K. Crammer, J.S. Kandola, Y. Singer, Online classification on a budget, Adv. Neural Inf. Process. Syst. (2003) 225–232.
[10] J. Weston, A. Bordes, L. Bottou, Online (and offline) on an even tighter budget, in: International Workshop on Artificial Intelligence and Statistics, 2005, pp. 413–420.
[11] L. Zhang, R. Jin, C. Chen, J. Bu, X. He, Efficient online learning for large-scale sparse kernel logistic regression, in: Association for the Advancement of Artificial Intelligence, 2012, pp. 1219–1225.
[12] L. Zhang, J. Yi, R. Jin, M. Lin, X. He, Online kernel learning with a near optimal sparsity bound, in: International Conference on Machine Learning, 2013, pp. 621–629.
[13] N. Cesa-Bianchi, A. Conconi, C. Gentile, On the generalization ability of on-line learning algorithms, IEEE Trans. Inf. Theory 50 (2004) 2050–2057.

[14] A. Rahimi, B. Recht, Random features for large-scale kernel machines, Adv. Neural Inf. Process. Syst. 20 (2007) 1177–1184.
[15] J. Xu, P. Pokharel, K. Jeong, J. Principe, An explicit construction of a reproducing gaussian kernel hilbert space, in: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 5, 2006, p. V.
[16] A. Cotter, J. Keshet, N. Srebro, Explicit approximations of the gaussian kernel, arXiv preprint arXiv:1109.4603 (2011).
[17] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, Z.-H. Zhou, Nyström method vs random fourier features: a theoretical and empirical comparison, Adv. Neural Inf. Process. Syst. (2012) 485–493.
[18] E. Nyström, Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben, Acta Math. 54 (1930) 185–204.
[19] C. Williams, M. Seeger, Using the nyström method to speed up kernel machines, Adv. Neural Inf. Process. Syst. (2001) 682–688.
[20] P. Drineas, M. Mahoney, On the nyström method for approximating a gram matrix for improved kernel-based learning, J. Mach. Learn. Res. 6 (2005) 2153–2175.
[21] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[22] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the nyström method, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 214–225.
[23] M. Ouimet, Y. Bengio, Greedy spectral embedding, in: International Workshop on Artificial Intelligence and Statistics, 2005, pp. 253–260.
[24] A. Rahimi, B. Recht, Weighted sums of random kitchen sinks: replacing minimization with randomization in learning, Adv. Neural Inf. Process. Syst. (2008) 1313–1320.
[25] C. Yang, R. Duraiswami, L. Davis, et al., Efficient kernel machines using the improved fast gauss transform, Adv. Neural Inf. Process. Syst. 17 (2005) 1561–1568.
[26] B. Schölkopf, P. Simard, V. Vapnik, A. Smola, Improving the accuracy and speed of support vector machines, Adv. Neural Inf. Process. Syst. 9 (1997) 375–381.
[27] Y. Lee, O. Mangasarian, Rsvm: Reduced support vector machines, in: The first SIAM International Conference on Data Mining, 2001, pp. 5–7.
[28] S. Keerthi, O. Chapelle, D. DeCoste, P. Bennett, Building support vector machines with reduced classifier complexity, J. Mach. Learn. Res. 7 (2006) 1493–1515.
[29] B. Scholkopf, S. Mika, C. Burges, P. Knirsch, K. Muller, G. Ratsch, A. Smola, Input space versus feature space in kernel-based methods, IEEE Trans. Neural Netw. 10 (1999) 1000–1017.
[30] A. Cotter, S. Shalev-Shwartz, N. Srebro, Learning optimally sparse support vector machines, in: International Conference on Machine Learning, 2013, pp. 266–274.
[31] E.E. Osuna, F. Girosi, Advances in Kernel Methods, MIT Press, Cambridge, MA, USA, 1999.
[32] M. Wu, B. Schölkopf, G. Bakır, A direct method for building sparse kernel learning algorithms, J. Mach. Learn. Res. 7 (2006) 603–624.
[33] O. Dekel, S. Shalev-Shwartz, Y. Singer, The forgetron: a kernel-based perceptron on a budget, SIAM J. Comput. 37 (2008) 1342–1372.
[34] G. Cavallanti, N. Cesa-Bianchi, C. Gentile, Tracking the best hyperplane with a simple budget perceptron, Mach. Learn. 69 (2007) 143–167.
[35] F. Orabona, J. Keshet, B. Caputo, The projectron: a bounded kernel-based perceptron, Int. Conf. Mach. Learn. (2008) 720–727.
[36] J. Wang, S.C. Hoi, P. Zhao, J. Zhuang, Z. Liu, Large scale online kernel classification, in: International Joint Conferences on Artificial Intelligence, 2013, pp. 1750–1756.
[37] Y. Engel, S. Mannor, R. Meir, Sparse online greedy support vector regression, Eur. Conf. Mach. Learn. (2002) 84–96.
[38] K. Zhang, J. Kwok, Density-weighted nyström method for computing large kernel eigensystems, Neural Comput. 21 (2009) 121–146.
[39] S. Kumar, M. Mohri, A. Talwalkar, Sampling methods for the nyström method, J. Mach. Learn. Res. 13 (2012) 981–1006.
[40] A. Talwalkar, A. Rostamizadeh, Matrix coherence and the nyström method, arXiv preprint (2010) arXiv:1004.2008.
[41] L. Mackey, A. Talwalkar, M.I. Jordan, Divide-and-conquer matrix factorization, Adv. Neural Inf. Process. Syst. (2011) 1134–1142.
[42] A. Gittens, The spectral norm error of the naive nyström extension, arXiv preprint (2011). arXiv:1110.5305.
[43] R. Jin, T. Yang, M. Mahdavi, Improved bound for the nyström's method and its application to kernel classification, arXiv preprint (2011)arXiv:1111.2262.
[44] C. Cortes, M. Mohri, A. Talwalkar, On the impact of kernel approximation on learning accuracy, in: Conference on Artificial Intelligence and Statistics 9 (2010) 113–120.
[45] S. Shalev-Shwartz, O. Shamir, N. Srebro, K. Sridharan, Learnability, stability and uniform convergence, J. Mach. Learn. Res. 11 (2010) 2635–2670.
[46] S. Ross, J.A. Bagnell, Stability conditions for online learnability, arXiv preprint (2011)arXiv:1108.3154.
[47] A. Rakhlin, O. Shamir, K. Sridharan, Making gradient descent optimal for strongly convex stochastic optimization, in: International Conference on Machine Learning, 2012, pp. 449–456.
[48] E. Candes, J. Romberg, Sparsity and incoherence in compressive sampling, Inverse Probl. 23 (2007) 969–985.
[49] S. Shalev-Shwartz, Y. Singer, N. Srebro, Pegasos: primal estimated sub-gradient solver for svm, in: International Conference on Machine Learning, 2007, pp. 807–814.
[50] P. Mallapragada, R. Jin, A. Jain, Non-parametric mixture models for clustering, Struct. Syntactic Stat. Pattern Recognit. (2010) 334–343.
[51] T. Abrahamsen, L. Hansen, A cure for variance inflation in high dimensional kernel principal component analysis, J. Mach. Learn. Res. 12 (2011) 2027–2044.
[52] P.L. Bartlett, V. Dani, T.P. Hayes, S. Kakade, A. Rakhlin, A. Tewari, High-probability regret bounds for bandit online linear optimization, in: Annual Conference on Learning Theory, 2008, pp. 335–342.
[53] M. Mahdavi, T. Yang, R. Jin, An improved bound for the nyström method for large eigengap, CoRR, 2012.

Ming Lin received his B.S. degree in the Department of Automation from Tsinghua University, Beijing, China, in 2008. He is currently a 5th year Ph.D. in the Department of Automation, Tsinghua University. His interests include optimization and statistics.

Lijun Zhang received the B.S. and Ph.D. degrees in Computer Science from Zhejiang University, China, in 2007 and 2012. Currently, he is working as a postdoc at Engineering & Computer Science, Michigan State University. His research interests include machine learning, information retrieval, and data mining.



Rong Jin received the B.A. degree in Engineering from Tianjin University, Tianjin, China, in 1993, the M.S. degree in Physics from Beijing University, Beijing, China, in 1996, and the M.S. and Ph.D. degrees in Computer Science from Carnegie Mellon University, Pittsburgh, PA, in 2000 and 2003, respectively. He is currently an Associate Professor with the Department of Computer Science and Engineering, Michigan State University, East Lansing. He is working on the areas of statistical machine learning and its application to information retrieval. He has published more than 80 conference and journal articles on related topics. Jin received the U.S. National Science Foundation Career Award in 2006.

Shifeng Weng was born in 1978. He received his Ph.D. in Pattern Recognition from Department of Automation, Tsinghua University, in 2005. He worked in Shanghai Baosteel Research Institute and Motolora China Lab from year 2005 to 2009. Currently, he is a vice-professor in Zhejiang Wanli University.

Changshui Zhang received his B.S. degree in Mathematics from the Peking University, Beijing, China, in 1986, and Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, China, in 1992. He is currently a Professor of the Department of Automation, Tsinghua University. He is an Associate Editor of the journal Pattern Recognition. His interests include artificial intelligence, image processing, pattern recognition, machine learning, evolutionary computation, etc.