Memory Efficient Kernel Approximation

Si Si SSI@CS.UTEXAS.EDU
Cho-Jui Hsieh CJHSIEH@CS.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
Department of Computer Science, The University of Texas, Austin, TX 78721, USA

Abstract

The scalability of kernel machines is a big challenge when facing millions of samples due to storage and computation issues for large kernel matrices, that are usually dense. Recently, many papers have suggested tackling this problem by using a low-rank approximation of the kernel matrix. In this paper, we first make the observation that the structure of shift-invariant kernels changes from low-rank to block-diagonal (without any low-rank structure) when varying the scale parameter. Based on this observation, we propose a new kernel approximation algorithm – Memory Efficient Kernel Approximation (MEKA), which considers both low-rank and clustering structure of the kernel matrix. We show that the resulting algorithm outperforms state-of-the-art low-rank kernel approximation methods in terms of speed, approximation error, and memory usage. As an example, on the mnist2m dataset with two-million samples, our method takes 550 seconds on a single machine using less than 500 MBytes memory to achieve 0.2313 test RMSE for kernel ridge regression, while standard Nyström approximation takes more than 2700 seconds and uses more than 2 GBytes memory on the same problem to achieve 0.2318 test RMSE.

1. Introduction

Kernel methods (Schölkopf & Smola, 2002) are a class of machine learning algorithms that map samples from input space to a high-dimensional feature space. In the high-dimensional feature space, various methods can be applied depending on the machine learning task, for example, kernel support vector machine (SVM) (Cortes & Vapnik, 1995) and kernel ridge regression (Saunders et al., 1998). A key issue in scaling up kernel machines is the storage and computation of the kernel matrix, which is usually dense. Storing the dense matrix takes $O(n^2)$ space, while computing it takes $O(n^2 d)$ operations, where n is the number of data points and d is the dimension. A common solution is to approximate the kernel matrix using limited memory storage. This approach not only resolves the memory issue, but also speeds up kernel machine solvers, because the time complexity for using the kernel is usually proportional to the amount of memory used to represent the kernel. Most kernel approximation methods aim to form a low-rank approximation $G \approx CC^T$ for the kernel matrix G, with $C \in \mathbb{R}^{n \times k}$ and rank $k \ll n$. Although it is well known that Singular Value Decomposition (SVD) can compute the best rank-k approximation, it often cannot be applied to kernel matrices as it requires the entire kernel matrix to be computed and stored. To overcome this issue, many methods have been proposed to approximate the best rank k approximation of kernel matrix, including Greedy basis selection techniques (Smola & Schölkopf, 2000), incomplete Cholesky decomposition (Fine & Scheinberg, 2001), and Nyström methods (Drineas & Mahoney, 2005).

However, it is unclear whether low-rank approximation is the most memory efficient way to approximate the kernel matrix. In this paper, we first make the observation that for practically used shift-invariant kernels, the kernel structure varies from low-rank to block-diagonal as the scaling parameter γ varies from 0 to ∞. This observation suggests that even the best rank-k approximation can have extremely large approximation error when γ is large, so it is worth exploiting the block structure of the kernel matrix. Based on this idea, we propose a Memory Efficient Kernel Approximation (MEKA) method to approximate the kernel matrix. Our proposed method considers and analyzes the use of clustering in the input space to efficiently exploit the block structure of shift-invariant kernels. We show that the blocks generated by kmeans clustering have low-rank structure, which motivates us to apply Nyström low-rank approximation to each block separately. Between-cluster blocks are then approximated in a memory-efficient manner. Our approach only needs $O(nk + (ck)^2)$ memory to store a rank-ck approximation(where $c \ll n$ is the number of clusters), while traditional low-rank methods need $O(nk)$ space to store a rank-k approximation. Therefore,

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

using the same amount of storage, our method can achieve lower approximation error than the commonly used low-rank methods. Moreover, our proposed method takes less computation time than other low-rank methods to achieve a given approximation error.

Theoretically, we show that under the same amount of storage, the error bound of our approach can be better than standard Nyström, if the gap between $k+1^{st}$ and $ck+1^{st}$ singular values for G is large and values in the between-cluster blocks are relatively small compared to the within-cluster blocks. On real datasets, using the same amount of memory, our proposed algorithm achieves much lower reconstruction error using less time, and is faster for kernel regression. For example, on the mnist2m dataset with 2 million samples, our method takes 550 seconds on a single machine using less than 500 MBytes memory to achieve accuracy comparable with standard Nyström approximation, which takes more than 2700 seconds and uses more than 2 GBytes memory on the same problem.

The rest of the paper is outlined as follows. We present related work in Section 2, and then motivate our algorithm in Section 3. Our kernel approximation algorithm MEKA is proposed and analyzed in Section 4. Experimental results are given in Section 5. We present our conclusions in Section 6.

2. Related Research

To approximate the kernel matrix using limited memory, one common way is to use a low-rank approximation. The best rank-k approximation can be obtained by the SVD, but it is computationally prohibitive when n grows to tens of thousands. One alternative is to use randomized SVD (Halko et al., 2011), but it still needs computation and storage of the entire dense kernel matrix.

In order to overcome the prohibitive time and space complexity of SVD, the Nyström and Random Kitchen Sinks (RKS) methods have been proposed. The Nyström method (Williams & Seeger, 2001) generates a low-rank approximation based on a sampled subset of columns of the kernel matrix. Many strategies have been proposed to improve over the basic Nyström approximation, including ensemble Nyström (Kumar et al., 2009), Nyström with k-means to obtain benchmark points (Zhang et al., 2008; Zhang & Kwok, 2010), and randomized Nyström (Li et al., 2010). Different Nyström sampling strategies are analyzed and compared in (Kumar et al., 2012; Gittens & Mahoney, 2013). Another way to approximate the kernel matrix is to use Random Kitchen Sinks (RKS) (Rahimi & Recht, 2007; 2008), which approximates the Gaussian kernel function based on its Fourier transform. Recently (Le et al., 2013) sped up the RKS by using fast Hadamard matrix multiplications. (Yang et al., 2012) showed that the Nyström method has better generalization error bound than the RKS approach if the gap in the eigen-spectrum of the kernel matrix is large. Another approach proposed by (Cotter et al., 2011) approximates the Gaussian kernel by the t-th order Taylor expansion, but it requires $O(d^t)$ features, which is computationally intractable for large d or t.

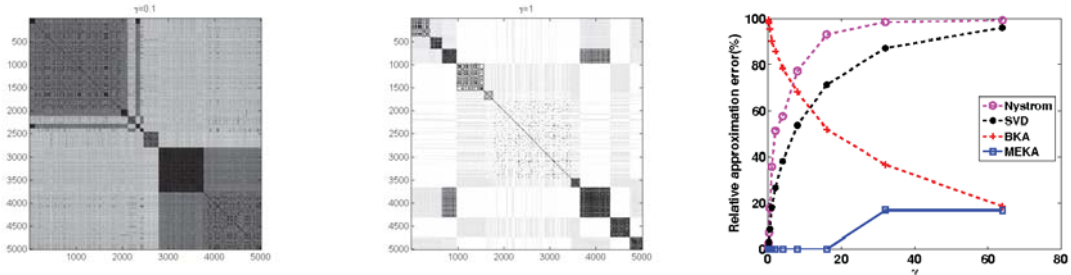
Many of the above methods can be viewed as faster ways to approximate the top-k SVD of the kernel matrix, so the approximation error is always larger than top-k SVD when using $O(nk)$ memory. As we show in Section 3, the kernel matrix typically changes from low-rank to block structure as the scaling parameter γ increases. However, the block structure of the kernel matrix has *never* been considered in dense kernel approximation, although it has been studied for approximation of other types of matrices. For example, (Savas & Dhillon, 2011) applied Clustered Low Rank Approximation (CLRA) to approximate large and sparse social networks. CLRA applies spectral clustering to the adjacency matrix, runs SVD on each diagonal block, and uses matrix projection to capture off-diagonal information. All these steps require storage of the whole matrix, thus is infeasible for large-scale dense kernel matrices. For example, computing the whole kernel matrix of the mnist2m dataset requires about 16 TBytes of memory; moreover the time for computing the SVD and projection steps is prohibitive. On the same dataset our proposed approach can achieve good results in 10 minutes with only 500 MBytes memory (as shown in Section 5). Thus, we need totally different algorithms for clustering and approximating blocks to yield our memory-efficient scheme.

3. Motivation

We use the Gaussian kernel as an example to discuss the structure of the kernel matrix under different scale parameters. Given two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the Gaussian kernel is defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}$, where γ is a scale or width parameter; the corresponding kernel matrix G is $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Low-rank approximation has been widely used to obtain an approximation for kernel matrices. However, under different scales, the kernel matrix has quite different structures, suggesting that different approximation strategies should be used for different γ.

Let us examine two extreme cases of the Gaussian kernel: when $\gamma \to 0$, $G \to \boldsymbol{e}\boldsymbol{e}^T$ where $\boldsymbol{e} = [1, \ldots, 1]^T$. As a consequence, G is close to low-rank when γ is small. However, on the other extreme as $\gamma \to \infty$, G changes to the identity matrix, which has full rank with all eigenvalues equal to 1. In this case, G does not have a low-rank structure, but has a block/clustering structure. This observation motivates us to consider both low rank and clustering structure of the kernel. Figure 1(a) and 1(b) give an example of the structure of a Gaussian kernel with different γ on a real dataset by randomly sampling 5000 samples from covtype dataset.

Before discussing further details, we first contrast the use of block and low-rank approximations on the same dataset. We compare approximation errors for different methods



(a) RBF kernel matrix with $\gamma = 0.1$ (b) RBF kernel matrix with $\gamma = 1$ (c) Comparison of different kernel approximation methods on various γ.

Figure 1. (a) and (b) show that the structure of the Gaussian kernel matrix $K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2}$ for the covtype data tends to become more block diagonal as γ increases(dark regions correspond to large values, while lighter regions correspond to smaller values). Plot (c) shows that low-rank approximations work only for small γ, and Block Kernel Approximation (BKA) works for large γ, while our proposed method MEKA works for small as well as large γ.

when they use the same amount of memory in Figure 1(c). Clearly, low rank approximation methods work well only for very small γ values. Block Kernel Approximation (BKA), a naive way to use clustering structure of G, as proposed in Section 4.1, is effective for large γ. Our proposed algorithm, MEKA, considers both block and low-rank structure of the kernel, and thus performs better than others under different γ values as seen in Figure 1(c).

4. Our proposed framework

In this section, we first introduce Block Kernel Approximation (BKA), a simple way to exploit the clustering structure of kernel matrices, and then propose our main algorithm, Memory Efficient Kernel Approximation(MEKA), which overcomes the shortcoming of BKA by considering both clustering and low-rank structure of the kernel matrix.

4.1. Clustering structure of shift-invariant kernel matrices

There has been substantial research on approximating shift-invariant kernels (Rahimi & Recht, 2007). A kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is shift-invariant if the kernel value depends only on $\boldsymbol{x}_i - \boldsymbol{x}_j$, that is, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(\eta(\boldsymbol{x}_i - \boldsymbol{x}_j))$ where $f(\cdot)$ is a function that maps $\mathbb{R}^d$ to $\mathbb{R}$, and $\eta > 0$ is a constant to determine the "scale" of the data. η is very crucial to the performance of kernel machines and is usually chosen by cross-validation. We further define $g_{\boldsymbol{u}}(t) = f(\eta t \boldsymbol{u})$ to be a one variable function along $\boldsymbol{u}$'s direction. We assume the kernel function satisfies the following property:

Assumption 1. *$g_{\boldsymbol{u}}(t)$ is differentiable in t when $t \neq 0$.*

Most of the practically used shift-invariant kernels satisfy the above assumption, for example, the Gaussian kernel ($K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}$), and the Laplacian kernel ($K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|_1}$). It is clear that η^2 is equivalent to γ for the Gaussian kernel. When η is large, off-diagonal blocks of shift-invariant kernel matrices will become zero, and all the information is concentrated in diagonal blocks. To approximate the kernel matrix by exploiting this "clustering structure", we first present a simple Block Kernel Approximation (BKA) as follows. Given a good partition $\mathcal{V}_1, \ldots, \mathcal{V}_c$ of the data points, where each $\mathcal{V}_s$ is a subset of $\{1, \ldots, n\}$, BKA approximates the kernel matrix as:

$$G \approx \tilde{G} \equiv G^{(1,1)} \oplus G^{(2,2)} \oplus \cdots \oplus G^{(c,c)}. \quad (1)$$

Here, $\oplus$ denotes direct sum and $G^{(s,s)}$ denotes the kernel matrix for block $\mathcal{V}_s$ – note that this implies that diagonal blocks $\tilde{G}^{(s,s)} = G^{(s,s)}$ and all the off-diagonal blocks, $\tilde{G}^{(s,t)} = 0$ with $s \neq t$.

BKA is useful when η is large. By analyzing its approximation error, we now show that k-means in the input space can be used to capture the clustering structure in shift-invariant kernel matrices. The approximation error equals $\|\tilde{G} - G\|_F^2 = \sum_{i,j} K(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 - \sum_{s=1}^{c} \sum_{i,j \in \mathcal{V}_s} K(\boldsymbol{x}_i, \boldsymbol{x}_j)^2$. Since the first term is fixed, to minimize the error $\|\tilde{G} - G\|_F^2$, it is equivalent to maximizing the second term, the sum of squared within-cluster entries $D = \sum_{s=1}^{c} \sum_{i,j \in \mathcal{V}_s} K(\boldsymbol{x}_i, \boldsymbol{x}_j)^2$.

However, directly maximizing D will not give a useful partition – the maximizer will assign all the data into one cluster. The same problem occurs in graph clustering (Shi & Malik, 2000; von Luxburg, 2007). A common approach is to normalize D by each cluster's size $|\mathcal{V}_s|$. The resulting spectral clustering objective (also called ratio association) is:

$$D^{\text{kernel}}(\{\mathcal{V}_s\}_{s=1}^c) = \sum_{s=1}^{c} \frac{1}{|\mathcal{V}_s|} \sum_{i,j \in \mathcal{V}_s} K(\boldsymbol{x}_i, \boldsymbol{x}_j)^2. \quad (2)$$

Maximizing (2) usually yields a balanced partition, but the computation is expensive because we have to compute all the entries in G. In the following theorem, we derive a lower bound for $D^{\text{kernel}}(\{\mathcal{V}_s\}_{s=1}^c)$:

Theorem 1. *For any shift-invariant kernel that satisfies Assumption 1,*

$$D^{kernel}(\{\mathcal{V}_s\}_{s=1}^c) \geq \bar{C} - \eta^2 R^2 D^{kmeans}(\{\mathcal{V}_s\}_{s=1}^c) \quad (3)$$

where $\bar{C} = \frac{nf(0)^2}{2}$, R is a constant depending on the kernel function, and $D^{kmeans} \equiv \sum_{s=1}^{c} \sum_{i \in \mathcal{V}_s} \|\boldsymbol{x}_i - \boldsymbol{m}_s\|_2^2$ is the k-means objective function, where $\boldsymbol{m}_s = (\sum_{i \in \mathcal{V}_s} \boldsymbol{x}_i)/|\mathcal{V}_s|$ for $s = 1, \cdots, c$ are the cluster centers.

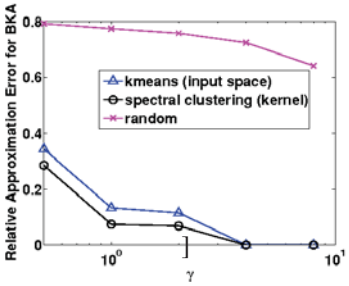


Figure 2. The Gaussian kernel approximation error of BKA using different ways to generate five partitions on 500 samples from covtype. K-means in the input space performs similar to spectral clustering on kernel matrix, but is much more efficient.

The proof is given in Appendix 7.1. Interestingly, the right hand side of (3) can be maximized when the k-means objective function D^{kmeans} is minimized. Therefore, although optimal solutions for k-means and ratio association might be different (consider the two circles non-linear case), Theorem 1 shows that conducting k-means in the input space will provide a reasonably good way to exploit the clustering structure of shift-invariant kernels, especially when it is infeasible to perform spectral clustering on G as it needs to precompute the entire kernel matrix. Figure 2 shows that the partition from k-means works as well as spectral clustering on G, which directly optimizes D^{kernel}. One advantage of conducting k-means is that the time complexity of each iteration is $O(ndc)$, which is much less than computing the kernel when the dimensionality d is moderate.

4.2. Memory Efficient Kernel Approximation

However, there are two main drawbacks to the above approach: (i) BKA ignores all the off-diagonal blocks, which results in large error when η is small (as seen in Figure 1(c)); (ii) for large-scale kernel approximation, it is too expensive to compute and store all the diagonal block entries. We now propose our new framework: Memory Efficient Kernel Approximation (MEKA) to overcome these two drawbacks.

To motivate using low-rank representation in our proposed method, we first present the following bound:

Theorem 2. *Given data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and a partition $\{V_1, \ldots, \mathcal{V}_c\}$, then for any s, t ($s = t$ or $s \neq t$)*

$$\|G^{(s,t)} - G_k^{(s,t)}\|_F \leq 4Ck^{-1/d}\sqrt{|\mathcal{V}_s||\mathcal{V}_t|}\min(r_s, r_t),$$

where $G_k^{(s,t)}$ is the best rank-k approximation to $G^{(s,t)}$; C is the Lipschitz constant of the shift-invariant function; r_s is the radius of the s-th cluster.

The proof is in Appendix 7.2. Theorem 2 suggests that when we apply k-means, r_s will be reduced and can be quite small, so the diagonal blocks and off-diagonal blocks tend to be low-rank. We also observe this phenomenon on real datasets: the rank of each block generated by k-means clustering is much smaller than by random clustering (see Appendix 7.4).

Memory Efficient Kernel Approximation (MEKA): Based on the above observation, we propose a fast and memory efficient scheme to approximate shift-invariant kernel matrices. As suggested by Theorem 2, each block tends to be low-rank after k-means clustering; thus we can form rank-k approximation for each of the c^2 blocks separately to achieve low error; however, this approach would require $O(cnk)$ memory, which can be prohibitive. Therefore, our proposed method first performs k-means clustering, and after rearranging the matrix according to clusters, it computes the low-rank basis only for diagonal blocks (which are more dominant than off-diagonal blocks) and uses them to approximate off-diagonal blocks. Empirically, we observe that the principal angles between basis of diagonal blocks and off-diagonal blocks are small(shown in Appendix 7.5). By using our proposed approach, we focus on diagonal blocks, and spend less effort on the off-diagonal blocks. Assume the rank-k_s approximation of the s^{th} diagonal block is $W^{(s)}L^{(s,s)}(W^{(s)})^T$, we form the following memory-efficient kernel approximation:

$$\tilde{G} = WLW^T, \tag{4}$$

where $W = W^{(1)} \oplus W^{(2)} \oplus \cdots \oplus W^{(c)}$; L is a "link" matrix consisting of c^2 blocks, where each $k_s \times k_t$ block $L^{(s,t)}$ captures the interaction between the s^{th} and t^{th} clusters. Let us first assume $k_s = k$, and we will discuss different stategies to choose k_s later. Note that if we were to restrict L to be a block diagonal matrix, $\tilde{G}$ would still be a block diagonal approximation of G. However, we consider the more general case that L is a dense matrix. In this case, each off-diagonal block $G^{(s,t)}$ is approximated as $W^{(s)}L^{(s,t)}(W^{(t)})^T$, and this approximation is memory efficient as only $O(k^2)$ additional memory is required to represent the (s,t) off-diagonal block. If a rank-k approximation is used within each cluster, then the generated approximation has rank ck, and takes a total of $O(nk+(ck)^2)$ storage.

Computing $W^{(s)}$. Since we aim to deal with dense kernel matrices of huge size, we use Nyström approximation to compute low-rank "basis" for each diagonal block. When applying the standard Nyström method to a $n_s \times n_s$ block $G^{(s,s)}$, we sample m columns from $G^{(s,s)}$, evaluate their kernel values, compute the rank-k pseudo-inverse of an $m \times m$ matrix, and form $G^{(s,s)} \approx W^{(s)}(W^{(s)})^T$. The time required per block is $O(n_s m(k+d) + m^3)$, and thus our method requires a total of $O(nm(k+d) + cm^3)$ time to form W. We can replace Nyström by any other kernel low-rank approximation method discussed in Section 2, but empirically we observe that the classical Nyström method combined with MEKA gives excellent performance.

Computing $L^{(s,t)}$. The optimal least squares solution for $L^{(s,t)}(s \neq t)$ is the minimizer of the local approximation error $\|G^{(s,t)} - W^{(s)}L^{(s,t)}(W^{(t)})^T\|_F$. However, forming the entire $G^{(s,t)}$ block can be time consuming. For example, computing the whole kernel matrix for mnist2m

Table 1. Memory and time analysis of various kernel approximation methods, where T_L is the time to compute the L matrix and T_C is the time for clustering in MEKA.

Method	Storage	Rank	Time Complexity
RKS	$O(cnk)$	ck	$O(cnkd)$
Nyström	$O(cnk)$	ck	$O(cnm(ck+d)+(cm)^3)$
SVD	$O(cnk)$	ck	$O(n^3+n^2d)$
MEKA	$O(nk+(ck)^2)$	ck	$O(nm(k+d)+cm^3)+T_L+T_C$

with 2 million data points takes more than a week. Therefore, to compute $L^{(s,t)}$, we propose to randomly sample a $(1+\rho)k \times (1+\rho)k$ submatrix $\bar{G}^{(s,t)}$ from $G^{(s,t)}$, and then find $L^{(s,t)}$ that minimizes the error on this submatrix. If the row/column index set for the subsampled submatrix $\bar{G}^{(s,t)}$ in $G^{(s,t)}$ is v_s/v_t, then $L^{(s,t)}$ can be computed in closed form:

$$L^{(s,t)} = ((W^{(s)}_{v_s})^T W^{(s)}_{v_s})^{-1} (W^{(s)}_{v_s})^T \bar{G}^{(s,t)} W^{(t)}_{v_t} ((W^{(t)}_{v_t})^T W^{(t)}_{v_t})^{-1},$$

where $W^{(s)}_{v_s}$ and $W^{(t)}_{v_t}$ are formed by the rows in $W^{(s)}$ and $W^{(t)}$ with row index sets v_s and v_t respectively.

Since there are only k^2 variables in $L^{(s,t)}$, we do not need too many samples for each block, and the time to compute $L^{(s,t)}$ is $O((1+\rho)^3 k^3)$. In practice, we observe that setting ρ to be 2 or 3 is enough for a good approximation, so the time complexity is about $O(k^3)$. In practice, many values in off-diagonal blocks are close to zero, and only few of them have large values as shown in Figure 1. Based on this observation, we further propose a thresholding technique to reduce the time for storing and computing $L^{(s,t)}$. Since the distance between cluster centers is a good indicator for the values in an off-diagonal block, we can set the whole block $L^{(s,t)}$ to 0 if $K(\boldsymbol{m}_s, \boldsymbol{m}_t) \leq \epsilon$ for some thresholding parameter $\epsilon > 0$. Obviously, to choose ϵ, we need to achieve a balance between speed and accuracy. When ϵ is small, we will compute more $L^{(s,t)}$; while when ϵ is large, we will set more $L^{(s,t)}$ to be 0, but increase the approximation error. The influence of ϵ on real datasets is shown in Appendix 7.8.

Choosing the rank k_s for each cluster. We need to decide the rank for the $s^{th}(s = 1, \cdots, c)$ cluster, k_s, which can be done in different ways: (i) the same k for all the clusters; (ii) singular value based approach. For (ii), suppose $M^{(s)}$ is the $m_s \times m_s$ matrix consisting of the intersection of m_s sampled columns in $G^{(s,s)}$, and M is the $cm \times cm (\sum_{s=1}^{c} m_s = cm)$ block-diagonal matrix with $M^{(s)}$ as diagonal block. We can choose k_s such that the set of top-ck singular values of M is the union of the singular values of $M^{(s)}$ in each cluster, that is, $[\sigma_1(M), \ldots, \sigma_{ck}(M)] = \cup_{s=1}^{c} [\sigma_1(M^{(s)}), \ldots, \sigma_{k_s}(M^{(s)})]$. To use (ii), we can oversample points in each cluster, e.g., sample $2k$ points from each cluster, perform eigendecompostion of a $2k \times 2k$ kernel matrix, sort the eigenvalues from c clusters, and finally select the top-ck eigenvalues and their corresponding eigenvectors. (i) achieves lower memory usage and is faster, while (ii) is slower but achieves lower error for diagonal blocks. In the experiment, we set all the clusters to have the same rank k. We show that this simple choice of k_s already outperforms state-of-the-art kernel approximation methods.

Our main algorithm is presented in Algorithm 1. In Table 1, we compare the time and storage for our method with SVD, standard Nyström, and RKS. We can see that MEKA is more memory efficient. For the time complexity, both T_L (time for computing off-diagonal L) and T_C (time for clustering) are small because (1) we use thresholding to force some $L^{(s,t)}$ blocks to be zero, and perform least squares on small blocks, which means T_L can at most be $O(\frac{1}{2}c^2k^3)$; (2)T_C is proportional to the number of samples. For a large dataset, we sample 20000 points for k-means, and thus the clustering is more efficient than working on the entire data set. We show the empirical time cost for each step of our algorithm in Appendix 7.9.

Algorithm 1: Memory Efficient Kernel Approximation (MEKA)

Input : Data points $\{(\boldsymbol{x}_i)\}_{i=1}^{n}$, scaling parameter γ, rank k, and number of cluster c.

Output : The rank-ck approximation $\tilde{G} = WLW^T$ using $O(nk + (ck)^2)$ space

Generate the partition $\mathcal{V}_1, \ldots, \mathcal{V}_c$ by k-means;
for $s = 1, \ldots, c$ **do**
  Perform the rank-k approximation
  $G^{(s,s)} \approx W^{(s)}(W^{(s)})^T$ by standard Nyström;
forall $(s,t)(s \neq t)$ **do**
  Sample a submatrix $\bar{G}^{(s,t)}$ from $G^{(s,t)}$ with row index set v_s and column index set v_t;
  Form $W^{(s)}_{v_s}$ by selecting the rows in $W^{(s)}$ according to index set v_s;
  Form $W^{(t)}_{v_t}$ by selecting the rows in $W^{(t)}$ according to index set v_t;
  Solve the least squares problem:
  $\bar{G}^{(s,t)} \approx W^{(s)}_{v_s} L^{(s,t)} (W^{(t)}_{v_t})^T$ to obtain $L^{(s,t)}$;

4.3. Analysis

We now bound the approximation error for our proposed method. We show that when $\sigma_{k+1} - \sigma_{ck+1}$ is large, where σ_{k+1} and σ_{ck+1} are the $k+1^{st}$ and $ck+1^{st}$ singular values of G respectively, and entries in off-diagonal blocks are small, MEKA has a better approximation error bound compared to standard Nyström that uses similar storage.

Theorem 3. *Let Δ denote a matrix consisting of all off-diagonal blocks of G, so $\Delta^{(s,t)} = G^{(s,t)}$ for $s \neq t$ and all zeros when $s = t$. We sample cm points from the dataset uniformly at random without replacement and split them according to the partition from k-means, such that each cluster has m_s benchmark points and $\sum_{s=1}^{c} m_s = cm$. Let G_{ck} be the best rank-ck approximation of G and $\tilde{G}$ be the rank-ck approximation from MEKA. Suppose we choose the rank k_s for each block using the singular value based*

Table 2. Comparison of approximation error of our proposed method with six other state-of-the-art kernel approximation methods on real datasets, where γ is the Gaussian scaling parameter; c is the number of clusters in MEKA; k is the rank of each diagonal-block in MEKA and the rank of the approximation for six other methods. Note that for a given k, every method has roughly the same amount of memory. All results show relative kernel approximation errors for each k.

Dataset	k	γ	c	Nys	RNys	KNys	ENys	RKS	fastfood	MEKA
pendigit	128	2	5	0.1325	0.1361	0.0828	0.2881	0.4404	0.4726	**0.0811**
ijcnn1	128	1	10	0.0423	0.0385	0.0234	0.1113	0.2972	0.2975	**0.0082**
covtype	256	10	15	0.3700	0.3738	0.2752	0.5646	0.8825	0.8920	**0.1192**

Table 3. Comparison of our proposed method with six other state-of-the-art kernel approximation methods on real datasets for kernel ridge regression, where λ is the regularization constant. All the parameters are chosen by cross validation, and every method has roughly the same amount of memory as in Table 2. All results show test RMSE for regression for each k. Note that k for fastfood needs to be larger than d, so we cannot test fastfood on mnist2m when $k = 256$.

Dataset	k	γ	c	λ	Nys	RNys	KNys	ENys	RKS	fastfood	MEKA
wine	128	2^{-10}	3	2^{-4}	0.7514	0.7555	0.7568	0.7732	0.7459	0.7509	**0.7375**
cadata	128	2^{2}	5	2^{-3}	0.1504	0.1505	0.1386	0.1462	0.1334	0.1502	**0.1209**
cpusmall	256	2^{2}	5	2^{-4}	8.8747	8.6973	6.9638	9.2831	9.6795	10.2601	**6.1130**
census	256	2^{-4}	5	2^{-5}	0.0679	0.0653	0.0578	0.0697	0.0727	0.0732	**0.0490**
covtype	256	2^{2}	10	2^{-2}	0.8197	0.8216	0.8172	0.8454	0.8011	0.8026	**0.7106**
mnist2m	256	2^{-5}	40	2^{-5}	0.2985	0.2962	0.2725	0.3018	0.3834	na	**0.2667**

approach as mentioned in Section 4.2, then with probability at least $1-\delta$, the following inequalities hold for any sample of size cm:

$$\|G-\tilde{G}\|_2 \leq \|G-G_{ck}\|_2+\frac{1}{\sqrt{c}}\frac{2n}{\sqrt{m}}G_{max}(1+\theta)+2\|\Delta\|_2,$$

$$\|G-\tilde{G}\|_F \leq \|G-G_{ck}\|_F+\left(\frac{64k}{m}\right)^{\frac{1}{4}} nG_{max}(1+\theta)^{\frac{1}{2}}+2\|\Delta\|_F$$

where $\theta = \sqrt{\frac{n-m}{n-0.5}\frac{1}{\beta(m,n)}\log\frac{1}{\delta}}d^G_{max}/G^{\frac{1}{2}}_{max}$; $\beta(m,n) = 1-\frac{1}{2\max\{m,n-m\}}$;$G_{max} = max_i G_{ii}$; *and* d^G_{max} *represents the distance* $max_{ij}\sqrt{G_{ii}+G_{jj}-2G_{ij}}$.

The proof is given in Appendix 7.3. When $k_s(s = 1,\cdots,c)$ is balanced and n is large, MEKA provides a rank-ck approximation using roughly the same amount of storage as rank-k approximation by standard Nyström. Interestingly, from Theorem 3, if $\|G-G_k\|_2-\|G-G_{ck}\|_2 \geq 2\|\Delta\|_2$, then

$$\|G-\tilde{G}\|_2 \leq \|G-G_k\|_2+\frac{1}{\sqrt{c}}\frac{2n}{\sqrt{m}}G_{max}(1+\theta).$$

The second term in the right hand side of above inequality is only $\frac{1}{\sqrt{c}}$ of that in the spectral norm error bound for standard Nyström that uniformly samples m columns without replacement in G to obtain the rank-k approximation as shown in (Kumar et al., 2009). Thus, if there is a large enough gap between σ_{k+1} and σ_{ck+1}, the error bound for our proposed method is better than standard Nyström that uses similar storage. Furthermore, when γ is large, G tends to have better clustering structure, suggesting in Theorem 3 that $\|\Delta\|$ is usually quite small. Note that when using the same rank k for all the clusters, the above bound can be worse because of some extreme cases, e.g., all the top-ck eigenvalues are in the same cluster. In practice we do not observe those extreme situations.

Table 4. Data set statistics (n: number of samples).

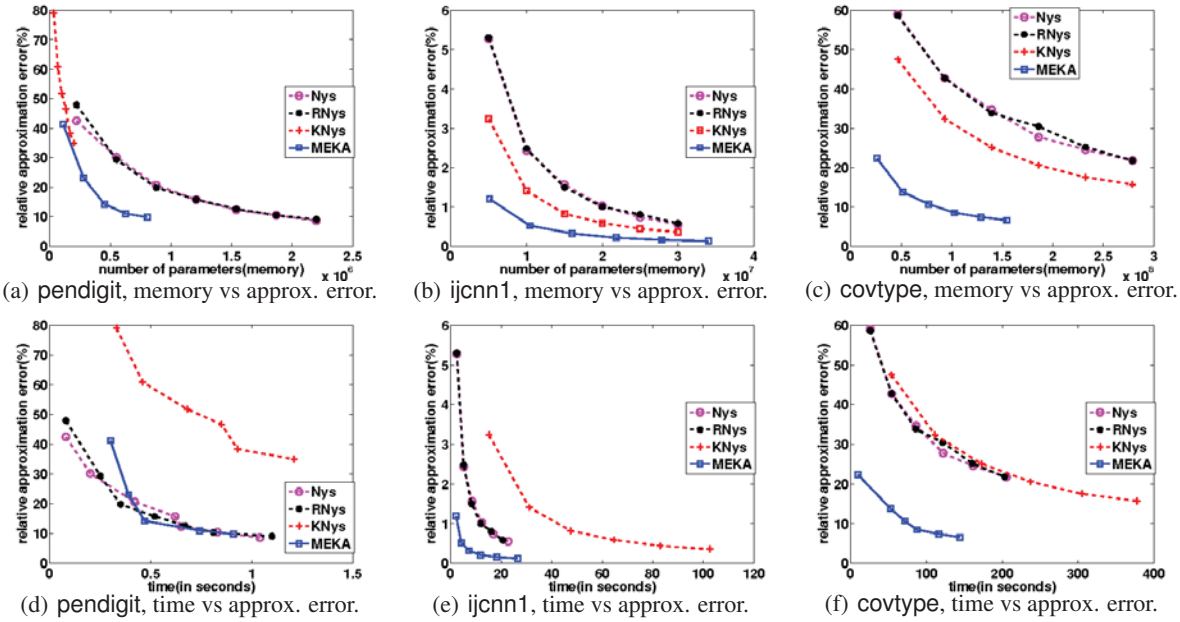
Dataset	n	d	Dataset	n	d
wine	6,497	11	census	22,784	137
cpusmall	8,192	12	ijcnn1	49,990	22
pendigit	10,992	16	covtype	581,012	54
cadata	20,640	8	mnist2m	2,000,000	784

5. Experimental Results

In this section, we empirically demonstrate the benefits of our proposed method, MEKA on various data sets[1] that are listed in Table 4. All experiment results here are based on the Gaussian kernel, but we observe similar behavior on other shift-invariant kernels (see Appendix 7.7 for results on other kernels). We compare our method with six state-of-the-art kernel approximation methods:

1. The standard Nyström(Williams & Seeger, 2001) method(denoted by Nys). In the experiment, we uniformly sample $2k$ columns of G without replacement, and run Nyström for rank-k approximation.
2. Kmeans Nyström(Zhang & Kwok, 2010) (denoted by KNys), where the landmark points are the cluster centroids. As suggested in (Zhang et al., 2012), we sample 20000 points for clustering when the total number of data samples is larger than 20000.
3. Random Kitchen Sinks(Rahimi & Recht, 2008)(denoted by RKS), which approximates the shift-invariant kernel based on its Fourier transform.
4. Fastfood with "Hadamard features"(Le et al., 2013)(denoted by fastfood).
5. Ensemble Nyström (Kumar et al., 2009)(denoted by ENys). Due to concern for the computation cost, we set the number of "experts" in ENys 3.
6. Nyström using randomized SVD(Li et al., 2010) (denoted by RNys). We set the number of power iterations $q = 1$ and oversampling parameter $p = 10$.

[1]All the datasets are downloaded from `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets` and UCI repository(Bache & Lichman, 2013).



(a) pendigit, memory vs approx. error. (b) ijcnn1, memory vs approx. error. (c) covtype, memory vs approx. error.

(d) pendigit, time vs approx. error. (e) ijcnn1, time vs approx. error. (f) covtype, time vs approx. error.

Figure 3. Low-rank Gaussian kernel approximation results. Methods with approximation error above the top of y-axis are not shown.

We compare all the methods on two different tasks: kernel low-rank approximation and kernel ridge regression. Note that both Nyström approximation and our method perform much better than incomplete Cholesky decomposition with side information (Bach & Jordan, 2005) (shown in Appendix 7.6), so we do not include the latter method in our comparison. All the experiments are conducted on a machine with an Intel Xeon X5440 2.83GHz CPU and 32G RAM.

5.1. Kernel approximation quality

We now compare the kernel approximation quality for the above methods. The parameters are listed in Table 2. The rank (k) varies from 100 to 600 for ijcnn1 and covtype and from 20 to 200 for the pendigit data.

Main results. The kernel approximation results are shown in Table 2 and Figure 3. We use relative kernel approximation error $\|G - \tilde{G}\|_F / \|G\|_F$ to measure the quality. We randomly sampled 20000 rows of G to evaluate the relative approximation error for ijcnn1 and covtype. In Table 2, we fix the rank k, or the memory usage of low-rank representation, and compare MEKA with the other methods in terms of relative approximation error. As can be seen, under the same amount of memory, our proposed method consistently yields lower approximation error than other methods. In Figure 3, we show the kernel approximation performance of different methods by varying k. Our proposed approximation scheme always achieves lower error with less time and memory. The main reason is that using similar amount of time and memory, our method aims to approximate the kernel matrix by a rank-ck approximation, while all other methods are only able to form a rank-k approximation.

Robustness to the Gaussian scaling parameter γ. To show the robustness of our proposed algorithm with different γ as explained in Section 2, we test its performance on the ijcnn1 (Figure 5(a)) and sampled covtype datasets (Figure 1(c)). The relative approximation errors for different γ values are shown in the figures using a fixed amount of memory. For large γ, the kernel matrix tends to have block structure, so our proposed method yields lower error than other methods. The gap becomes larger as γ increases. Interestingly, Figure 1(c) shows that the approximation error of MEKA is even superior to the exact SVD, as it is much more memory efficient. Even for small γ where the kernel exhibits low-rank structure, our proposed method performs better than Nyström based methods, suggesting that it can get the low rank structure of the kernel matrix.

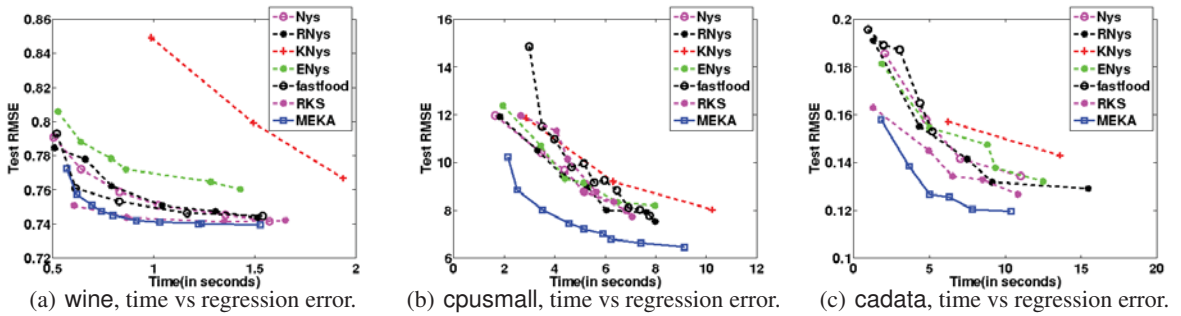
Robustness to the number of clusters c. Compared with Nyström, one main extra parameter for our method is the number of clusters c. In Figure 5(b), we test our method with different values of c on ijcnn1 dataset. In this experiment, c ranges from 5 to 25 and the rank $k = 100$ in each cluster. The memory usage is $nk + (ck)^2$, so the storage increases as c increases. For a fair comparison, we increase the rank of other methods as c increases, so that all the methods use the same amount of memory. Figure 5(b) shows that the performance of our proposed method is stable for varying choices of c.

5.2. Kernel Ridge Regression

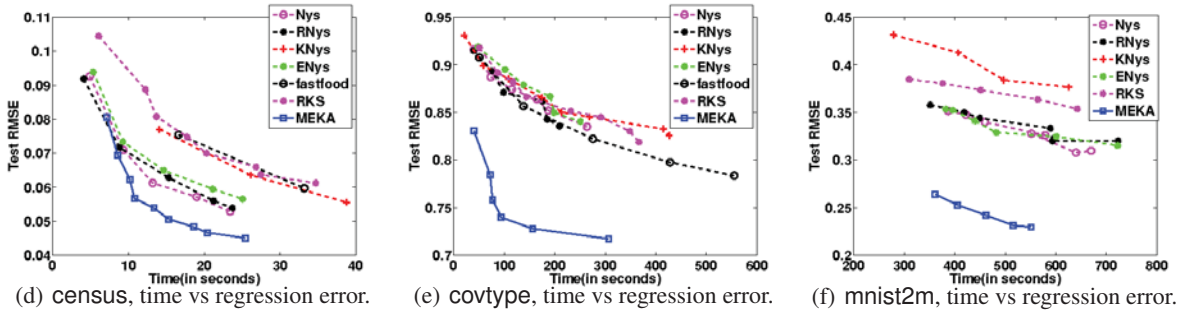
Next we compare the performance of various methods on kernel ridge regression (Saunders et al., 1998):

$$\min_{\boldsymbol{\alpha}} \|G\boldsymbol{\alpha} - \boldsymbol{y}\|^2 + \lambda \|\boldsymbol{\alpha}\|^2, \tag{5}$$

where G is the kernel matrix formed by training samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$, and $\boldsymbol{y} \in \mathbb{R}^l$ are the targets. For each kernel approximation method, we first form the approximated

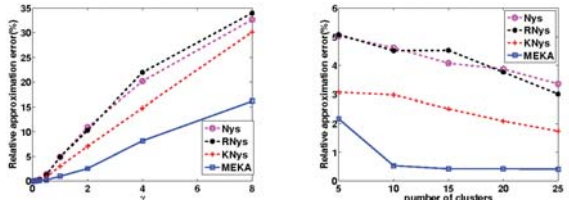


(a) wine, time vs regression error. (b) cpusmall, time vs regression error. (c) cadata, time vs regression error.



(d) census, time vs regression error. (e) covtype, time vs regression error. (f) mnist2m, time vs regression error.

Figure 4. Kernel ridge regression results for various data sets. Methods with regression error above the top of y-axis are not shown. All the results are averaged over five independent runs.



(a) Comparison under different γ. (b) Comparison under different numbers of clusters.

Figure 5. The kernel approximation errors for different Gaussian scaling parameter γ and c on ijcnn1 dataset.

kernel $\tilde{G}$, and then solve (5) by conjugate gradient (CG). The main computation in CG is the matrix vector product $\tilde{G}\boldsymbol{v}$. Using low-rank approximation, this can be computed using $O(nk)$ flops. For our proposed method, we compute $WLW^T\boldsymbol{v}$, where $W^T\boldsymbol{v} = \sum_{s=1}^{c} W^{(s)}\boldsymbol{v}^{(s)}$ requires $O(nk)$ flops, $L(W\boldsymbol{v})$ requires $O(\|L\|_0)$ flops, and $W(LW^T\boldsymbol{v})$ requires $O(nk)$ flops. Therefore, the time complexity for computing the matrix vector product for both MEKA and low-rank approximation methods are proportional to the memory for storing the approximate kernel matrices.

The parameters are chosen by five fold cross-validation and shown in Table 3. The rank for these algorithms is varied from 100 to 1000. The test root mean square error (test RMSE) is defined as $\|\boldsymbol{y}^{te} - G^{te}\boldsymbol{\alpha}\|$, where $\boldsymbol{y}^{te} \in \mathbb{R}^u$ is testing labels and $G^{te} \in \mathbb{R}^{u\times l}$ is the approximate kernel values between testing and training data. The covtype and mnist2m data sets are not originally designed for regression, and here we set the target variables to be 0 and 1 for mnist2m and -1 and 1 for covtype. Table 3 compares the kernel ridge regression performance of our proposed scheme with six other methods given the same amount of memory or same k in terms of test RMSE. It shows that our proposed method consistently performs better than other methods. Figure 4 shows the time usage of different methods for regression by varying the memory or rank k. As we can see that using the same amount of time, our proposed algorithm always achieves the lowest test RMSE. The total running time consists of the time for obtaining the low-rank approximation and time for regression. The former depends on the time complexity for each method, and the latter depends on the memory requirement to store the low-rank matrices. As shown in the previous experiment, MEKA is faster than the other methods while achieving lower approximation error and using less memory. As a consequence, it achieves lower test RMSE in less time compared to other kernel approximation methods.

6. Conclusions

In this paper, we have proposed a novel method, Memory Efficient Kernel Approximation (MEKA) for approximating shift invariant kernel matrices. We observe that the structure of the shift invariant kernel matrix changes from low rank to block diagonal as the scale parameter is changed. Our method exploits both low-rank and block structure present in the kernel matrix, and thus performs better than previously proposed low-rank based methods in terms of approximation and regression error, speed and memory usage.

Acknowledgements This research was supported by NSF grants CCF-1320746 and CCF-1117055. C.-J.H also acknowledges support from an IBM PhD fellowship.

References

Bach, Francis R. and Jordan, Michael I. Predictive low-rank decomposition for kernel methods. In *ICML*, 2005.

Bache, K. and Lichman, M. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

Cotter, A., Keshet, J., and Srebro, N. Explicit approximations of the Gaussian kernel. *arXiv:1109.47603*, 2011.

Cucker, F. and Smale, S. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2001.

Drineas, P. and Mahoney, M. W. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

Fine, S. and Scheinberg, K. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

Gittens, A. and Mahoney, M. W. Revisiting the Nyström method for improved large-scale machine learning. In *ICML*, 2013.

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

Kumar, S., Mohri, M., and Talwalkar, A. Ensemble Nyström methods. In *NIPS*, 2009.

Kumar, S., Mohri, M., and Talwalkar, A. Sampling methods for the Nyström method. *Journal of Machine Learning Research*, 13:981–1006, 2012.

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.

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

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

Saunders, C., Gammerman, A., and Vovk, V. Ridge regression learning algorithm in dual variables. In *ICML*, 1998.

Savas, B. and Dhillon, I. S. Clustered low rank approximation of graphs in information science applications. In *SDM*, 2011.

Schölkopf, B. and Smola, A. J. *Learning with kernels*. MIT Press, 2002.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Smola, A. J. and Schölkopf, B. Sparse greedy matrix approximation for machine learning. In *ICML*, 2000.

Stewart, G.W. and Ji-Guang, Sun. *Matrix Perturbation Theory*. Academic Press, Boston, 1990.

von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

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

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. Nyström method vs random Fourier features: A theoretical and empirical comparison. In *NIPS*, 2012.

Zhang, K. and Kwok, J. T. Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE Trans. Neural Networks*, 21(10):1576–1587, 2010.

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

Zhang, K., Lan, L., Wang, Z., and Moerchen, F. Scaling up kernel SVM on limited resources: A low-rank linearization approach. In *AISTATS*, 2012.