Matrix Approximation Algorithms and Its Applications

A dissertation submitted in partial fulfillment of the requirement for the degree of
Master of Technology
In

Computer Science &Engineering

under the Supervision of

Mr. Suman Saha

By

Lokendra Singh Patel

Enrollment no: 132213

Jaypee University of Information Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that dissertation report entitled **"Matrix Approximation Algorithms and Its Application"**, submitted by **Lokendra Singh Patel** in partial fulfillment for the award of degree of Master of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Supervisor's Name…………………**

Designation…………………………

Acknowledgement

This may seen long but the task of my thesis work both theoretically and practically may not have been completed without the help, guidance and mental support of the following person. Firstly I would like to thank my guide **Mr. Suman Saha**, Assistant Professor, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan(H.P), who provided me the related material and idea for the project proposal. He indeed guided me to do the task for my thesis in such a way that it seems to be research work encouraged me a lot for doing my thesis in very smooth manner. Even if I made the mistake sometime he always tried to correct those mistakes and endeavor always to take me in the right direction.

Secondly, I would like to thank **my parents** who have always been with me for inspiring me that I can do the good thesis task with the hard work. Their instigation always helped me to grow my mind focused towards the hard work for implementation of thesis with having the research work in the mind.

Once again thanks a ton to all mentioned people in my life.

Date ………………………… **Signature………………………**

Name……………………………

Table of Contents

List of tables

List of figures

Abstract

Large data sets have tens of thousands to millions of training instances, which suffers from high time and space complexity. To reduce the time and space complexity, we propose efficient Nystrom method to approximate kernel matrix, which is used in many machine learning methods such as kernel-based methods, e.g. Kernel Ridge Regression, Kernel Principle Component Analysis and Support Vector Machine. This thesis focuses on sampling based matrix approximation methods. Matrix approximation will help to speed up the kernel based algorithms to large data set. We give the desirable error bound both in the Frobenius and spectral norm for the quality of approximation. Based on these error bounds, we analyze the quality of approximation in kernel based algorithms. We present guarantees on approximation accuracy based on various matrix properties and analyze the effect of matrix approximation on actual kernel-based algorithms. Our proposed algorithm gives the lower error bound for the low rank approximation of the kernel matrix.

Chapter 1

Introduction

In this chapter, motivation, objective and problem statement briefly explained. This chapter presented overview of the thesis and presented the organization of the thesis.

1.1 Motivation

In machine learning, there is a problem with large scale data. So the problem with large scale data is storage and time complexity. Due to high complexity we have to find the efficient approximation of a matrix. That matrix be a kernel matrix, which is use in support vector machine, kernel principle component analysis and kernel ridge regression. Large kernel matrix having millions of entries, such large data sets creates problem both in storing and operating. So we have to find the efficient solution for kernel matrix to speed up the kernel methods. We find the good approximation of kernel matrix using the efficient Nystrom method. Efficient Nystrom method generates the low rank approximation of matrix. Sampling based approximation method select the subset of columns and using the subset of columns generate the efficient solution of the problem.

Suppose having symmetric positive semidefinite matrix A, n is very large. So it requires $O(n^2)$ space complexity to store the kernel matrix. And operating the kernel matrix, it requires $O(n^3)$ finding the singular value decomposition (SVD) of matrix. Suppose B is the approximate matrix with rank k, so we have to minimize the $\|A - B\|$ in terms of Frobenius norm and spectral norm respectively $\|W\|_F$ and $\|W\|_2$ for any matrix W. Rank of B much smaller than n. We have to find the orthogonal vector and diagonal matrix using Eigen value and Eigen vector. In singular value decomposition, we have to decompose the matrix in right orthogonal vector, left orthogonal vector and diagonal matrix. Previously, we have various without sampling approximation method such as truncated SVD but without sampling based method takes more time as compare the sampling based methods. Sampling based approximation leads to inaccuracy as compare truncated SVD. Nystrom method was use for numerical integration of quadrature method. In 2000 William and Seeger introduced Nystrom method for kernel methods to reduce the time and space complexity. Matrix approximation is very useful in large data set. For large data set, kernel methods take large amount of time to solve the problems of machine learning.

In this thesis, we look the efficient solution of this problem that generates efficient matrix approximation of kernel methods. We introduced sampling based efficient Nystrom method.

1.2 Objective

The main objective of matrix approximation is to speed up the kernel methods. We have to generate the low rank of the given matrix A. Given matrix A with rank n, we have to find the good approximation of A and the rank of matrix A is less than n. Matrix A has good spectral feature that helps in finding the good approximation. Approximate matrix is able to speed up the kernel methods such as support vector machine, kernel ridge regression, kernel principle component analysis. But sampling based approximation suffers from inaccuracy so we have to minimize the approximation error. Approximation error calculated using Frobenius norm and Spectral norm. Measure of approximation is based on quality of error bound. We have to generate both theoretical and experimental error bound using Frobenius norm and Spectral Norm. We have to choose columns using good sampling methods. And that columns help to generate good approximation. We have various sampling based such as Nystrom methods and column sampling methods to approximate the matrix. For better approximation this thesis deals with following objectives:

1 For a given matrix, we have to develop low rank matrix approximation based on sampling methods.
2 We have to select the columns for approximation based on efficient sampling methods. Efficient Nystrom method uses these sampled columns to generate the efficient matrix approximation.
3 Using the Frobenius norm and spectral norm, we will show the quality of matrix approximation.
4 Here, we show that how approximation works with the kernel methods such as support vector machine.

1.3 Problem Statement

To found the accuracy of the algorithms in terms of error bound. Suppose we have a matrix A ∈ $R^{n \times m}$ be a matrix and $A_k$is the best k rank approximation.

$$\|A - A_k\|_2 = \delta \tag{1.1}$$

$$\|A - A_k\|_F = \delta \tag{1.2}$$

Where $\|.\|_2$ represents the spectral norm and $\|.\|_F$ represents the Frobenius norm of a matrix.

δ is the represents a tolerable level of error for the given application. We have to minimize the tolerable error δ.

1.4 Thesis Organization

In this thesis, there are seven chapters. In the first chapter, we discuss about the motivation, objective and problem statement. In the second chapter, we discuss review of linear algebra because concepts of linear algebra uses in the low rank approximation. Chapter 3 presents the literature survey of the matrix approximation. Various low rank approximation methods are discussed in chapter 3. In the chapter 4, we discuss the application of the low rank approximation. This chapter tells hoe the low rank approximation works. Chapter 5 presents the proposed approach and the algorithm. Chapter 6 describes the implementation results of the standard Nystrom method and the efficient Nystrom method and also discusses results of the application of the low rank approximation. Chapter 7 provides the conclusion and outline the most promising directions for future work.

Chapter 2

Review of linear algebra

In this chapter, we have to discuss about basics of linear algebra. Suppose X be an $m \times n$ matrix, entries of matrix are non-negative. Independent rows (or columns) are the rank of the matrix X; thus, $rank(X) \leq \min\{m, n\}$. A square $n \times n$ matrix all of whose off-diagonal elements are zero is called a diagonal matrix; rank of diagonal matrix is equal to number of non-zero diagonal elements. If n diagonal elements of diagonal matrix are one, such type of matrix called identity matrix of dimension n and identity matrix represented by I.

2.1 Vector Terminology

2.1.1 Vector Length

Squaring each component, add all the square components and taking the square root of the sum. If $\vec{q}$ is a vector, length of vector denoted by $|\vec{q}|$.

$$|\vec{q}| = \sqrt{\sum_{i=1}^{n} q_i^2} \tag{2.1}$$

For example, if $\vec{q} = [4,11,8,10]$, then

$$|\vec{q}| = \sqrt{4^2 + 11^2 + 8^2 + 10^2} = \sqrt{301} = 17.35$$

2.1.2 Vector Addition

Addition of two vectors means adding each component in its corresponding position.

Suppose, if $A = [a_1, a_2, \dots, a_n]$ and $B = [b_1, b_2, \dots, b_n]$, then $A + B = [a_1 + b_1, a_2 + b_2, \dots, a_n + b_n]$.

2.1.3 Scalar Multiplication

Multiply a scalar (real number) times a vector means multiplying each elements by the scalar to get new vector. Scalar multiplication means if a is a real number and $\vec{q}$ is a vector $[v_1, v_2, \dots \dots \dots, v_n]$ then $a * \vec{q} = [av_1, av_2, \dots \dots \dots, av_n]$ [1].

2.1.4 Inner Product

Multiplication of two vectors is called inner product (also called scalar product or dot product). Scalar value is the calculated by multiplying each component in $\vec{q}_1$by the component in $\vec{q}_2$ in the same position and adding the product of each component, we get a scalar. Dimensions of both vectors should be same, and then only inner product is possible. The inner product of the two vectors is denoted as $(\vec{q}_1, \vec{q}_2)$ or $\vec{q}_1.\vec{q}_2$(the dot product)[1].

$$(\vec{q}_1, \vec{q}_2) = \vec{q}_1.\vec{q}_2 = \sum_{i=1}^{n} q_1 q_2 \qquad (2.2)$$

2.1.5 Orthogonality

If inner product of two vectors equals to zero then they called orthogonal to each other. In two dimensional space, they called as vectors are perpendicular or the angle between two vectors are 90^0.

$$(\vec{q}_1, \vec{q}_2) = \vec{q}_1.\vec{q}_2 = \sum_{i=1}^{n} q_1 q_2 = 0 \qquad (2.3)$$

2.1.6 Normal Vector

Length of the vector is 1, called normal vector. We can normalized the vector by dividing each component in it by the vector's length.

For example, if = [2,4,1,2], then

$$|\vec{q}| = \sqrt{2^2 + 4^2 + 1^2 + 2^2} = \sqrt{25} = 5$$

Then $\vec{r} = \left[{}^2/_5, {}^4/_5, {}^1/_5, {}^2/_5\right]$ is a normal vector because

$$|\vec{r}| = \sqrt{({}^2/_5)^2 + ({}^4/_5)^2 + ({}^1/_5)^2 + ({}^2/_5)^2} = \sqrt{{}^{25}/_{25}} = 1$$

2.1.7 Orthonormal Vectors

Orthonormal vectors are those vectors which are of unit length and are orthogonal to each other. For example,

$$\vec{q} = \left[{}^2/_5, {}^1/_5, {}^{-2}/_5, {}^4/_5\right]$$

and

$$\vec{r} = \left[{}^{3}/_{\sqrt{65}}, {}^{-6}/_{\sqrt{65}}, {}^{4}/_{\sqrt{65}}, {}^{2}/_{\sqrt{65}}\right]$$

are orthonormal because

$$|\vec{q}| = \sqrt{(2/5)^2 + (2/5)^2 + (-2/5)^2 + (4/5)^2} = 1$$

$$|\vec{r}| = \sqrt{\left({}^{3}/_{\sqrt{65}}\right)^2 + \left({}^{-6}/_{\sqrt{65}}\right)^2 + \left({}^{4}/_{\sqrt{65}}\right)^2 + \left({}^{2}/_{\sqrt{65}}\right)^2} = 1$$

$$\vec{q}.\vec{r} = 0$$

2.2 Matrix terminology

2.2.1 Square Matrix

Equal number of rows and columns are called square matrix. If matrix has n row and columns called n square. For example, the matrix with 2-square

$$\begin{bmatrix} 2 & 3 \\ 4 & 4 \end{bmatrix}$$

2.2.2 Transpose

Changing the row into columns is called transpose of the matrix. Transpose of the matrix denoted by subscript^T . Given a matrix A, then transpose of the matrix A is A^T. For example, if

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Then transpose of matrix A is

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Size of matrix A is 2×3 then sixe of A^T become 3×2.

2.2.3 Matrix multiplication

Before multiplying to matrix, we have to check compatibility of these two matrixes. These matrixes are compatible only if first matrix has the same number of columns as the second matrix has rows. After the multiplication, size of the resulting matrix depends on first matrix rows and second matrix columns. Multiplications of two matrixes are determined by finding the inner product of each row of first matrix and each column of second matrix.

If A is a $m \times n$ matrix and B is $n \times p$ matrix, then AB is $n \times p$ matrix.

For example,

$$A = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 3 & 2 \\ -1 & 4 \\ 1 & 2 \end{bmatrix}$$

$$AB = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ -1 & 4 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 16 \\ 0 & 26 \end{bmatrix}$$

2.2.4 Identity Matrix

Any square matrix with elements of diagonal matrix equal to one and remaining elements of matrix equal to zero is called identity matrix. The diagonal entries are a_{ij} iff $i = j$, i.e., $a_{11}, a_{22}, \dots\dots, a_{mm}$. Identity matrix denoted by I_n or $I_{n\times n}$ for n-square matrix or for convenience it simply denoted by I. Every identity matrix must follow the properties [1]

$$AI = A, \text{ where } A \text{ is a matrix}$$

For example,

$$A = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} \text{ and } I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$AI = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} = A$$

2.2.5 Orthogonal Matrix

Matrix A is orthogonal iff we multiply its transpose then get an identity matrix.

$$AA^T = A^T A = I$$

For example $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix}$

A is orthogonal matrix then,

$$AA^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2.6 Diagonal Matrix

A matrix A is said to be diagonal matrix iff all the elements a_{ij} are zero where $i \neq j$.

$$A = \begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{mm} \end{bmatrix}$$

2.3 Eigenvector and eigenvalue

Eigen vector and Eigen value are calculated by the characteristic equation. Characteristic equation is given below:

$$A\,\vec{x} = \nu\vec{x}$$

where A is a square matrix, $\vec{x}$ is the eigen vector and ν is a scalar. ν is the eigen value. Using the ν, we have to find corresponding eigen vector. Eigen vector and eigen value also called as characteristic vectors and characteristic roots, respectively [5].

Properties of eigenvalues:

1. Theorem: Eigenvector with dissimilar eigenvalues are linearly independent.

Suppose, square matrix A and set of eigenvectors is $X = \{x_1, x_2, x_3, \dots \dots \dots, x_p\}$ with eigenvalues $v_1, v_2, v_3, \dots \dots \dots, v_p$ where $v_i \neq v_j$ whenever $i \neq j$. Then X said to be linearly independent set.

2. Theorem: Singular matrix has no eigenvalue.

Suppose, square matrix A. If A has zero eigenvalue then A said to be a singular matrix. And determinant of singular matrix is zero, so inverse of singular matrix don't exists.

3. Theorem: Nonsingular matrix has eigenvalues.

Suppose A is a square matrix with size n. They follow some properties:

1. A is a nonsingular matrix.

2. A is row-reduce to identity matrix.

3. Null space of A contains only zero vector, $\aleph(A) = \{0\}$

5. Unique solution exists for every possible choice of a, a exists in linear system (A, a).

6. There exists a set of linearly independent for the columns A.

7. Matrix A invertible, means A is nonsingular matrix there exists inverse of matrix A.

8. Rank of matrix A is n, $r(A) = n$.

9. The nullity of matrix A is zero, $n(A) = 0$.

10. Determinant of matrix A is non-zero, $\det(A) \neq 0$.

11. If $v = 0$, then there is no eigenvalue exists.

4. Theorem: Eigenvalues of the polynomial matrix.

Suppose A is a square matrix and $v$is the eigenvalue of A. Let $q(t)$ be a polynomial in the variable t. Then $q(v)$ is an eigenvalue of the matrix $q(A)$.

5. **Theorem: Eigenvalues of inverse of a matrix.**

Suppose A is a nonsingular matrix and ν is the eigenvalue of A. Then ν^{-1} is an eigenvalues of the matrix A^{-1}.

6. Theorem: Eigenvalues of the transpose of a matrix.

Suppose A is a square matrix with size n. Then ν is the eigenvalue of the matrix A^T.

7. Theorem: Eigenvalues of the real matrix.

Suppose A is a square matrix with real elements and x is an eigenvector of A for the eigenvalue ν. Then $\bar{x}$ is an eigenvector of A for the eigenvalue $\bar{\nu}$.

8. Theorem: Eigenvalues of hermitian matrices Hermitian matrices have real eigenvalues. Suppose A is a hermitian matrix and ν is the eigenvalues of A, $\nu \in \mathbb{R}$.

9. Hermitian matrices have orthogonal eigenvalues. Suppose, A is a hermitian matrix and if there exists two eigenvectors of A for different eigenvalues. Than x and y are orthogonal vectors to each other [5].

2.4 Matrix Norms

$L^{a\times b}$ is a vector space of dimension ab, magnitude of matrices $A \in L^{a\times b}$ can be measured by employing any vector norm on L^{ab}.

2.4.1 Frobenius norm

The Frobenius norm or Hilbert-Schmidt norm of $A \in L^{m\times n}$ is defined by the equations

$$\|A\|_F = \sqrt{\sum_{i=1}^{a}\sum_{j=1}^{b}\left|c_{ij}\right|^2} \tag{2.4}$$

$$\|A\|_F = \sqrt{Trace(AA^*)}$$

$$\|A\|_F = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2}$$

where A^* represents the conjugate transpose of A, σ_i is the singular values of A. Singular value is the square root of the eigenvalue. If $AA^* = A^*A = I$ then it called as unitary matrix. The function trace is the sum of the diagonal of the matrix A.Trace has a cyclic nature [6]

$$Trace(XYZ) = Trace(ZXY).$$

Properties of trace:

$$trace(\lambda A) = \lambda Trace(A)$$

and

$$Trace(A + B) = Trace(A) + Trace(B)$$

2.4.2 Spectral Norm

Spectral norm of matrix defined as square root of maximum eigenvalue of A^*A (where A^* is the conjugate transpose of matrix A) [6].

$$\|A\|_2 = (maximum\ eigenvalue\ of\ A^*A)^{1/2}$$

$$\|A\|_2 = \sqrt{\lambda_{max}(A^*A)}$$

$$\|A\|_2 = \sigma_{max}(A) \qquad (2.6)$$

2.4.3 Max Norm

The max norm defined as elementwise norm.

$$\|A\|_{max} = \max\{|a_{ij}|\} \qquad (2.7)$$

Max norm is not sub-multiplicative.

2.5 Vector Norm

2.5.1 Euclidean norm (or 2-norm)

Euclidean norm (or L^2) l^2 Norm is a vector norm defined for a complex vector-

$$x = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix}$$

by

$$\|x\| = \sqrt{\sum_{r=1}^{n} |x_r|^2} \tag{2.8}$$

Where $|x_r|$ on the right denotes the complex modulus.

2.5.2 L^1 Norm

A L^1 vector norm defined for a vector

$$x = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix}$$

With complex entries by

$$\|x\|_1 = \sum_{r=1}^{n} |x_r| \tag{2.9}$$

L^1 norm also called Taxicab norm or Manhattan norm. distance calculated by L^1 norm is called as Manhattan distance or L^1 distance [6].

2.5.3 L^{Infinity} Norm

A $L\text{^}Infinity$ vector norm defined for a vector

$$x = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix}$$

With complex entries by

$$\|x\|_\infty = \underset{i}{max} |x_i| \tag{2.10}$$

2.5.4 p-norm

For any real number, $p \geq 1$

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \tag{2.11}$$

If $p = 1$, called as taxicab norm (or L^1 norm). if $p = 2$, known as Euclidean norm. And $p = \infty$, then the p −norm is infinity norm (or maximum norm). Holder mean is related by p −norm.

By the definition, $0 < p < 1$, then norm is not define as the resulting function because it violates the triangular properties[12].

2.6 Matrix Decomposition (or Matrix Factorization)

In linear algebra, matrix decomposition (or matrix factorization) is defined as factorization of matrix into product of matrices. There are many different matrix decompositions methods exist. Each of the decomposition methods has specific feature to solve the different class of problems[4].

2.6.1 Decomposition based on solving linear equation

2.6.1.1 LU decomposition

In 1948, the great mathematician Alan Turing was introduced LU decomposition. LU decomposition (LU stands for lower upper and also known as LU factorization) factors the matrix into product of two parts, first is a lower triangular matrix and an upper triangular matrix. Gaussian elimination of matrix is viewed by LU decomposition.

Suppose, A be a square matrix. Factors of A refers as L and U, where L is the lower triangular matrix and U is refers as upper triangular matrix.

$$A = LU$$

In lower triangular matrix, L, all entries above the diagonal are zero. In upper triangular matrix, U, all entries below the diagonal are zero. For example, A is 3×3 matrix, LU decomposition of A,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Application of LU decomposition:

1. Solving linear equations

2. Inverting the matrix
3. Computing the determinant

2.6.1.2 LU reduction

LU reduction is an algorithm that is another version of LU factorization. Super computing and highly parallel computing used LU decomposition. LU decomposition used as benchmark algorithm, that provides to measure the speed for various computers. LU decomposition is the another kind of parallelized version of an LU decomposition algorithm [4].

2.6.1.3 QR decomposition

In 1959, British computer scientist John G. F. Francis and Soviet mathematician Vera Kublanovskay discovered the QR algorithm for eigenvalues.

In the field of linear algebra, QR decomposition (also called QR factorization) of matrix is the decompose into two matrix, first, Q and second, R then the product of Q and R equals to A.

$$A = QR \tag{2.12}$$

where Q is the orthogonal matrix and R is the upper triangular matrix.

Linear least squares problem solved by QR decomposition.

QR decomposition for square matrix:

Let A be the square matrix, decomposed as

$$A = QR$$

where Q is an orthogonal matrix (its columns are orthogonal unit vectors, $Q^T Q = I$) and R is the upper triangular matrix(right triangular matrix). Factorization is unique if A is invertible, so the diagonal elements of R is positive.

QR decomposition for rectangular matrix: Suppose A is rectangular matrix with $n \times m$ size (where $m \geq n$), orthogonal matrix Q is $m \times m$ and upper triangular matrix is $m \times n$. In upper triangular matrix, bottom $(n - m)$ rows of $m \times n$, consists only zeroes.

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = [Q_1, Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1$$

where R_1 is an $n \times n$ upper triangular matrix, zero is an $(n - m) \times n$ zero matrix, Q_1 is $m \times n$, Q_2 is $m \times (m - n)$ and Q_1 and Q_2 both have orthogonal columns [4].

2.6.1.4 Rank factorization (or rank decomposition)

Matrix A have size $m \times n$ with rank of A is r. Rank decomposition of A is

$$A = CF$$

where c is an $m \times r$ matrix and F is an $r \times n$ matrix. Rank decomposition is possible for every finite-dimensional matrix. For matrix A, whose column rank is r. Columns rank of matrix indicates there are r linearly independent columns in A.

$$rank(A) = rank(A^T)$$

Rank factorization can be achieved by row echelon forms. In this, we can compute B, reduced form of A. Removing the non-pivot columns, C is obtain and eliminating all zero rows of B [4].

2.6.1.5 Interpolative decomposition

An interpolative decomposition is the product of two matrixes. One of the matrixes contains selected columns form the original matrix. And another matrix contains a subset of columns that is identity matrix.

An interpolative decomposition of matrix A of rank $r \leq \min\{m, n\}$ is factorization

$$A\Pi = [A\Pi_1 \; A\Pi_2] = A\Pi_1[I \;\; T],$$

Where $\Pi = [\Pi_{1,}\Pi_2]$ is a permutation matrix, i.e., $A\Pi_2 = A\Pi_1 T$. We can write it as $A = BP$, where $B = A\Pi_1$ and $P = [I, T]\Pi^T$ are the skeleton and interpolation matrices, respectively

If A has not exact rank r, then A can be approximated by interpolative decomposition such that $A = BP + E$, where $\|E\| \sim \ \sigma_{r+1}$ is the largest singular value of A [4].

2.6.2 Decomposition based on eigenvalues and eigenvectors

2.6.2.1 Eigendecomposition

Canonical from of factorization is called as Eigendecomposition (or spectral decomposition), and the matrix is shown by the eigenvalues and eigenvectors. Eigendecomposition is possible only for square matrix.

For square matrix A, vector x of n dimensions that satisfied the linear equation

$$Ax = \upsilon x \tag{2.13}$$

where υ is the scalar, named as eigenvalue corresponding to x. Equation() called characteristic equation or eigenvalue problem.

$$p(\upsilon) = \det(A - \upsilon I) = 0 \tag{2.14}$$

where $p(\upsilon)$ is the characteristic polynomial and the equation called as characteristic equation.

For each eigenvalue, υ_i, there is specific equation

$$(A - \upsilon_i I)x = 0$$

There are linearly independent solutions to each eigenvalue equation.

Eigendecomposition of a square matrix

Let matrix A is size $n \times n$ with n linearly independent eigenvectors. Matrix A can be factorized as

$$A = G\Sigma G^{-1} \tag{2.16}$$

where G is the square matrix whose column is the eigenvectors of A and Σ is the diagonal matrix whose diagonal entries are corresponding to the each eigenvalues [5].

Functional calculus

Power series of matrices is computed by using eigendecomposition. $f(x)$ is given by

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots \ldots \ldots \ldots.$$

For matrix A,

$$f(A) = G f(\Sigma) G^{-1}$$

Because Σ is a diagonal matrix, function of Σ calculated:

$$[f(\Sigma)]_{ii} = f(v_i)$$

Similar method works with the holomorphic functional calculus, using

$$A^{-1} = G\Sigma^{-1}G^{-1}$$

from above,

$$[f(\Sigma)]_{ii} = f(v_i)$$

For example,

$$A^2 = (G\Sigma G^{-1})(G\Sigma G^{-1}) = G\Sigma(G^{-1}\mathrm{G})\Sigma G^{-1} = G\Sigma^2 G^{-1}$$

$$A^n = G\Sigma^n G^{-1} \tag{2.17}$$

2.6.2.2 Schur decomposition

In linear algebra, if square matrix A has size of n with complex entries, then decomposition of A define as,

$$A = GUG^{-1} \tag{2.18}$$

where G is a unitary matrix (for unitary matrix, G^{-1} and conjugate transpose G^* of G), and U is an upper triangular matrix, called a schur form of A. So U is same as A, U has the same multiset of eigenvalues, and U is a triangular, eigenvalues of A is the diagonal entries of U.

The Schur decomposition says that there is a nested sequence of A −invariant subspaces $\{0\} = V_0 \subset V_1 \subset V_2 \subset \cdots \ldots\ldots\ldots.. \subset V_n = C^n$ and there is first i basis vector span V_i fro each nested sequence [12].

Generalized schur decomposition

For given two matrices A and B, generalized schur decomposition factorize both matrices as

$$A = QRZ^* \tag{2.19}$$

And

$$B = QPZ^* \tag{2.20}$$

where Q and Z are unitary, and R and P are upper triangular. The generalized schur decomposition is known as QZ decomposition.

Eigenvalue υ that solve the generalized eigenvalues decomposition problem $Ax = \nu Bx$(where x is an unknown nonzero vector) is calculated as the ratio of the diagonal elements of R to those P. For the i^{th} generalized eigenvalue ν_i satisfies

$$\nu_i = \frac{R_{ii}}{P_{ii}} \tag{2.21}$$

2.6.2.3 Singular value decomposition

In linear algebra, the singular value decomposition (SVD) is the factorization method for real or complex matrix. Suppose we have given real or complex matrix A with $n \times m$ size, then singular value decomposition of A(Golub and van loan, 1996; Watkins, 1991),

$$A = U\Sigma V^* \tag{2.22}$$

where U is an $n \times n$ orthonormal matrix, columns of U called as left singular vectors of A. And V^*(Conjugate transpose of V for real or complex matrix) is an $m \times m$ the orthonormal matrix, columns of V called as right singular values of A [8].

Relation with eigenvalue and singular value decomposition:

1. The left singular vectors of A calculated by eigenvalues of AA^*.
2. The right singular vectors of A calculated by eigenvalues of A^*A.
3. The non-singular values of A calculated by both A^*A and A^*A.

The properties of SVD matrix are given below:

1. The Singular values of a real valued rectangular matrix A are equal to the square roots of the eigenvalues $v_1, v_2, v_3, \dots \dots \dots .., v_m$ of matrix AA^T.
2. Positive singular values is same as the rank of matrix A
$$rank(A) = r,\ r \leq \text{ m.}$$
3. The Euclidean norm of A is equal to the largest singular value,
$$\|A\|_2 = \sigma_1$$

Basic Idea of SVD: Data is in high dimensional space, highly variable set of data points so need to be reducing it to low dimensional space losing the less information [13]. SVD is based on the theorem of linear algebra. Theorem says that any rectangular matrix can be factorize in three matrices – orthogonal matrix U, a diagonal matrix Σ and transpose of orthogonal matrix. We can represent the theorem as

$$A_{mn} = U_{mm}\Sigma_{mn}V_{nn}^T$$

where $U^TU = I,\ V^TV = I$, here, the columns of U are orthonormal eigenvectors of AA^T, the columns of V are orthonormal eigenvectors of AA^T and S is a diagonal matrix containing the square roots of eigenvalues from $U\ or\ V$ in descending order[2].

Theorem 1: (Matrix diagonalization theorem) Let A be a square real-valued $n \times n$ matrix with n linearly independent eigenvectors. Then there exists an eigendecomposition

$$A = U\Sigma U^{-1} \qquad (2.23)$$

Where the columns are the eigenvectors of A and Σ are a diagonal` matrix whose diagonal entries are the eigenvalues of A in decreasing order.

$$\begin{pmatrix} \nu_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \nu_n \end{pmatrix}, \nu_i \geq \nu_{i+1}$$

Decomposition is unique if eigenvalues are distinct.

Theorem 2 (Symmetric diagonalization theorem) Let A be a square, symmetric real-valued matrix with n linearly independent eigenvectors. Then there exists an eigen decomposition

$$A = U\Sigma U^T \tag{2.24}$$

For symmetric matrix, left singular vector and right singular vector both are same.

Application of Singular value decomposition

1. Pseudoinverse

Pseudoinverse of the matrix can be computing using singular value decomposition.

Pseudo-inverse of symmetric matrix A as

$$A^+ = \sum_{t=1}^{r} {\sigma_t}^{-1} U^{(t)} {U^{(t)}}^T \tag{2.25}$$

And

pseudo-inverse of complex matrix A as

$$A^+ = V\Sigma^+ U^* \tag{2.26}$$

where Σ^+ is the pseudoinverse of Σ, which is calculated by replacing the non-zero diagonal entry by the reciprocal of the singular value and transposing the resulting matrix. Pseudoinverse is used to solve the linear least squares problems[2].

2. Low rank approximation

Low rank approximation problem is solved by the SVD. Suppose the $\tilde{A}$ is the approximation of A

$$\tilde{A} = U\tilde{\Sigma}U^T$$

In low rank approximation, sort the singular values in the decreasing order and then select the top k largest singular values. we have to minimize the Frobenius norm of the difference between A and $\tilde{A}$. This theorem is also known as Eckart-Young theorem.

2.6.2.4 CUR decomposition

CUR decomposition are the class of randomized algorithms which is used to approximate the matrix A by taking only small number of columns of A.

Given a matrix A ∈ $R^{n\times m}$ decompose into three matrices C, U and Ras

$$A = CUR$$

where ∈ $R^{m\times k}$, $U \in R^{k\times k}$,$R \in R^{n\times k}$, C contains exactly k columns of A, R consists of row of A and U is a small matrix that says the product of CUR very close to A.

CUR decomposition can be used as SVD for low rank approximation. But CUR has high inaccuracy as compare to SVD still CUR approximation is easy to compute.

2.7 Kernel functions

Kernel functions can be used in many applications as they provide a simple bridge from linearly to non-linearly for algorithms which can express in terms of dot products. Below is a list of some kernel function available [7].

2.7.1 Linear kernel

The linear kernel is the simplest kernel function. It is given by the inner product $< x, y >$ plus an optional constant c [7].

$$k(x, y) = x^T y + c$$

2.7.2 Polynomial kernel

The polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized.

$$k(x, y) = (\alpha x^T y + c)^d$$

Where alpha is the slope, constant term c and the polynomial degree d [7].

2.7.3 Gaussian kernel

The Gaussian kernel is example of radial basis function kernel.

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Alternatively, it could also be implemented using

$$k(x, y) = \exp(-\gamma\|x - y\|^2)$$

Sigma plays the important a major role in the performance of kernel, it depends on the problem [7].

2.7.4 Exponential kernel

The Laplace kernel is equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter. Being equivalent, it is also a radial basis function kernel [7].

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{\sigma}\right)$$

2.7.5 Power kernel

The power kernel is also known as the triangular kernel.

$$k(x, y) = -\|x - y\|^d$$

2.7.6 Log kernel

The log kernel seems to be particularly interesting for images, but it only conditionally positive define.

$$k(x, y) = -\log(\|x - y\|^d + 1)$$

2.8 Summary

In this chapter, we present the basic terminology of matrix and vector. Various norm discussed that will use to find the error bound of the various low rank approximation method. We also discussed some kernel function which uses in the implementation In next chapter we present the various low rank approximation algorithms and their theoretical error bound based on Frobenius norm and spectral norm.

Chapter 3
Literature survey

In this chapter, we discuss various low rank approximation methods to generate the good approximation of the matrix. We discuss sampling algorithms and some other low rank approximation methods. Contents of this chapter is linear time SVD, constant time SVD and various sampling based low rank approximation methods such as standard Nystrom and column sampling method.

3.1 Introduction

Given matrix X, find the good approximation of X that has low rank. There are several techniques for fast computation of the approximation of X. And X has good spectral feature that help to capture the data from the original matrix. Quality of the matrix approximation is measured by the Frobenius norm and spectral norm. That is the fundamental result of linear algebra for matrix X and k is the any positive integer. Best k rank approximation is denoted by X_k. Efficient computation of the low rank approximation measure by the $\|X - X_k\|$. C is the any rank k matrix then it satisfies

$$\|X - C\| \leq \|X - X_k\| + \delta \tag{3.1}$$

Tolerable error δ is adding to X a matrix G whose entries are independent Gaussian random variable with mean 0 and standard deviation σ. That is, σ is not too big, the optimal rank k approximation to $\tilde{X} = X + G$ will approximate X nearly as well as X_k. This stability of low rank approximations with respect to Gaussian noise is well understood and stems from the fact that no low dimensional subspace accommodates G well, i.e., $\|G_k\|$ is small for small k. Low rank approximations are frequently used with the explicit purpose of removing Gaussian noise [30].

A fundamental result in random matrix theory is that the G_{ij} being Gaussian is not essential in the above example. Rather, G is innocuous by virtue of the following three properties of its entries: independence, mean zero, and small variance. If N is a random matrix whose entries N_{ij} are zero-mean, independent random variables with variance bounded by σ^2, then $\|N_k\| \sim \|G_k\|$. There is a proof of the quantity $\|N_k\|$ bounds the influence that N may apply over the optimal rank k approximation to $X + N$. Specifically, to the extent that $\|X_k\| \gg \|N_k\|$, the matrix $(X + N)_k$ will be largely determined by X [31].

3.1.1 Statement of results

Definitions of the Frobenius norm and 2- norm,

$$\|X\|_F = \left(\sum_{i,j} X^2\right)^{1/2} \text{ and } \|X\|_2 = \max_{\|x\|=1} \|X_x\|$$

For matrix X and any k, $\|X\|_F \leq \sqrt{k}\|X\|_2$, and $\|X_k\|_2 = \|X\|_2$

This analysis based on observing that acts of sampling and quantization cab be viewed as adding random matrix N to X, whose entries are independent random variables with zero-mean and bounded variance. Since, with high probability, N has very weak spectral features, shows the effect sampling and quantization nearly vanishes when low rank approximation to $X + N$ is computed. The quality of approximation is given by the Frobenius norm and 2-norm [17].

Next statement a lemma formalizing that perturbation matrices which are poorly approximable in k dimensions have little influence on the optimal rank k approximation.

Lemma 1. *Let X and N be any matrices and write $\tilde{X} = X + N$. then*

$$\left\|X - \tilde{X}_k\right\|_2 \leq \|X - X_k\|_2 + 2\|N_k\|_2 \text{ and}$$

$$\left\|X - \tilde{X}_k\right\|_F \leq \|X - X_k\|_F + \|N_k\|_F + 2\sqrt{\|N_k\|_F \|X_k\|_F}$$

Notice that all error terms above scale with $\|N_k\|$ and thus whenever N is poorly approximated in k dimensions, i.e., $\|N_k\|$ is small, the error caused by adding N to X must be small.

According to lemma 1, take an example of a Gaussian perturbation matrix. This will provide a sense of scale for our results, stated in theorems 1-3 below

Fact 1. *Let* X be an $n \times m$ matrix, where $m \leq n$, whose entries are independent Gaussian random variables with mean 0 and variance σ^2. With probability $1 - e^{-\theta(n)}$,

$$\|G_k\|_2 \le 4\sigma\sqrt{n} \text{ and } \|G_k\|_F \le 4\sigma\sqrt{kn}.$$

To put these two bounds in perspective consider the trivial rank k approximation, D, that results from zeroing-out all but the first k rows of G. With high probability we have $\|D\|_F \approx \sigma\sqrt{kn}$. Moreover, since D has rank at most k, $\|D\|_2 \ge \|D\|_F/\sqrt{k}$. Fact 1 asserts that the optimal rank k approximation of G only improves upon this trivial approximation by at most a factor of 4, attesting to the near-orthogonally of the rows of G. In contrast, for general $m \times n$ matrices A with $|A_{ij}| = \sigma$, $\|A_k\|$ can easily be as large as $\sigma\sqrt{mn}$, in either norm[25].

Results show that it is possible to find a good low rank approximation to A even after randomly quantizing its entries. In theorem 1 below we quantize each entry to a single bit, representing a 32 to 64 factor of compression over standard floating point numbers. Naturally, one can generalize the quantization process to larger set of numbers, trading representation length for error.

Theorem 1. *Let A be any $m \times n$ matrix where $m \le n$, and let $b = max_{ij}|A_{ij}|$. Let $\hat{A}$ be a random $m \times n$ matrix whose entries are independently distributed as*

$$\hat{A}_{\boldsymbol{ij}} = \begin{cases} +b \ with \ probability \frac{1}{2} + \frac{A_{ij}}{2b}, \\ -b \ with \ probability \frac{1}{2} - \frac{A_{ij}}{2b}. \end{cases} \tag{3.2}$$

For all sufficiently large n, with probability at least $1 - \exp(-19(\log n)^4)$, the matrix $N = A - \hat{A}$ satisfies

$$\|N_k\|_2 \le 4\sigma\sqrt{n} \text{ and } \|N_k\|_F \le 4\sigma\sqrt{kn}.$$

Our second result asserts it is possible to find a good low rank approximation to A even after randomly omitting many of its entries. In particular, the stronger the spectral features of A the more of its entries we can afford to omit.

Theorem 2. *Let A be any* $m \times n$ *matrix where* $76 \leq m \leq n$, *and let* $b = \max_{ij}|A_{ij}|$. *For* $p \geq (8\log n)^4/n$, *let* $\hat{A}$ *be a random* $m \times n$ *matrix whose entries are independently distributed as*

$$\hat{A}_{ij} = \begin{cases} \frac{A_{ij}}{p} \; with\, probility\; p, \\ 0 \qquad otherwise \end{cases} \tag{3.3}$$

With probability at least *with probability at least* $1 - \exp(-19(\log n)^4)$, *the matrix* $N = A - \hat{A}$ *satisfies*

$\|N_k\|_2 \leq 4\sigma\sqrt{n/p}$ and $\|N_k\|_F \leq 4\sigma\sqrt{kn/p}$.

As mentioned earlier, we can improve upon the uniform sparsification process by retaining entries with probability that depends on their magnitude. This yields greater sparsification when entry magnitudes vary, without increasing the error bounds.

Theorem 3. *let* $\hat{A}$ *be a random* $m \times n$ *matrix where* $76 \leq m \leq n$, *and let* $b = \max_{ij}|A_{ij}|$. *For any p>0, define* $\tau_{ij} = p(A_{ij}/b)^2$ *and let*

$$p_{ij} = \max\left\{\tau_{ij},\; \sqrt{\tau_{ij} \times (8\log n)^4/n}\right\}$$

Let $\hat{A}$ *be a random* $m \times n$ *matrix whose entries are independently distributed as*

$$\hat{A}_{ij} = \begin{cases} \frac{A_{ij}}{p} \; with\, probility\; p_{ij}, \\ 0 \qquad otherwise \end{cases} \tag{3.4}$$

1. *With probability at* least *with probability at least* $1 - \exp(-19(\log n)^4)$, *the matrix* $N = A - \hat{A}$ *satisfies*

   $\|N_k\|_2 < 4\sigma\sqrt{n/p}$ and $\|N_k\|_F < 4\sigma\sqrt{kn/p}$.
2. *The expected number of non-zero entries in* $\hat{A}$ *is bounded by (pmn)*$\times$ $Avg\left[\left(\frac{A_{ij}}{b}\right)^2\right] + m(8\log n)^4$[29] .

Proof of lemma 1

Now, prove two lemmas relating $\|A - B_k\|$ to $\|A - A_k\|$ for arbitrary matrices A, B, in the Frobenius and 2-norm. specifically, lemma 2 compares $\|A - B_k\|_2$ to $\|A - A_k\|_2$, while lemma 4 compares $\|A - B_k\|_F$ to $\|A - A_k\|_F$ [32].

Lemma 2. *For any matrices A and B*

$$\|A - B_k\|_2 \text{ to } \|A - A_k\|_2 + 2\|(A - B)_k\|_2.$$

Proof. Starting with $\|A - B_k\|_2$ and applying the triangle inequality we get (3.5). Using that for any rank k matrix D, $\|B - B_k\|_2 \leq \|B - D\|_2$ we get (3.6). applying the triangle inequality again gives (3.7)

$$\|A - B_k\|_2 \leq \|A - B\|_2 + \|A - B_k\|_2 \tag{3.5}$$

$$\leq \|A - B\|_2 + \|A - A_k\|_2 \tag{3.6}$$

$$\leq \|A - B\|_2 + \|B - A\|_2 + \|A - A_k\|_2 \tag{3.7}$$

Finally, we note that $\|B - A\|_2 = \|A - B\|_2 = \|(A - B)_k\|_2$, which concludes the proof .

In order to prove the Frobenius norm bound we need to introduce the following notion. Given a matrix, let P_M denote the projection matrix onto the space spanned by the columns of M_k (we suppress the dependence of P_M on k to simplify notation). An important consequences of the Singular Value Decomposition is that $M_k = P_M M$ and, as a result, that for any matrices A and B[30],

$$\|P_A A\|_F \geq \|P_B A\|_F \tag{3.8}$$

To prove our stated bounds for the Frobenius norm we will first show that for any matrices A, B, if $\|(A - B)_k\|_F$ is small, then projecting A onto P_B is almost as good as projecting it onto P_A in terms of capturing Frobenius norm.

Lemma 3. *For any matrices A and B*

$$\|P_B A\|_F \geq \|P_A A\|_F - 2\|(A-B)_k\|_F.$$

Proof. Starting with $\|P_B A\|_F$ and applying the triangle inequality we get (3.9). Applying (3.8) yields (3.10). Applying the triangle inequality again gives (3.11)

$$\|P_B A\|_F \geq \|P_B A\|_F - \|P_B(A-B)\|_F \tag{3.9}$$

$$\geq \|P_A A\|_F - \|P_B(A-B)\|_F \tag{3.10}$$

$$\geq \|P_A A\|_F - \|P_A(B-A)\|_F - \|P_B(A-B)\|_F \tag{3.11}$$

Finally, we applying (3.8) to bound the $\|P_X(A-B)\|_F$ terms in (3.11) by $\|(A-B)_k\|_F$.

We now use Lemma 3 to prove that if $\|(A-B)_k\|_F$ is small, then $\|A-B_k\|_F$ is not much larger than $\|A-A_k\|_F$. In order words, B_k can be a good surrogate for A_k with respect to the Frobenius norm even when $\|A-B\|_F$ is large, so long as $\|(A-B)_k\|_F$ is small [30].

Lemma 4. *For any matrices A and B*,

$$\|A-B_k\|_F \leq \|A-A_k\|_F + 2\sqrt{\|(A-B)_k\|_F\,\|A_k\|_F} + \|(A-B)_k\|_F.$$

Proof. By the fact $P_B B = B_k$ and the triangle inequality we get

$$\|A-B_k\|_F \leq \|A-P_B A\|_F + \|P_B(A-B)\|_F. \tag{3.12}$$

Applying the Pythagorean inequality to each column of A implies that for any projection matrix P_B,

$$\|A-P_\mathrm{B}A\|_F^2 = \|A\|_F^2 - \|P_\mathrm{B}A\|_F^2 \tag{3.13}$$

Inserting (3.12) in (3.13), we get

$$\|A-B_k\|_F \leq (\|A\|_F^2 - \|P_\mathrm{B}A\|_F^2)^{\frac{1}{2}} + \|P_B(A-B)\|_F. \tag{3.14}$$

To bound the right hand side of (3.14) we first invoke the lower bound for $\|P_B A\|_F$ provided by Lemma 3 to get (3.15). We then use (3.13) to get (3.16) to which we apply the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ to get (3.17).

$$\|A - B_k\|_F \leq (\|A\|_F^2 - \|P_A A\|_F^2 + 4\|(A-B)_k\|_F \|P_A A\|_F)^{\frac{1}{2}} + \|P_B(A-B)\|_F \tag{3.15}$$

$$= (\|A - A_K\|_F^2 + 4\|(A-B)_k\|_F \|A_k\|_F)^{\frac{1}{2}} + \|P_B(A-B)\|_F \tag{3.16}$$

$$\leq \|A - B_k\|_F + (4\|(A-B)_k\|_F \|A_k\|_F)^{\frac{1}{2}} + \|P_B(A-B)\|_F \tag{3.17}$$

To wrap up, we use (3.8) again to bound $\|P_B(A-B)\|_F$ by $\|(A-B)_k\|_F$.

3.2 Comparison of the various error bounds

Now, we focus on the various error bounds given by Drines et al[19]**,** as it gives the best known bounds for computing near-optimal low rank matrix approximation by column / row sampling. Approach given by Achlioptas et al [18], i) sample c columns from A, selecting each columns with probability proportional to its squared 2-norm, ii) determine the optimal k- dimensional subspace D. Intuitively, as the number of columns sampled grows, the sample approaches the distribution of columns of A. In particular, [] show that if $c = O(1/\epsilon^2)$ columns are drawn, then with constant probability the resulting rank k matrix satisfies

$$\|A - D\|_{\boldsymbol{F}}^{\boldsymbol{2}} \leq \|A - A_k\|_{\boldsymbol{F}}^{\boldsymbol{2}} + \boldsymbol{\epsilon}\|A\|_F^2 \tag{3.18}$$

$$\|A - D\|_{\boldsymbol{F}}^{\boldsymbol{2}} \leq \|A - A_k\|_{\boldsymbol{F}}^{\boldsymbol{2}} + \boldsymbol{\epsilon}\sqrt{k}\|A\|_F^{\boldsymbol{2}} \tag{3.19}$$

More generally, if $c = O((\log z)^2/\epsilon^2)$ columns are drawn then (3.18), (3.19) holds with probability $1 - 1/z$. We note that, in the case of the Frobenius norm, the bound in (3.19) is meaningful only if $\epsilon\sqrt{k} < 1$.

By Comparison, given the same matrix A, invoking sample $(\frac{16n}{\epsilon^2}, n)$ yields an $m \times n$ matrix $\hat{A}$ with $O(\frac{n}{\epsilon^2} + m(\log n)^4)$ non-zero entries, sampled from the distribution of theorem 3 with $p = 16nb^2/(\epsilon\|A\|_F)^2$. Thus, by Lemma 1 and theorem 3, with probability $1 - \exp(-19(\log n)^4)$,

$$\left\|A - \hat{A}_k\right\|_2 \leq \|A - A_k\|_2 + \epsilon\|A\|_F \tag{3.20}$$

$$\left\|A - \hat{A}_k\right\|_F \leq \|A - A_k\|_F + 3\sqrt{\epsilon}k^{1/4}\|A\|_F \tag{3.21}$$

where in deriving (3.21) we assumed that $\epsilon\sqrt{k} < 1$ so that the bound in (3.19) is meaningful.

Unfortunately, making a direct comparison of the two results at this point is hindered by the fact (3.18),(3.19) bound $\|A - D\|^2 - \|A - A_k\|^2$ while (3.20), (3.21) bound $\left\|A - \hat{A}\right\| - \|A - A_k\|$. If the bound the right hand side of (3.18), (3.19) using the inequality $a^2 + b^2 < (a + b)^2$, observed that the Frobenius bounds are comparable, while 2-norm decays at the rate of [6].

Such a comparison is oversimplified, to be sure: $a^2 + b^2 < (a + b)^2$ is inappropriate when $\|A - A_k\| \gg \epsilon\|A\|_F$. Also, equating $O(1/\epsilon^2)$ columns of A with $O(\frac{n}{\epsilon^2} + m(\log n)^4)$ non-zero entries is justified only when each column of A contributes $\theta(n)$ non-zero entries and $\frac{n}{\epsilon^2}$ is not dominated by $m(\log n)^4$[18].

3.3 Low rank approximation

3.3.1 Truncated SVD

The singular value decomposition is a classical mathematical technique for factorizing a matrix. The SVD for a matrix $A \in \mathbb{R}^{m\times n}$ is the decomposition $A = USV^T$, where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and $S \in \mathbb{R}^{m\times n}$. The matrices U, V are orthogonal, with their columns being eigenvectors of XX^T and X^TX respectively. S is a diagonal matrix $\text{diag}(\sigma_1, \sigma_2, \ldots \ldots, \sigma_p\ )$ where $p = \min\{m, n\}$. The $\sigma_i,'s$ are sorted in decreasing order and are known as singular values of A. The $\sigma_i's$ is satisfy $\sigma_i \geq 0$ for all i, and the squares of nonzero $\sigma_i's$ are the eigenvalues of XX^T(or equivalently X^TX).One of the most common operations one performs using the decompositions USV^T is to form the truncated SVD

$$A_k = U_kS_kV_k^T,$$

Where we take first k columns of U, V and consider the $k \times k$ submatrix S_k. Due to the orthogonally of $U, V,$ one can equivalently write A_k as $U_k U_k^T A$ or $A V_k V_k^T$. This matrix has rank at most k, and a classic theorem of Eckart-Young-Mirsky says that it is a good approximation to the matrix A in the following specific sense[30].

Theorem 4 [EACKART AND YOUNG 1936; MIRSKY 1960]. *For a given matrix $A \in \mathbb{R}^{m \times n}$, let A_k be its truncated SVD. Let $\|.\|_M$ be a unitarily invariant matrix norm. then, for all $k \leq rank(A)$,*

$$\|A - A_k\|_M = \min_{rank(B)=k} \|A - B\|_M$$

To see the intuition behind the theorem, another way to express the SVD is to write each elements of the matrix A as

$$A_{ij} = \sum_l U_{il} V_{jl} S_{ll} = \sum_l U_{il} V_{jl} \sigma_l^2.$$

Now suppose that we keep just the top k singular values of A, and significantly treat the rest as being 0: this creates a rank k approximation to A. Since $\sigma_1 \geq \sigma_2 \ldots \ldots \geq \sigma_p$, intuitively this approximation should be quite good, because we are keeping the terms that contribute the most to A_{ij}. What the theorem says is that this approximation is not only good, but optimal [27].

Two commonly invariant norms are the Frobenius and 2-norm. the Frobenius norm of matrix, $\|.\|_F$, is an analogue of the l_2 norm for vectors :

$$\|A\|_F = \sqrt{\sum_i \sum_j A_{ij}^2} = \sqrt{\sum_i \sigma_i^2} .$$

The 2-norm (or spectral norm) of the matrix, $\|.\|_2$ is an induced that the largest singular value of a matrix:

$$\|A\|_2 = \sigma_1 = \max_i \sigma_i$$

An important connection between the 2-norm and the singular values of a matrix A is the following result:

$$\|A - A_k\|_2 = \sigma_{k+1}$$

It is well known that $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}.\ \|A\|_2$ [11].

3.3.2 Computing the SVD

In general, the problem of computing the SVD of a matrix reduces to that of computing the Eigenvalue decomposition of a symmetric matrix. Eigenvalue decomposition is deeply connected to the SVD because of the following fact.

Fact 2. Let $A \in \mathbb{R}^{m \times n}$ have the decomposition USV^T, where $S = \text{daig}(\sigma_i)$. Let $u_i,\ v_i$ denote the columns of U, V respectively. Then,

i. $C = A^T A$ has eigenvalue σ_i^2 with corresponding eigenvectors v_i.
ii. $B = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$ has eigenvalues $\pm\sigma_i$ with corresponding eigenvalues $\frac{1}{\sqrt{2}}\begin{bmatrix} u_i \\ \pm v_i \end{bmatrix}$.

Therefore, common methods for computing the SVD of a matrix are standard eigensolvers such as QR iteration and Arnoldi/Lanczos iteration, or are slight modifications of the same, such as the modified Golub-Reinsch method [Chan 1982]. We note that any algorithm for SVD computation can only produce an approximation to the true SVD of a matrix. The reason for this is an impossibility result which says that no algorithm can find the exact eigenvalues of a general matrix. The proof of this claim is that computing the eigenvalues of a matrix is equivalent to finding the roots of its characteristic polynomial, and further, that every polynomial is the characteristic polynomial of some matrix (known as the *companion matrix*). However, the Abel-Ruffini theorem [37] says that there is no formula for finding the roots of polynomials of degree ≥ 5. This implies that no general algorithm exists for finding eigenvalues, which shows the claim [8].

For most classical SVD algorithms, we have the following error guarantees due to finite precision arithmetic [11]. Throughout, we let ϵ denote the machine-precision constant, that is, the maximum relative error in storing a real number on a computer.

(1) The reported singular values $\tilde{\sigma}_i$ are close to the true ones σ_i:

$$| \sigma_i - \tilde{\sigma}_i | \leq \epsilon . \sigma_i$$

(2) The reported left-singular vectors $\tilde{u}_i$ are close in angle $\theta(\tilde{u}_i, u_i)$ to the true left-singular vectors u_i:

$$\theta(\tilde{u}_i, u_i) = \frac{\epsilon \|A\|_2}{\min_{j \neq i} |\sigma_i - \sigma_j|}$$

(3) The reported matrix $\tilde{U}, \tilde{\Sigma}, \tilde{V}$ are precisely the SVD of a matrix $A + \Delta A$, where

$$\|\Delta A\|_2 = \epsilon \|A\|_2$$

On dense datasets, the classical methods have complexity of $O(mn \min\{m, n\})$ to compute the thin (or economy) SVD, defined as:

$$A = U_n S_n V_n^T$$

that is, the truncated SVD with $k = n$. This makes them infeasible if $\min\{m, n\}$ is large. This the motivation to approximation to the SVD.

3.3.3 Computing Approximations to the SVD

The truncated SVD of a matrix is mathematically guaranteed to be the optimal low rank approximation in the sense described in Theorem 4. However, a natural question is whether we can find a suboptimal approximation much quicker: provided the additional error is not too great, one can typically use this instead of the truncated SVD. For a norm $\|.\|_M$, the quantity of interest is

$$\delta_M(A, \hat{A}) = \|A - \hat{A}\|_M - \|A - A_k\|_M$$

Where A_k is the truncated SVD of A and $\hat{A}$ is an approximation to A_k. If $\delta_M(A, \hat{A})$ is small, that means that $\hat{A}$ is close to being an optimal low rank approximation to A [8].

3.3.4 Linear time SVD approximation algorithm

3.3.4.1 The Algorithm

Given a matrix $A \in \mathbb{R}^{m \times n}$ we wish to approximate its top k singular values and the corresponding singular vectors in a constant number of passes through the data and

$O(c^m + c^2)$ additional space and $O(c^2m + c^3)$ additional time. The strategy behind the LinearTimeSVD algorithm is to pick c columns of the matrix A, rescale each by an appropriate factor to form a matrix $A \in \mathbb{R}^{m \times c}$, and then compute the singular values and corresponding left singular vectors of the matrix C, which will be approximations to the singular values and left singular vectors of A, in a sense we make precise later. These are calculated by performing an SVD of the matrix C^TC to compute the right singular vectors of C and from them calculating the left singular vectors of C.

The LinearTimeSVD algorithm is described in figure 1; it takes as input a matrix A and returns as output an approximation to the top k left singular values and the corresponding singular vectors. Note that by construction the SVD of C is $C = H\Sigma_C Y^T$.

LinearTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m \times n}, c, k \in \mathbb{Z}^+$ such that $1 \leq k \leq c \leq n, \{p_i\}_{i=1}^{n}$ such that $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

Output: $H_k \in \mathbb{R}^{m \times k}$ and $\sigma_t(C), t = 1, \dots . k$.

```
1. For $t = 1$ to $c$,
   (a) Pick $i_t \in 1, \dots .. n$ with Pr$[i_t = \alpha] = p_\alpha, \alpha = 1, \dots \dots ., n$.
   (b) Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_i}$.
2. Compute $C^TC$ and its SVD
3. Compute $h^t = \frac{Cy^t}{\sigma_t(C)}$ for $t = 1, \dots ., k$.
4. Return $H_k$, where $H_k^{(t)} = h^t$, and $\sigma_t(C), t = 1, \dots \dots, k$.
```

Figure 1: The LinearTimeSVD algorithm.

3.3.4.2 Analysis of the implementation and running time

Assuming thatnearly optimal sampling probabilities are used, then in the LinearTimeSVD algorithm the sampling probabilities P_k can be used to select columns to be sampled in one pass and $O(c)$ additional space and time using the Select algorithm of [13]. Given the elements to be sampled, the matrix C can then be constructed in one additional pass; this requires additional space and time that is $O(mc)$. Given $A \in \mathbb{R}^{m\times n}$, computing C^TC requires $O(mc)$ additional space and $O(mc^2)$ additional time, and computing the SVD of C^TC requires $O(c^3)$ additional time. Then computing H_k requires k matrix-vector multiplications for a total of $O(mck)$ additional space and time. Thus, overall $O(cm + c^2)$ additional space and $O(c^2m + c^3)$ additional time are required by the LinearTimeSVD algorithm. Note that the "description" of the solution that is computable in the allotted additional space and time is the explicit approximation to the top k singular values and corresponding left singular vectors [31].

3.3.5 Constant time SVD approximation algorithm

3.3.5.1 The algorithm

Given a matrix $A \in \mathbb{R}^{m\times n}$ we now wish to approximate its top k singular values and the corresponding singular vectors in a constant number of passes through the data and additional space and time that are $O(1)$, independent of m and n. The strategy behind the ConstantTimeSVD algorithm is to pick c columns of the matrix A, rescale each by an appropriate factor to form a matrix $A \in \mathbb{R}^{m\times c}$ and then compute approximations to the singular values and left singular vectors of the matrix C, which will then be approximations to the singular values and left singular vectors of A. In the LinearTimeSVD algorithm of section 3.3.4, the left singular vectors of the matrix C are computed exactly; as the analysis of section 3.3.4.2 showed, this computation takes additional space and time that is linear in $m + n$ (assuming that n is constant). With the ConstantTimeSVD algorithm, in order to use only a constant $O(1)$ additional space and time, sampling is performed again, drawing rows of C to construct a matrix $W \in \mathbb{R}^{w\times c}$. The SVD of W^TW is then computed; let $W^TW = Z\Sigma_{W^TW}Z^T = Z\Sigma_W^2Z^T$. The singular values and corresponding singular vectors so obtained are with high probability approximations to the singular values and singular vectors of $C^T C$ and thus to the singular values and right singular vectors of C. Note that this is simply using the LinearTimeSVD algorithm to approximate the right singular vectors of C by randomly sampling rows of C.

ConstantTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $c, k \in \mathbb{Z}^+$ such that $1 \leq w \leq m,\ 1 \leq w \leq m$ and$1 \leq k \leq \min(w, c)$, and $\{p_i\}_{i=1}^{n}$ such that $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

Output: $\sigma_t(W), t = 1, \dots\dots, l$ and a "description" of $\widetilde{H}_l \in \mathbb{R}^{m \times l}$.

```
1. For $t = 1$ to $c$,
   (a) Pick $i_t \in 1, \dots..n$ with $\Pr[i_t = \alpha] = p_\alpha, \alpha = 1, \dots\dots, n$, and save $\{(i_t, p_{ji}) : t = 1, \dots., c\}$.
   (b) Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_i}$ (Note that Set $C$ is not explicitly constructed in RAM)
2. Choose Set $\{q_j\}_{j=1}^{m}$ such that $q_j = \left|C_{(j)}\right|^2/\|C\|_F^2$.
3. For $t = 1$ to $w$,
   (a) Pick $i_t \in 1, \dots..m$ with $\Pr[j_t = \alpha] = q_\alpha, \alpha = 1, \dots\dots, m$,
   (b) Set $W_{(t)} = C_{(ji)}/\sqrt{wq_{jt}}$
4. Compute $W^T W$ and its SVD.
5. If a $\|.\|_F$ bound is desired, set $\gamma = \epsilon/100\, k$,
   Else if a $\|.\|_2$ bound is desired, set $\gamma = \epsilon/100$.
6. Let $l = \min\{k, \max\{t : \sigma_t^2(W) \geq \gamma \|W\|_F^2\}\}$
7. Return singular values $\{\sigma_t(W)\}_{t=1}^{l}$ and their corresponding singular vectors $\{z^t\}_{t=1}^{l}$.
```

Figure 2: The ConstantTimeSVD algorithm [31].

The ConstantTimeSVD algorithm is described in Figure 2; it takes as input a matrix A and returns as output a “description” of an approximation to the top k left singular values and the corresponding singular vectors. This “description” of the approximations to the left singular vectors of A may, at the expense of one additional pass and linear additional space and time, be converted into an explicit approximation to the lest singular vectors of A by using $C = \breve{H}\Sigma_W Z^T$ to compute $\breve{H}$, whose columns are approximations of the left singular vectors of C. Note that γ in the ConstantTimeSVD algorithm is introduced to bound small singular values of C that may be perturbed by the second level of sampling; as indicated, the particular value of γ that is chosen depends on the norm bound which is desired [31].

3.3.5.2 Analysis of the implementation and running time

Assuming that optimal sampling probabilities are used, then in the ConstantTimeSVD algorithm the sampling probabilities pk can be used to select columns to be sampled in one pass and $O(c)$ additional space and time using the Select algorithm of [13]. Given the columns of A to be sampled, we do not explicitly construct the matrix C but instead perform a second level of sampling and select w rows of C with probabilities $\{q_i\}_{i=1}^{m}$ (described in ConstantTimeSVD algorithm) in order to construct the matrix W. We do this by performing a second pass and using $O(w)$ additional space and time, again using the Select algorithm. Then in a third pass we explicitly construct W; this requires additional space and time that is $O(cw)$. Then, given W computing $W^T W$ requires $O(cw)$ additional space and $O(c^2 w)$ additional time, and computing the SVD of $W^T W$ requires $O(c^3)$ additional time. The singular values and corresponding singular vectors thus computed can then be returned as the “description” of the solution. The total additional time for the ConstantTimeSVD algorithm is then $O(c^3 + cw^2)$; this is a constant if c and w are assumed to be a constant. To explicitly compute $\breve{H}_k$ would require k matrix-vector multiplications which would require another pass over the data and $O(mck)$ additional space and time [28].

3.4 Sampling-based techniques for matrix approximation

In this section, we introduce the two most common sampling-based techniques for matrix approximation and compare their performance on a variety of tasks.

Notations

For a matrix T ∈ $R^{a\times b}$, we define $T^{(j)}, j = 1 \dots\dots..b$ as the j^{th} column vector of T and $T_{(i)}, i = 1 \dots\dots..a$ as the i^{th} row vector of T. We denote by T_k the best rank k approximation to T, that is $T_k = argmin_{V\in R^{a\times b}}, rank(V) = \|T - V\|_{\xi,}$ where $\xi \in \{2, F\}$, $\|.\|_2$ denoted the spectral norm and $\|.\|_F$ the Frobenius norm of a matrix. Assuming that $rank(T) = r$ we can write thin singular value decomposition (SVD) of this matrix as $T = U_T\Sigma_T V_T^T$ where Σ_T is a diagonal and contains the singular values of T sorted in decreasing order and $U_T \in R^{a\times r}$ and $V_T \in R^{b\times r}$ are corresponding to left and right singular vector of T. Then we can describe T_k in terms of its SVD as $T_k = U_{T,K}\Sigma_{T,K}V_{T,K}^T$. Let $K \in R^{n\times n}$ be a symmetric positive semidefinite (SPSD) kernel and Gram matrix with $rank(K) = r \leq n$. We will write the SVD of K as $K = U\Sigma U^T$, and pseudo-inverse of $K\ as\ K^+ = \sum_{t=1}^{r} {\sigma_t}^{-1}\, U^{(t)}U^{(t)^T}$ and $K^+ = K^{-1}$ when K is full rank. For $k < r$ $K_k = \sum_{t=1}^{r} \sigma_t\, U^{(t)}U^{(t)^T} = U_K\Sigma_K U_K^T$ is the 'best' rank-*k* approximation to K, i.e. $K_k = argmin_{K\prime\in R^{n\times n}}, rank(K') = \|K - K'\|_{\xi\in\{2,F\}}$ with

$$\|K - K_k\|_2 = \sigma_{k+1} \tag{3.1}$$

$$\|K - K_k\|_F = \sqrt{\textstyle\sum_{t=k+1}^{r} \sigma_t^2} \tag{3.2}$$

We assume that we sample columns uniformly without replacement, though various methods have been proposed to select columns [14].

Let C denote the $n \times l$ matrix formed by these columns and W the $l \times l$ matrix consisting of intersection of these l columns with the corresponding l rows of K. Note that W is the SPSD since K is SPSD [7]. Without loss of generality, the columns and rows of K can be rearranged based on this sampling so that K and C can be written as Follows:

$$K = \begin{bmatrix} W & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix} \text{ And } C = \begin{bmatrix} W \\ K_{21} \end{bmatrix} \tag{3.3}$$

3.4.1 Nystrom Method

The Nystrom method was initially introduced as a quadrature method for numerical integration, used to approximate Eigen function solutions [15]. More recently, it was presented in Williams and Seeger (2000) to speed up kernel algorithms and has been used in applications ranging from manifold learning to image segmentation Nystrom method uses **W** and **C** from (4.3) to approximate K, and for uniform sampling of columns, the Nystrom method generates a rank – k approximation $\widetilde{K}$ of K for $k < n$ defined by :

$$\widetilde{K}_k^{nys} = CW_k^{+}C^T \approx K \tag{3.4}$$

Here W_k is the best rank k approximation of W for Frobenius norm and W_k^{+}denotes the pseudo-inverse of W_k. If we write the SVD of W as $W = U_w \Sigma_w U_w^T$, plugging into equation 4.4 we can

$$\widetilde{K}_k^{nys} = CU_{w,k}\Sigma_{w,k}^{+}U_{w,k}^T C^T$$

$$= (\sqrt{\frac{l}{n}}CU_{w,k}\Sigma_{w,k}^{+})(\frac{n}{l}\Sigma_{w,k})(\sqrt{\frac{l}{n}}CU_{w,k}\Sigma_{w,k}^{+})^T \tag{3.5}$$

And hence the Nystrom method approximates the top k singular values (Σ_k) and singular vectors (U_k) of as:

$$\tilde{\Sigma}_{nys} = (\frac{n}{l})\Sigma_{w,k} \text{ and } \widetilde{U}_{nys} = \sqrt{\frac{l}{n}}CU_{w,k}\Sigma_{w,k}^{+} \tag{3.6}$$

The time complexity of compact SVD on W is $O(l^2k)$ matrix multiplication C takes $O(nlk)$ hence the total complexity of Nystrom method is $O(nlk)$ [15].

3.4.2 Column sampling method

The Column sampling method was introduced to approximate the SVD of any rectangular matrix. It generates approximations of K by using the SVD of C. If we write the SVD of C as $C = U_C\Sigma_C V_C^T$ then the column sampling method approximate the top k singular values (Σ_k) and singular vector (U_k) of K as [27]:

$$\tilde{\Sigma}_{col} = \sqrt{\frac{l}{n}}\,\Sigma_{C,K} \text{ And } \widetilde{U}_{col} = U_c = CV_{c,k}\Sigma_{c,k}^{+} \tag{3.7}$$

The runtime of column sampling method is dominated by the SVD of C. The algorithm takes $O(nlk)$ time to perform compact SVD on c but still more expensive than the Nystrom method as the constants for SVD are greater than those for the $O(nlk)$ matrix multiplication step in the Nystrom method [15].

Low rank approximation

We will focus on the accuracy of low rank approximation of kernel matrices is tied to the performance of kernel- based learning algorithms. Furthermore, the connection between kernel matrix approximation and the hypothesis generated by several widely used kernel-based learning algorithms has been theoretically analyzed. Hence, accurate low-rank approximations are of great practical interest in machine learning. The optimal is K_k is given by:

$$K_k = U_k \Sigma_k U_k^T = U_k U_k^T K = K U_k U_k^T \tag{3.8}$$

where the columns of U_k are the k singular vectors of K corresponding to the top k singular values of K. We refer to $U_k \Sigma_k U_k^T$ as Spectral Reconstruction, since it uses both the singular values and vectors of K and $U_k U_k^T K$ as Matrix Projection, since it uses only singular vectors to compute the projection of K onto the space spanned by vectors U_k. These two low-rank approximations are equal only if $\Sigma_k$and $U_k$contains the true singular values and singular vectors of K. Since this is not the case of approximate methods such as Nystrom and Column sampling these two measures generally give different errors. Thus we analyze each measure separately in the following sections [22].

Matrix projection

For column sampling using (3.7), the low rank approximation via matrix projection is

$$\widetilde{K}_k^{col} = \widetilde{U}_{col,k} \widetilde{U}_{col,k}^T K = U_{C,k} U_{C,k}^T K = C((C^T C)_k)^+ C^T K \tag{3.9}$$

Where

$$(C^T C)_k^{-1} = V_{C,k} (\Sigma_{C,k}^2)^+ V_{C,k}^T. \text{ Clearly, if } k = l, (C^T C)_k = C^T C \ .$$

Similarly, using (3.6), the Nystrom matrix projection is

$$\widetilde{K}_k^{nys} = \widetilde{U}_{nys,k}\,\widetilde{U}_{nys,k}^T K = \frac{l}{n} C (W_k^2)^+ C^T K \qquad (4.10)$$

As shown in (3.9) and (3.10), the two methods have similar expressions for matrix projection, except that $C^T C$ is replaced by a scaled W^2. The scaling term appears only in the expression for the Nystrom method. We now present theorem 1 and observations 1 and 2, which provide further insights about these two methods in the context of matrix projection [14].

Theorem 1 *The Column sampling and Nystrom matrix projections are of the form* $U_C R U_C^T K$*, where* $R \in R^{l\times l}$ *is SPSD. Further, Column sampling gives the lowest reconstruction error (measured in* $\|.\|_F$*) among all such approximations if* $k = l$*.*

Observation 1 For $k = l$ *matrix projection for column sampling reconstruction C exactly. This can be seen by block- decomposition K as :* $[C\ \ \bar{C}]$*, where* $\bar{C} = [K_{21} K_{22}]^T$*, and using (3.9)*

$$\widetilde{K}_l^{col} = C(C^T C)^+ C^T K = [C\ \ C(C^T C)^+ C^T \bar{C}] = [C\ \ \bar{C}].$$

Observation 2 For $k = l$, the span of the orthogonalized Nystrom singular vectors equals the span of $\widetilde{U}_{col}$. Hence, matrix projection is identical for Column sampling and Orthonormal Nystrom for $k = l$.

Matrix projection approximations are not necessarily symmetric and require storage of and multiplication with K. Hence, although matrix projection is often analyzed theoretically, for large-scale problems, the storage and computational requirements may be inefficient or even infeasible [14].

Spectral reconstruction

Using (4.6), the Nystrom reconstruction is :

$$\widetilde{K}_k^{nys} = \widetilde{U}_{nys,k} \tilde{\Sigma}_{nys,k} \widetilde{U}_{nys,k}^T = C W_k^+ C^T \qquad (3.11)$$

Where $k = l$, this approximation perfectly reconstructs three blocks of k, and K_{22} is approximated by the Schur Complement of W in K. The Column sampling spectral reconstruction has a similar from [14] (4.6):

$$\widetilde{K}_k^{col} = \widetilde{U}_{col,k}\widetilde{\Sigma}_{col,k}\widetilde{U}_{col,k}^T = \sqrt{\frac{n}{l}}\,C((C^TC)_k^{\frac{1}{2}})^+C^T \qquad (3.12)$$

In contrast of matrix projection, the scaling term now appears in the column sampling reconstruction. To analyze the two approximations, we consider an alternative characterization using the fact that $K = X^TX$ for some $X \in R^{N\times n}$. We define a zero-one sampling matrix, $S \in R^{n\times l}$, that selects l columns from K, that is, $C = KS$. Further, $W = S^TKS = X'^TX'$, where $X' \in R^{N\times l}$ contains l sampled columns of X and $X' = U_{X'}\Sigma_{X'}V_{X'}^T$ is the SVD of X' . We now present two results. Theorem 2 shows that the optimal spectral reconstruction is data dependent and may differ from the Nystrom and column sampling approximations. And theorem 3 reveals that in certain instances the Nystrom method is optimal, while the column sampling methods enjoy no such guarantee [12].

Theorem 2 *Column sampling and Nystrom spectral reconstruction of rank k are of the form $X^TU_{X',k}ZU_{X',k}^TX$ where $Z \in R^{k\times k}$ is SPSD. Further, among all approximations of this, neither the Column sampling nor the Nystrom approximation is optimal (in $\|.\|_F$).*

Theorem 3 Let $r = \text{rank}(K) \le k \le l$ and $\text{rank}(W) = r$. Then, the Nystrom approximation is exact for spectral reconstruction. In contrast, Column sampling is exact *iff* $W = ((l/n)C^TC)^{1/2}$.

3.4.3 Modified Nystrom Approximation

Given a $m \times m$ symmetric matrix A, one needs to select $c(\ll m)$ columns of A to form a matrix $\mathrm{C} \in \mathbb{R}^{m\times c}$ to construct the standard or modified Nystrom approximation. Without loss of generality, A and C can be permuted such that

$$K = \begin{bmatrix} W & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix} \text{ And } C = \begin{bmatrix} W \\ K_{21} \end{bmatrix} \qquad (3.3)$$

Where w is of size $c \times c$. The standard Nystrom approximation is defined by

$$\widetilde{K}_k^{nys} = CW_k^+C^T \approx K \qquad (3.4)$$

And the modified Nystrom approximation is [24]

$$\tilde{A}_c^{mod} = CU^{mod}C^T = C(C^TA(C^+)^T)C^T$$

Here the $c \times c$ matrices $U^{nys} \triangleq W^+$ and $U^{mod} = C^+A(C^+)^T$ are called the intersection matrices. We see that the only difference between the two models is their intersection matrices [15].

For the approximation CUC^T constructed by either of the methods, given a target rank k, we hope the error ratio

$$f = \|A - CUC^T\|_\xi \big/ \|A - A_k\|_\xi, \quad (\xi = F \; or \; 2), \tag{3.5}$$

is small as possible. For the standard Nystrom method, whatever a column selection algorithm is used, the ratio f must grow with the matrix size m when c is fixed [17].

Lemma 1 (Lower Error Bound of the Standard Nystrom Method). *Whatever* a *column sampling algorithm is used, there exists an $m \times m$ SPSD matrix A such that the error incurred by the standard Nystrom method obeys:*

$$\|A - CW^+C^T\|_F^2 \geq \Omega\left(1 + \frac{mk}{c^2}\right)\|A - A_k\|_F^2,$$
$$\|A - CW^+C^T\|_2 \geq \Omega\left(\frac{m}{c}\right)\|A - A_k\|_2$$

Here k is an arbitrary target rank, and c is the number of selected columns.

Thus, when the matrix size m is large, the standard Nystrom approximation is very inaccurate unless a large number of columns are selected. By comparison, for the modified Nystrom method, the error ratio f remains constant for a fixed c and a growing m. Therefore, the modified Nystrom method is more accurate than the standard Nystrom method [16].

However, the accuracy gained by modified Nystrom method is the cost of higher time and space complexities. Computing the intersection matrix $U^{nys} = W^+$ only takes time $O(c^3)$ and space$O(c^2)$, while computing $U^{mod} = C^+A(C^+)^T$ naively takes time $O(c^3)$, and space $O(c^2)$, while computing $U^{mod} = C^+A(C^+)^T$ naively takes time $O(mc^2) + T_{Multiply}(m^2c)$ and space $O(mc)$ [21].

Approximation algorithm

Data: $n \times n$ Gram matrix G and $c \leq n$.

Result: $n \times n$ matrix $\tilde{G}$.

- Pick c columns of G, uniformly at random with replacement; Let I be the set of indices of the sampled columns.
- Let C be the $n \times c$ matrix containing the sampled columns.
- Let W be the $c \times c$ submatrix of G whose entries are $G_{ij}, i \in I, j \in I$.
- Return $\tilde{G} = CW^{+}C^{T}$ [8].

3.5 Summary

In this section, we discussed various low rank approximations algorithms. We discussed two algorithms to compute the SVD of a matrix $A \in \mathbb{R}^{m \times n}$ which do not require that A be stored in RAM, but additional space required is either linear in m + n or is a constant independent of m and n; error bounds for both algorithms are proven with respect to both the Frobenius and spectral norms. We also present sampling based matrix approximation i.e, standard Nystrom method and column sampling method for the selection of representative columns. In the next chapter we presented application of the approximation. Matrix approximation helps to speed up the kernel methods such as support vector machine, kernel ridge regression, kernel principle component analysis.

Chapter 4
Applications

In previous chapter, we discussed various low rank approximation techniques based on sampling and non-sampling that generate approximation of the kernel matrices. We analyzed the effectiveness of these algorithms. In this chapter, we discuss specific application of these approximations particularly in context of large scale applications. We discus Nystrom low rank approximation for efficient linearization of a non-linear SVM, and provide theoretical error analysis.

4.1 Support vector machine

Support Vector Machine (SVM) delivers state-of-the-art results in non-linear classification, but the need to maintain a large number of support vectors poses a challenge in large scale training and testing. To scale up kernel SVM on limited resources, we propose a low rank linearization approach that transforms a non-linear SVM to a linear one via a novel, approximate empirical kernel map computed from efficient low-rank approximation of kernel matrices. Support vector machine is introduced by Vapnik and Cortes in 1995 for classification has been widely used in various scientific domains. The use of kernels allows the input samples to be mapped to a Reproducing Kernel Hilbert Space (RKHS), which is crucial to solving linearly nonseparable problems. While kernel SVMs deliver the state-of-the-art results, the need to manipulate the kernel matrix imposes significant computational bottleneck, making it difficult to scale up on large data[32].

Here, we discus general approach towards linearization kernel SVM for large scale problems. This is achieved by low rank approximation to the kernel matrix of the kernel matrix where the low-rank factors can be deemed as providing a novel, approximate empirical kernel map that explicitly transforms the kernel SVM into a linear space; the resultant linear SVM can then be solved efficiently using state-of-the-art linear solvers. This framework has several desirable properties. First, it can be applied to any (nonlinear) SVM variations and any Positive semi-definite (PSD) kernel; second, both the dimension of the approximate kernel map and the number of “basis” in the decision function can be freely controlled by the user, therefore guaranteeing efficient training and testing; third, theoretical bounds can established on the approximation, which in turn provides important guidance on sampling based low-rank approximation; last and most important, this approach inherits the rich repesentability of kernel SVM as well as the high efficiency of linear SVM, and ideally can be applied to arbitrarily large problems with limited computing resources via advanced incremental learning techniques[33].

4.1.1 Transforming Non-linear SVM into Linear SVM

We shoe that a nonlinear SVM can be cast exactly as a linear SVM using symmetric decomposition of kernel matrices. Suppose we are given a set of training pairs (x_i, y_i), where $x_i \in \mathbb{R}^{d\times 1}$'s are concatenated as row in the $n \times d$ training data matrix X_r and $y_i \in \pm 1$'s are stored in the training level $y_r \in \mathbb{R}^{n\times 1}$. Similarly we have m testing samples in $X_e \in \mathbb{R}^{m\times d}$. Assume we use a positive semi-definite(PSD) kernel function $k(x_i, x_j) = \langle \psi(.), \psi(.) \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where $\psi(x_i)$ is the associated mapping function that implicitly maps the data point from the input space to feature space. Define the kernel matrix on the training and testing data in blocks as $K = \begin{bmatrix} K_{rr} & K_{re} \\ K_{er} & K_{ee} \end{bmatrix}$, $k_{rr} \in \mathbb{R}^{n\times n}$ is the kernel matrix defined on X_r, $K_{ee} \in \mathbb{R}^{m\times m}$ is defined on $X_e$and $K_{er} \in \mathbb{R}^{m\times n}$ is defined on X_e and X_r. Training a kernel SVM is to find the classifier $f(x) = sign(w^T\psi(x) + b)$ by solving the optimization[35]

$$\min_{w,\xi,b} \quad \frac{1}{2}\|w\|^2 + C\sum \xi_i \tag{4.1}$$

$$y_i(w^T\psi(x) + b) \geq 1 - \xi_i.$$

where $C > 0$ is the regularization parameter. In the following we discuss how to transform the non-linear (kernel) SVM into linear SVM via decomposition of the PSD kernel matrix.

Proposition 1 *Given training data X_r and y_r, and test data X_e. A kernel SVM (4.1) trained on X_r, y_r, and tested on X_e is equivalent to a linear SVM trained on F_r, y_r and tested on F_e, where*

$$K = \begin{bmatrix} F_r \\ F_e \end{bmatrix} [{F_r}^T \quad {F_e}^T] \tag{4.2}$$

is any decomposition of the psd kernel matrix K evaluated on (X_r, X_e), and the factor $F_r \in \mathbb{R}^{n \times p}$ and $F_r \in \mathbb{R}^{m \times p}$can be deemed as "virtual samples" whose dimensionality p is the rank of K.

Proof 1 *The dual of the kernel SVM optimization (4.1) can be written as*

$$\min_\alpha \frac{1}{2} \alpha^T Q_{rr} \alpha - \sum \alpha_i \tag{4.3}$$

$$s.t.\ 0 \le \alpha_i \le C, \sum \alpha_i y_i = 0$$

$$Q_{rr} = K_{rr} \odot (y_r y_r^T),$$

where α is the Lagrangian multipliers and $\odot$ is the entry-wise product between matrices. The prediction on the testing data can be written as

$$\hat{y}_e = K_{er}.(\alpha \odot y_r), \tag{4.4}$$

Let $F = \begin{bmatrix} F_r \\ F_e \end{bmatrix}$, and be the i^th columnin F^T. Assume we train a linear SVM using F_r and y_r, with the primal form

$$\min_{\bar{w}, \bar{\xi}, b} \frac{1}{2} \|\bar{w}\|^2 + C \sum \bar{\xi}_i \tag{4.5}$$

$$y_i(\bar{w}^T F_i + b) \ge 1 - \bar{\xi}_i$$

The dual can then be written as

$$\min_{\bar{\alpha}} \frac{1}{2} \bar{\alpha}^T \bar{Q}_{rr} \bar{\alpha} - \sum_i \bar{\alpha}_i \tag{4.6}$$

$$s.t.\ 0 \le \bar{\alpha}_i \le C, \sum \bar{\alpha}_i y_i = 0$$

$$\bar{Q}_{rr} = (F_r F_r^T) \odot (y_r y_r^T).$$

Then the prediction on F_e is

$$\bar{\hat{y}}_e = F_e^T F_r . (\bar{\alpha} \odot y_r) \tag{4.7}$$

Comparing (4.3) and (4.6), we can see these two problems are equivalent and lead to the identical optimal solution $\alpha^* = \bar{\alpha}^*$ since $F_r F_r^T = K_{rr}$ (4.2). Plugging the optimal solutions into (4.4) and (4.7), and nothing the fact $F_e^T F_r = K_{er}$ (4.2), we can see the prediction in (4.4) and (4.7) are identical, i.e., $\hat{y}_e = \bar{\hat{y}}_e$. The kernel SVM (4.1) and linear SVM (4.5) are equivalent.
Proposition 1 shows that any kernel SVM can as an equivalent linear SVM by decomposition of the kernel matrix $\mathrm{K} = \mathrm{F}F^T$(4.2), where F serves as an empirical kernel map or virtual samples. The positive semi-definiteness of the kernel matrix guarantees that decomposition (4.2) always exists. When only training data is used, the decomposition [34]

$$K_{rr} = F_r F_r^T \tag{4.8}$$

That allows to recover the Langrangian multipliers in the original nonlinear decision function (4.3).

Motivated by this observation, we consider learning large scale kernel SVM in two stages: first, transform

it to a linear SVM using kernel eigenvalue decomposition; second, solve a linear SVM efficiently. Obviously, the key to the success of such linearization is an efficient decomposition of the PSD kernel matrix to obtain the empirical kernel map F_r (4.8).

4.2 SVM Low-rank Linearization

The kernel matrix is the key building block of kernel methods: its entries recover the inner product of the samples in the kernel induced feature space. This avoids explicit computation of the mapping $\psi(x)$ (which can be potentially infinite dimensional) but instead one only needs to perform kernel evaluations in the input space. Such "kernel trick" allows the model to capture highly non-linear classification concepts, but at the cost of manipulating the $n \times n$ kernel matrix. In comparison, linear SVM assumes a simple and explicit mapping (i.e., $\psi(x) = x$) which renders great potential computational efficiency[32].

Proposition 1 provides a new perspective on the kernel map embodied through the empirical kernel matrix K. It shows that any exact decomposition of the kernel matrix can preserve the dot products among feature induced kernel mapping $\psi(x_i)'$s via a new, empirical kernel map F_i's, as

$$K_{ij} = \langle \psi(x_i), \psi(x_j) \rangle = \langle F_i, F_j \rangle$$

This is the key to transforming a non-linear SVM into an explicit linear counterpart. It bridges the gap between non-linear and linear SVMs and opens the possibility of training large scale non-linear SVM by advanced linear solvers.

Given an $n \times n$ kernel matrix on the training set, with the eigenvalue decomposition

$$K_{rr} = U_r \Sigma_r U_r^T \tag{4.9}$$

where $U_r \in \mathbb{R}^{n \times n}$ contains orthogonal eigenvectors such that $U_r^T U_r = I_{n \times n}$, and Σ_r is a diagonal matrix whose diagonal entries are eigenvalues (λ) in descending order. Then the empirical kernel map on training data (4.8) can be chosen as

$$F_r = U_r \Sigma_r^{1/2} \tag{4.10}$$

Theoretically, the eigenvalue decomposition provides the optimal rank-$k$approximation of the kernel matrix

$$\min_{rank(\widetilde{K}_{rr})=k} \left\| K_{rr} - \widetilde{K}_{rr} \right\|_F^2 = \sum_{i=k+1}^{n} \lambda_i^2 \tag{4.11}$$

where $\widetilde{K}_{rr}$ is the rank-k approximated matrix. In other words, given a dimension, k, the feature map

$$F_r^{(k)} = U_r^{(k)} (\lambda_r^{(k)})^{1/2} \tag{4.12}$$

composed of top eigenvectors/values is the optimal since the inner products it recovers is the closest to K_{rr} among all rank-k kernel maps, which equals sum of squared minimum n−k eigenvalues as shown in (4.11) [34].

However, exact computation of the top k eigenvectors requires $O(n^2k)$ time and $O(n^2)$ space, which is not suitable for large problems. So we seek an approximate decomposition here. We are interested in the Nystrom method that has gained great popularity recently in scaling up kernel based algorithms [36]. Given a set of training samples X_r and the kernel matrix K_{rr}, the Nystrom method chooses a subset of k samples z and provides a rank-k approximation of the kernel matrix as

$$\widetilde{K}_{rr} = K_{rz}K_{zz}^{-1}K_{rz}^T \tag{4.13}$$

where $K_{rz} \in \mathbb{R}^{n\times k}$ is the kernel matrix on X_r and X_z, and $K_{zz} \in \mathbb{R}^{k\times k}$ is the kernel matrix on z.

Next we show how to approximate the optimal kernel map (4.12) using the Nystrom low-rank approximation (4.13) [37]. Let the eigenvalue decomposition of K_{zz} be $U_z\Sigma_z U_z^T$, then (4.13) can be written as

$$\widetilde{K}_{rr} = \tilde{F}_r\tilde{F}_r^T$$

$$\tilde{F}_r = K_{rz}U_z\Sigma_z^{-1/2} \tag{4.14}$$

the rank-k approximation by the Nystrom method (4.13) provides a natural approximation to the optimal kernel map F_r (4.12). Consider the extreme case where the landmark set z in the Nystrom method is chosen as the whole data set: then $U_z \to U_r$, $\Sigma_z \to \Sigma_r$, $K_{rz} \to K_{rr}$ and as a result we have, when $|z| \to n$

$$\begin{aligned} K_{rz}U_z\Sigma_z^{-1/2} &= K_{rr}U_r\Sigma_r^{-1/2} \\ &= U_r\Sigma_r U_r^T U_r\Sigma_r^{-1/2} \\ &= U_r\Sigma_r^{1/2} \end{aligned}$$

Nystrom's approximation error in the form of Frobenius norm

$$\delta_F = \left\| K_{rr} - \widetilde{K}_{rr} \right\|_F \tag{4.15}$$

or in the form of spectral norm

$$\delta_2 = \left\| K_{rr} - \widetilde{K}_{rr} \right\|_2 \tag{4.16}$$

These are the error bound in the rank$-k$ approximation[36]. We analyze the quality of SVM classifier using Nystrom low rank approximation.

4.3 Summary

In this chapter we discussed support vector machine for large scale datasets. Kernel methods suffer from highly time complexity. We discussed a non-linear SVM to a linear one via a novel, approximate empirical kernel map computed form efficient low-rank approximation of kernel matrices. We showed the effectiveness of the approximation in theoretically. In next chapter, we discuss the implementation result of matrix approximation and the low rank support vector machine using real data sets.

Chapter 5

Proposed Approach: Efficient Nystrom Method

In this section, we introduce sampling based technique for matrix approximation. We assume that we sample columns uniformly without replacement. Our proposed method, Efficient Nystrom method reduced the error for approximated matrix in term of Frobenius norm and spectral norm.

Notation

Let $H \in \mathbb{R}^{i\times j}$ be an arbitrary matrix. $H^{a}, a = 1,2 \ldots \ldots i$, as the a^{th} row vector of H and $H_b, b = 1,2, \ldots \ldots, b$, as the b^{th} column vector of H $\|.\|$ represents the norm of the vector. Moreover, $H^{(a:b)}$ refers to the a^{th} through b^{th} columns of H and $H_{(a:b)}$ refers to the a^{th} through b^{th} rows of H. If $rank(H) = r$, then the thin Singular Value Decomposition of the H as

$$H = U_H \Sigma_H V_H^T \tag{5.1}$$

where Σ_H is the diagonal matrix that contains the singular values of H in decreasing order and $U_H \in \mathbb{R}^{i\times r}$ and $V_H \in \mathbb{R}^{j\times r}$ both are orthogonal columns that contains the left singular vectors and right singular vectors of H for its singular values. We show that H_k the best $rank - k$ approximation for H [22].

$\mathrm{H}_k = argmin_{V\in\mathbb{R}^{i\times j}, rank(V)=k} \|\mathrm{H} - \mathrm{V}\|_{\xi}$, where $\xi \in \{2, F\}$ and $\|.\|_2$ represents the spectral norm and $\|.\|_F$ represents the Frobenius norm of the matrix. We resents the SVD for top k singular values of H as

$$H_k = \mathrm{U}_{H,k} \Sigma_{H,k} \mathrm{V}_{H,k}^T \tag{5.2}$$

where $\Sigma_{H,k}$ represents the diagonal matrix of top k singular values of H and $U_{H,k}$ is the left singular vector and $V_{H,k}$ is the right singular vector.

Now let $M \in \mathbb{R}^{N\times n}$ matrix, then we have to make it symmetric positive semidefinite (SPSD) kernel or Gram matrix using linear kernel such that $X = M^T M$ where $X \in \mathbb{R}^{n\times n}$ and X be a SPSD matrix. SVD for the matrix X as $X = U\Sigma U^T$, where U is the orthogonal vector of $X$and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \dots \dots, \sigma_r)$ is the diagonal matrix of X. rank of the matrix X is r $\leq n$. We have to find the pseudo-inverse of X as

$$\mathrm{X}^{+} = \sum_{l=1}^{r} \sigma_l{}^{-1}\, \mathrm{U}^{(l)}\mathrm{U}^{(l)^{\mathrm{T}}} \tag{5.3}$$

And X^{+}= X^{-1}, both are equal when X is full rank matrix.

Let k $<$ r, then

$$\mathrm{X_k} = \sum_{l=1}^{r} \sigma_l\, \mathrm{U}^{(l)}\mathrm{U}^{(l)^{\mathrm{T}}} = \mathrm{U}_k \Sigma_k \mathrm{U}_k^{\mathrm{T}} \tag{5.4}$$

$\mathrm{X_k}$ is the best rank $-k$ approximation to X. Quality of approximation is measure by the Frobenius norm and spectral norm.

$$\mathrm{X}_k = \mathrm{argmin}_{\mathrm{X}' \in \mathbb{R}^{n \times n}, \mathrm{rank}(\mathrm{X}')=k} \|\mathrm{X} - \mathrm{X}'\|_{\xi} \tag{5.5}$$

where $\xi \in \{2, F\}$, with $\|X - X_k\|_2 = \sigma_{k+1}$ and $\|\mathrm{X} - \mathrm{X}_k\|_F = \sqrt{\sum_{l=k+1}^{r} \sigma_l^2}$

Now we focuses on the generating an low rank approximation $\widetilde{\mathrm{X}}$ of X based on sampling algorithm. We have to choose the column from the original matrix such as $l \ll n$. There is a assumption, we sample the columns uniformly without replacement, there are various method to select the columns. We have to choose sampling matrices such as D and Q, where D denote the $n \times l$ matrix formed by the sampling columns. And In this section, we introduce sampling based technique for matrix approximation. We assume that we sample columns uniformly without replacement. Efficient Nystrom method reduced the error for approximated matrix in term of Frobenius norm and spectral norm [18].

Now we focuses on the generating an low rank approximation $\widetilde{\mathrm{X}}$ of X based on sampling algorithm. We have to choose the column from the original matrix such as $l \ll n$. We take an assumption, we sample the columns uniformly without replacement, there are various method to select the columns. We have to choose sampling matrices such as D and Q, where D denote the $n \times l$ matrix formed by the sampling columns. Q denote $l \times l$ such as intersection of l rows with the l columns of X. X is the SPSD matrix so Q also be a SPSD matrix [25].

Now we can write the D and Q such as

$$\mathrm{X} = \begin{bmatrix} Q & X_{21}^T \\ X_{21} & X_{22} \end{bmatrix} \text{ And } D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \tag{5.6}$$

We select l columns from X_{21}^Tnamed as D_1 and l columns from X_{22} named as D_2. Then combine D_1 and D_2 form as D matrix with $n \times l$. We change the sampling technique as describe in Nystrom method [12]. To generate the low rank approximation of X, we need D and SVD of Q.

5.1 Efficient Nystrom method

When Nystrom method introduced, it was used as quadrature method for numerical integration and eigenfunction solution approximated by Nystrom method. Recently, to speed up the kernel methods and used in the application of manifold learning to image segmentation Nystrom method introduced by Williams and Seeger [20]. Accuracy of efficient Nystrom method is calculated by the Frobenius norm$\left\| X - \widetilde{X_k} \right\|_F$. Efficient Nystrom method uses the Q and D to approximate the kernel matrix X. Efficient Nystrom method is effectively able to generate the rank k approximation $\widetilde{\mathrm{X}}$ of X for $k < n$, defined by:

$$X_k^{enys} = DQ_k^+ D^T \approx X \tag{5.7}$$

where Q_k is the best k- rank approximation of Q for the Frobenius norm and the Q_k^+ denotes the the pseudo-inverse of Q_k. SVD of the Q as $\mathrm{Q} = \mathrm{U}_Q \Sigma_Q \mathrm{U}_Q^{\mathrm{T}}$, then put it in equation (5.7) we can write

$$X_k^{enys} = D\mathrm{U}_{Q,k} Q_{Q,k}^+ U_{Q,k}^T D^T$$

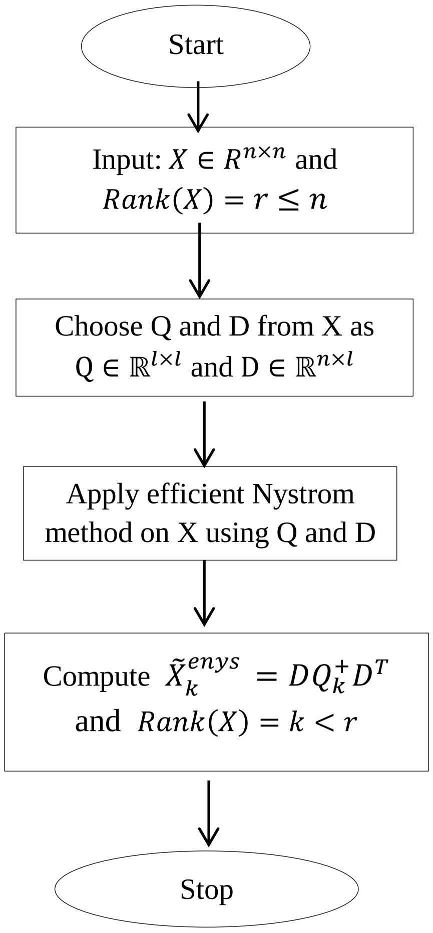
$$= (\sqrt{\frac{l}{n}} DU_{Q,k}\Sigma_{Q,k}^+)(\frac{n}{l}\Sigma_{Q,k})(\sqrt{\frac{l}{n}} DU_{Q,k}\Sigma_{Q,k}^+)^T$$

Top k singular values (Σ_k) and singular vectors $(U_k)$of X approximated by efficient Nystrom method as:

$$\tilde{\Sigma}_{enys} = (\frac{n}{l})\Sigma_{Q,k} \text{ and } \widetilde{U}_{enys} = \sqrt{\frac{l}{n}} D U_{Q,k} \Sigma^{+}_{Q,k} \tag{5.8}$$

Time complexity of SVD of Q is $O(l^3)$ and multiplication with Q takes $O(kln)$, so the time complexity of the efficient Nystrom method is $O(l^3 + kln)$. .

5.2 System design



Start
Input: $X \in R^{n\times n}$ and $Rank(X) = r \leq n$
Choose Q and D from X as $Q \in \mathbb{R}^{l\times l}$ and $D \in \mathbb{R}^{n\times l}$
Apply efficient Nystrom method on X using Q and D
Compute $\tilde{X}_k^{enys} = DQ_k^{+}D^T$ and $Rank(X) = k < r$
Stop

Figure 1 Flow Chart of the Efficient Nystrom Method

5.3 Algorithm (Efficient Nystrom method)

Input: $M \in \mathbb{R}^{N\times n}$ or $X \in \mathbb{R}^{n\times n}$($X$ SPSD matrix) , rank (M) = r

Output: $\tilde{X} \in \mathbb{R}^{n\times n}$ and $\operatorname{rank}(\tilde{K}_k) = k, k \ll r$

```
1. Check given matrix is SPSD
   if yes then pass it for approximation
   else make it SPSD using linear kernel($X = M^T M$)
2. for i=1 to $l$.
   Pick $l$ columns of $X$ and $\operatorname{rank}(\tilde{K}_k) = k$, $l$ rows
3. $Q$ is the SPSD matrix containing the $l$ columns and $l$ rows
4. for i= $l + 1$ to $2l$
   pick next $l$ columns.
5. D is the matrix with $n \times l$ size
6. Compute the SVD of the Q as $Q = U_Q \Sigma_Q U_Q^T$
7. Select top $k$ singular values and vector
8. $\Sigma_k$ and $U_k$ are the top k singular values and singular vector
9. Return $\tilde{\Sigma}_{enys} = (\frac{n}{l})\Sigma_{w,k}$ and $\tilde{U}_{enys} = \sqrt{\frac{l}{n}} D U_{Q,k} \Sigma_{Q,k}^{+}$
10. Compute $\tilde{X}_{enys} = \tilde{U}_{enys} \tilde{\Sigma}_{enys} \tilde{U}_{enys}^T$
11. Return $\tilde{X}_{enys}$
```

Figure 2: Efficient Nystrom algorithm for generating the low rank approximation

5.4 Summary

In this chapter we discussed our proposed algorithm efficient Nystrom method. Efficient Nystrom method uses different sampling method from standard Nystrom method. We discussed algorithm and the flowchart of efficient Nystrom method. In the next chapter, we show the implementation results of the standard Nystrom and efficient Nystrom method.

Chapter 6
Implementation and results

6.1 Datasets

In the implementation, we used 3 types of data sets. First, random generated data with 1000 instances and 1000 attribute using linear kernel. Second, letter data set having 16000 of instances and 16 attribute. For making the kernel matrix (SPSD) of the letter data set, need to be kernel function so we use radical basis function (sigma=.1). We have to covert the matrix into kernel matrix because efficient Nystrom method is applicable only for kernel matrix. Third, abalone data set having 4177 instances and 8 attribute. For making the kernel matrix (SPSD) of the abalone data set, need to be a kernel function [].

Efficient Nystrom method gives low construction error as compare to the standard Nystrom method.

Table 1 Description of the datasets used in our experiments comparing sampling-based matrix approximations

Data set	No. of Instance	No. of attribute	Kernel
Random data	1000	10000	Linear
Letters	16000	16	RBF
Abalone	4177	8	RBF

We compare the results of the all three datasets for sampling methods (standard Nystrom and efficient Nystrom). Random data having large error as compare to real datasets. All the same datasets used in the kernel methods such as support vector machine. In the application part, we will show how approximation works for kernel methods. Table 1 represents the three data set, random data set, letters and abalone. We will compare standard Nystrom method and efficient Nystrom method based on the results of these data sets. Random data is self-generated and letter data set and abalone data set is available on the UCI repository. UCI repository is online resource for data sets [9][10].

6.2 System requirements

1. Hardware requirements

- Windows 7 operating system or Ubuntu 12.04
- 2 GB RAM
- 2.4 GHz dual core processor
- 160 GB hard drive

2. Software requirements

For statistical computing and graphics, there exists a programming language named R. T language is widely used by the data miners and the statisticians for analyzing the data and developing statistical software.

R was developed by the Ross Ihaka and Robert Gentleman and R Development Core Team at the University of Auckland, New Zealand. R is named partly after the first letter of first name of its first two authors and partly as the generalization of S language.

R is a GNU project. R is primarily written in C and FORTRAN. It is freely available under the GNU General Public License. Pre-compiled versions of R are provided for different operating systems. A command line interface is used by R. Graphical front-ends are available for developing the user-friendly application in R. Various GUIs are available for R programming like RStudio, Deduvcer, and Java GUI for R, Rattle GUI, R Comander, RGUI, RWeka RKWard etc.

RStudio is used for simulating the proposed approach

RStudio is a free and cross-platform open source IDE (integrated development environemnt) for R. Two editions are available for RStudio: RStudio Desktop and RStudio Server. RStudio Desktop runs locally as a regular desktop application. Via RServer, RStudio can be accessed using web browser. The R-Server runs on remote Linux server. RStudio desktop is available for Microsoft Windows, Linux and Mac OS X. RStudio is written in C++. Its GUI is developed by using Qt framework [26].

6.3 Results of matrix approximation

In the results section, we compare sampling methods for low rank approximation. Comparisons of the approximation methods based on the error bound. The error bound is calculated suing the Frobenius norm and spectral norm. We will show the comparisons of the sampling methods based on the three types of data set. One is random data set and reaming two is the real data sets. Results of the low rank approximation methods are simulated in R. To implement the sampling based approximation, we have to need some R-package such as matrix, kernlab, etc.

Table 2 Analysis of two sampling methods based on random generated data with various ranks of the matrix in Frobenius norm

Target rank	Standard Nystrom method	Efficient Nystrom method
90	313.8549	305.1442
80	319.1229	310.2205
70	326.0917	315.2669
60	332.444	322.5728
50	340.3372	330.8423

Table 2 shows the results based on the Frobenius norm. Frobenius norm shows the quality of the approximation. Efficient Nystrom method has good quality of low rank approximation as compared the standard Nystrom method. As the rank of the matrix increase the approximation error becomes less. Means we select more spectral feature as increase the rank. In the table 2, we take the rank form 50 to 90. When the rank is 90 the Frobenius error for efferent Nystrom is 305.1442 and for standard Nystrom is 313.8549. And decrease the rank up to 50 then this error become 330.8423 for the efficient Nystrom method and 340.3372 for standard Nystrom method. So the results show that efficient Nystrom method has better quality bound.

Figure 3 represents the comparison graph between the standard Nystrom method and efficient Nystrom method. Plots show that efficient Nystrom method has low reconstruction error as compare to the standard Nystrom method. These errors known as the Frobenius error that gives the quality of low rank approximation.

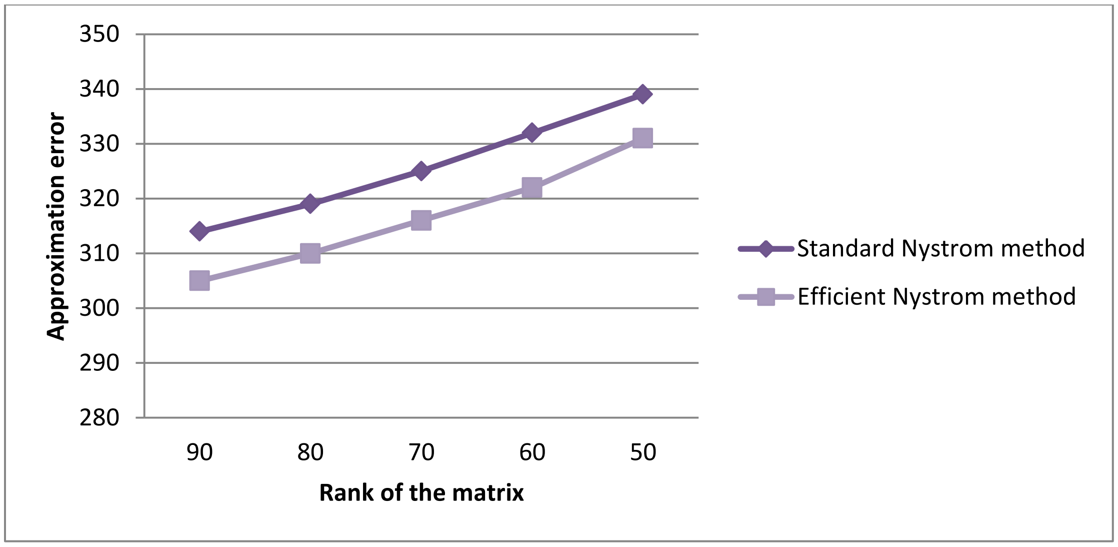


Figure 3 Plot between standard Nystrom method and efficient Nystrom method using Frobenius norm for the random data

Efficient Nystrom method and standard Nystrom method are based on sampling method. We perform the approximation only same columns of the original matrix instead of the whole matrix. We select the sampled columns using uniform sampling algorithms without replacement. If we found the 100% accuracy, then the Frobenius error becomes zero. Means there is no error in the approximation.

Table 3 Analysis of two sampling methods based on random generated data with various ranks of the matrix in spectral norm

Target rank	Standard Nystrom method	Efficient Nystrom method
90	30.20235	27.44825
80	31.46203	28.92563
70	34.61651	32.02353
60	34.20472	33.62095
50	36.94372	35.45737

Table 3 shows the analysis of the results based on spectral norm. It shows the comparison between standard Nystrom method and the efficient Nystrom method based on the spectral norm.

We show the result based on the Frobenius norm and spectral norm. Both of norms show the quality of the approximation so efficient Nystrom method has the good approximation because it has a low reconstruction error. Frobenius and spectral norm shows the error bound with various ranks the approximation.

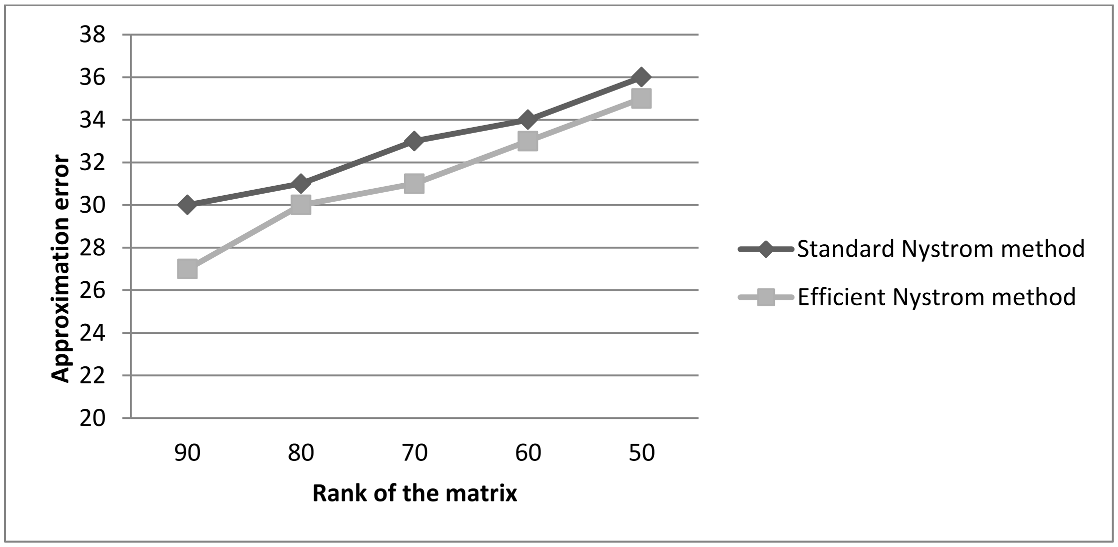


Figure 4 Plot between standard Nystrom method and efficient Nystrom method using spectral norm for the random data

Figure 4 shows the plots of the spectral error. The efficient Nystrom method has a low reconstruction error as compared to the standard Nystrom method. Here, we only show the comparisons between the sampling based methods. There exist other approximation methods, but they have a high time complexity, so we prefer only the sampling methods. Sampling methods are much faster than other approximation methods such as truncated SVD. But sampling methods suffer from high inaccuracy, but that inaccuracy is tolerable so we prefer sampling methods and try to improve the quality of bounds in Frobenius norm and spectral norm.

Table 4 Analysis of two sampling methods based on letter data set with various ranks of the matrix in Frobenius norm

Target rank	Standard Nystrom method	Efficient Nystrom method
50	35.97377	34.26584
60	35.30443	33.93821
70	34.83593	33.65025
80	34.48521	33.37971
90	34.14554	33.00249
100	33.82525	32.76092

Table 4 shows the analysis of the sampling based methods on the letter data set. Description about the letter date set discussed in the 6.1. It shows the error bound based on the Frobenius norm and compare both the sampling methods so efficient Nystrom methods has a low reconstruction error. For the letter data set, we use radial basis function kernel to make kernel matrix with sigma value 0.1.

If the target rank is 50 then the standard Nystrom method gives 35.97277 error bound while efficient Nystrom method gives 34.2684. And when we choose the target rank 100 then qualities of approximation improve. We have done the simulation on the different target range, and then analyze the results.

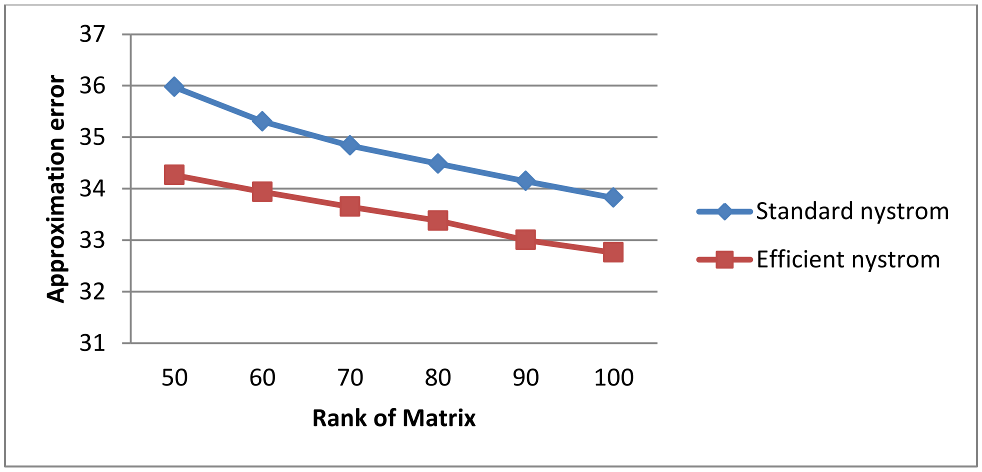


Figure 5 : Plot between standard Nystrom method and efficient Nystrom method using Frobenius norm for the letter data set

Figure 5 shows the plot between the standard Nystrom method and efficient Nystrom method for the letter data set based on the Frobenius norm. Figure 4 represents the spectral error for the sampling methods. If the target rank increase then the spectral error decrease. If we increase the target rank then approximated matrix become very close to the original matrix.

Table 5 Analysis of two sampling methods based on letter data set with various ranks of the matrix in Frobenius norm

Target rank	Standard Nystrom method	Efficient Nystrom method
50	5.837304	5.074329
60	5.048819	4.601283
70	5.031492	4.538082
80	5.03018	4.489118
90	5.029517	4.453372
100	5.026945	4.282975

Table 5 shows the analysis of the results for the sampling methods. It shows the spectral error for the standard Nystrom method and efficient Nystrom method. If target rank is 50 then spectral error for the standard Nystrom is 5.837304 and spectral error for the efficient Nystrom method. As we increase the target rank then spectral error decreases. For the target rank is 100 then spectral errors for standard Nystrom is 5.026945 and efficient Nystrom method is 4.282975. These results show the quality of the approximation in terms of the spectral norm. We use radial basis function kernel to make for letter data set as kernel matrix with sigma=0.1.

Efficient Nystrom method has low construction error for the real data sets as compare to the standard Nystrom method. If matrix having good spectral feature then we get good approximation. Our proposed sampling based low rank approximation method having low construction error.

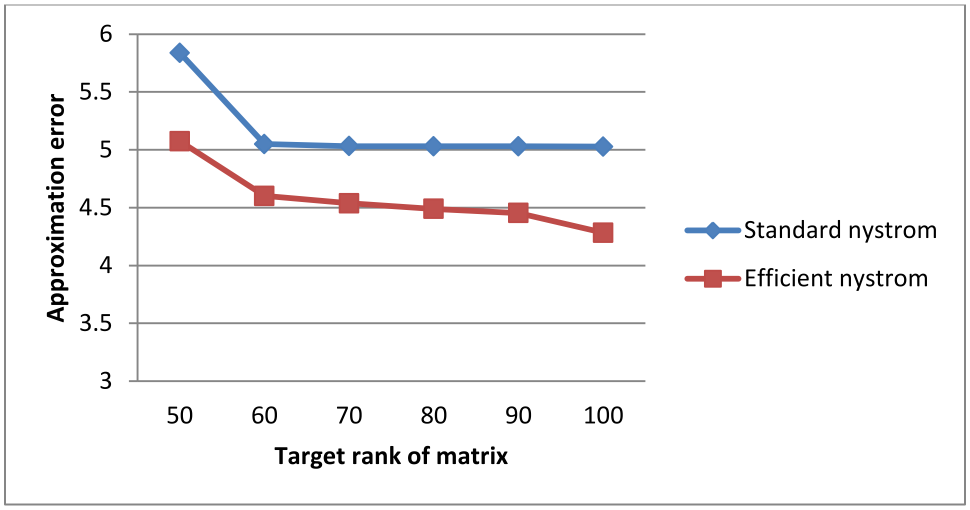


Figure 6 Plot between standard Nystrom method and efficient Nystrom method using spectral norm for the letter data set

Figure 6 shows the plot between the standard Nystrom method and efficient Nystrom method. It shows error bound for the sampling methods. These results are based on the letter data sets using spectral norm. Approximation error shows the quality of the approximation. Target rank of the matrix is increased then spectral error will decrease so that shows the approximation error of the targeted low rank matrix.

All these results show the quality of matrix approximation based on the sampling method. The efficient Nystrom method has a low construction error as compared the standard Nystrom method. As compared to previous data sets for the spectral error, letter data set has a low reconstruction error. Because letter data set having high dimensions so it gives less error. Matrix approximation works well for the large data sets. For the application point of view, we use the small matrix instead of large matrix. Matrix approximation helps to speed up the kernel method because we use the small matrix to find the kernel of the matrix. For example, support vector machine used for the classification. For large data, SVM does not work well, so we have to apply the matrix approximation over the large data. Matrix approximation helps to speed up the kernel method such as support vector machine, kernel principal component analysis and kernel ridge regression.

Table 6 Analysis of two sampling methods based on abalone data set with various ranks of the matrix in Frobenius norm

Target rank	Standard Nystrom method	Efficient Nystrom method
50	56.20786	54.2193
60	51.36662	49.03168
70	43.0483	42.3812
80	40.4295	39.0392
90	38.59548	37.18314
100	36.05404	34. 489118

Table 6 shows the analysis of standard Nystrom method and efficient Nystrom method based on the Frobenius norm for the abalone data set. If the target rank is 50 then Frobenius error for the standard Nystrom method is 56.20786 and for the efficient Nystrom method is 54.2193. So Frobenius norm gives the error bound for the quality of matrix approximation for the abalone data set. For abalone data, we use radial basis function kernel with sigma = 1.

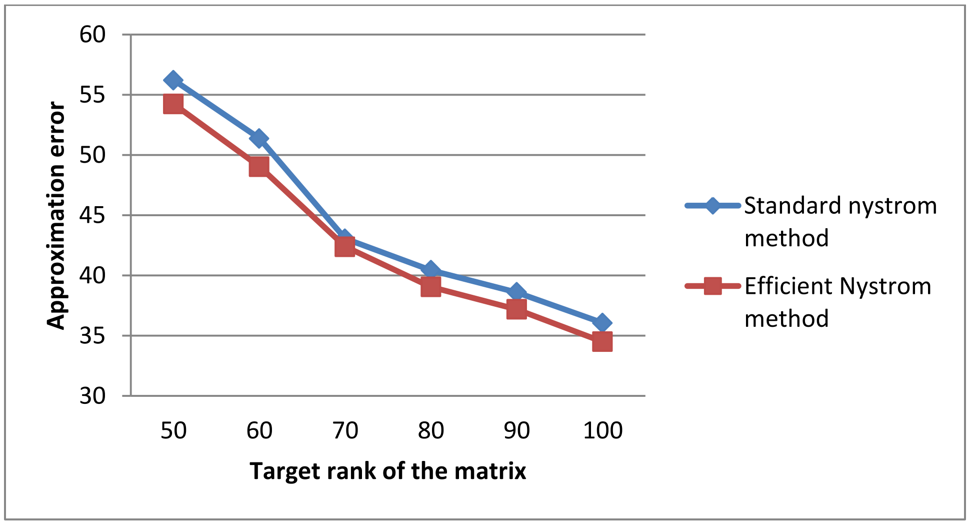


Figure 7 Plot between standard Nystrom method and efficient Nystrom method using Frobenius norm for the abalone data set

Figure 7 shows the plot between the standard Nystrom and efficient Nystrom method for the abalone dataset using Frobenius norm. After the analysis of the plot, efficient

Nystrom has low reconstruction error as compared standard Nystrom for the abalone data set based on the Frobenius norm. We increase the target rank for the approximation, then we get low reconstruction error for the abalone data set. After the increasing the rank of the approximated matrix, we get the quality of the approximation. Means highly rank with minimum error, but our target is low rank and minimum error. We have to minimize the error of the approximation so our proposed algorithms having minimum reconstruction error.

Table 7 Analysis of two sampling methods based on abalone data set with various ranks of the matrix in spectral norm

Target rank	Standard Nystrom method	Efficient Nystrom method
50	38.71982	36.03871
60	37.31203	35.4216
70	29.60764	25.52193
80	28.04498	24.83014
90	26.55803	22.01872
100	24.10108	21.28443

Table 7 shows the error analysis of the standard Nystrom and efficient Nystrom method based on spectral norm for the abalone data set. Abalone data set also gives the low construction error for the efficient Nystrom method. These three data sets used in the application to show the effectiveness of the approximation in the kernel based methods. We will extract the small amount of information from the original training data. Spectral norm gives the information related to the error of the approximation. But Frobenius norm gives the best approximation error as compared to the spectral norm. Real data set works well for the approximation as compared to the random dataset. In the implementation, we show the approximation error with the three datasets. After analyze the results, efficient Nystrom method has minimum reconstruction error as compared to the standard Nystrom method. If target rank is 50 then the spectral error for the standard Nystrom method is 38.71982 and for efficient Nystrom method are 36.03871. When target rank is 100 then spectral norms for the standard Nystrom method is 24.10108 and for the efficient Nystrom method is 21.28443. According these norms we find the conclusion that efficient Nystrom gives better accuracy.

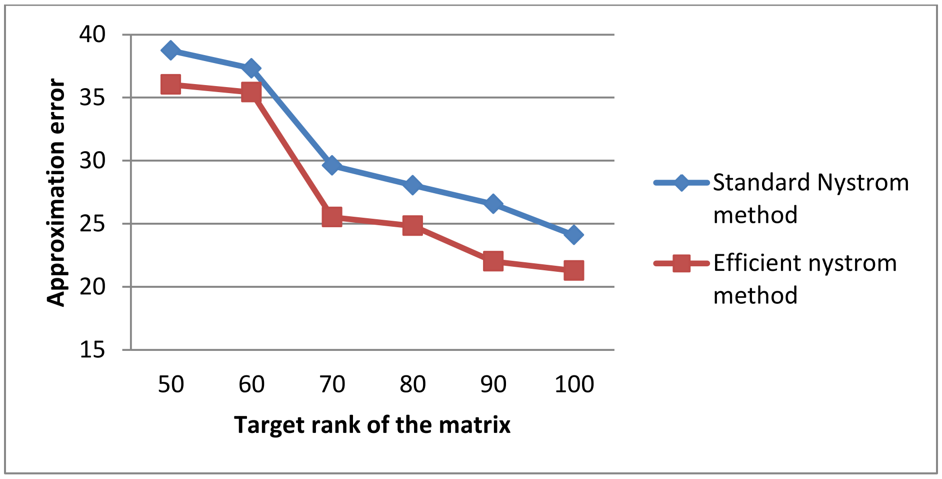


Figure 8 Plot between standard Nystrom method and efficient Nystrom method using norm for the abalone data set

Figure 8 shows the results of the abalone data set based on the spectral norm. After analysis the graph, we found that efficient Nystrom method has a good approximation for the kernel matrix. Here, Radius Basis Function is used in the implementation.

6.4 Results of the support vector machine

Here, we show the application of the matrix approximation. Matrix approximation helps to speed up the kernel algorithms such as support vector machine, kernel ridge regression, and kernel principal component analysis. Here, we show the computation time of the support vector machine. We take approximated matrix instead of original matrix as the training data. And the approximated matrix is small as comparison of the original matrix. So it reduced the computation time. At the time of calculating the kernel of the training data, we use small matrix for calculating the kernel. Size of the small matrix is $n \times k$ and it contains the orthogonal matrix and square root of the diagonal matrix. After calculating the kernel of the matrix, small kernel matrix multiplies by its transpose that is same as the kernel of the original matrix. But there is some error in the training data. And these errors are tolerable for the application. We will show computation with three data sets. Description of the data sets already explained in the section 6.1. First is random data and remaining two are real data sets.

Table 8 Computation time of Support vector machine without approximation and using the standard Nystrom method and efficient Nystrom method

Target Rank	Computation time of SVM using standard Nystrom method (in seconds)	Computation time of SVM using efficient Nystrom method (in seconds)
50	1.356364	1.4600258
60	1.4280241	1.4990289
70	1.4590261	1.5270288
80	1.479027	1.581033
90	1.5410309	1.594034
100	1.553031	1.600034

Table 8 shows the computation time of the approximation methods. Efficient Nystrom has better accuracy as compare to the standard Nystrom method. But for random data set, efficient Nystrom method is having more computation time as compare to the standard Nystrom. Computation time for the support vector machine is 4.471313 seconds without approximation. After applying the standard Nystrom and efficient Nystrom method, computation time becomes very less. In table 8, computation time for various rank of the matrix using Nystrom method and efficient Nystrom method. As we increase the rank of approximated matrix, the computation time increase. When target rank of the matrix 50 then computation time is 1.356364 seconds using standard Nystrom method and computation time for the efficient Nystrom method is 1.4600258 seconds. If we increase the target rank 100 then the computation time is 1.553031 seconds using standard Nystrom method and computation time is 1.600034 second for the efficient Nystrom method.

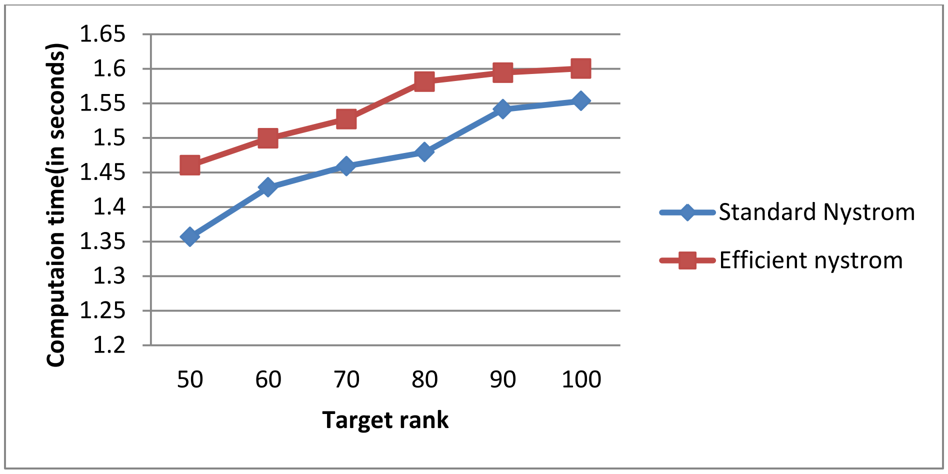


Figure 9 Plot between the computation time of standard Nystrom method and efficient Nystrom for random data in SVM

Figure 9 represents the computation time of the support vector machine using the approximated matrix. For the random data set, standard Nystrom work well in terms of the computation time. Regression problems are solved by the support vector machine. For large scale data set, support vector machine having high time complexity. Due to such reason, need to be an efficient implementation using the matrix approximation.

Table 9 Computation time of Support vector machine using the standard Nystrom method and efficient Nystrom method for abalone dataset

Target Rank	Standard Nystrom	Efficient Nystrom
50	1.2580152	1.2560141
60	1.295017	1.2800162
70	1.292016	1.2740159
80	1.327019	1.292017
90	1.3360188	1.301017
100	1.35602	1.3100178

Table 9 represents the computation time of the support vector machine for the abalone dataset. Efficient Nystrom method takes low computation time as compare to the standard Nystrom method if such approximation used for the matrix approximation. If Support Vector Machine (SVM) uses original training data set for classification then it takes 3.596205 seconds in computation. And with approximation it takes 1.2580152 seconds if the target rank is 50 and if target rank is 100 then it takes 1.35602 seconds.

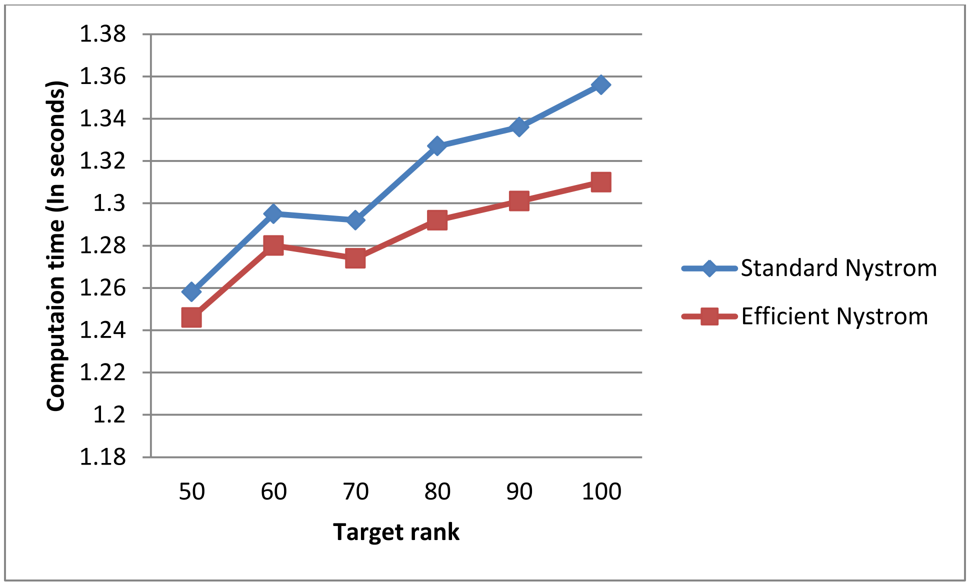


Figure 10 Plot between the computation time of standard Nystrom method and efficient Nystrom for abalone data set in SVM

Figure 10 represents the computation time of the efficient Nystrom method and standard Nystrom method in the abalone dataset. Efficient Nystrom method is having low computation cost. If we use original training dataset in the classification using support vector machine. Then support vector machine is having high computation cost so we have to apply low rank approximation on the training data to speed up the kernel methods.

Table 10 Computation time of Support vector machine using the standard Nystrom method and efficient Nystrom method for letter dataset

Target Rank	Standard Nystrom	Efficient Nystrom
50	1.2690151	1.2590139
60	1.2750161	1.272016
70	1.2800159	1.2870159
80	1.3290188	1.2860172
90	1.3320189	1.2940171
100	0.34202	1.3160191

Table 10 shows the results of the computation time of the support vector machine for the letter data set. Original dataset takes 3.3070168 seconds to classify the data using support vector machine. After the approximating, the training dataset is becomes smaller as compared to the original dataset. So, efficient implementation of the SVM takes 1.2690151 seconds to classify the data.

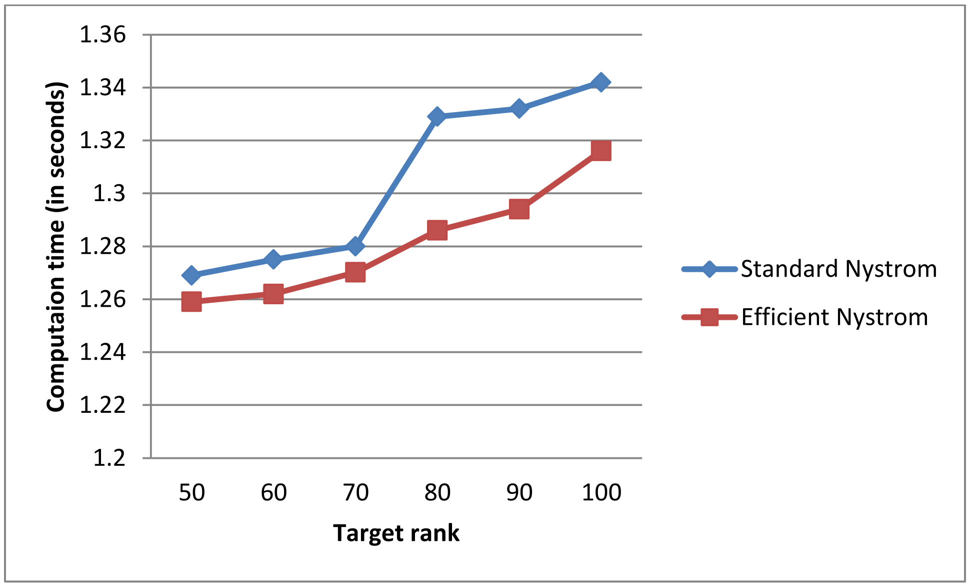


Figure 11 Plot between the computation time of standard Nystrom method and efficient Nystrom for letter data set in SVM

Figure 11 represents the comparison between the computation time of the efficient Nystrom method and the standard Nystrom method. Both of the sampling based approximation methods helps to speed up the kernel methods. As we increase the target rank of the matrix, computation time will increase. Without approximation it takes 3.3070168 seconds.

6.5 Summary

In this chapter, we discuss the effectiveness of low rank approximation in support vector machine. We compare various sampling methods for matrix approximation. Our proposed method is superior for other sampling based methods. We have done the experiments on various dataset such as letter dataset and abalone dataset. Based on the results of these datasets, we claim that efficient Nystrom method is superior than standard Nystrom.

Chapter 7
Conclusion

We addressed the question how can large scale data handled by the machine learning algorithm. We focused on this problem and find the effective solution of this problem; we have to generate the low rank approximations based on sampling methods. In chapter 5, we discussed efficient Nystrom method to generate the low rank approximation. In chapter 6, our result shows that efficient Nystrom method is superior for large datasets. We showed the effectiveness of low rank approximation on the kernel methods. We showed the various comparisons of the sampling based methods. In chapter 2, we discussed various decomposition methods. SVD is superior method for matrix decomposition. In chapter 3, we discussed previous work as literature survey. In efficient Nystrom method, we introduced new sampling method which is superior as previously discussed.

Though efficient Nystrom method reduces the error but still some improvements are required which will be considered in future. I will try to implement efficient sampling method to reduce the error.

References

[1] Olver, Peter J., and Chehrzad Shakiban. "Applied linear algebra." *Upper Saddle River journal*, 2006.

[2] Krzysztof simek , “Properties of a singular value decomposition based dynamical model of gene expression data, “ *International Journals Applied Math. Computer Science*, Vol. 13, No. 3, pp. 337–345, 2003.

[3] Forsythe, George Elmer, and Cleve B. Moler. “Computer solution of linear algebraic systems” *Englewood Cliffs*, Prentice-Hall, vol. 7, 1967.

[4] Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., & Willsky, A. S., “Rank-sparsity incoherence for matrix decomposition”, *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572-596, 2011.

[5] Parlett, Beresford N., “The symmetric eigenvalue problem”, *Englewood Cliffs*, Prentice-Hall, Vol. 7, 1980.

[6] Belitskii, Genrikh Ruvimovich, and Yurii I. Lyubich. "Matrix norms and their applications”, Springer, 1988.

[7] Hofmann, T., Schölkopf, B., & Smola, A. J.” Kernel methods in machine learning”, *The annals of statistics*, pp. 1171-1220, 2008.

[8] Aditya Krishna Menon and Charles Elkan “Fast Algorithms for Approximating the Singular Value Decomposition,” ACM Transactions on Knowledge Discovery from Data, TKDD- 2011, vol. 5, no. 2, article no. 13, pp. 1-36, feb. 2011.

[9] David J. Slate,”Letter Recognition Data Set”, *UCI Machine Learning Repository*, available: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

[10] Warwick J Nash, Tracy L Sellers, Andrew J Cawthorn and Wes B Ford(1994),” Abalone DataSet”, Availeble: https://archive.ics.uci.edu/ml/datasets/Abalone.

[11] Golub, G. H. and Van loan, C. F., “Matrix Computations,*”* 3rd Ed. *Johns Hopkins University Press, Baltimore,* MD, USA pp. 374-426, 1996.

[12] Watson, G. Alistair, "Characterization of the subdifferential of some matrix norms", *Linear Algebra and its Applications*, vol. 170, 33-45, 1992.

[13] Zhang, Jimeng Sun Yinglian Xie Hui, and Christos Faloutsos, "Less is more: Compact matrix decomposition for large sparse graphs," *Proceedings of the Seventh SIAM International Conference on Data Mining*. Society for Industrial Mathematics, Vol. 127, p.366, 2007.

[14] Ameet Talwalkar, Sanjiv Kumar, Mehryar Mohri and Henry Rowley, “Large-scale SVD and Manifold Learning,*”* Journal of Machine Learning Research vol. 14, pp. 3129-3152,jan 2013.

[15] Petros Drineas , Michael W. Mahoney, “On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning,” *Journal of Machine Learning Research,* vol. 6,pp. 2153-2175, 2005.

[16] Zhang, K., Tsang, I., & Kwok, J., “Improved Nystrom low-rank approximation and error analysis,” Proceedings of the 25th international conference on Machine learning, *pp.* 1232-1239, 2008.

[17] Dimitris Achlioptas and McSherry ,“Fast Computation of Low Rank Matrix Approximations,*” Journal of the ACM*,vol. 54 pp. 601-618, 2007.

[18] Krzysztof Simek ,“properties of a singular value decomposition based dynamical model of gene expression data,“ *International Journal of applied mathematics and computer science*, Vol. 13 , No. 3, 337–345, 2003.

[19] Williams, C. K. I., & Seeger, M., “Using the Nystrom method to speed up kernel machines,” *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, pp. 682-688, 2000.

[20] Fowlkes, Charless, "Spectral grouping using the Nystrom method," Pattern Analysis and Machine Intelligence, IEEE Transactions, vol.26 pp. 214-225, 2004.

[21] Sanjiv kumar, Meheyar mohri, Ameet Talwalkar, “Sampling methods for the Nyström method,*” The journal of Machine Learning Research* vol 13, pp. 981-1006. 2012.

[22] Petros Drineas, Michael W. Mahoney, **“**On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning,*” Journal of Machine Learning Research*,vol. 6,pp. 2153-2175, 2005.

[23] Zhang, K., Tsang, I. & Kwok, J.,” Improved Nystr¨om low-rank approximation and error analysis,” *International Conference on Machine Learning*, pp. 1232-1239 2008.

[24] Aditya Krishna Menon and Charles Elkan, “Fast Algorithms for Approximating the Singular Value Decomposition,” ACM transaction knowledge discovery data,vol. 5,pp. 1556-4681, feb, 2011.

[25] Kumar, Sanjiv, Mehryar Mohri, and Ameet Talwalkar, "Ensemble nystrom method," *Advances in Neural Information Processing System*,vol. 22, pp. 1060-1068, 2009.

[26] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar, “On sampling-based approximate spectral decomposition,” *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 553-560, 2009

[27] Baatz, Wolfgang, Massimo Fornasier, and Jan Haskovec, "Mathematical methods for spectral image reconstruction," *Scientific Computing and Cultural Heritage*. Springer Berlin Heidelberg,pp. 3-10, 2013

[28] Zhang, Kai, Ivor W. Tsang, and James T. Kwok, "Improved Nyström low-rank approximation and error analysis," *Proceedings of the 25th international conference on Machine learning*, pp. 1232-1239, 2008

[29] Achlioptas, Dimitris, and Frank McSherry, "Fast computation of low rank matrix approximations," *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 611-618, 2001

[30] Drineas, Petros, Ravi Kannan, and Michael W. Mahoney, "Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 158-183, 2006

[31] LB Chang, Z Bai, SY Huang, CR Hwang, "Asymptotic error bounds for kernel-based Nyström low-rank approximation matrices," *Journal of Multivariate Analysis,vol 120*, pp. 102-119, 2013

[32] Zhang, K., Lan, L., Wang, Z., & Moerchen, F. "Scaling up kernel SVM on limited resources: A low-rank linearization approach", *International Conference on Artificial Intelligence and Statistics* pp. 1425-1434, 2012.

[33] Nguyen, XuanLong, Ling Huang, and Anthony D. Joseph. "Support vector machines, data reduction, and approximate kernel matrices." *Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, pp. 137-153, 2008.

[34] Collobert, Ronan, and Samy Bengio. "SVMTorch: Support vector machines for large-scale regression problems." *The Journal of Machine Learning Research*, vol. 1, 143-160, 2001.

[35] Kulis, Brian, Mátyás Sustik, and Inderjit Dhillon. "Learning low-rank kernel matrices." *Proceedings of the 23rd international conference on Machine learning*, pp. 505-512, ACM, 2006.

[36] Ross Ihaka and Robert(2010), "R-Stdio",Available: www.rstudio.com

[37] Auslander, Maurice, "Representation theory of Artin algebras", Cambridge University Press, Vol. 36,1997.