Computationally Efficient Nyström Approximation using Fast Transforms

Si Si[⋆] SSI@CS.UTEXAS.EDU
Cho-Jui Hsieh[†] CHOHSIEH@UCDAVIS.EDU
Inderjit S. Dhillon[⋆] INDERJIT@CS.UTEXAS.EDU
[⋆]Department of Computer Science, Unigersity of Texas at Austin
[†] Departments of Statistics and Computer Science, University of California at Davis

Abstract

Our goal is to improve the *training* and *prediction* time of Nyström method, which is a widely-used technique for generating low-rank kernel matrix approximations. When applying the Nyström approximation for large-scale applications, both training and prediction time is dominated by computing kernel values between a data point and all landmark points. With m landmark points, this computation requires $\Theta(md)$ time (flops), where d is the input dimension. In this paper, we propose the use of a family of fast transforms to generate structured landmark points for Nyström approximation. By exploiting fast transforms, e.g., Haar transform and Hadamard transform, our modified Nyström method requires only $\Theta(m)$ or $\Theta(m \log d)$ time to compute the kernel values between a given data point and m landmark points. This improvement in time complexity can significantly speed up kernel approximation and benefit prediction speed in kernel machines. For instance, on the webspam data (more than 300,000 data points), our proposed algorithm enables kernel SVM prediction to deliver 98% accuracy and the resulting prediction time is 1000 times faster than LIBSVM and only 10 times slower than linear SVM prediction (which yields only 91% accuracy).

1. Introduction

In recent years, significant effort has been expended in trying to scale up kernel machines; however, standard kernel machines such as kernel SVM or kernel ridge regression suffer from slow training and prediction which prohibits their use in real-world large-scale applications. One of the key issues is to compute and store the kernel matrix, which usually takes $\Theta(n^2 d)$ time and $\Theta(n^2)$ space, where n is the number of training samples and d is the dimensionality. A widely-used approach to alleviate this problem is to approximate the kernel matrix by its low-rank approximation, as a result of which kernel methods can be solved efficiently.

There has been a vast body of work on low-rank approximation methods, among which Nyström method has drawn considerable attention for very large-scale datasets. It has been shown in (Williams & Seeger, 2001; Li et al., 2010; Rudi et al., 2015; Hsieh et al., 2014a) that Nyström based kernel approximation is efficient for scaling up kernel machines in both training and prediction phases. In Nyström approximation, both training and prediction time is dominated by the time to compute the kernel values between a data point and all landmark points. With m landmark points, assume each kernel evaluation takes $\Theta(d)$ time, then $\Theta(md)$ time is needed to compute m kernel values. There has been substantial work in the literature (Drineas & Mahoney, 2005; Zhang et al., 2008) towards improving the approximation for a fixed number of landmark points m by exploiting different landmark points selection methods.

However, the **computational complexity for both training and testing phases** has not been taken into consideration when forming landmark points. For a fixed m landmark points, popular Nyström methods need $\Theta(nmd)$ to form the kernel values when forming the low-rank approximation and $\Theta(md)$ for prediction on one testing point. In this paper, we propose the use of fast transforms, e.g., fast Haar and Hadamard transforms, to generate structured landmark points. Based on these new structured landmark points, the kernel values can be quickly evaluated, which reduces the time complexity of forming the Nyström approximation and speeds up the prediction. More specially, to test a new sample, for m landmark points, traditional Nyström methods take $\Theta(md)$ time to make a prediction and due to the benefit of fast transforms, the time complexity can be reduced to $\Theta(m \log d)$ or $\Theta(m)$ via our method. Although fast transforms have been widely used in machine

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

learning community, e.g., speed up random projection in (Le et al., 2013) and (Gittens & Mahoney, 2013b); their benefit for constructing structured landmark points to improve Nyström methods has never been investigated.

In this paper, we make the following contributions:

- We propose a family of fast transform based Nyström approximation algorithms (Fast-Nys). By enforcing the structure of the landmark points, with a fixed time budget, our proposed algorithms can compute kernel approximation with many more landmark points compared to traditional Nyström approximation.
- We cast the landmark points learning problem into an optimization problem with structural constraints, which can be solved efficiently in an alternating minimization scheme.
- We compare our proposed algorithm with state-of-the-art kernel approximation and fast prediction algorithms on real datasets and show that our proposed algorithm can achieve lower approximation error and higher prediction accuracy under a fixed time budget. When combined with the divide-and-conquer framework, our proposed algorithm performs better than the best known fast kernel SVM prediction algorithm.

In Section 2, we review state-of-the-art kernel approximation and fast prediction methods. In Section 3, we explain the Nyström method and show how to apply it to speed up the training and prediction of kernel machines. The main algorithm is described in Section 4. Section 5 shows the experimental results. We conclude our paper in Section 6.

2. Related Work

To speed up kernel methods, a family of widely-used techniques is to approximate the kernel matrix. Among them, one popular way is to generate random features from the kernel function to approximate kernel matrix. Random kitchen sinks (RKS) (Rahimi & Recht, 2007; 2008; Dai et al., 2014) approximate the shift-invariant kernel based on its Fourier transform. (Yang et al., 2014) considers using Quasi-Monte Carlo to generate features. Other than shift-invariant kernels, (Kar & Karnick, 2012) considers constructing random features for inner product kernels and (Pennington et al., 2015) generalizes random features for polynomial kernel for data on the unit sphere. Fastfood (Le et al., 2013) applies the Hadamard transform to speed up RKS. Several work focuses on exploiting the structure of the kernel matrix, e.g., (Hsieh et al., 2014b) considers the block structure of the kernel matrix and (Si et al., 2014) considers both low-rank and block structure of the shift-invariant kernel matrix to reduce the memory usage.

Other than random feature and block structure based methods, Nyström approximation (Williams & Seeger, 2001) is an efficient way to generate the low-rank representation to approximate the kernel matrix. The basic idea for Nyström approximation is to select some data points as landmark points and use the kernel values between these landmark points and all the data points to generate the low-rank approximation. There are many landmark points selection algorithms proposed in the literature. Standard Nyström (Williams & Seeger, 2001) uniformly samples landmark points from the dataset. More sophisticated landmark points selection algorithms have been proposed, including kmeans sampling (Zhang et al., 2008), sampling based on norm of corresponding kernel columns (Drineas & Mahoney, 2005), ensemble Nyström (Kumar et al., 2009), entropy-based selection (Brabanter et al., 2010), subspace distance (Lim et al., 2015) and leverage score based sampling (Gittens & Mahoney, 2013a;b). (Gittens & Mahoney, 2013b) further proposed to use Hadamard transform to speedup the leverage score computation. Different from previous work, we propose a family of Fast Transform Landmark points Selection algorithms that focuses on both "approximation error" and "computational time" through learned structured landmark points.

Prediction time for kernel methods. Kernel methods typically need $\Theta(nd)$ prediction time for each test sample, which could be expensive when the size of training set is large. Therefore, speeding up the prediction have become a popular research direction. For example, (Jose et al., 2013; Choromanska & Langford, 2015) proposed to speed up prediction time using tree-based approaches. Besides using trees, Nyström approximation can also significantly speed up the prediction. (Hsieh et al., 2014a) constructs pseudo landmark points in Nyström approximation and combines with divide-and-conquer strategy to achieve fast prediction. Our proposed algorithms utilize the property of fast transforms to speed up the kernel values evaluation, and thus can be applied to speed up prediction time. With m landmark points, the prediction time for our approach is $\Theta(m \log d)$ or $\Theta(m)$ while traditional Nyström methods require $\Theta(md)$.

3. Background

In this section, we will present the training and prediction time complexity when applying Nyström approximation to kernel machines. Given a set of instance-label pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$ for classification and $y_i \in \mathbb{R}$ for regression, we use $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to denote the kernel function. The kernel matrix is given by $G \in \mathbb{R}^{n \times n}$, where $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. One famous kernel machine is kernel SVM:

$$\boldsymbol{\alpha}^* \leftarrow \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \boldsymbol{e}^T \boldsymbol{\alpha} \quad \text{s.t. } 0 \le \boldsymbol{\alpha} \le B,$$

where $\boldsymbol{\alpha}^*$ is the dual optimal solution of the kernel machine; Q is an $n \times n$ matrix with $Q_{ij} = y_i y_j G_{ij}$; B is the

balancing parameter between loss and regularization. The prediction for a testing sample $\boldsymbol{x}$ needs to compute the decision value: $\sum_{i=1}^{n} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x})$. Due to the $\Theta(n^2 d)$ time and $\Theta(n^2)$ space complexity of forming and storing kernel matrix, kernel machines are slow in both training and prediction phases.

To scale up kernel machines, a promising way is to form the low-rank approximation of G by Nyström approximation. Given $m \ll n$ landmark points $\{\boldsymbol{u}_j\}_{j=1}^{m}$, Nyström methods (Williams & Seeger, 2001) form the following low-rank approximation for the kernel matrix:

$$G \approx \bar{G} := C W^{\dagger} C^T, \tag{1}$$

where $C \in \mathbb{R}^{n \times m}$ contains kernel values between all the data points and the landmark points ($C_{ij} = K(\boldsymbol{x}_i, \boldsymbol{u}_j)$), and $W \in \mathbb{R}^{m \times m}$ contains kernel values between landmark points ($W_{ij} = K(\boldsymbol{u}_i, \boldsymbol{u}_j)$), and $W^{\dagger}$ is the pseudo-inverse of W. The time complexity for traditional Nyström is $\Theta(nmd + m^3)$, where $\Theta(nmd)$ is used for forming the kernel value matrix C, and $\Theta(m^3)$ is used for forming $W^{\dagger}$. We can see that in Nyström methods computing C is the bottleneck for large-scale applications.

After forming the Nyström approximation, the training speed can be sped up since the matrix-vector product $\bar{G}\boldsymbol{v}$ can be computed efficiently. More interestingly, Nyström approximation is also a promising way to speed up the testing phase. For a testing sample $\boldsymbol{x}$, with Nyström approximation the decision value can be computed by

$$\bar{\boldsymbol{x}}(W^{\dagger} C^T \boldsymbol{\alpha}) = \bar{\boldsymbol{x}}\boldsymbol{\beta}, \tag{2}$$

where $\boldsymbol{\alpha}$ is the kernel SVM (or kernel ridge regression) model, and $\bar{\boldsymbol{x}} = [K(\boldsymbol{x}, \boldsymbol{u}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{u}_m)]$. $\boldsymbol{\beta}$ is independent of $\boldsymbol{x}$, so can be precomputed and stored. Therefore, the time complexity for predicting a testing sample is equal to the time complexity for computing $\bar{\boldsymbol{x}}$, which requires $\Theta(md)$ time/flops for computing the m kernel values. It has been shown in many previous papers (Hsieh et al., 2014a) that Nyström approximation is an efficient way to improve the prediction time, and can achieve state-of-the-art performance if used in conjunction with divide-and-conquer SVM or Memory Efficient Kernel Approximation frameworks (Hsieh et al., 2014b; Si et al., 2014).

However, existing landmark points selection approaches aim to minimize the "kernel approximation error", and none of them brought the "computational time" into the picture when forming the Nyström approximation. In this work, we propose a family of Fast Transforms Landmark points Selection algorithms (Fast-Nys) that considers both "approximation error" and "computational time". As a result, we are able to improve the Nyström approximation in both training and prediction phases.

4. Fast Transform landmark points selection (Fast-Nys)

We consider a family of kernel functions that have the following form:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(\boldsymbol{x}_i) f(\boldsymbol{x}_j) g(\boldsymbol{x}_i^T \boldsymbol{x}_j), \tag{3}$$

where $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ and $g(\cdot) : \mathbb{R} \to \mathbb{R}$. Some widely used kernel functions belong to this family: in Gaussian kernel, $f(\boldsymbol{x}) = e^{-\gamma \|\boldsymbol{x}\|^2}, g(z) = e^{2\gamma z}$, in polynomial kernel $f(\boldsymbol{x}) = 1, g(z) = (c + z)^p$, and in homogeneous kernel $f(\boldsymbol{x}) = 1, g(z) = z^p$.

Given a set of m landmark points $U \in \mathbb{R}^{m \times d}$ where each row of U is a landmark point $\boldsymbol{u}_j$, in the training phase the main computation of forming Nyström approximation (1) is to compute $C \in \mathbb{R}^{n \times m}$ where $C_{ij} = K(\boldsymbol{x}_i, \boldsymbol{u}_j)$. The computational time for each row of C can be written as

$$T_f + T_g \times m + T_{U\boldsymbol{x}}, \tag{4}$$

where T_f and T_g are function evaluation time for $f(\cdot)$ and $g(\cdot)$ respectively, and $T_{U\boldsymbol{x}}$ is the time complexity for computing $U\boldsymbol{x}$, i.e., the inner product between an arbitrary $\boldsymbol{x} \in \mathbb{R}^d$ and U. In the prediction phase for kernel SVM, as shown in (2), the bottleneck is to compute $\bar{\boldsymbol{x}}$, the kernel values between a test sample and all landmark points. So the prediction time complexity is the same as (4).

Now we analyze the time complexity for computing kernel values between one data point $\boldsymbol{x}$ and all landmark points (Eq. (4)). Typically the matrix-vector product $U\boldsymbol{x}$ has $\Theta(md)$ time complexity, and both T_g and T_f are very fast. Therefore, the third term (computation of $T_{U\boldsymbol{x}}$) dominates the time complexity: computing each $\bar{\boldsymbol{x}}$ requires $\Theta(md)$ time complexity. Table 1 shows the time segmentation for standard Nyström on MNIST dataset with $60,000$ samples, and $T_{U\boldsymbol{x}}$ clearly dominates the computational time.

Now we are ready to describe the main idea of this paper. *Can we compute $U\boldsymbol{x}$ using less than $\Theta(md)$ floating point operations with m landmark points?* The answer is yes if the landmark matrix U has some special structure. If we force the landmark points to collectively have some special structure, then the kernel approximation and prediction time can both be sped up. We propose two examples of the Fast Transform Landmark Points Selection methods, where for each one we force U to have a special structure to benefit the computation of Ux.

4.1. Haar Landmark Points

For the first example of our proposed framework, we make use of the fast Haar discrete wavelet transform. Haar discrete wavelet transform associates with Haar matrix H_d.

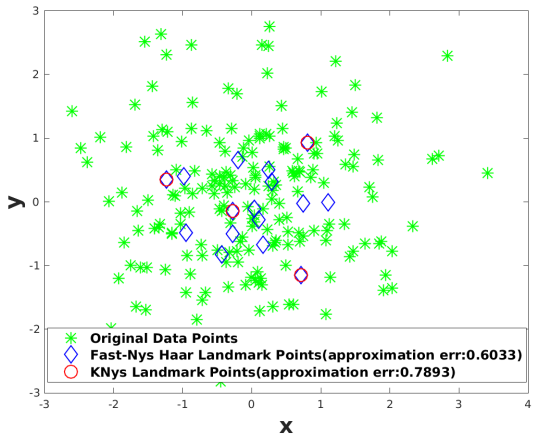


Figure 1. Comparisons of landmark points and approximation error using our proposed method (Fast-Nys with Haar landmark points) and Kmeans Nystrom (KNys). In the figure, green asterisk are the data points generated from Gaussian distribution. Red circles are the landmark points in KNys and blue diamonds are the generated Haar landmark points from Fast-Nys. Note that Fast-Nys uses fast transform to form the kernel values, so with the similar amount of time, Fast-Nys can handle more landmark points than traditional Nyström methods (here, KNys).

As an example, when $d = 4$,

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}. \quad (5)$$

Generally, the $2d \times 2d$ Haar matrix can be derived as:

$$H_{2d} = \begin{pmatrix} H_d \otimes [1, 1] \\ I_d \otimes [1, -1] \end{pmatrix}, \quad (6)$$

where I_d is $d \times d$ identity matrix; $\otimes$ is the Kronecker product. The time complexity for fast wavelet transform is $\Theta(d)$, which means $d \times d$ Haar matrix multiplies a $d \times 1$ vector can be done in $\Theta(d)$ time.

In the Haar landmark points approach, we force the $m \times d$ landmark matrix U to have the following structure:

$$U = [V_1 H_d^T,\ V_2 H_d^T, \ldots,\ V_s H_d^T]^T, \quad (7)$$

where $s = m/d$, each V_i is a d by d diagonal matrix, and H_d is the $d \times d$ Haar matrix.

Given a sample $\boldsymbol{x}$, the computation of $U\boldsymbol{x}$ can be done by

$$U\boldsymbol{x} = [(H_d V_1 \boldsymbol{x})^T,\ (H_d V_2 \boldsymbol{x})^T, \ldots,\ (H_d V_s \boldsymbol{x})^T]^T. \quad (8)$$

If we write $V_i = \text{diag}(\boldsymbol{v}_i)$ where $\boldsymbol{v}_i$ is a d-dimensional vector, called 'seed', then each $\bar{\boldsymbol{v}} = V_i \boldsymbol{x}$ can be computed by d multiplications. Also, using the properties of Haar transform, $H_d \bar{\boldsymbol{v}}$ can be computed in $\Theta(d)$ time (and only with plus and minus operations) using Haar transform. Therefore, with m landmark points, using Haar landmark points, the computation of $U\boldsymbol{x}$ can be done in $\Theta(ds) = \Theta(m)$ time, which is significantly faster than the $\Theta(md)$ time in standard Nyström. When d is not 2^l for some l, we can append zero features, which will not change the order of time complexity.

Using this structure, we have the freedom in choosing 'seeds' vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$. Note that given a set of s classical landmark points $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_s$ that can be generated by any landmark points selection approach (random sampling, kmeans clustering, leverage score based sampling, or others), we can set the seeds $\boldsymbol{v}_1 = \boldsymbol{m}_1, \ldots, \boldsymbol{v}_s = \boldsymbol{m}_s$. Since the first row of the Haar matrix is all 1s, the first row of each $H_d V_i$ will be $\boldsymbol{m}_i$. This implies that the original landmark points $\{\boldsymbol{m}_i\}_{i=1}^s$ will be a subset of the Haar landmark points $\{\boldsymbol{u}_i\}_{i=1}^m$. Also, it is not difficult to show that the new set of landmark points will decrease the kernel approximation error compared with using original Nyström landmark points (see Theorem 1 in the Appendix).

In term of complexity, the time complexity for using $\{\boldsymbol{m}_i\}_{i=1}^s$ and $\{\boldsymbol{u}_i\}_{i=1}^m$ are the same (both are $\Theta(m)$) to form one row of C in Nyström approximation. For the training time, there is small computation overhead in computing $W^\dagger$ as shown in Table 1. Furthermore, computing $W^\dagger$ can be further sped up by randomized SVD (Li et al., 2010).

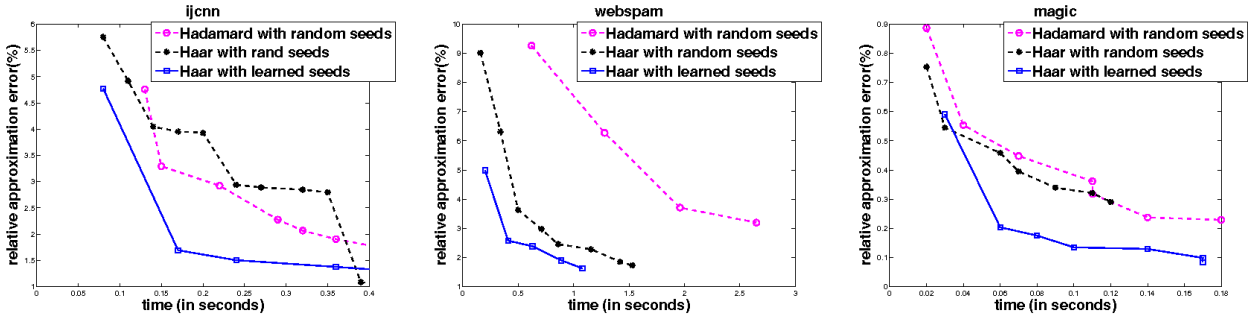
Figure 1 is a toy example comparing our proposed method with a state-of-the-art Nyström approximation method, Kmeans Nyström (KNys). As we can see in the figure, the landmark points generated from Fast-Nys cover more space than the landmark points from KNys, so it can achieve much lower approximation error. Note that although Fast-Nys uses more landmark points than KNys, the training and prediction time are similar since the structure of landmark points allow us to speed up the kernel value evaluation by fast wavelet transform.

In conclusion, the benefits of our proposed Haar landmark points are:

- For the same number of landmark points, using Haar landmark points will speed up training and prediction.
- Using the same $\Theta(m)$ time, Haar landmark points allow us to generate more landmark points, and is guaranteed to achieve better performance than the original landmark points.

4.2. Hadamard Landmark Points

Instead of using the Haar matrix in (7), we can also use other fast transforms such as Hadamard transform, which



(a) ijcnn. (b) webspam. (c) magic.

Figure 2. Comparisons of Haar landmark points and Hadamard landmark points for Gaussian kernel approximation on three real datasets. We also compare random seeds with learning seeds with structural constraint (Haar with structural constraint) in Nyström approximation. x-axis shows the time (in seconds), and y-axis shows the relative kernel approximation error. Since Haar landmark points perform better than Hadamard landmark points in (12), we do not show the result of Hadamard with structural constraint in these plots.

associates with the Walsh-Hadamard matrix defined as

$$\bar{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } \bar{H}_{2d} = \begin{bmatrix} \bar{H}_d & \bar{H}_d \\ \bar{H}_d & -\bar{H}_d \end{bmatrix}.$$

By exploiting the structure, the fast Hadamard transform allows the computation of matrix-vector product $\bar{H}_d \boldsymbol{v}$ to be done in $\Theta(d \log d)$ time. We define the Hadamard landmark points to have the following structure:

$$U = [V_1 \bar{H}_d^T,\ V_2 \bar{H}_d^T, \ldots,\ V_s \bar{H}_d^T]^T. \tag{9}$$

Using a similar derivation with the previous subsection, we can see that with m landmark points, the matrix-vector product $U\boldsymbol{x}$ can be done in $\Theta(m \log d)$ time if landmark points matrix U follows the form of (9).

Similar to the Haar landmark points case, given a set of original landmark points $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_s\}$, we can set $\boldsymbol{v}_i = \boldsymbol{m}_i$ for all $i = 1, \ldots, s$, the resulting kernel approximation has lower approximation error than just using the original s landmark points, with little computational overhead. With the same number of landmark points, approximation using Haar landmark points usually has better time complexity but worse approximation quality compared with using Hadamard landmark points. A comparison between Hadamard landmark points and Haar landmark points is shown in Figure 2. We can see that Haar landmark points in most cases achieves lower approximation error than Hadamard landmark points using the same amount of time.

4.3. Searching for Seeds

In this section, we propose another way to learn the seeds $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_s$ in our algorithm by minimizing the upper bound of kernel approximation error. We consider the upper bound derived in (Zhang et al., 2008):

Proposition 1 (Zhang et al., 2008)**.** *If the kernel function* $K(\cdot, \cdot)$ *satisfies*

$$K(\boldsymbol{a}, \boldsymbol{b})^2 - K(\boldsymbol{c}, \boldsymbol{d})^2 \leq \eta(\|\boldsymbol{a} - \boldsymbol{c}\|^2 + \|\boldsymbol{b} - \boldsymbol{d}\|^2) \tag{10}$$

for some constant η*, then the error of Nyström approximation is bounded by* $\|\bar{G} - G\|_F \leq C_1\sqrt{e} + C_2 e$*, where*

$$e = \sum_{i=1}^{n} \left(\min_j \|\boldsymbol{x}_i - \boldsymbol{u}_j\|^2 \right). \tag{11}$$

It is easy to show that the kernel function considered in our paper (3) satisfies condition (10) (see Lemma 1 in the Appendix). This motivates us to find the optimal seeds to minimize the error e.

Optimal Seeds Selection by Minimizing the Error. In our proposed algorithms, the fast transform landmark points (7) and (9) are parameterized by the seed vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$, so here the goal is to find the best seed vectors to minimize the upper bound of the approximation error, which leads to the following optimization problem:

$$\underset{\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s\}, \boldsymbol{t}}{\operatorname{argmin}} \quad \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{u}_{t_i}\|^2 \tag{12}$$

$$\text{s.t.} \quad U = [V_1 H_d^T, V_2 H_d^T, \ldots, V_s H_d^T]^T, \tag{13}$$

where $\boldsymbol{u}_i$ is the i-th row of U, H can be Haar or Hadamard matrices, and $\boldsymbol{t}$ is the indicator vector.

In the following we propose an alternating minimization scheme to solve the above optimization problem. We iteratively update the indicator vector $\boldsymbol{t}$ and the seed vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$. A naive algorithm will have $\Theta(nmd)$ time complexity because there are n points, m landmark points in the d-dimensional space. However, we show that by using Haar/Hadamard transforms our algorithm only requires $\Theta(md + nd)$ or $\Theta(md + nd \log d)$ time.

When $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s\}$ are fixed, the cluster indicator can be computed by

$$t_i = \underset{q=1,\ldots,m}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{u}_q\|^2 = \underset{q=1,\ldots,m}{\operatorname{argmin}} \|\boldsymbol{u}_q\|^2 - 2\boldsymbol{x}_i^T \boldsymbol{u}_q.$$

$\|\boldsymbol{u}_q\|$ is independent of n so can be computed efficiently, and the only bottleneck is to compute $U\boldsymbol{x}_i$. Luckily, due to

number of landmark points	methods	$T_{U\boldsymbol{x}}$	T_g	pseudo-inverse of W	Learning seed	Rest	Approx. Error	Total time
$m = 40$	Nys	2.79	0.17	0.01	0	0.06	0.0063	3.03
	Fast-Nys	0.77	0.17	0.01	0.33	0.07	0.0068	1.35
$m = 80$	Nys	5.38	0.18	0.01	0	0.08	0.0033	5.65
	Fast-Nys	1.36	0.18	0.01	0.50	0.06	0.0038	2.11
$m = 160$	Nys	10.81	0.20	0.02	0	0.12	0.0014	11.15
	Fast-Nys	2.65	0.20	0.02	0.76	0.08	0.0020	3.71

Table 1. Segment the time (in seconds) for each step of standard Nyström (Nys) and Fast-Nys with Haar landmark points for the same number of landmark points m, run on MNIST dataset with 60,000 samples. Note that 'Rest' includes the time for computing T_f. We can see that computing $U\boldsymbol{x}$ dominate the whole procedure of Fast-Nys and Fast-Nys can significantly speed up the computation and achieve similar accuracy. Another thing we want to mention is that under the similar amount of time $m = 40$ for Nys (3.03 seconds) and $m = 160$ for Fast-Nys (3.71 seconds), Fast-Nys achieves much lower approximation error (0.0068 vs.0.0020).

Table 2. Comparisons of Fast-Nys and the standard Nyström approximation under m landmark points. $n \gg m$ and $n \gg d$. T_{seed} is the time for learning the seeds that is $\Theta(nd + md)$ for Haar landmark points, and $\Theta(nd \log d + md)$ for Hadamard landmark points.

	Time to form Kernel Approximation	Prediction Time in kernel methods
Standard Nyström	$\Theta(nmd + m^3)$	$\Theta(md)$
Fast-Nys with Haar landmark points	$\Theta(nm + m^3) + T_{seed}$	$\Theta(m)$
Fast-Nys with Hadamard landmark points	$\Theta(nm \log d + m^3) + T_{seed}$	$\Theta(m \log d)$

the structure of U in (13), the matrix-vector product can be computed efficiently, so the time complexity for this step is

$$nT_{U\boldsymbol{x}} + md.$$

When $\boldsymbol{t}$ is fixed, the update of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ can be separated into several subproblems. Note that there are s groups in the structure of (13) and each group has d landmark points. Assume each t_i (indicator of $\boldsymbol{x}_i$) is in group $g(i)$ and offset $q(i)$, then we can write (12) as

$$\underset{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_s}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{d} ((\boldsymbol{x}_i)_j - H_{q(i),j}(\boldsymbol{v}_{g(i)})_j)^2. \qquad (14)$$

The objective function of (14) is just a sum of m one-variable quadratic function, which can be solved in closed form with time complexity of $\Theta(nd)$.

In summary, running the above alternating minimization for a fixed number of iterations takes $\Theta(nd + md)$ for Haar landmark points, and $\Theta(nd \log d + md)$ for Hadamard landmark points. This alternating minimization scheme can scale to large problems because the time complexity is linear to n. In practice, the algorithm usually converges to a reasonably good solution in 10 iterations, so we fix the number of iterations to be 10 for all the experiments. In Figure 2, we compare learning seeds with the one without learning in Fast-Nys, where the initial seeds are selected randomly. Clearly the former achieves lower kernel approximation error.

Note that although Kmeans Nyström proposed in (Zhang et al., 2008; Zhang & Kwok, 2010) also used Proposition 1 to find landmark points, in our algorithm we are learning the "seeds" for fast transforms, so we cannot directly use kmeans. We further reduce the time complexity by exploiting the structure of landmark points.

Algorithm 1 Fast Transforms for Nyström Approximation

Input : Data points $\{(\boldsymbol{x}_i)\}_{i=1}^n$, number of initial seeds s, the number of landmark points $m = sd$

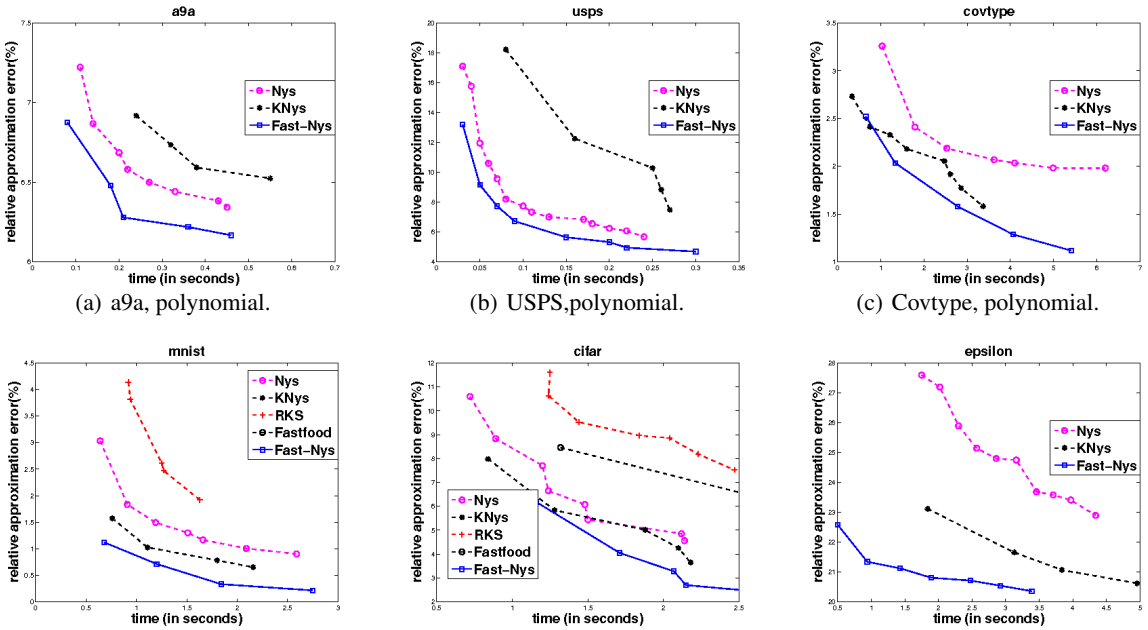
Output: The rank-m approximation $CW^\dagger C^T$

1. Select seed vectors $\{\boldsymbol{v}_i\}_{i=1}^s$ using Section 4.1 or 4.3.
2. Form U by (7) where H_d can be Haar matrix or Hadamard matrix
3. Form the C, W using fast matrix-vector product with Haar or Hadamard transform as shown in (8)
4. The rank-m approximation $G \approx CW^\dagger C^T$

4.4. Summary of the Proposed Algorithm

Our method is described in Algorithm 1. Note that (1) Similar to other Nyström based methods (Li et al., 2010), when m is large, randomized SVD can be applied to speed up computing pseudo inverse of the $m \times m$ intersection matrix W; (2) To further improve the speed, in the experiments, we randomly sample 2000 data points to learn the seeds; (3) Number of landmark points does not have to be fixed as sd. We can truncate the fast transforms to get different numbers of landmark points.

The time complexity is summarized in Table 2. Since we can use the fast discrete wavelet (Fourier) transform without explicitly forming the matrix, Fast-Nys is also memory efficient. In Table 1, we segment the time for Fast-Nys and compare it with standard Nyström with the same amount of landmark points. We can see that with the same amount



(a) a9a, polynomial. (b) USPS,polynomial. (c) Covtype, polynomial.

(d) MNIST, Gaussian. (e) CIFAR, Gaussian. (f) Epsilon, homogeneous.

Figure 3. Low-rank kernel approximation results. x-axis is the time and y axis shows the relative kernel approximation error. Methods with approximation error above the top of y-axis are not shown. Note that (1) the time for learning the landmark points is included in the running time. (2) two random feature based methods, RKS and Fastfood can not directly apply for homogeneous and polynomial kernel. So we only show their kernel approximation results on Gaussian kernel in (d) and (e). More comparison between Fast-Nys with other Nyström based methods can be found in Figure 5 in the Appendix.

Table 3. Data set statistics (n: number of samples; d: dimension of samples).

Dataset	n	d	Dataset	n	d
USPS	9298	256	Covtype	581,012	54
a9a	48,842	123	MNIST	60,000	784
Letter	18,000	16	CIFAR	60,000	400
Epsilon	25,000	2,000	webspam	350,000	254

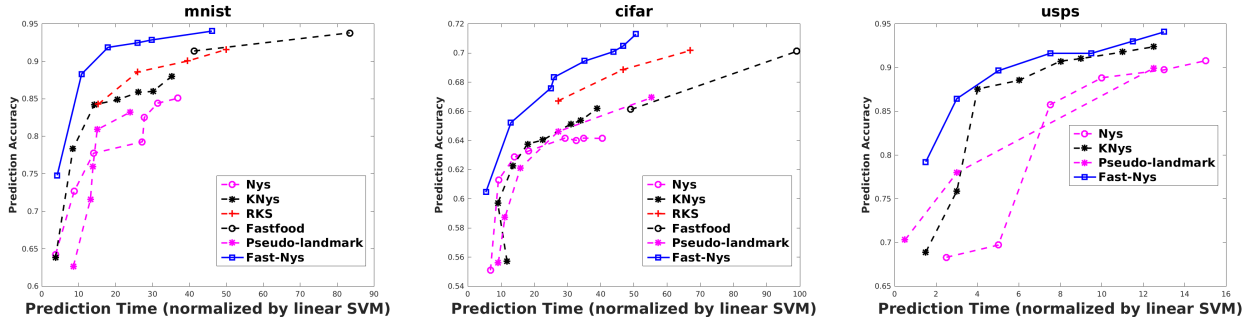
of landmark points, (1) time for computing $U\boldsymbol{x}$, i.e., inner product between data points and landmark points, dominates the whole procedure; (2) learning seeds takes a small portion of the overall time; (3) Fast-Nys is faster than standard Nyström method, while achieving similar accuracy.

5. Experimental Results

In this section, we empirically demonstrate the benefits of our proposed method (Fast-Nys) on eight data sets listed in Table 3. We consider two different tasks and compare their computational time: time for forming the kernel approximation and prediction time for kernel SVM. We test three types of widely used kernels: Gaussian kernel ($k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$, polynomial kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^p$, and homogeneous kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^p$. The degree p is set to be 3 in the experiment. For comparison, we compare our method with seven state-of-the-art methods for kernel approximation or kernel SVM prediction:

1. The standard Nyström method (Nys), where the landmark points are sampled uniformly at random from the dataset (Williams & Seeger, 2001).
2. Kmeans Nyström (KNys), where the landmark points are the kmeans centroids (Zhang & Kwok, 2010).
3. Pseudo Landmark points (Pseudo), where pseudo-landmark points are generated from the sampled columns of kernel matrix to improve the Nyström approximation (Hsieh et al., 2014a).
4. Random Kitchen Sinks (RKS), which approximates the shift-invariant kernel based on its Fourier transform (Rahimi & Recht, 2008).
5. Fastfood with "Hadamard features" (Fastfood), which uses the Hadamard transform to speed up matrix multiplication (Le et al., 2013).
6. The Local Deep Kernel Learning method (LDKL), which learns a tree-based primal feature embedding to achieve faster prediction speed (Jose et al., 2013).
7. Divide-and-Conquer based fast Prediction (DC-Pred++): pseudo landmark points with divide-and-conquer strategy for fast prediction (Hsieh et al., 2014a).

The first five methods are kernel approximation methods, which are also used for speeding up the prediction in kernel SVM, and the last two (LDKL and DC-Pred++) are two state-of-the-art algorithms specifically designed for fast prediction. The setting and parameters used in the experiments are shown in the Appendix.



(a) MNIST, Gaussian (b) CIFAR, Gaussian (c) USPS, polynomial

Figure 4. Fast prediction results. Methods with prediction accuracy below the bottom of y-axis are not shown. x-axis shows the prediction time which is normalized by the linear SVM prediction time, and the y-axis shows the kernel SVM prediction accuracy. Since RKS and Fastfood can not directly apply for approximating polynomial kernel, we show their performance on Gaussian kernel in (a) and (b)

Table 4. Comparison of kernel SVM prediction on four real-world datasets. Note that the actual prediction time is normalized by the linear SVM prediction time. For example, 10x means the actual prediction time $= 10\times$ (time for linear SVM prediction time). Some prediction measurement is used in (Hsieh et al., 2014a).

Dataset	Metric	DC-Fast-Nys	DC-Pred++	LDKL	KNys	RKS	Fastfood
Letter	Prediction Time	**7.6x**	11.71x	12.81x	140x	61x	50x
	Accuracy	**95.4%**	95.14%	95.36%	87.58%	89.93%	89.9%
USPS	Prediction Time	**7.31x**	14.8x	7.50x	200x	72.5x	80x
	Accuracy	94.9%	94.82%	**95.71%**	92.56%	91.33%	94.39%
webspam	Prediction Time	**11.21x**	20.5x	23x	200x	34.5x	80x
	Accuracy	98.0%	**98.4%**	95.15%	95.01%	96.4%	96.7%
a9a	Prediction Time	**7.35x**	7.93x	7.91x	50x	15x	80x
	Accuracy	**84.70%**	83.90%	84.54%	83.9%	84.32%	61.9%

5.1. Kernel Approximation

The kernel approximation results are shown in Figure 3. We use relative kernel approximation error $\|G - \tilde{G}\|_F / \|G\|_F$ to measure the quality, and test on three types of kernels. In these plots, we vary the number of landmark points m for different Nyström methods and each marker represents the result corresponds to one choice of m. We can observe from Figure 3 that, with the same amount of time, our proposed method significantly and consistently yields lower approximation error than other Nyström based methods for various kernels. This means that our proposed method can achieve faster training speed for speeding up kernel machines. Fast-Nys also shows better performance than leverage score based (Gittens & Mahoney, 2013a) and entropy based (Brabanter et al., 2010) landmark points in Nyström approximation in Figure 5 in the Appendix. Since our method is a basic Nyström method, it can combine with ensemble Nyström and MEKA (Si et al., 2014) to achieve lower approximation error.

5.2. Fast Prediction for Kernel SVM

Next we show the benefit of using our method to improve the kernel SVM prediction time. First, in Figure 4 we show the comparison between our proposed algorithm and other kernel approximation methods for kernel SVM prediction. Since Fastfood and RKS can only be used for shift-invariant kernel, we do not compare with them on USPS. As can be seen in Figure 4, with the same prediction time, Fast-Nys always achieves higher accuracy. For example, on MNIST, our proposed method takes around 40x linear SVM's prediction time to achieve accuracy of 95%, while the second best algorithm takes more than 80x linear SVM's prediction time to achieve similar accuracy.

Furthermore, we can even boost the performance of our proposed algorithm with divide-and-conquer-framework (DC-Fast-Nys). The divide-and-conquer strategy has been shown to be useful in speeding up the training and prediction in kernel methods (Si et al., 2014; Hsieh et al., 2014a;b). The performance of DC-Fast-Nys are shown in Table 4. All the results are based on the Gaussian kernel. We can see that DC-Fast-Nys is faster than other methods, especially DC-Pred++ and LDKL which are two state-of-the-art fast prediction algorithms.

6. Conclusion

In this paper, we propose a family of fast transform based Nyström approximation algorithms, Fast-Nys, which constructs easy-to-compute landmarks to reduce the kernel value evaluation time between these landmark points and the given data points. Using the same amount of computational time, Fast-Nys can deal with more landmark points compared with all the previous Nyström methods, and this lead to fast training and prediction in kernel machines.

Acknowledgements This research was supported by NSF grants CCF-1320746 and IIS-1546459.

References

Brabanter, K. De, Brabanter, J. De, Suykens, J.A.K., and Moor, B. De. Optimized fixed-size kernel models for large data sets. *Computational Statistics & Data Analysis*, 54(6):1484–1504, 2010.

Choromanska, Anna and Langford, John. Logarithmic time online multiclass prediction. In *NIPS*, 2015.

Dai, Bo, Xie, Bo, He, Niao, Liang, Yingyu, Raj, Anant, florina F. Balcan, Maria, and Song, Le. Scalable kernel methods via doubly stochastic gradients. In *NIPS*. 2014.

Drineas, Petros and Mahoney, Michael W. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 6:2153–2175, 2005.

Gittens, Alex and Mahoney, Michael. Revisiting the Nyström method for improved large-scale machine learning. In *ICML*, 2013a.

Gittens, Alex and Mahoney, Michael W. Revisiting the Nyström method for improved large-scale machine learning. *CoRR*, abs/1303.1849, 2013b.

Halko, N., Martinsson, P. G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

Hsieh, C.-J., Si, S., and Dhillon, I. S. Fast prediction for large-scale kernel machines. In *NIPS*, 2014a.

Hsieh, C.-J., Si, S., and Dhillon, I. S. A divide-and-conquer solver for kernel support vector machines. In *ICML*, 2014b.

Jose, C., Goyal, P., Aggrwal, P., and Varma, M. Local deep kernel learning for efficient non-linear SVM prediction. In *ICML*, 2013.

Kar, P. and Karnick, H. Random feature maps for dot product kernels. In *AISTATS*, 2012.

Kumar, Sanjiv, Mohri, Mehryar, and Talwalkar, Ameet. Ensemble Nyström methods. In *NIPS*, 2009.

Le, Q. V., Sarlos, T., and Smola, A. J. Fastfood – approximating kernel expansions in loglinear time. In *ICML*, 2013.

Li, Mu, Kwok, James T., and Lu, Bao-Liang. Making large-scale Nyström approximation possible. In *ICML*, 2010.

Lim, Woosang, Kim, Minhwan, Park, Haesun, and Jung, Kyomin. Double Nyström method: An efficient and accurate Nyström scheme for large-scale data sets. In *ICML*, 2015.

Pennington, Jeffrey, Yu, Felix X., and Kumar, Sanjiv. Spherical random features for polynomial kernels. In *NIPS*, 2015.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, 2008.

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

Rudi, Alessandro, Camoriano, Raffaello, and Rosasco, Lorenzo. Less is more: Nyström computational regularization. In *NIPS*, 2015.

Si, S., Hsieh, C.-J., and Dhillon, I. S. Memory efficient kernel approximation. In *ICML*, 2014.

Williams, Christopher K. I. and Seeger, Matthias. Using the Nyström method to speed up kernel machines. In *NIPS*, 2001.

Yang, J., Sindhwani, V., Avron, H., and Mahoney, M. W. Quasi-Monte Carlo Feature Maps for Shift-Invariant Kernels. In *ICML*, 2014.

Zhang, K., Tsang, I. W., and Kwok, J. T. Improved Nyström low rank approximation and error analysis. In *ICML*, 2008.

Zhang, Kai and Kwok, James T. Clustered Nyström method for large scale manifold learning and dimension reduction. *Trans. Neur. Netw.*, 21(10):1576–1587, 2010.

7. Appendix

7.1. Proof of lower approximation error

Theorem 1. *Using Algorithm 1 to generate m landmark points, we can guarantee that the approximation quality will become better than the traditional Nyström approximation with initial s landmark points:*

$$\|G - \bar{G}\| \leq \|G - \tilde{G}\|, \tag{15}$$

where $\tilde{G}$ and $\bar{G}$ are the approximation of G from standard Nyström and Algorithm 1 respectively.

Proof. Let us first compare our method with standard Nyström. The generalization to other sampling strategies based Nyström is straight forward. Let G denote the kernel matrix form on the n data points, and suppose s landmark points $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_s$ are selected uniformly at random from the data. Let us define the sampling matrix $S \in R^{n\times s}$ to be a zero-one matrix where $S_{ij} = 1$ if i-th sample in the dataset is selected as landmark point. C is a $n \times s$ matrix consisting of the corresponding s columns selected from G and W consists of the kernel matrix formed by these s landmark points. So by standard Nyström, $\tilde{G} = CW^+G^T$, $C = GS$ and $W = S^TGS$.

Using $\boldsymbol{m}_1, \cdots, \boldsymbol{m}_s$ as initial landmark points in Algorithm 1, after fast transforms, we totally have $m = sd$ landmark points $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$, of which the last s points are the original landmark points and the rest $m - s$ are new landmark points. Assume the new kernel matrix G_H is the kernel matrix on the union of the original n data points and $m - s$ new added landmark points. So G is a block in G_H. Similarly we define S_H, C_H, and W_H as sampling matrix, m sampled columns in G_H and kernel matrix formed by m landmark points respectively. So $C_H = G_HS_H$ and $W_H = S_H^TG_HS_H$. Let the decomposition of G_H be $G_H = L_H^TL_H$. So

$$G_H = L_H^TL_H = \begin{bmatrix} \bar{L}^T \\ L^T \end{bmatrix}\begin{bmatrix} \bar{L} & L \end{bmatrix} = \begin{bmatrix} \bar{L}^T\bar{L} & \bar{L}^TL \\ L^T\bar{L} & L^TL \end{bmatrix}. \tag{16}$$

Since G is a block in G_H, the decomposition of G is L^TL.

Since $C_H = G_HS_H = L_H^TL_HS_H$ and let the singular value decomposition of L_HS_H be $U_H\Sigma_HV_H^T$, $C_H = L_H^TU_H\Sigma_HV_H^T$. Also we have

$$W_H = S_H^TG_HS_H \quad = S_H^TL_H^TL_HS_H = V_H\Sigma_H^2V_H^T. \tag{17}$$

The Nyström approximation on G_H is written as

$$\begin{aligned} G_H &= C_HW_H^+C_H^T \\ &= L_H^TU_H\Sigma_HV_H^TV_H\Sigma_H^{-2}V_H^TV_H\Sigma_HU_H^TL_H \\ &= L_H^TU_HU_H^TL_H. \end{aligned} \tag{18}$$

So we have

$$\begin{aligned} G_H - C_HW_H^+C_H^T &= L_H^TL_H - L_H^TU_HU_H^TL_H \\ &= (L_H - U_HU_H^TL_H)^T(L_H - U_HU_H^TL_H). \end{aligned} \tag{19}$$

The Nyström approximation error on the original n data points or G part is

$$\begin{aligned} (G_H - C_HW_H^+C_H^T)_G &= L^TL - L^TU_HU_H^TL \\ &= (L - U_HU_H^TL)^T(L - U_HU_H^TL). \end{aligned} \tag{20}$$

According to Lemma 1 in (Drineas & Mahoney, 2005), we have the standard Nyström approximation on G as

$$\begin{aligned} G - CW^+C^T &= L^TL - L^TUU^TL \\ &= (L - UU^TL)^T(L - UU^TL). \end{aligned} \tag{21}$$

where LS's SVD is $U\Sigma V^T$.

Since U is the basis for the range space of LS and U_H is the basis for the range space of L_HS_H, so $range(U) \subseteq range(U_H)$. According to the proposition 8.5 in (Halko et al., 2011), we have

$$\|L - U_HU_H^TL\|_2 \leq \|L - UU^TL\|_2, \tag{22}$$

so

$$\|(G_H - C_HW_H^+C_H^T)_G\| \leq \|G - CW^+C^T\|, \tag{23}$$

or

$$\|G - \bar{G}\| \leq \|G - \tilde{G}\|. \tag{24}$$

□

7.2. Lemma 1

Lemma 1. *If the kernel function can be written as* (3)*, assume the maximum distance between the samples and the original point is a bounded number R, and f, g are differentiable, then*

$$K(\boldsymbol{a}, \boldsymbol{b})^2 - K(\boldsymbol{c}, \boldsymbol{d})^2 \leq \eta(\|\boldsymbol{a} - \boldsymbol{c}\|^2 + \|\boldsymbol{b} - \boldsymbol{d}\|^2) \tag{25}$$

for any $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, where

$$\eta = 4M_f^4L_g^2R^2 + 4M_f^2M_g^2L_f^2,$$

where $M_f = \max_{\|\boldsymbol{x}\|\leq R} |f(\boldsymbol{x})|$, $M_g = \max_{\|\boldsymbol{u}\|\leq R} |g(\boldsymbol{u})|$, $L_f = \max_{\|\boldsymbol{x}\|\leq R} |f'(\boldsymbol{x})|$, $L_g = \max_{\|\boldsymbol{u}\|\leq R} |g'(\boldsymbol{u})|$.

Proof. For any $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, we have

$$
\begin{aligned}
&(K(\boldsymbol{a},\boldsymbol{b}) - K(\boldsymbol{c},\boldsymbol{d}))^2 \\
=&\bigg(f(\boldsymbol{a})f(\boldsymbol{b})g(\boldsymbol{a}^T\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d})g(\boldsymbol{c}^T\boldsymbol{d}) \bigg)^2 \\
=&\bigg((f(\boldsymbol{a})f(\boldsymbol{b})g(\boldsymbol{a}^T\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d})g(\boldsymbol{a}^T\boldsymbol{b})) \\
&+ (f(\boldsymbol{c})f(\boldsymbol{d})g(\boldsymbol{a}^T\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d})g(\boldsymbol{c}^T\boldsymbol{d})) \bigg)^2 \\
\leq& 2\bigg(g(\boldsymbol{a}^T\boldsymbol{b})(f(\boldsymbol{a})f(\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d})) \bigg)^2 \\
&+ 2\bigg(f(\boldsymbol{c})f(\boldsymbol{d})(g(\boldsymbol{a}^T\boldsymbol{b}) - g(\boldsymbol{c}^T\boldsymbol{d})) \bigg)^2 \\
\leq& 2M_g^2\bigg(f(\boldsymbol{a})f(\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d}) \bigg)^2 \\
&+ 2M_f^4\bigg(g(\boldsymbol{a}^T\boldsymbol{b}) - g(\boldsymbol{c}^T\boldsymbol{d}) \bigg)^2.
\end{aligned}
$$

We can then bound each term by

$$
\begin{aligned}
&\bigg(f(\boldsymbol{a})f(\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d}) \bigg)^2 \\
\leq& \bigg(f(\boldsymbol{a})f(\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{b}) + f(\boldsymbol{c})f(\boldsymbol{b}) - f(\boldsymbol{c})f(\boldsymbol{d}) \bigg)^2 \\
\leq& 2(f(\boldsymbol{a}) - f(\boldsymbol{c}))^2 f(\boldsymbol{b})^2 + 2(f(\boldsymbol{b}) - f(\boldsymbol{d}))^2 f(\boldsymbol{c})^2 \\
\leq& 2M_f^2\bigg((f(\boldsymbol{a}) - f(\boldsymbol{c}))^2 + (f(\boldsymbol{b}) - f(\boldsymbol{d}))^2 \bigg) \\
=& 2M_f^2\bigg(f'(\xi_1)^2\|\boldsymbol{a} - \boldsymbol{c}\|^2 + f'(\xi_2)^2\|\boldsymbol{b} - \boldsymbol{d}\|^2 \bigg) \\
\leq& 2M_f^2 L_f^2(\|\boldsymbol{a} - \boldsymbol{c}\|^2 + \|\boldsymbol{b} - \boldsymbol{d}\|^2)
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
&(g(\boldsymbol{a}^T\boldsymbol{b}) - g(\boldsymbol{c}^T\boldsymbol{d}))^2 \\
=& (g'(\xi)(\boldsymbol{a}^T\boldsymbol{b} - \boldsymbol{c}^T\boldsymbol{d}))^2 \\
\leq& L_g^2(\boldsymbol{a}^T\boldsymbol{b} - \boldsymbol{c}^T\boldsymbol{b} + \boldsymbol{c}^T\boldsymbol{b} - \boldsymbol{c}^T\boldsymbol{d})^2 \\
=& L_g^2((\boldsymbol{a} - \boldsymbol{c})^T\boldsymbol{b} + (\boldsymbol{b} - \boldsymbol{d})^T\boldsymbol{c})^2 \\
\leq& 2L_g^2(\|(\boldsymbol{a} - \boldsymbol{c})^T\boldsymbol{b}\|^2 + 2\|(\boldsymbol{b} - \boldsymbol{d})^T\boldsymbol{c}\|^2) \\
\leq& 2L_g^2 R^2(\|\boldsymbol{a} - \boldsymbol{c}\|^2 + \|\boldsymbol{b} - \boldsymbol{d}\|^2)
\end{aligned}
$$

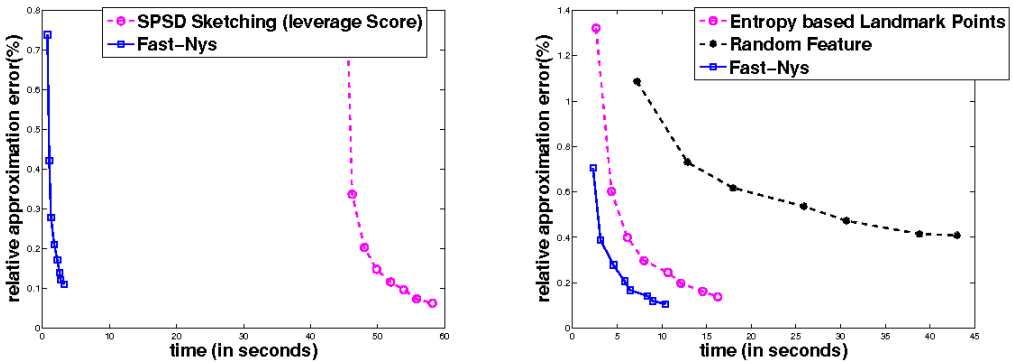
This proves (25). □

7.3. Parameters for the experimental results

- All the experiments were conducted on a machine with an Intel Xeon X5440 2.83GHz CPU and 32G RAM. We tried to have the best implementation for each algorithm. Fast-Nys, DC-Pred++, Nys, KNys, RKS, Fastfood are all implemented in C sharing the same modules. LDKL is the highly optimized C++ implementation published along with the original paper (Jose et al., 2013).
- The degree for the polynomial kernel and homogeneous kernel is set to be 3.
- We do data normalization with mean to be 0 and variance to be 1 before running our algorithms.
- When working on fast prediction experiments, we first form the low-rank approximation for the kernel matrix and apply liblinear to perform the classification.
- For fast prediction parameters(γ is the width parameters for Gaussian kernel and C is the regularization term in Liblinear SVM):
  - cifar: $\gamma = 2^{-10}$,$C = 64$;
  - mnist: $\gamma = 2^{-10}$,$C = 128$;
  - a9a: $C = 32$;
- For kernel approximation:
  - magic: $\gamma = 0.01$
  - ijcnn: $\gamma = 0.01$
  - webspam: $\gamma = 1$
- When working on prediction, we use random samples as the initial landmarks for Fast-Nys. The number of initial landmarks ranges from 2 to 10.
- When using kmeans Nyström, we randomly sample 10000 data samples to perform clustering.
- For LDKL, for a fair comparison, we disable the SSD operation.
- We use an alternating minimization algorithm to find the seeds in our algorithm. The algorithm usually converges to a reasonably good solution in 10 iterations, so we fix the number of iterations to be 10 for all the experiments. For example, on MNIST dataset with k=10, the initial objective function value (using random samples) is 1750260, after 10 iterations it drops to 90041, and the converged solution has objective function value 89872.

7.4. Comparison with other kernel approximation methods

We show the comparison between fast-Nys with leverage score (Gittens & Mahoney, 2013b) and entropy based landmark points (Brabanter et al., 2010) in Nyström approximation and random feature (Rahimi & Recht, 2007).



(a) MNIST, Gaussian (b) CIFAR, Gaussian

Figure 5. Low-rank kernel approximation results. x-axis is the time and y axis shows the relative kernel approximation error. Methods with approximation error above the top of y-axis are not shown. (a) compares Fast-Nys with sampling landmark points based on leverage score (Gittens & Mahoney, 2013a). Since this method needs to compute the entire kernel, it is much slower than our method. (b) compares Fast-Nys with entropy based landmark points based Nyström approximation (Brabanter et al., 2010) and random feature (Rahimi & Recht, 2008). We can also observe Fast-Nys achieves much lower approximation error than these two methods.