Kernel Approximation[Mixed]

Atul [15817161]
Manish Kumar Bera [15807381]
Mohd Abbas Zaidi [150415]

***Abstract*—This document is part of the submission for final project report for the term paper of the course EE609: Convex Optimization [IIT Kanpur, Spring 2019].**

References: [Drineas and Mahoney, 2005] [Si et al., 2017] [Yang et al., 2014] [Kar and Karnick, 2012] [Hamid et al., 2014]

I. Introduction

The very famous 'kernel trick' is one of the most widely used method used in machine learning and statistical sciences to work on non-linear data using linear methods. Essentially, kernel replaces the euclidean inner-product with a non-linear inner-product: $\mathbf{x}^\top \mathbf{y} \to k(\mathbf{x}, \mathbf{y})$. Another common notation for kernels is $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$, where $\phi : \chi \to \chi'$ is the mapping to the transformed vector space. There are many popular kernels, like:

1) RBF (radial basis function) kernel
   $k(\mathbf{x}, \mathbf{y}) = \text{expe}\left\{-\gamma \left\|\mathbf{x} - \mathbf{y}\right\|^2\right\}$
2) polynomial kernel
   $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y} + c)^d$

However, the kernel trick is notoriously infamous for the amount of space required for storing the parameter values, and the time complexity for prediction. This is because, in the kernel trick, the parameters are generated by explicitly storing 'important' data instances. The kernels are instance based learners, as they remember the 'important' instances, which they use as parameters for the model. For example, if $\mathbf{w}$ is a learnt parameter, then for a test data point $\mathbf{x}_{\text{test}}$:

$$\mathbf{w}^\top \phi(\mathbf{x}_{\text{test}}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_{\text{test}})$$

This motivates the need for a kernel approximation technique. It is impractical, and sometimes impossible to compute the exact *kernel representation vector* ($\phi(\mathbf{x})$). This is because, the dimension of the *representation vector* may be too high (*representation vector* for RBF kernel has infinite dimensions). However, if we can use an approximation of the *representation vector*, it may prove to be useful.

Thus, we aim to find an approximation ($\hat{\phi}(\mathbf{x})$) of the *kernel representation vector* ($\phi(\mathbf{x})$), s.t., for any two data points in the data set, $\mathbf{x}$ and $\mathbf{y}$:

$$\hat{\phi}(\mathbf{x})^\top \hat{\phi}(\mathbf{y}) \approx \phi(\mathbf{x})^\top \phi(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$$

A very important structure in kernel methods is the Gram matrix. Let us define the gram matrix. Let $X \in \mathbb{R}^{n \times d}$ is the matrix representing certain dataset, where n is the number of data points and d is the number of features for each data point. Then the gram matrix is defined as:

$$G = \phi(X)\phi(X)^\top$$

In this report, we will discuss some methods of approximating the gram matrix.

$$\tilde{X}\tilde{X}^\top \approx G = \phi(X)\phi(X)^\top$$

A. Kernel Approximation Techniques

Kernel approximation techniques can be broadly classified into 3 categories:

1) **Kernel approximation for shift invariant kernels**: These methods are applicable for shift invariant kernels, like the *RBF kernel*. Methods including *Random Fourier Features* and *Quasi-Random Fourier Features* lie in this category.
2) **Kernel approximation for dot product kernels**: These methods are applicable for dot product kernels, like the *polynomial kernels*. Methods like *Random Maclaurin Feature Maps* and *Compact Random Feature Maps*
3) **General kernel approximation**: These methods are applicable for all types of kernels in general. Methods like the *Nystrom kernel approximation*, *Block Kernel Approximation*, and *Memory Efficient Kernel Approximation* fall in this category.

In this report we will primarily focus on the following methods:

1) Nystrom method
2) BKA (Block Kernel Approximation)
3) MEKA (Memory Efficient Kernel Approximation)

II. Nystrom Method[Drineas and Mahoney, 2005]

A. Introduction

Nystrom method is a very well known and well established technique used in kernel approximation. The primary reasons for its popularity are its robust proof of bounds and simplicity of implementation.

We have provided a brief review of the mathematical background and theorems required in the analysis of the Nystrom method, along with the nalysis, in appendix A.

B. The method

In this section we will describe the exact algorithm that is used in Nystrom method for kernel approximation.

ALGORITHM: Nystrom method for approximating kernel

- INPUT:
  - dataset $X \in \mathbb{R}^{n \times d}$
  - probability distribution over rows of X : $\{p_i\}_{i=1}^n$, s.t. $\sum_{i=1}^n p_i = 1$
  - a number $c \leq n$
  - a number $k \leq c$
  - a kernel k, s.t., $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$
- OUTPUT: $\tilde{X}$ s.t. $XX^\top \approx \tilde{X}\tilde{X}^\top$

1) Sample row indices of X according to the distribution $\{p_i\}_{i=1}^n$. The sampling is done i.i.d. and with replacement. Let the sampled indices be collected in the index set $S = \{s_1, s_2, \ldots, s_c\}$.
2) Construct matrix C such that:
$$C_{i,j} = \frac{G_{i,s_j}}{\sqrt{cp_{s_j}}} = \frac{k(X_i, X_{s_j})}{\sqrt{cp_{s_j}}}$$
3) Construct matrix W s.t.:
$$W_{i,j} = \frac{G_{s_i,s_j}}{c\sqrt{p_{s_i}p_{s_j}}}$$
4) compute W_k the best rank-k approximation of W
5) Construct $\tilde{X} = C(W_k^\dagger)^{1/2}$
6) return $\tilde{X}$

The analysis of this method is given in appendix A

III. Block Kernel Approximation [Si et al., 2017]

The BKA method essentially clusters the data points according to k-means and then applies matrix approximation to the gram matrices of the clusters separately.

ALGORITHM: Block Kernel Approximation

1) Cluster the data points using k-means. Let the cluster-sets be $\mathcal{N}_1, \ldots, \mathcal{N}_c$.
2) for each cluster $\mathcal{N}_s (s \in \{1, \ldots, c\})$, performs Nystrom method in the gram matrix of the cluster. The gram matrix of cluster s is denoted by $G^{(s)}$ s.t. $G^{(s)}_{i,j} = k(\phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}))$ where $i, j \in \mathcal{N}_s$.

The analysis for this method is given in appendix B.

IV. Memory Efficient Kernel Approximation[Si et al., 2017]

This is an extension of the BKA method. In this method the off-diagonal matrices are also factorized.

ALGORITHM: Memory Efficient Kernel Approximation

1) Cluster the data points using k-means. Let the cluster-sets be $\mathcal{N}_1, \ldots, \mathcal{N}_c$.
2) for each cluster $\mathcal{N}_s (s \in \{1, \ldots, c\})$, performs Nystrom method in the gram matrix of the cluster. The gram matrix of cluster s is denoted by $G^{(s)}$ s.t. $G^{(s)}_{i,j} = k(\phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}))$ where $i, j \in \mathcal{N}_s$. Let the decomposition look like this: $G^{(s)} = C_s W_s^\dagger C_s^\top$.
3) for each of the diagonal blocks $G^{(s,t)}$, s.t., $G^{(s,t)}_{i,j} = k(\phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}))$ where $i \in \mathcal{N}_s$ and $j \in \mathcal{N}_t$; the block is approximated as:
$$G^{(s,t)}_{i,j} = C_s W^\dagger_{s,t} C_t^\top$$

V. Experiment

We fully implemented the three kernel approximation methods that have been discussed namely Nystrom, Block Kernel Approximation(BKA) and Memory Efficient Kernel Approximation(MEKA). We have used Gaussian Kernel with Iris dataset in order to perform the comparisons. The error metric used for evaluation is the relative kernel approximation error $\left\|G - \tilde{G}\right\|_F / \|G\|_F$ which is the same as that used in meka.

We compared our Nystrom method with the inbuilt module of sklearn Nystrom to check our implementation. Figure 1 shows the comparison as we vary the gamma values in the kernel. It was observed that our implementation was very near in performance to the inbuilt Nystrom.

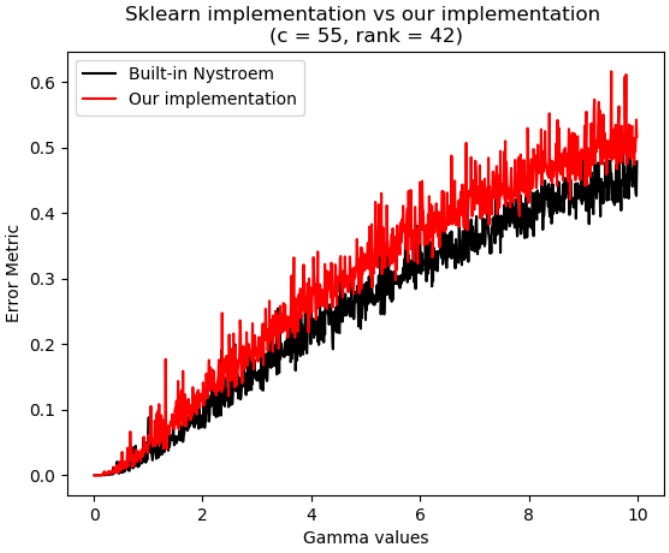

Fig. 1: Comparison while varying the γ values

Figure 2 provides the same comparison as we vary the rank(k in k-rank approximation).

In all the subsequent comparisons, Nystrom refers to our implementation. This is more suitable since both MEKA and

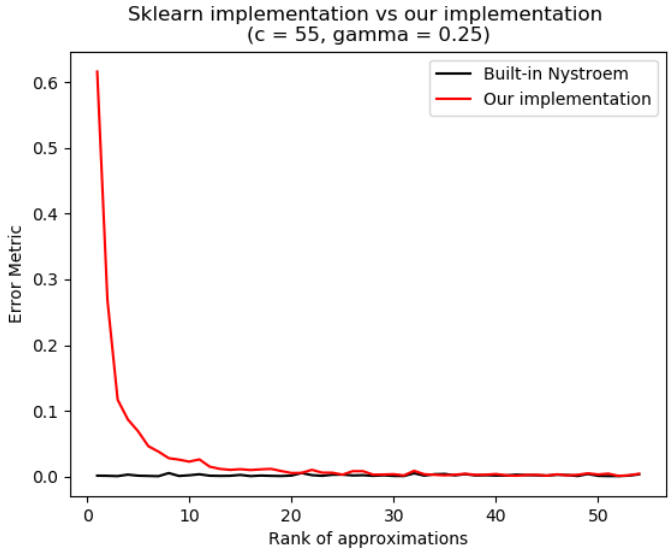

Fig. 2: Comparison while varying the rank in k-rank approximation

BKA use the implemented Nystrom in order to find the kernel approximation.

Next, we observed that BKA performs worse as compared to Nystrom(for same rank measure). This is expected since it only approximates the diagonal blocks. This can be seen in Figure 3.

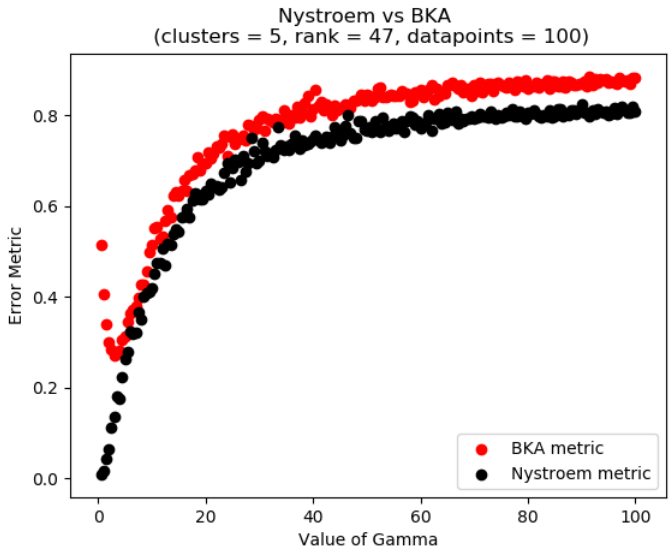

Fig. 3: Comparison while varying the rank in k-rank approximation

Interestingly it was also observed that for higher rank, BKA outperforms Nystrom when k is increased to a higher value as observed in Figure 4.

Next, we provide comparisons with MEKA. It was verified that MEKA outperforms the nystrom algorithm, the error in case of MEKA is lesser as compared to nystrom as seen in Figure 5

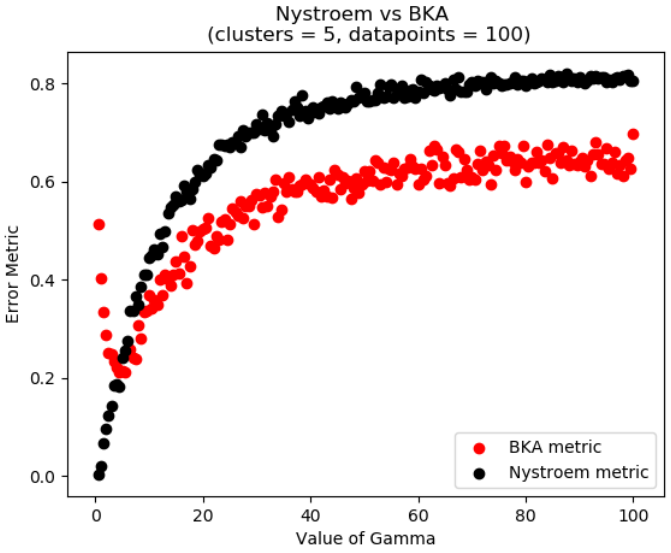

Fig. 4: Comparison while varying the rank in k-rank approximation

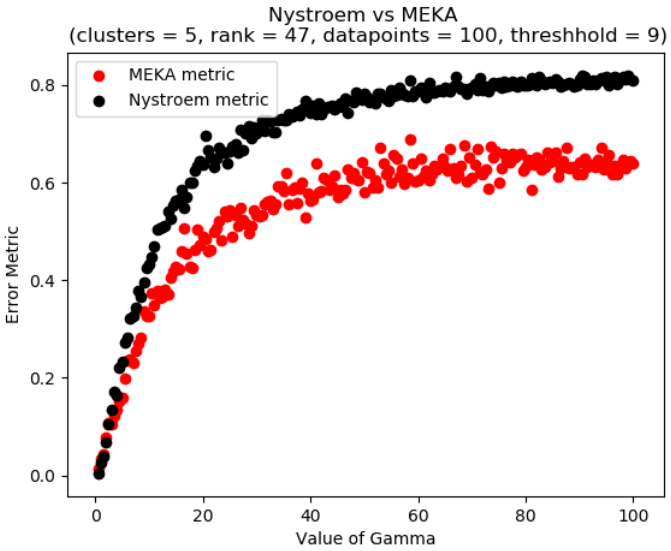

Fig. 5: Kernel Approximation under varying γ

Figure 6 performs the comparison while varying the number of clusters in MEKA.

We can see in Figure 7, that the performance of both Nystrom and MEKA improves as we allow more number of points, therefore increase the value of c.

Figure 8 shows that we can allow Nystrom to perform as good as MEKA if we increase the allowable rank in k-rank approximation. MEKA, as we will discuss later is able to perform better since it approximates the block matrices with higher rank under same error and time constraints.

MEKA uses the distance between the cluster centers to decide whether two clusters are sufficiently identical or not. If the cluster centers are far apart, the off-diagonal block corresponding to them is not computed. Therefore, a threshold is used to decide this. Taking threshold = 0 would mean we

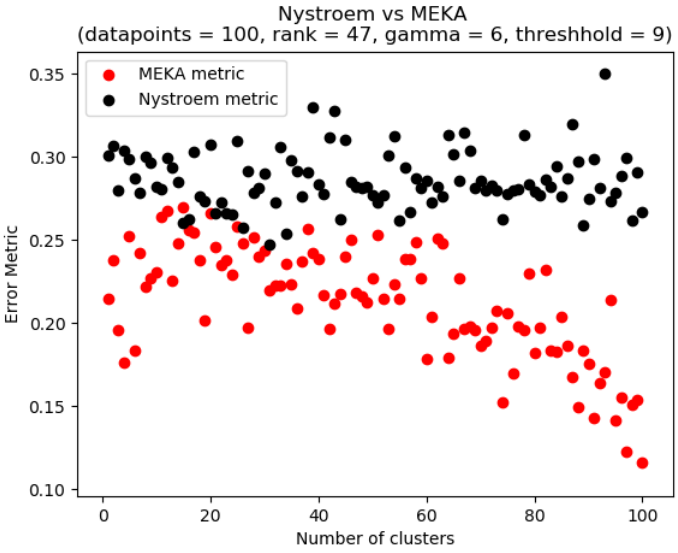

Fig. 6: Kernel Approximation under varying no of clusters for MEKA

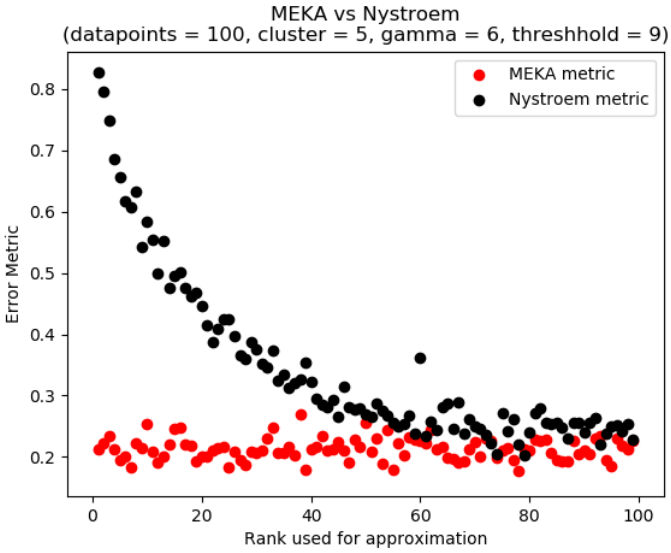

Fig. 8: Kernel Approximation under varying rank for Kernel approximation in Nystrom

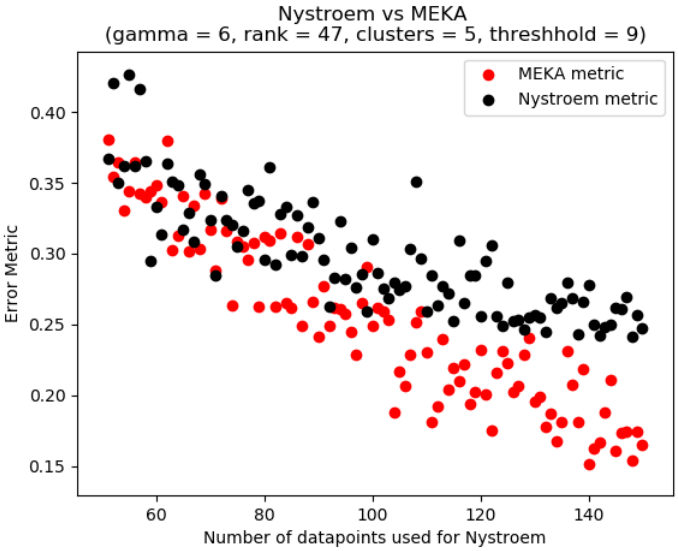

Fig. 7: Kernel Approximation under varying no of datapoints for Nystrom

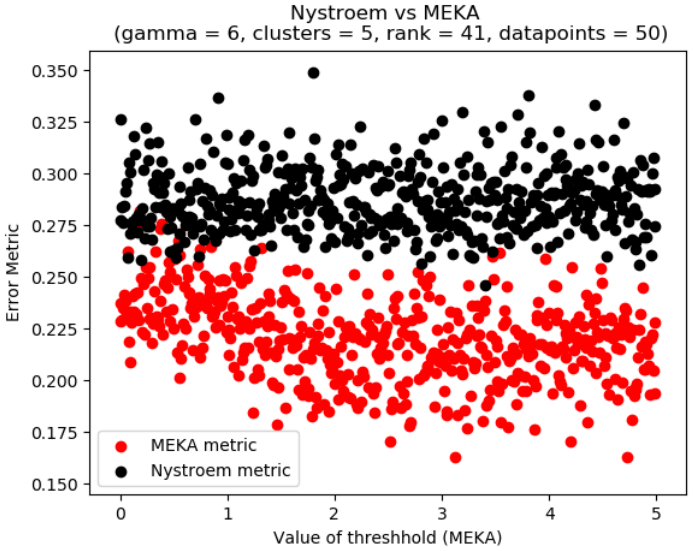

Fig. 9: Kernel Approximation under varying thresh-hold for MEKA

perform BKA. Figure 9 shows the variation in performance of MEKA as threshold is varied. As threhsold increases, more and more off-diagonal blocks are computed, hence the error reduces. In order to choose the range of threshold values, we used the observed pairwise cluster distances.

VI. Discussion

We also tried to explore various rank options as given in MEKA. As one would expect, the performance of MEKA is highly sensitive to the rank chosen . Finally, unlike the format reported in paper, we have chosen k as a linear function of the cluster size.

Althogh, MEKA is an approximated version of Nystrom, it still outperforms it. The main reason is that in case of MEKA, we can use a higher k(where k is the rank of approximation), while satisfying same error and time constraints. Therefore, we get a better approximation in MEKA.

A. Proposal for further work

It may be noted that the Nystrom method picks with high probability only those points that are far from the origin. But, intuitively, the purpose of the approximation method is to select the appropriate points that represent the dataset. So, taking inspiration from the k-means clustering employed in BKA, we wish to propose that the cluster centers may be used as the representatives for the dataset. Another issue is that for kernels like gaussian kernel, the diagonal values are exactly same in the Gram matrix, and hence selection of points

becomes completely random. Using clustering will solve this issue as well.

The probability distribution given in the paper, however, helps to prove bounds on the error in Nystrom. However, for practical purposes other methods should also be explored, as they might give better empirical performance.

We can also use a Markov model type sampling method in Nystrom which will ensure that the knowledge of proviously sampled datapoints is also put to use.

Another issue with MEKA and BKA is that we should know the number of clusters in our dataset. In online settings, we can use non-paramteric methods to decide the number of clusters. As each point comes in, we can decide whether or not to create a new cluster.

REFERENCES

[Drineas et al., 2006] Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157.

[Drineas and Mahoney, 2005] Drineas, P. and Mahoney, M. W. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175.

[Hamid et al., 2014] Hamid, R., Xiao, Y., Gittens, A., and DeCoste, D. (2014). Compact random feature maps. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 19–27. JMLR.org.

[Kar and Karnick, 2012] Kar, P. and Karnick, H. (2012). Random feature maps for dot product kernels. In Lawrence, N. D. and Girolami, M. A., editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, Spain, April 21-23, 2012*, volume 22 of *JMLR Proceedings*, pages 583–591. JMLR.org.

[Si et al., 2017] Si, S., Hsieh, C.-J., and Dhillon, I. S. (2017). Memory efficient kernel approximation. *J. Mach. Learn. Res.*, 18(1):682–713.

[Yang et al., 2014] Yang, J., Sindhwani, V., Avron, H., and Mahoney, M. W. (2014). Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 485–493. JMLR.org.

APPENDIX A
ANALYSIS OF NYSTROM METHOD

A. review: approximate matrix multiplication

The work described in this section is from [Drineas et al., 2006].

ALGORITHM: Approximate matrix multiplication

- INPUT: $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, positive integer c, probabilities $\{p_i\}_{i=1}^n$
- OUTPUT: matrices C and R s.t. $CR \approx AB$

1) for $t \in 1,\dots,c$:
   - pick $i_t \in \{1,\dots,n\}$ with $\Pr[i_t = k] = p_k$, (i.i.d., with replacement)
   - Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)}/\sqrt{cp_{i_t}}$
2) return C and R

Theorem 1. *Let $A \in \mathbb{R}^{m\times n}$, c is a positive integer and $\{p_i\}_{i=1}^n$ are s.t.:*

$$p_i = \frac{\left\|A^{(k)}\right\|_2^2}{\|A\|_F^2}$$

Then if the matrix C is constructed using above algorithm to approximate matrix multiplication $AA^\top \approx CC^\top$ then:

1) $\mathbb{E}\left[\left\|AA^\top - CC^\top\right\|_F\right] \le \frac{1}{\sqrt{c}}\|A\|_F^2$
2) $\Pr\left[\left\|AA^\top - CC^\top\right\|_F \ge \frac{\eta}{\sqrt{c}}\|A\|_F^2\right] \le \delta$ *where* $\delta \in (0,1)$ *and* $\eta = 1 + \sqrt{8\log(1/\delta)}$

We will now prove an extension of this theorem.

Lemma 2.

$$\mathbb{E}[\left\|AA^\top AA^\top - CC^\top CC^\top\right\|_F] \le \frac{2}{\sqrt{c}}\|A\|_F^4$$

$$\Pr[\left\|AA^\top AA^\top - CC^\top CC^\top\right\|_F > \frac{2\eta}{\sqrt{c}}\|A\|_F^4] < \delta$$

where $\delta \in (0,1)$ and $\eta = 1 + \sqrt{8\log(1/\delta)}$.

proof: Note that

$$AA^\top AA^\top - CC^\top CC^\top$$
$$= AA^\top(AA^\top - CC^\top) + (AA^\top - CC^\top)CC^\top$$

By submultiplicitivity and subadditivity, we get:

$$\left\|AA^\top AA^\top - CC^\top CC^\top\right\|$$
$$(\le \|A\|_F^2 + \|C\|_F^2)\left\|AA^\top - CC^\top\right\|$$
$$= 2\|A\|_F^2\left\|AA^\top - CC^\top\right\| \text{ (since the norms of C and A are equal)}$$

The lemma follows from the application of theorem 1. □

B. review: approximate SVD

ALGORITHM: Approximate SVD

- INPUT: $A \in \mathbb{R}^{m\times n}$, integers c, k $(1 \le k \le c \le n)$, probability distribution $\{p_i\}_{i=1}^n$
- OUTPUT: orthogonal matrix $H_k \in \mathbb{R}^{m\times k}$ and numbers $\sigma_t(C)$ $(t = 1,\dots,k)$

1) for $t \in \{1,\dots,c\}$:
   - pick $i_t \in \{1,\dots,n\}$ $\Pr[i_t = \alpha] = p_\alpha$
   - set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$
2) compute $C^\top C$ andt its SVD $C^\top C = \sum_{t=1}^c \sigma_t^2(C) y^t (y^t)^\top$
3) compute $h^t = Cy^t/\sigma_t(C)$ for $t = 1,\dots,k$
4) return H_k where $H_k^{(t)} = h^t$ and $\sigma_t(C)$, $t = 1,\dots,k$

Theorem 3. *Let $A \in \mathbb{R}^{m\times n}$ and let H_k and C be the matrices constructed in the above algorithm. Then:*

$$\|A - H_k H_k A\|_F^2 \le \|A - A_k\|_F^2 + 2\sqrt{k}\left\|AA^\top - CC^\top\right\|_F$$

C. Analysis: main nystrom method based kernel approximation

Before proceeding to the main theorem, we will first introduce some notations and state some lemmas which will be used to prove the main theorem.

Notation 4.

$$G = X^\top X$$
$$C = GSD$$
$$C_X = XSD$$
$$C_X = \hat{U}\hat{\Sigma}\hat{V}^\top (\textit{ singular value decomposition})$$
$$\hat{U}_k = \textit{ first k columns of } \hat{U}$$
$$\tilde{G}_k = CW_k^\dagger C^\top$$
$$W = C_X^\top C_X$$

Lemma 5.

$$\left\|G - \tilde{G}_k\right\| = \left\|X^\top X - X^\top \hat{U}_k \hat{U}_k^\top X\right\|$$

Lemma 6.

$$\left\|X^\top X - X^\top \hat{U}_k \hat{U}_k^\top X\right\|_F^2$$
$$= \left\|X^\top X\right\|_F^2 - 2\left\|XX^\top \hat{U}_k\right\|_F^2 + \left\|\hat{U}_k^\top XX^\top \hat{U}_k\right\|_F^2$$

Lemma 7.

$$\left|\left\|XX^\top \hat{U}_k\right\|_F^2 - \sum_{t=1}^{k}\sigma_t^4(C_X)\right|$$
$$\leq \sqrt{k}\left\|XX^\top XX^\top - C_X C_X^\top C_X C_X^\top\right\|_F$$

Lemma 8.

$$\left\|\hat{U}_k^\top XX^\top \hat{U}_k\right\|_F^2 - \sum_{t=1}^{k}\sigma_t^4(C_X)$$
$$\leq \sqrt{k}\|k\|\left\|XX^\top XX^\top - C_X C_X^\top C_X C_X^\top\right\|_F$$

Lemma 9.

$$\left|\sum_{t=1}^{k}\sigma_t^4(C_X) - \sigma_t^2(X^\top X)\right|$$
$$\leq \sqrt{k}\left\|XX^\top XX^\top - C_X C_X^\top C_X C_X^\top\right\|_F$$

Now we will state and prove the main theorem.

Theorem 10.

$$\mathbb{E}[\left\|G - \tilde{G}_k\right\|_F] \leq \|G - G_k\| + \epsilon\sum_{i=1}^{n} G_{i,i}^2$$
$$Pr[\left\|G - \tilde{G}_k\right\|_F > \|G - G_k\| + \epsilon\sum_{i=1}^{n} G_{i,i}^2] \leq \delta$$

Proof: Define $E = XX^\top XX^\top - C_X C_X^\top C_X C_X^\top$. Then we have:

$$\left\|G - \tilde{G}_k\right\|_F^2$$
$$= \left\|X^\top X\right\|_F^2 - 2\left\|XX^\top \hat{U}_k\right\|_F^2 + \left\|\hat{U}_k^\top XX^\top \hat{U}_k\right\|_F^2$$
$$\leq \left\|X^\top X\right\|_F^2 - \sum_{t=1}^{k}\sigma_t^4(C_X) + 3\sqrt{k}\|E\|_F$$
$$\leq \left\|X^\top X\right\|_F^2 - \sum_{t=1}^{k}\sigma_t^2(X^\top X) + 4\sqrt{k}\|E\|_F$$

The first statement follows from lemma 5 and 6. The second statement follows from 7 and 8. The 3rd statement follows from 9. Since $\left\|X^\top X\right\|_F^2 - \sum_{t=1}^{k}\sigma_t^2(X^\top X) = \|G - G_k\|_F^2$, we get that:

$$\left\|G - \tilde{G}_k\right\|_F^2 \leq \|G - G_k\|_F^2 + 4\sqrt{k}\left\|XX^\top XX^\top - C_X C_X^\top C_X C_X^\top\right\|$$

This can be combined with theorem 1 and lemma 2 along with Jensen's inequality to get theorem 10.

APPENDIX B
ANALYSIS OF BLOCK KERNEL APPROXIMATION

Notation 11.

$$k(\mathbf{x}, \mathbf{y}) = f(\eta(\mathbf{x} - \mathbf{y}))$$
$$g_{\mathbf{u}}(t) = f(\eta t\mathbf{u})$$
$$D_{kernel}(\{\mathcal{N}_s\}_{s=1}^c) = \sum_{s=1}^{c}\frac{1}{|\mathcal{N}_s|}\sum_{i,j\in\mathcal{N}_s} k(x^i, x^j)^2$$

Theorem 12. *For shift-invariant kernels:*

$$D_{kernel}(\{\mathcal{N}_s\}_{s=1}^c) \geq \bar{C} - \eta^2 R^2 D_{kmeans}(\{\mathcal{N}_s\}_{s=1}^c)$$

where $\bar{C} = \frac{nf(0)^2}{2}$, *R is a constant depending on the kernel function, and* $D_{kmeans}(\{\mathcal{N}_s\}_{s=1}^c) = \sum_{s=1}^{c}\sum_{i\in\mathcal{N}_s}\left\|\mathbf{x}^{(i)} - \mathbf{m}^{(s)}\right\|$, $m^{(s)}$ *are the cluster centers.*

Proof: By mean value theorem we have:

$$k(x^i, x^j) = g_u(\eta\left\|x^i - x^j\right\|_2) = g_u(0) + \eta g'(t)\left\|x^i - x^j\right\|_2$$
$$\implies f(0) \leq k(s^i, x^j) + \eta R\left\|x^i - x^j\right\|_2$$
$$\text{where } R = \max_t |g'(t)|$$

Squaring both sides and applying AM-GM inequality we get:

$$\frac{f(0)^2}{2} \leq k(x^i, x^j)^2 + \eta^2 R^2\left\|x^i - x^j\right\|_2^2$$

Plugging into the definition of D_{kernel} we get:

$$D_{kernel}(\{\mathcal{N}_s\}_{s=1}^c) \geq \frac{nf(0)^2}{2} - \eta^2 R^2 D_{kmeans}(\{\mathcal{N}_s\}_{s=1}^c)$$

□