Kernel Approximation

Aniket Anand Deshmukh
Department of EECS
University of Michigan
Ann Arbor, MI 48105
aniketde@umich.edu

Abstract

In this project, I explore the kernel approximation techniques based on randomization. In the classification tasks when data is not linearly separable in the current feature space, kernel methods project input data points into high dimensional feature space (infinite-dimensional in cases of kernels like Gaussian kernel) and find the optimal hyperplane in that feature space. But if number of data points are large, size of the kernel matrix is going to be large and it could affect the efficiency of the algorithms. Using kernel approximation technique such as random Fourier features, random feature maps, Nystrom approximation, data points are projected into some approximated low dimensional space (compared to infinite or very high dimensional space in kernel trick). I explore some of these kernel approximation techniques and compare their results.

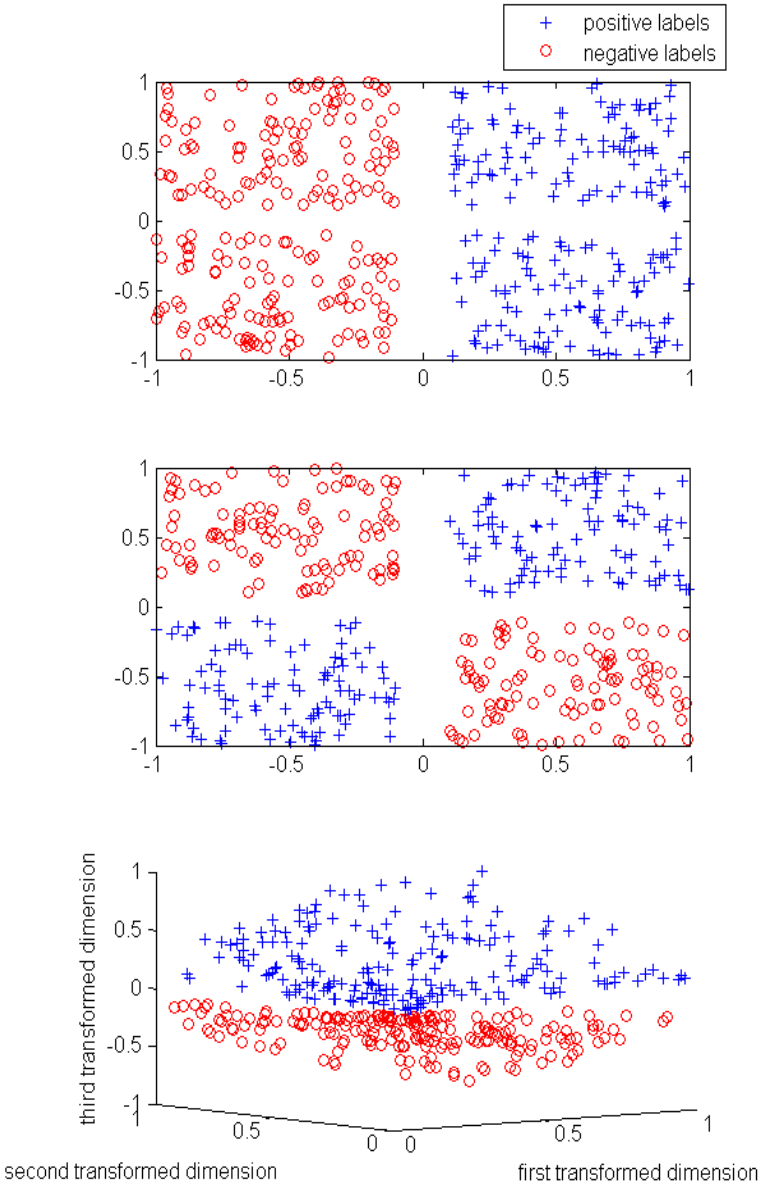
1 Introduction

Ideas like reproducing kernel Hilbert space (RKHS) have been used extensively in statistical learning theory from last few decades. In the context of binary classification, When the data is not linearly separable in the current space, non linear feature mapping is used to map the data points into high dimensional space (infinite dimensional in cases of kernels like Gaussian) and find the optimal hyperplane in that feature space. For example let's look at Fig. 1. In the topmost plot we have 2 dimensional data which is linearly separable. In the middle plot we again have 2 dimensional data but in this case, it's not linearly separable. So we transform the data into 3 dimensional space using transformation $(x_1, x_2) \rightarrow (x_1^2, x_2^2, x_1x_2)$. Resulting 3 dimensional data is shown in the bottom most plot in Fig. 1. As one can notice easily, transformed data is now linearly separable.

Transformations like $(x_1, x_2) \rightarrow (x_1^2, x_2^2, x_1x_2)$ may blow up the number of dimensions and can be impractical to use. Kernel trick allows us to transform the data in high dimensional (potentially infinite) using inner products, without actually using the non linear feature mapping. For example Let's say we have a transformation $x \rightarrow \phi_x$. Using kernel trick we can write $k(x, y) = \langle \phi_x, \phi_y \rangle$, where k is appropriate kernel [6]. In this case we don't even need to know the non linear feature mapping ϕ and we can easily map our input data to potentially infinite dimensional space.

Though we may not need to calculate the ϕ, this reduces our ability to play with the data directly, because to be able to use kernel trick we need all the relations between input data points in terms of it's inner products. Given n sample points, we need to calculate the $n \times n$ kernel matrix. The resultant complexities in terms of both space (quadratic) and time (usually cubic) can be quite demanding for large problems, posing a big challenge on practical applications. So given a kernel $k(x, y)$ can we find approximate feature mapping

Figure 1: Kernel Trick



such that $k(x, y) \approx z(x)^T z(y)$? Idea is to map the data in high dimension space using explicit feature mapping $z : R^d \rightarrow R^D$. If we can do this, we can solve two problems: 1) We do not blow up the feature space as in just using non linear feature mapping, and 2) We are not constrained to use non linear feature mapping only through it's inner products.

This problem has been addressed in variety of settings. Ali Rahimi proposed a random feature maps for shift invariant kernels (like Gaussian exp $\frac{\|x-y\|^2}{2\sigma^2}$) [8] using Theorem 1 and then this work has been extended by Yang using Quasi random features [13]. For dot product kernels (of the form $f(\langle x, y\rangle)$, polynomial kernels are special case with $(\langle x, y\rangle)^p$, where p is a positive integer), Kar proposed a random Maclaurin feature maps [4] and then Pham proposed a new technique using tensor sketching [7]. Hamid extended this work by proposing more compact feature map using random projections [3]. There is one more general family of kernel approximation technique called Nystrom approximation. This method applies to any kernels (which satisfies all the properties of kernel) and uses low rank matrix approximation technique [12, 2]. Later there have been more extensions and detailed error analysis for the Nystrom approximation [16, 14, 5, 11, 10, 9]. All the above methods come under the category of randomization and there is one more broad family where features are obtained using optimization [15]. In this project I try to implement most of these methods and compare their results. Unless otherwise stated I assume the following - $x, y \in R^d$ are any two data points from $x_1, ..., x_n$, and $k(x_i, x_j)_{i,j}$ forms $n \times n$ positive semi definite kernel matrix. Our goal is to find the feature mapping $z : R^d \rightarrow R^L$, where $L > d$, such that $k(x, y) \approx z(x)^T z(y)$.

The rest of the report is organized as follows. Section 2 gives the kernel approximation methods for shift invariant kernels (random Fourier features - RFF, quasi random Fourier features - QRFF). Section 3 describes the kernel approximation methods for dot product kernels (random Maclaurin feature maps - RMFM and compact random feature maps - CRFM). Section 4 describes broader category of Nystrom approximation and section 5 gives the results. Final section concludes the report.

2 Shift Invariant Kernels

2.1 Random Fourier Features

Random Fourier features use sine/cosine as their basis function to approximate the kernel and take advantage of Theorem 1 to approximate the kernel

Theorem 1 *A continuos kernel $k(x, y) = k(x - y)$ on R^d is positive definite iff $k(\delta)$ is the Fourier transform of a non-negative measure.*

If a shift variant kernel $k(\delta)$ is properly scaled then Theorem 1 guarantees that $p(w)$ in the 1 is a proper probability distribution.

$$k(x - y) = \int_{R^d} p(w) e^{jw^T(x-y)} dw = E_w(\zeta_w(x)\zeta_w(y)^*), \tag{1}$$

[8]. Basically random Fourier features approximate the integral in 1 using samples drawn from $p(w)$. Let's say we draw L samples $w_1, w_2, ..., w_L$ samples from $p(w)$.

$$\begin{aligned} k(x - y) &= \int_{R^d} p(w) e^{jw^T(x-y)} dw \\ &= \int_{R^d} p(w) cos(w^T x - w^T y) dw \\ &\approx \frac{1}{L} \sum_{i=1}^{L} cos(w_i^T x - w_i^T y) \\ &= \frac{1}{L} \sum_{i=1}^{L} cos(w_i^T x) cos(w_i^T y) + sin(w_i^T x) sin(w_i^T y) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{L}\sum_{i=1}^{L}[cos(w_i^T x), sin(w_i^T x)]^T[cos(w_i^T y), sin(w_i^T y)] \\
&= z(x)^T z(y)
\end{aligned}
$$

where $z(x) = \frac{1}{\sqrt{L}}[cos(w_i^T x), sin(w_i^T x)] \in R^{2L}$ is an approximate non linear feature mapping (mapped to $2L$ dimensional space).

Algorithm 1 Random Fourier Features - RFF

1: Input: A positive definte shift invariant kernel $k(x, y) = k(x - y)$
2: Compute: Fourier transform p of the kernel k: $p(w) = \frac{1}{2\pi}\int_{R^d} e^{-jw^T\delta}k(\delta)d\Delta$
3: Draw L iid $w_1, w_2, ..., w_L \in R^d$ samples from p
4: Return: $z(x) = \frac{1}{\sqrt{L}}[cos(w_i^T x), sin(w_i^T x)]_{i=1}^{L} \in R^{2L}$

2.2 Quasi Random Fourier Features

Quasi random Fourier features extend the idea of random Fourier features using Quasi-Monte-Carlo (QMC) techniques. In the last section we sample from $p(w)$ using Monte Carlo (MC) technique. Let's discuss MC technique in more detail so that we can introduce the idea of QMC [13].

Let's consider the calculating following integral

$$I_d[f] = \int_{[0,1]^d} f(x)dx \tag{2}$$

If x is a random vector uniformly distributed over $[0, 1]^d$, then $I_d[f] = E[f(x)]$. An empirical approximation of this expectation can be computed by drawing random samples from uniform distribution $[0, 1]^d$ independently. Let $w_1, w_2, ..., w_L$ are L independent samples from uniform distribution $[0, 1]^d$. So approximation to above integral is

$$I_d[f] \approx I_w[f] = \frac{1}{L}\sum_{i=1}^{L} f(w) \tag{3}$$

This is the MC method. Now let's define the integration error:

$$\epsilon_w(f) = |I_d[f] - I_w[f]| \tag{4}$$

when w are drawn randomly, the central limit theorem asserts that is L is large enough then $\epsilon_w(f) \approx \sigma(f)L^{-\frac{1}{2}}v$, where v is a standard normal random variable, and $\sigma(f)$ is the square root of the variance of f.

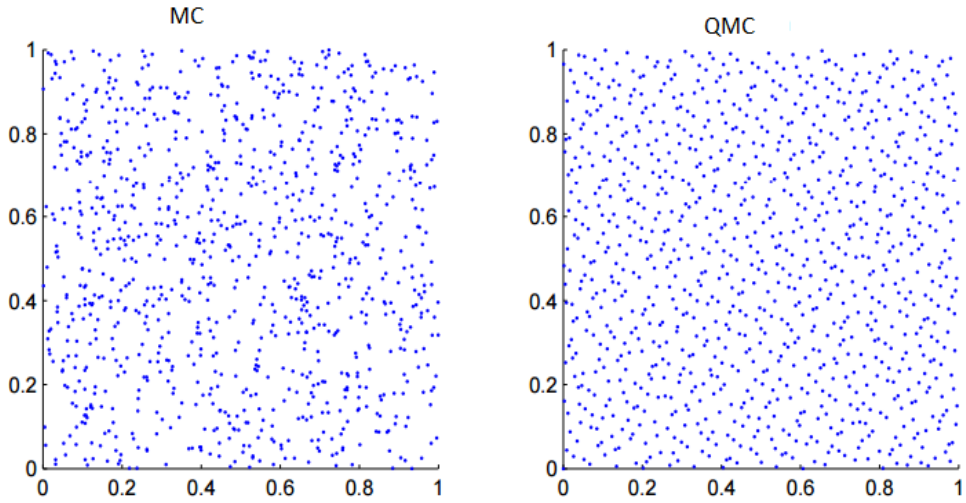
$$\sigma^2(f) = \int_{[0,1]^d} (f(x) - I_d(f))^2 dx \tag{5}$$

In other words, the root mean square error of the Monte Carlo method is

$$(E_w[\epsilon_w(f)^2])^{1/2} \approx \sigma(f)s^{-1/2} \tag{6}$$

Therefore, the Monte Carlo method converges at a rate of $O(s^{-1/2})$. The aim of the QMC methods is to improve this convergence rate by defining deterministic low discrepancy sequence to construct $w_1, w_2, ..., w_L$, instead of randomly sampling points. The intuition behind QMC is explained through the figure 2. In the case of MC random sampling may lead to a clustered data points (even though it should form a uniform distribution in 2 dimensions) but QMC method gives better realizations of uniform distribution. This is backed by Koksma-Hlawka inequality [13] which along with such low discrepancy sequence guarantees the faster convergence rate.

Figure 2: QMC vs MC, figure from [13]



Comparison of MC and QMC sequences.

Now, assume that $p(w)$ can be written in as $p(w) = \Pi_{j=1}^{d} p_j(x_j)$, where $p_j(\cdot)$ is a univariate density function. The density functions associated to kernels like Gaussian, Cauchy ets admit such a form. As QMC method is applicable to integrals over unit cube, integrals such as in Equation 1 are computed by first generating low discrepancy sequence $t_1, t_2, ..., t_L$ (refer [1] for more details) and transforming it into a sequence $w_1, w_2, ..., w_L$ using CDF $\phi_j(\cdot)$ of $p_j(\cdot)$ (by setting $w = \phi^{-1}(t)$). I use publicly available low discrepancy sequence generator - http://people.cs.kuleuven.be/ dirk.nuyens/qmc-generators/.

Algorithm 2 Quasi Random Fourier Features - QRFF

1: Input: A positive definte shift invariant kernel $k(x, y) = k(x - y)$
2: Compute: Fourier transform p of the kernel k: $p(w) = \frac{1}{2\pi} \int_{R^d} e^{-jw^T\delta} k(\delta) d\Delta$
3: Generate: low discrepancy sequence $t_1, t_2, ..., t_L$
4: Compute: $w_1, w_2, ..., w_L \in R^d$ samples using $w = \phi^{-1}(t)$)
5: Return: $z(x) = \frac{1}{\sqrt{L}}[cos(w_i^T x), sin(w_i^T x)]_{i=1}^{L} \in R^{2L}$

3 Dot Product kernels - Polynomial Kernels

3.1 Random Maclaurin Feature Maps

Random feature maps for polynomial kernels (or any dot product kernel) use classical result in harmonic analysis (Maclaurin expansion). Maclaurin series of $f(x) = \exp(x)$

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

Theorem 2 *A function $f[-1, 1] \rightarrow R$ constituents a positive definite kernel $K : S_\infty \times S_\infty \rightarrow R, K : (x, y) \rightarrow f(\langle x, y \rangle)$ iff f is an analytic function admitting a Maclaurin expansion with only non-negative coefficients ie. $f(x) = \sum_{n=0}^{\infty} a_n x^n, a_n \geq 0, n = 0, 1, 2, ...$ Here $S_\infty = \{x \in H : \|x\|_2 = 1\}$ for some hilbert space H [4].*

Lemma 1 *Let $w \in R^d$ be a vector each of whose coordinates have been chosen pairwise indepedently using a fair coin tosses from the set $\{-1, 1\}$ and consider the feature map $z : R^d \to R$, $z : x \to w^T x$. Then for all $x, y \in R^d$, $E_w[z(x)z(y)] = \langle x, y \rangle$.*

Proof is straightforward and details can be seen in [4]. Using the Theorem 2 and Lemma 1, we can find the real valued explicit feature map. First, randomly pick a number $N \in \mathbb{N} \cup \{0\}$ with $\mathbb{P}[N = n] = \frac{1}{p^{n+1}}$. After doing this, we pick N independent Rademacher vectors $w_1, w_2, ..., w_N$ and output the feature map $z : R^d \to R$, $z : x \to \sqrt{a_N p^{N+1}} \Pi_{j=1}^{N} w_j^T x$. Note here that a_N comes from a Maclaurin expansion of $f(x)$ which defines the dot product kernel. For example for polynomial kernel $(\langle x, y \rangle)^m$, $f(x) = x^m$.

Lemma 2 *Let $z : R^d \to R$ be the feature map constructed above. Then for all x, y, we have $E[(z(x)z(y)] = K(x, y)$, where the expectation is over the choice of the Rademacher vectors.*

Algorithm 3 Random Maclaurin Feature Maps - RMFM

1: Input: A positive definte dot product kernel $k(x, y) = f(\langle x, y \rangle)$
2: Compute: Maclaurin expansion of $f(x) = \sum_{n=0}^{\infty} a_n x^n$ by setting $a_n = \frac{f^{(n)}(0)}{n!}$
3: Fix a value of $p > 1$
4: **for** i = 1 to L **do**
5: Choose $N \in \mathbb{N} \cup \{0\}$ with $\mathbb{P}[N = n] = \frac{1}{p^{n+1}}$
6: Choose N vectors $w_1, w_2, ..., w_N \in \{-1, 1\}^d$ selecting each co-ordinate using fair coin tosses.
7: Let feature map $z_i : x \to \sqrt{a_N p^{N+1}} \Pi_{j=1}^{N} w_j^T x$
8: **end for**
9: Return: $z : x \to \frac{1}{L}(z_1(x), ..., z_L(x))$

3.2 Compact Random Feature Maps

The above approach or previous approaches for polynomial kernel approximation create maps that are rank deficient and therefor do not completely utilize the capacity of projected feature space. To address this challenge [3] came up with the compact random feature maps (CRFM) to approximate polynomial kernels more accurately and compactly. The basic crux of the method is to project the data into very high dimensional space (i.e. high L) using techniques like RMFM and then down project using Johnson-Lindenstrauss random projections.

Algorithm 4 Compact Random Feature Maps - CRFM

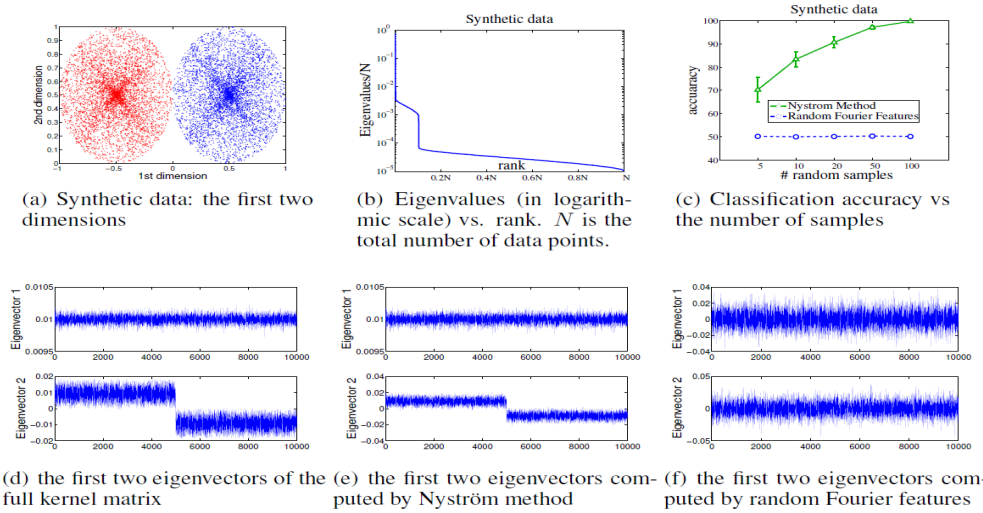
1: Input: A positive definte shift invariant kernel $k(x, y) = k(x - y)$, up and down projection dimensionalities L and D such that $D < L$
2: Up Project: $z : R^d \to R^L$ using Algorithm 3
3: Down Project: $zD : R^L \to R^D$ using Johnson-Lindenstrauss random projections.
4: Return: zD

4 General - Nystrom Approximation

In the last 2 sections, we saw the kernel approximation techniques random Fourier features for shift invariant kernels and random feature maps for the polynomial kernels. In this, we discuss Nystrom approximation which uses low rank approximation techniques and hence can be used for any type of kernel (but mostly we will restrict our analysis to shift invariant kernels). Unlike random Fourier features, which uses data independent sine/cosine basis function, for Nystrom approximation basis functions are sampled from the training examples and are therefore data dependent [12, 2]. [14] shows that that when there

is a large gap in the eigen-spectrum of the kernel matrix, approaches based on the Nystrom method can yield impressively better generalization error bound than random Fourier features based approach (figure 1). But when eigen spectrum decays slowly, Nystrom approximation will be erroneous as it depends on a low rank approximation.

Figure 3: RFF vs Nystrom comparison when eigen gap is large figure from [14]



(a) Synthetic data: the first two dimensions

(b) Eigenvalues (in logarithmic scale) vs. rank. N is the total number of data points.

(c) Classification accuracy vs the number of samples

(d) the first two eigenvectors of the full kernel matrix

(e) the first two eigenvectors computed by Nyström method

(f) the first two eigenvectors computed by random Fourier features

The Nystrom method approximates kernel matrix $k(x_i, x_j)_{i,j}$ using a small subset of its columns (m, and this method is notot only useful for low rank approximation of kernels but also reduces the time complexities of many matrix operations (like inverse, eigenvalue decomposition) from $O(n^3)$ or $O(n^2k)$ to $O(nm^2)$ and space complexities from $O(n^2)$ to $O(nm)$, where k is the target rank, m is the number of selected columns, and it holds in general that $k < m << n$. In this way, time and space costs are only linearly in m, so many kernel methods can be efficiently solved even when m is large.

Let's say you have n samples $\{x_i\}_{i=1}^n \in R^d$. Let's say we have a Gaussian kernel, so kernel matrix would be -

$$K_{ij} = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}), \forall i, j = 1, ..., n$$

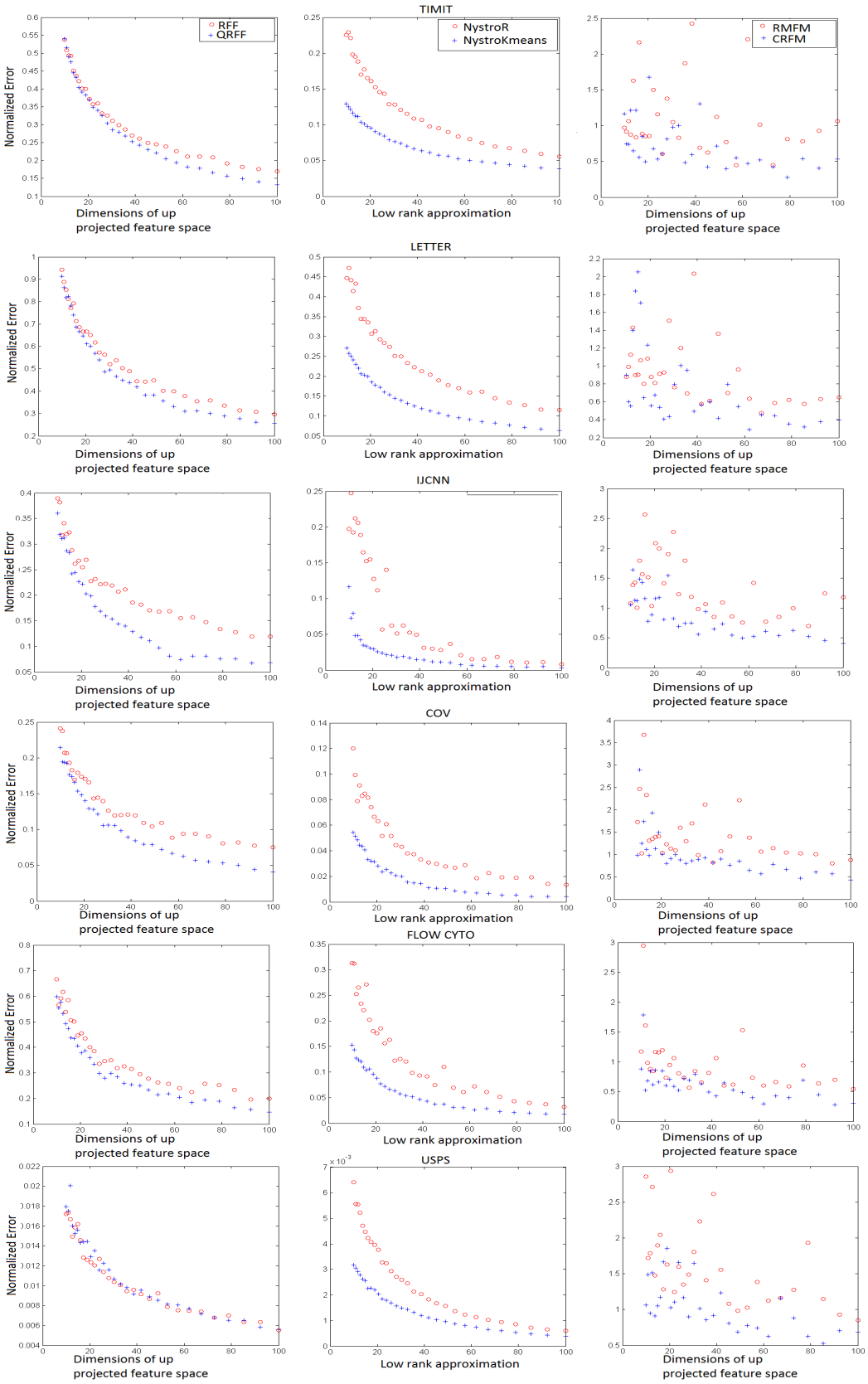
The Nystrom method approximates the K by first sampling m data points $\hat{x_1}, \hat{x_2}, ...\hat{x_m}$ and then constructing a low rank matrix by $\hat{K_r} = K_b \hat{K}^+ K_b^T$, where $K_b = [\kappa(x_i, \hat{x_j}]_{N \times m}, \hat{K} = \kappa(\hat{x_i}, \hat{x_j}]_{m \times m}$. Hence we get our final approximated feature mapping using vector representation in Equation 7, where $\hat{D}$ is eigenvalue matrix and $\hat{V}$ is corresponding eigenvector matrix.

$$z_n(x) = \hat{D}^{-\frac{1}{2}} \hat{V}^T \kappa(x, \hat{x_1}, ..., \kappa(x, \hat{x_m}) \tag{7}$$

Algorithm 5 Nystrom Approximation (Random sampling) - NystroR

1: Input: A positive definte skernel $k(x, y) = k(x-y)$, all data points $x_1, ..., x_n$, and m number of columns to sample
2: Choose m points randomly $\hat{x_1}, \hat{x_2}, ...\hat{x_m}$
3: Compute: $\hat{K} = \kappa(\hat{x_i}, \hat{x_j}]_{m \times m}$ and $K_b = [\kappa(x_i, \hat{x_j}]_{N \times m}$
4: Compute:$[V, D] = eig(\hat{K})$, where eig is the eigenvalue decomposition
5: Return: $z = K_b V D^{-\frac{1}{2}}$

Figure 4: Comparison



4.1 Improved Nystrom Methods - kmeans

In Nystrom method, we sample columns of data matrix randomly. [16] proposed Improved Nystrom Low-Rank Approximation and Error Analysis. It uses kmeans instead of random sampling and there is a good improvement in the kernel approximation. This has become a standard to compare with in future papers. There are other methods that use other sampling methods such as adaptive sampling [5].

Algorithm 6 Nystrom Approximation (Random sampling) - NystroKmeans

1: Input: A positive definte skernel $k(x, y) = k(x-y)$, all data points $x_1, ..., x_n$, and m number of columns to sample
2: Choose m points using (centers of) Kmeans of the data $\hat{x_1}, \hat{x_2}, ...\hat{x_m}$
3: Compute: $\hat{K} = \kappa(\hat{x_i}, \hat{x_j})_{m\times m}$ and $K_b = [\kappa(x_i, \hat{x_j}]_{N\times m}$
4: Compute:$[V, D] = eig(\hat{K})$, where eig is the eigenvalue decomposition
5: Return: $z = K_b V D^{-\frac{1}{2}}$

5 Results

From 4, shows the results for the 6 methods RFF, QRFF, NystorR, NystroKmeans, RMFM and CRFM. RFF is compared against QRFF, NystorR is compared against NystroKmeans and RMFM is compared against CRFM. RFF, QRFF, NystorR, and NystroKmeans are implemented for Gaussian kernel and bandwidth is set according to mean of all distances of points from the mean of the population. RMFM and CRFM are implemented for the polynomial of degree 2 i.e. $(\langle x, y\rangle)^2$. Results are obtained for the following 6 standard datasets - flow cytometry, cov, ijcnn, letter, timit and usps. Flow cytometry has 1000 data points and it's 6 dimensional. Cov data has 2000 data points and it's 52 dimensional. Ijcnn has 1000 data points and it's 22 dimensional. Letter has 1000 data points and it's 16 dimensional. Timit has 1000 data points and it's 39 dimensional and finally USPS has 400 data points and it's 256 dimensional (data is normalized in unit hypercube).

To get the error, accurate kernel matrix is calculated and then normalized error for each of the above approximations are calculated. For RFF and QRFF dimensionality L is varied, for NystorR and NystroKmeans low rank approximation m is varied and finally for RMFM and CRFM dimensionality both L, D are varied (For the purpose of experiments $L = 4D$). Results are plotted against L, m, and D respectively.

6 Conclusions

In this project, I studied few randomization techniques for the kernel approximation. I implemented some of these techniques and compared the results. As one can see QRFF performs better than RFF, NystroKmeans performs better than NystorR and CRFM performs better than RMFM almost all the time. Also with increasing L, m, and D, we get better approximations for the kernel. Plots of RMFM and CRFM look little noisy and it is because these techniques use random sampling. Though I averaged results over few runs (5 in the current experimental setup) it has a huge variance. To get better results we may need to run it for few more times. The results shown here are in tune with the results in the actual papers [13, 3, 16].

References

[1] J. Dick, F. Y. Kuo, and I. H. Sloan. High-dimensional integration: the quasi-monte carlo way. *Acta Numerica*, 22:133–288, 2013.

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

[3] R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact random feature maps. *arXiv preprint arXiv:1312.4626*, 2013.

[4] P. Kar and H. Karnick. Random feature maps for dot product kernels. *arXiv preprint arXiv:1201.6530*, 2012.

[5] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[6] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. 2012.

[7] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

[8] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[9] S. Si, C.-J. Hsieh, and I. Dhillon. Memory efficient kernel approximation. In *Proceedings of The 31st International Conference on Machine Learning*, pages 701–709, 2014.

[10] S. Wang, C. Zhang, H. Qian, and Z. Zhang. Improving the modified nyström method using spectral shifting. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–620. ACM, 2014.

[11] S. Wang and Z. Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[12] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161322, pages 682–688, 2001.

[13] J. Yang, V. Sindhwani, H. Avron, and M. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. *arXiv preprint arXiv:1412.8293*, 2014.

[14] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.

[15] F. X. Yu, S. Kumar, H. Rowley, and S.-F. Chang. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*, 2015.

[16] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.