Memory Efficient Kernel Approximation

Si Si, Cho-Jui Hsieh, Inderjit S. Dhillon

Chloe Baraille, Othmane Sebbouh

May 19, 2018

Abstract

In this report, we will review the article "Mean Efficient Kernel Approximation" (Si Si et al. 2015), which develops a new method for kernel approximation which is robust to different scaling parameters while being time and memory efficient. We will show the benefits of the developed method by comparing it against the standard Nyström method.

Contents

1 Introduction

In many machine learning algorithms, using kernel methods allows to create feature maps in high-dimensional, sometimes infinite-dimensional spaces where solving a particular task is possible and often easier. One of the main issues with such methods is the fact that they require computing, for n observations, a kernel matrix $K \in \mathbb{R}^{n\times n}$. When n is huge, computing the kernel becomes prohibitive in terms of computation time and memory usage. In this review, we will present some solutions to this particular issue. In particular, we will consider the case of shift invariant kernels, to which the well-known Gaussian kernel belongs. In all this review, we will consider the observations $x_1, ..., x_n \in \mathbb{R}^d$.

Definition 1. *A kernel $K : Z \times Z \to \mathbb{R}$ is called **shift-invariant** if it only depends on the difference between its arguments,*

$$\forall i, j \in [n], K(x_i, x_j) = f(\eta(x_i - x_j))$$

Where f is a function that maps $\mathbb{R}^d$ to $\mathbb{R}$.

In particular, given $\gamma > 0$, the Gaussian kernel is a shift-invariant kernel as it is defined $\forall i, j \in [n]$ by $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|_2^2}$. Now consider the Gaussian kernel. This kernel exhibits a particular structure which depends upon the values of γ:

- When γ is large, the kernel has a low-rank structure
- When γ is small, the kernel has a block-diagonal structure

As an illustration, consider the most extreme cases where:

- $\gamma = \infty$: then $K = I_d$, the identity matrix, which is diagonal.
- $\gamma = 0$: then $K = ee^T$, where $e = (1, ..., 1)^T$. It has rank 1.

Once these observations are stated, one can see that two types of approximations are well-suited for the shift-invariant kernels:

- Block-diagonal approximations when γ is large
- Low-rank approximations when γ is small

For the second case, although Singular Value Decomposition (SVD)(thresholded) seems to be a good way to approximate the kernel as it is provably the best rank-k ($\forall k$) approximation possible, it requires $O(n^2)$ storage space and computing it involves $O(n^2 d)$ operations. Indeed, computing the SVD requires already knowing the kernel we initially wanted to cheaply approximate.

Therefore, other solutions are needed. In the next section, we will present two methods, which each addresses one of the two cases presented. We will then show how the Mean Efficient Kernel Approximation (MEKA) method developed in the studied article (subsequently referred to as "our article") is heavily inspired from these two methods and demonstrate, through experiments on the ijcnn1 dataset, that MEKA is not only well-suited for different values of γ, but is also time and memory efficient.

2 Approximating shift-invariant kernel matrices

In order to better understand our article's method, a good understanding of Block Kernel Approximation (BKA) and Nystrôm Methods (NYS) appears to be necessary. Indeed, as we will see later, both the ideas of data clustering in BKA and low-rank approximation and randomness in NYS are determinant in MEKA. Hence, we will first present BKA, then NYS. Finally, we will see that MEKA succeeds to overcome the drawbacks of the two methods, while being more time and memory efficient than NYS.

2.1 Block Kernel Approximation

Heuristically. When γ is large, the kernel matrix, which we will note G, is almost block diagonal, up to a partition of the data points. Indeed, given a good partition of the data, the off-diagonal blocks will be almost zero. As a result, BKA aims to approximate G by calculating only the kernel matrices corresponding to each partition of the data points, which constitute the diagonal blocks, and setting the off-diagonal blocks to 0.

Formally: The only element which needs to be specified for BKA is the partition of the datapoints. Noting $\|.\|_F$ the Frobenius norm and $\tilde{G}$ the approximation of G by BKA, one needs to find a partition $\mathcal{V}_1, ...\mathcal{V}_c$ of $[n]$ such that the approximation error $\|G - \tilde{G}\|$ is small, where:

$$\tilde{G} = \begin{bmatrix} G^{(1,1)} & 0 & \cdots & 0 \\ 0 & G^{(2,2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G^{(c,c)} \end{bmatrix}$$

Then $\|G-\tilde{G}\|_F^2 = \sum_{i,j} K(x_i, x_j) - \sum_{s=1}^{c} \sum_{i,j \in \mathcal{V}_s} K(x_i, x_j)$ and we need to solve the following problem:

$$\min_{\mathcal{V}_1,...,\mathcal{V}_c} \|G - \tilde{G}\|_F^2 = \min_{\mathcal{V}_1,...,\mathcal{V}_c} \sum_{s=1}^{c} \sum_{i,j \in \mathcal{V}_s} K(x_i, x_j)$$

Directly minimizing the last quantity often leads to degenerate solutions where all the clusters are extremely imbalanced. Hence, using standard reasoning, we prefer to divide each term by the size of each cluster:

$$\min_{\mathcal{V}_1,...,\mathcal{V}_c} \sum_{s=1}^{c} \sum_{i,j \in \mathcal{V}_s} K(x_i, x_j)$$

While solving this problem allows indeed to find a good partition, it is computationally heavy. Instead, one can show that k-means clustering leads to a small enough approximation error. Hence we can now announce the BKA algorithm:

Algorithm 1 Block Kernel Approximation (BKA)

Input: Data points $\{(x_i)\}_{i=1}^n$, (big) scaling parameter γ, number of clusters c.
Output: The block-diagonal approximation $\tilde{G}$
Perform k-means clustering of the data points in c clusters
for $s = 1, ..., c$ **do**:
  Compute the cluster's kernel matrix $G^{(s,s)} = \{K(x_i, x_j)\}_{i,j \in \mathcal{V}_s}$
end

2.2 The standard Nyström method

The Nyström method applied to kernel matrix approximation was first developed in **[Ref]**. Many other variants of this method were later introduced, but we will restrict ourselves in this report to the original one, which was also studied in class. We now recall the method and the motivation behind it.

Motivation. Since kernel matrices often (small γ) exhibit a low rank structure, low rank approximations can be a solution for kernel approximation. As recalled in the introduction, using the best theoretical solution, SVD (or in the case of PSD kernels, eigendecomposition), is computationally costly. A solution to this problem is to approximate the eigenvectors and eigenvalues of G using the Nyström method which we describe below

Method. We sample $m(< n)$ columns of the kernel matric, wchih, since we don't have access to the kernel matrix, amounts to sampling m data points corresponding to a subset $J \in 1, ..., n, |J| = m$, and computing the columns $\{K(x_i, x_j)\}_{i \in [n], j \in J} = C$ and we compute the rank-k pseudo-inverse of the $m \times m$ matrix $L^{-1} = \{K(x_i, x_j\}_{i,j \in J}$. Finally, the rank-$k$ approximation of G is given by: $\tilde{G} = CLC^T$. This leads to the following algorithm:

Algorithm 2 Standard Nyström approximation (NYS)

Input: Data points $\{(x_i)\}_{i=1}^n$, scaling parameter γ, rank k, number of sampled columns m.
Output: The rank-k approximation $\tilde{G}$
Sample m data points $(x_i)_{i \in J}, |J| = m$ without replacement.
Compute $C = \{K(x_i, x_j)\}_{i \in [n], j \in J}$
Compute the rank-k pseudo-inverse of $L^{-1} = \{K(x_i, x_j\}_{i,j \in J}$ using SVD: L
Return $\tilde{G} = CLC^T$

3 Memory Efficient Kernel Approximation

Interestingly, MEKA draws from both the previous approaches to construct an approximation which is robust to different values of γ and time and memory efficient.

Similarly to BKA, MEKA attempts to capture the block-diagonal structure of the kernel matrix and hence clusters the data into c clusters. However to overcome the approximation error due to setting all off-diagonal blocks to 0, MEKA rather uses a low-rank approximation for each block. In MEKA, we note the matrix as follows:

$$\tilde{G} = \begin{bmatrix} \tilde{G}^{(1,1)} & \tilde{G}^{(1,2)} & \dots & \tilde{G}^{(1,c)} \\ \tilde{G}^{(2,1)} & \tilde{G}^{(2,2)} & \dots & \tilde{G}^{(2,c)} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{G}^{(c,1)} & \tilde{G}^{(c,2)} & \dots & \tilde{G}^{(c,c)} \end{bmatrix}$$

One observation that motivates the low-rank approximation of the blocks of the matrix is that it can be shown that with a suitable k-means clustering of the data (where each cluster has a small radius), all the blocks will be low-rank. A good idea would then be to perform a low-rank approximation of rach block and compute the resulting kernel matrix. However, this approach requires $O(cnk)$ storage space. The proposed alternative used in MEKA treats the block-diagonal

blocks (which are the most dominant) and the off-diagonal blocks differently.

The diagonal blocks. To compute the diagonal blocks approximation $(\tilde{G}^{(s,s)})_{s=1}^{c}$, we simply use the standard Nyström method to approximate each diagonal block $G^{(s,s)}, s = 1, ..., c$. As a result, we have: $\tilde{G}^{(s,s)} = W^{(s)} L^{(s,s)} {W^{(s)}}^T$. Hence, we have so far the following approximation:

$$\tilde{G} = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \begin{bmatrix} L^{(1,1)} & 0 & \cdots & 0 \\ 0 & L^{(2,2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L^{(c,c)} \end{bmatrix} \begin{bmatrix} {W^{(1)}}^T & 0 & \cdots & 0 \\ 0 & {W^{(2)}}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {W^{(c)}}^T \end{bmatrix}$$

This approximation still completely disregards the off-diagonal blocks of G, as so far $G^{(s,t)} = 0$ for all $s \neq t$.

The off-diagonal blocks. To compute the off-diagonal blocks of G, we need to compute the off-diagonal blocks of L. The best way to do so would be to take $L^{(s,t)}$ as the matrix minimizing the approximation error, i.e. as being the minimizer of the solution to the following problem:

$$\tilde{L}^{(s,t)} = \underset{L}{\operatorname{argmin}} \|G^{(s,t)} - W^{(s)} L (W^{(t)})^T\|$$

However, solving this problem is costly. Hence, we take the approximation a step further and randomly sample an $(1+\rho)k \times (1+\rho)k$ sub-matrix $\hat{G}^{(s,t)}$ of $G^{(s,t)}$ (in practice, we take $\rho = 2$ or 3) and solve:

$$\tilde{L}^{(s,t)} = \underset{L}{\operatorname{argmin}} \|\hat{G}^{(s,t)} - W^{(s)}_{v_s} L (W^{(t)})^T_{v_t}\|$$

where v_s are the sampled line of $W^{(s)}$ and v_t are the sampled columns of $W^{(t)}$. The solution to this problem is given by:

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

Hence, we now have the following approximation:

$$G \approx \tilde{G} = WLW^T$$

We can now announce the MEKA algorithm:

Algorithm 3 Mean Efficient Kernel Approximation (MEKA)

Input: Data points $\{(x_i)\}_{i=1}^{n}$, scaling parameter γ, rank k, number of clusters c, number of sampled columns for diagonal blocks m, ρ.
Output: The rank-k approximation $\tilde{G}$
Generate the partition $\mathcal{V}_1, ..., \mathcal{V}_c$ by k-means;
for s = 1 to c **do**:
  Perform the rank-k approximation $G^{(s,s)} \approx W^{(s)} L^{(s,s)} (W^{(s)})^T$ by standard Nyström.
end
forall $(s,t)(s \neq t)$ **do**
  Sample a sub-matrix $\bar{G}^{(s,t)}$ 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 $W^{(s)}$ according to index set v_t;
  Solve the least squares problem $\bar{G}^{(s,t)} \approx W^{(s)}_{v_s} L (W^{(t)})^T_{v_t}$ to obtain $L^{(s,t)}$;
end

Note that in this report we presented, for brievty, the most straightforward version of MEKA. Some alternatives exist. One can chose for example a different rank k_s for the approximation of the block-diagonal matrices. One can also set some off-diagonal blocks to 0 for faster computations. These two alternatives are implemented in our notebook.

4 Experiments

In our experiments, we show the benefits of MEKA, especially compared with the Nyström method.

For all the experiments, the measure of performance we use is the relative approximation error. Noting K the true kernel and $\tilde{K}$ its approximation by one of the proposed methods, the approximation error is defined as:

$$\frac{\|K - \tilde{K}\|_F}{\|K\|_F}$$

- **Robustness to** γ (Figure 1): We show that the MEKA method is robust to the choice of the parameter γ, which means that Meka performs well whether the Kernel matrix presents a block diagonal structure (high γ) or a low rank one (low γ). Note that this is not the case for Block Kernel Approximation (BKA), which fails when γ is high and for the standard Nyström method, which fails when γ is low. As we compare Meka with the standard Nyström method, we will see that this last observation is empirically verified.

- **Robustness to the number of clusters** c (Figure 2): Compared to the standard Nyström method, MEKA introduces an additional parameter: the number of clusters. We show that MEKA is robust to varying the number of clusters. Since increasing the number of clusters c increases the memory usage with MEKA, we increase the rank of the approximations for the standard Nyström method to have roughly the same memory usage to have a fair comparison.

- **Memory and time efficiency** (Figures 3 and 4): We compare the two methods in terms of their memory usage and computation time. The computation time is measured empirically. Noting M the memory usage of a method, we consider that for a rank ck approximation of the kernel matrix: $M_{NYS} = cnk$ and $M_{MEKA} = nk + (ck)^2$

- **Accuracy of SVM**: We compared the accuracy scores of SVM with a Gaussian kernel using MEKA and NYS. The goal of the approximations is speeding up kernel-SVM when all the training and testing data are available beforehand. Note that if this is not the case, our methods are of limited use. The results of this task are only presented in the notebook.

As it is was one of the main motivations of the article, we illustrate the results on low and high values of γ each time. For all these comparisons, we clearly see that MEKA outperforms the standard Nyström method.

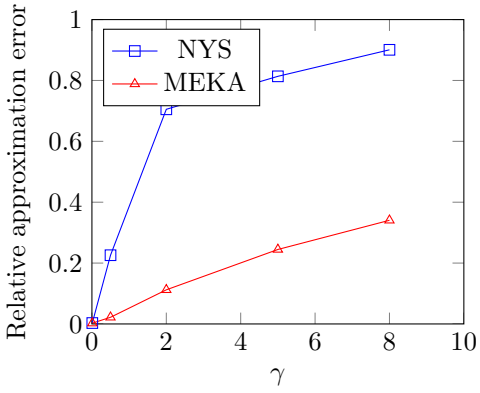

Figure1: Robustness to the scaling parameter γ

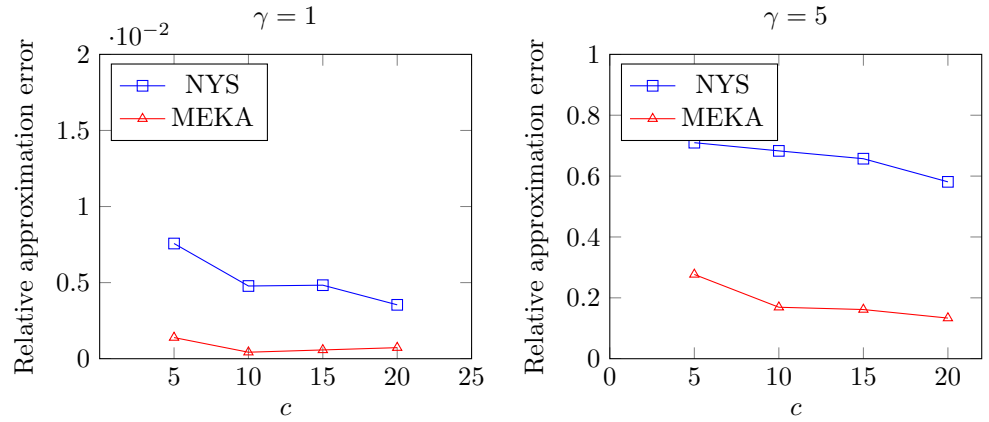

Figure 2: Robustness to the number of clusters c. The rank k is increased for NYS for a fair comparison.

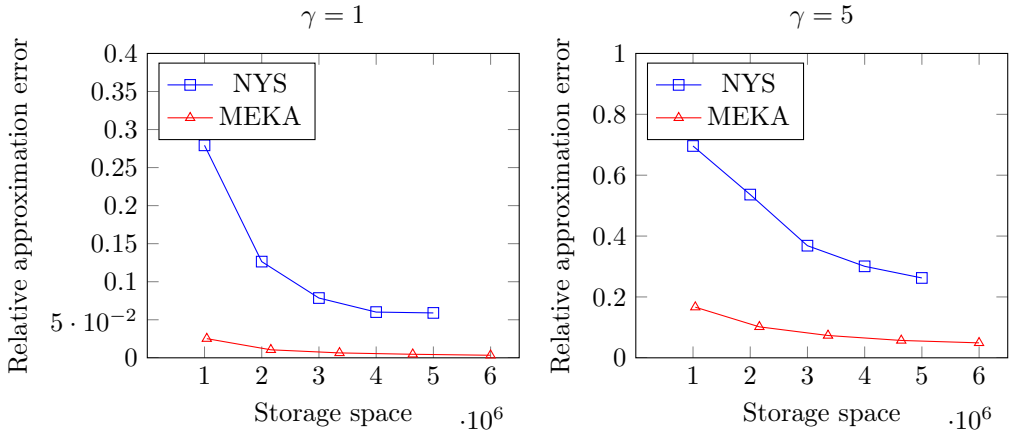

Figure 3: Memory efficiency of NYS and MEKA.

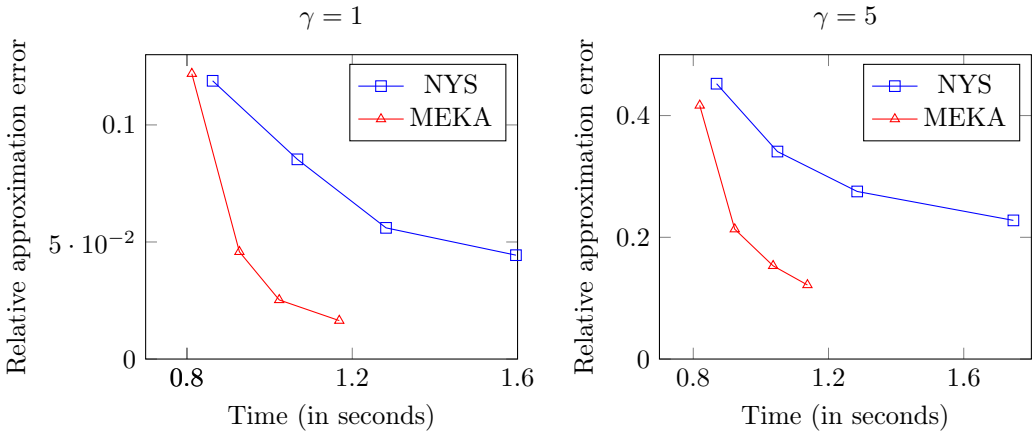

Figure 4: Time efficiency of NYS and MEKA.

Documentation and sources

[1] Si Si, Cho-Jui Hsieh, Inderjit S. Dhillon — *Memory Efficient Kernel Approximation*, ICML 2014.

[2] Christopher K. I. Williams, Matthias Seeger — *Using the Nyström method to speed-up kernel Machines*, Advances in Neural Information Processing Systems 13 (NIPS 2000).