Indian Institute of Technology Delhi

End Term Report : Randomized Kernel PCA

Deepak: 2019MT10685

Leaandro Rodricks: 2019MT10702

1 Abstract

The aim of our project is to find the most computationally efficient method that can be used to store and comprehend a given set of n data points, each lying in a d-dimensional space such that the number of elements required in its representation and thus, the space complexity of the representation is minimized. We have first considered a trivial case that can be represented by the simplest Representative-Coefficient method, followed by some non-trivial cases where we would require the Principal Component Analysis (PCA) method. We then discuss the algorithm and limitations of PCA and see how some of these limitations can be solved using the Kernel PCA method. We first examine the properties and construction of kernel functions and then discuss the algorithm and limitations of KPCA. Next, to tackle the computational inefficiency of KPCA, we introduce the Nystrom sampling approximation. This forms the base of our randomized approach toward Kernel PCA and the study of the tradeoff involved in this approach.

2 Representation Learning

Representation learning refers to a type of machine learning approach that involves automatically discovering or learning a feature space, or representation, that allows raw input data to be transformed into a more informative and useful format.

The goal of representation learning is to create a more compact and efficient representation of the data that captures the underlying patterns and structure in the input. Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensor data has not yielded to attempts to algorithmically define specific features. In supervised feature learning, features are learned using labeled input data .i.e. input-label pairs. In unsupervised feature learning, features are learned with unlabeled input data by analyzing the relationship between points in the dataset.

Thus the main goal of representation learning is to reduce the number of elements used in the representation of our data/features, which would indirectly help in taking up less storage space and be easier to process. To do this, we often use the representative coefficient method of representation as shown in the following examples :

Example 1. The input x_1, x_2, x_3, x_4 is a set of 4 points in R^2 where $x_1 = [-7, -14]^T, x_2 = [3, 6]^T, x_3 = [-4, 8]^T, x_4 = [1, 2]^T$. To store this input as it is, we would require 8 elements for each coordinate of the 4 points. But we can observe that all 4 points lie on a straight line with the equation $y = 2x$. Thus we can use a representative vector $r = [1, 2]^T$ and a coefficient vector $c = [-7, 3, -4, 1]$. In this representation, we need only $2 + 4 = 6$ elements instead of 8.

Example 2. Along with the above 4 points, we have an additional point $x_5 = [5, 5]^T$. Now there is no line in R^2 on which all the 5 pints would lie. However, we can take a proxy x_5' for the point x_5 such that this proxy lies on the line $y = 2x$ and thus leads to the new set of 5 points lying on the same line. To find out the proxy, we will find the component of x_5 on the line $y = 2x$, i.e. $(x_5.w)w$ where w is the unit vector in the direction of $y = 2x$. This is a compressed representation of the input dataset with the minimum possible total error in the representation of the points.

While these examples were pretty straightforward, there can be cases where no 3 points lie on a straight line, and finding the direction vector for minimizing the total error in representation would not be so easy. Thus, we introduce a standard method for this called PCA.

3 Principal component analysis (PCA)

Let X be a random variable distributed according to a probability measure $\mathbb{P}$ defined on a measurable space $\mathbb{X}$. Principal component analysis (PCA) deals with finding a direction $a \in \mathbb{X}(= \mathbb{R}^d)$ with $\|a\|_2 = 1$ such that $\mathrm{Var}[a^T X]$ is maximized. More generally, it provides a low-dimensional representation that retains as much variance as possible of X and is used as a popular statistical methodology for dimensionality reduction and feature extraction. In fact, the low-dimensional representation is the orthogonal projection of X onto the ℓ-eigenspace, i.e., the span of eigenvectors associated with top ℓ eigenvalues of the covariance matrix $\mathbb{E}(XX^T) - \mathbb{E}(X)\mathbb{E}(X^T)$ where $\ell < d$, resulting in a ℓ-dimensional representation.

Consider a data set of observations $\{x_n\}$ where $n = 1, 2, ..., N$, and x_n is a Euclidean variable with dimensionality D. Our goal is to project the data onto a space having dimensionality $M < D$ while maximizing the variance of the projected data. To begin with, consider the projection onto a one-dimensional space ($M = 1$). We can define the direction of this space using a D-dimensional vector u_1, which for convenience (and without loss of generality) we shall choose to be a unit vector so that $u_1^T u_1 = 1$. Each data point x_n is then projected onto a scalar value $u_1^T x_n$. The mean of the projected data is $u_1^T \bar{x}$ where $\bar{x}$ is the sample set mean given by $\bar{x} = 1/N \sum_{n=1}^{N} x_n$ and the variance of the projected data is given by $1/N \sum_{n=1}^{N} \{u_1^T x_n - u_1^T \bar{x}\}^2 = u_1^T S u_1$ where S is the data covariance matrix defined by $S = 1/N \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$. We now maximize the projected variance $u_1^T S u_1$ with respect to u_1. Instead of maximizing the term $u_1^T S u_1$ we have to maximize the term $u_1^T S u_1 + \lambda_1(1 - u_1^T u_1)$ to enforce the normalization condition of $u_1^T u_1 = 1$. By setting the derivative with respect to u_1 equal to zero, we see that this quantity will have a stationary point when $Su_1 = \lambda u_1$ which says that u_1 must be an eigenvector of S. If we left-multiply by u_1^T and make use of $u_1^T u_1 = 1$, we see that the variance is given by $u_1^T S u_1 = \lambda_1$ and so the variance will be a maximum when we set u_1 equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector is known as the first principal component. If we consider the general case of an M -dimensional projection space, the optimal linear projection for which the variance of the projected data is maximized is now defined by the M eigenvectors $u_1, u_2 ..., u_M$ of the data covariance matrix S corresponding to the M largest eigenvalues $\lambda_1, ..., \lambda_M$.

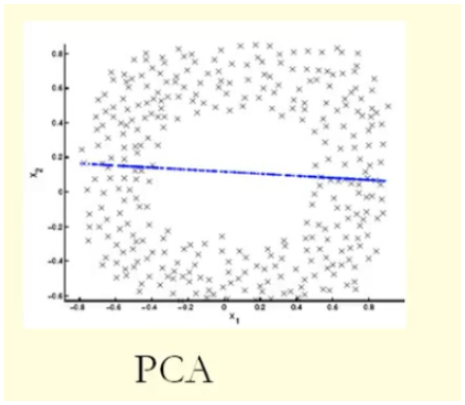
Algorithm

- Given $D = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}^d$ and compute the covariance matrix $C = 1/n \sum_{i=1}^{n} x_i x_i^T$.
- Given w which is a random unit vector ($\|w\|_2{}^2 = 1$), find another direction vector such that $w_1 = argmax\,(w^t C w)$.
- Compute the projections of the data points onto the w_1 vector to get the new data points x_i^1 like $x_1^1 \Rightarrow x_1 - (x_1^T w_1)w_1$,....$x_n^1 \Rightarrow x_n - (x_n^T w_1)w_1$
- Similarly compute the covariance matrix $C^1 = 1/n \sum_{i=1}^{n} x_i^1 (x_i^1)^T$ for the projected data set obtained and find a new direction vector such that $w_2 = argmax\,(w^T C^1 w)$

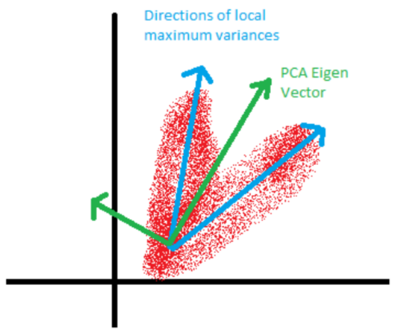
We can observe that the 2 directions obtained .i.e w_1 and w_2 are orthogonal to each other since w_2 minimizes the sum of errors w.r.t residues, which are orthogonal to w_1. Thus, $w_2^T w_1 = 0$. We continue these iterations till all the errors associated with $x_1, x_2, ..., x_n$ become 0 vectors. We know that after d rounds, all errors would be zero but if after k rounds the errors are approximately zero vectors (within a tolerance value), then we can say that $w_1, ..., w_k$ are the basis vectors and x_i can be represented as $x_i = (x_i^T w_1)w_1 + (x_i^T w_2)w_2 + ... + (x_i^T w_k)w_k$, where k < d. Thus, the number of elements needed in the representation would reduce to $k * (n + d)$ from $d * n$.

However, there are certain limitations of PCA like:

- **High Time Complexity:** The covariance matrix of the features would be a d*d matrix, and we need to find the k largest eigenvalues and corresponding eigenvectors. The time complexity for this process would be $O(d^3)$, which would exceed most limits in cases where the number of features(d) is very large.
- **Features having non-linear relationships:** The data may not lie in a low-dimensional **linear** subspace of the feature space and thus, we may need non-linear principal components. Consider the 2-dimensional data distribution shown in figure 1. The standard PCA always finds orthogonal principal components. However, the given data demands non-orthogonal principal components for its representation. This is evident by the directions of the local maximum variances.



(a) *Data distribution and the* 1st *principal component*



(b) *Principal Components and directions of Maximum Variance for the data*

Figure 1: *PCA failing in non-linear data distributions*

4 Kernel

In an ideal world, we want a simple model that is fast to train and that is complex enough to find complex relationships between inputs and outputs. Kernel methods do this by mapping the input space of the data to a higher dimensional feature space, in which simple linear models can be trained, resulting in efficient, low-bias low-variance models. So, a kernel function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the predictions are also based on linear combinations of a kernel function evaluated at the training data points. For models which are based on a fixed nonlinear feature space mapping $\phi(x) : X \to V$, the kernel function is given by the relation $k(x, x') =< \phi(x), \phi(x') >_V$. The key restriction is that $< ., . >_V$ must be a proper inner product. On the other hand, an explicit representation for ϕ is not necessary, as long as V is an inner product space.

Some examples of Kernel functions are as follows:

- **Linear Kernel:** If there are two vectors named x_1 and x_2, the linear kernel can be defined by the dot product of the two vectors: $K(x_1, x_2) = x_1 \cdot x_2$
- **Polynomial Kernel:** We can define a polynomial kernel with this equation: $K(x_1, x_2) = (x_1 \cdot x_2 + 1)^d$. Here, x_1 and x_2 are vectors and d represents the degree of the polynomial.

- **Gaussian Kernel:** The Gaussian kernel is an example of a radial basis function kernel. It can be represented with this equation: $k(x_i, x_j) = exp(-||x_i - x_j||^2/2\sigma^2)$. The given σ has a vital role in the performance of the Gaussian kernel. It should be carefully tuned according to the problem, neither overestimated and nor underestimated.

The concept of a kernel formulated as an inner product in a feature space allows us to build interesting extensions of many well-known algorithms by making use of kernel substitution. The general idea is that, if we have an algorithm formulated in such a way that the input vector x enters only in the form of scalar products, then we can replace that scalar product with some other choice of kernel.

In order to exploit kernel substitution, we need to be able to construct valid kernel functions. One approach is to choose a feature space mapping $\phi(x)$ and then use this to find the corresponding kernel. Here the kernel function is defined for a one-dimensional input space by $k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')^T$ where $\phi_i(x)$ are the basis functions. An alternative approach is to construct kernel functions directly. In this case, we must ensure that the function we choose is a valid kernel, in other words, that it corresponds to a scalar product in some (perhaps infinite dimensional) feature space.

Example: Consider a kernel function given by $k(x, z) = (x^T z)^2$. If we take the particular case of a two-dimensional input space $x = (x_1, x_2)$ we can expand out the terms and thereby identify the corresponding nonlinear feature mapping $k(x, z) = (x^T z)^2 = (x_1 z_1 + x_2 z_2)^2 = (x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2) = (x_1^2, \sqrt[2]{2}x_1 x_2, x_2^2)(z_1^2, \sqrt[2]{2}z_1 z_2, z_2^2) = \phi(x)^T \phi(z)$. We see that the feature mapping takes the form $\phi(x) = (x_1^2, \sqrt[2]{2}x_1 x_2, x_2^2)^T$ and therefore comprises all possible second order terms, with a specific weighting between them.

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without explicitly constructing the function $\phi(x)$. We require that the kernel $k(x, x')$ be symmetric and positive semidefinite and that it expresses the appropriate form of similarity between x and x' according to the intended application. We can construct complex kernels by modifying or existing kernels. If we consider the slightly generalized kernel $k(x, x') = (x^T x' + c)^2$ with $c > 0$, then the corresponding feature mapping $\phi(x)$ contains constant and linear terms as well as terms of order two. Similarly, $k(x, x') = (x^T x')^M$ contains all monomials of order M. This can similarly be generalized to include all terms up to degree M by considering $k(x, x') = (x^T x' + c)^M$ with c > 0.

5 Kernel PCA

PCA is a linear method. That is it can only be applied to datasets which are linearly separable. It does an excellent job for datasets, which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.

Consider a data set x_n of observations, where $n = 1, ..., N$, in a space of dimensionality D. The first step is to express conventional PCA in such a form that the data vectors x_n appear only in the form of the scalar products $x_n^T x_m$. Recall that the principal components are defined by the eigenvectors u_i of the covariance matrix $Su_i = \lambda_i u_i$, where $i = 1, 2, ..., D$. The $D * D$ sample covariance matrix S is given by $S = 1/N \sum_{n=1}^{N} x_n x_n^T$ and the eigenvectors are normalized such that $u_i^T u_i = 1$. Now consider a nonlinear transformation $\phi(x)$ into an M-dimensional feature space, so that each data point x_n is thereby projected onto a point $\phi(x_n)$. We can now perform standard PCA in the new feature space which implicitly defines a non-linear principal component model in the original space. Let the projected data have zero mean. The new $M * M$ covariance matrix in feature space would be given by $C = 1/N \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T$. The new eigenvector equation would be $Cv_i = \lambda_i v_i \Rightarrow 1/N \sum_{n=1}^{N} \phi(x_n)\{\phi(x_n)^T v_i\} = \lambda_i v_i$. Solving this eigenvector equation we obtain the equation $Ka_i = \lambda_i N a_i$ where K is the kernel gram matrix. Using the normalization condition of the eigenvectors v_i, we get $\lambda_i N a_i^T a_i = 1$. Thus the projection of a point x onto eigenvector i is given by $y_i(x) = \phi(x)^T v_i = \sum_{n=1}^{N} a_{in}\phi(x)^T\phi(x_n) = \sum_{n=1}^{N} a_{in} k(x, x_n)$.

So far we have assumed that the projected data set given by $\phi(x_n)$ has zero mean, which in general will not be the case. If this mean is not zero, we would have to centralize the projected data points to make it a zero mean data set given by $\tilde{\phi}(x_n) = \phi(x_n) - 1/N \sum_{l=1}^{N} \phi(x_l)$. The corresponding elements of the new Gram Matrix would be $\tilde{K}_{nm} = \tilde{\phi}(x_n)^T \tilde{\phi}(x_m) = k(x_n, x_m) - 1/N \sum_{l=1}^{N} k(x_n, x_l) - 1/N \sum_{l=1}^{N} k(x_l, x_m) + 1/N^2 \sum_{j=1}^{N} \sum_{l=1}^{N} k(x_j, x_l)$. This can be expressed in matrix notation as $\tilde{K} = K - 1_N K - K 1_N + 1_N K 1_N$ where 1_N denotes the $N * N$ matrix in which every element takes the value $1/N$.

Algorithm

- Compute the gram matrix $K \in R^{n*n}$ where $K_{ij} = k(x_i, x_j)\ \forall i, j$ and centralize it as shown in the above paragraph to get $\tilde{K}$.
- Compute the eigenvalues $n\lambda_1, n\lambda_2, ..., n\lambda_l$ and the eigenvectors $\beta_1, \beta_2, ..., \beta_l$ of $\tilde{K}$. Normalize the eigenvectors by the relation $\alpha_i = \beta_i/\sqrt{n\lambda_i}$ and obtain the set of vectors $\{\alpha_1, \alpha_2, ..., \alpha_l\}$
- If we write $w_k = \phi(x)\alpha_k$, we would have to compute $\phi(x)$ and reconstruct the eigenvectors of the covariance matrix. Instead compute $\sum_{j=1}^{N} \alpha_{kj} K_{ij}$ since the equation $\phi(x_i)^T w_k = \sum_{j=1}^{N} \alpha_{kj} K_{ij}$ holds true. Now we can use the following compressed representation $x_i \Rightarrow [\alpha_{1j} K_{ij}, \alpha_{2j} K_{ij}, ..., \alpha_{lj} K_{ij}]$. Thus we have changed the dimension from d to l, where l can be greater or lesser than d, but the non-linear structure of the original data set would be retained.

Now, there are some limitations of Kernel PCA too like:

- **Higher time complexity for large data sets:** It involves finding the eigenvectors of the $N * N$ matrix $\tilde{K}$ rather than the $D * D$ matrix S of conventional linear PCA giving it a time complexity of $O(n^3)$, and thus, approximations are often used in practice for large data sets.
- **Absence of an inverse mapping:** PCA provides an inverse mapping from the low-dimensional space back to the input space. So, input points can be approximately reconstructed from their low-dimensional images. Kernel PCA doesn't inherently provide an inverse mapping since the projection of points in feature space onto the linear PCA subspace in that space will typically not lie on the nonlinear D-dimensional manifold, although it's possible to estimate one using additional methods.

6 Sampling Methods

Let $T \in R^{axb}$ be an arbitrary matrix. We define $T^{(j)}, j = 1...b$, as the j^{th} column vector of T, $T_{(i)}, i = 1...a$, as the i^{th} row vector of T and $||.||$ the l_2 norm of a vector. Furthermore, $T^{(i:j)}$ refers to the i^{th} through j^{th} columns of T and $T_{(i:j)}$ refers to the i^{th} through j^{th} rows of T. If rank(T) = r, we can write the thin Singular Value Decomposition (SVD) of this matrix as $T = U_T \Sigma_T V_T^T$ where Σ^T is diagonal and contains the singular values of T sorted in decreasing order and $U_T \in R^{axr}$ and $V_T \in R^{bxr}$ have orthogonal columns containing the left and right vectors of T corresponding to its singular values. We denote by T_k the 'best' rank-k approximation to T, i.e., $T_k = argmin_{V \in R^{a \times b}, rank(V)=k} ||T - V||_\xi$, where $\xi \in \{2, F\}$ and $||.||_2$ denotes the spectral norm and $||.||_F$ the Frobenius norm of a matrix. We can describe this matrix in terms of its SVD as $T_k = U_{T,k} \Sigma_{T,k} V_T^T{}_{,k}$ where $\Sigma_{T,k}$ is a diagonal matrix of the top k singular values of T and $U_{T,k}$ and $V_{T,k}$ are the matrices formed by the associated left and right singular vectors.

Now let $K \in R^{nxn}$ be a symmetric positive semidefinite (SPSD) kernel or Gram matrix with $rank(K) = r <= n$, i.e. a symmetric matrix for which there exists an $X \in R^{Nxn}$ such that $K = X^T X$. We will write the SVD of K as $K = U \Sigma U^T$, where the columns of U are orthogonal and $\Sigma = diag(\sigma_1, ..., \sigma_r)$ is diagonal. The pseudo-inverse of K is defined as $K^+ = \Sigma_{t=1}^r \sigma_t^{-1} U^{(t)} U^{(t)T}$, and $K^+ = K^{-1}$ when K is full rank. For $k < r$, $K_k = \Sigma_{t=1}^k \sigma_t^{-1} U^{(t)} U^{(t)T} = U_k \Sigma_k U_k^T$ is the 'best' rank-k approximation to K,i.e., $K_k = argmin_{K' \in R^{nxn}, rank(K')=k} ||K - K'||_{\xi \in \{2,F\}}$, with $||K - K_k||_2 = \sigma_{k+1}$ and $||K - K_k||_F = \sqrt{\Sigma_{t=k+1}^r \sigma_t^2}$. We wish to get an approximation $\tilde{K}$ of K based on a sample $l << n$ of its columns.

7 Nystrom method

Let us assume that the sample of l columns is given to us, C denotes the nxl matrix formed by these columns and W the lxl matrix consisting of the intersection of these l columns with the corresponding l rows of K. Since K is SPSD, W is also SPSD. Thus, K and C can be written as- $K = \begin{bmatrix} W & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix}$ and $C = \begin{bmatrix} W \\ K_{21} \end{bmatrix}$. The Nystrom method uses W and C to approximate K. Assuming a uniform sampling of the columns, the Nystrom method generates a rank-k approximation $\tilde{K}$ of K for $k < n$ defined by $\tilde{K}_k^{nys} = C W_k^+ C \approx K$, where W_k is the best k-rank approximation of W with respect to the spectral or Frobenius norm and W_k^+ denotes the pseudo-inverse of W_k. When $k = l$ (or $k >= rank(C)$), this approximation perfectly reconstructs three blocks of K, and K_{22} is approximated by the Schur Complement of W in K. $\tilde{K}_k^{nys} = \begin{bmatrix} W & K_{21}^T \\ K_{21} & K_{21} W^+ K_{21}^T \end{bmatrix}$.

Since the Nystrom method operates on a small subset of K, i.e., C, the selection of columns can significantly influence the accuracy of the approximation. The most common class of sampling techniques that select columns using a fixed probability distribution. The types of fixed sampling techniques are-

- **Plain Nystrom**- The columns are sampled uniformly at random without any replacement.
- **Diagonal Nystrom**- The i^{th} column can be sampled non-uniformly with weight proportional to its corresponding diagonal element K_{ii}
- **Column-Norm Nystrom**- The i^{th} column can be sampled non-uniformly with weight proportional to the L_2 norm of the column.

Instead of sampling all l columns from a fixed distribution, **Adaptive sampling** alternates between selecting a set of columns and updating the distribution over all the columns. Starting with an initial distribution over the columns, $s < l$ columns are chosen to form a submatrix C. The probabilities are then updated as a function of previously chosen columns and s new columns are sampled and incorporated in C. This process is repeated until l columns have been selected.

8 Theoretical Analysis

We now present theoretical results that compare the quality of the Nystrom approximation to the 'best' low-rank approximation, i.e., the approximation constructed from the top singular values and singular vectors of K.

8.1 Method 1

Theorem 1

Let $Z_1, ..., Z_l$ be a sequence of random variables sampled uniformly without replacement from a fixed set of $l+u$ elements Z, and let $\phi : Z^l \to R$ be a symmetric function such that for all $i \in [1, l]$ and for all $z_1, ..., z_l \in Z$ and $z'_1, ..., z'_l \in Z, |\phi(z_1, ..., z_l) - \phi(z_1, ..., z_{i-1}, z'_i, z_{i+1}, ..., z_l)| <= c$. Then, for all $\in > 0$, the following inequality holds: $Pr[\phi - E[\phi] >= \in] <= exp(-2 \in^2 / \alpha(l, u)c^2)$ where $\alpha(l, u) = (lu)/(l + u - 1/2) * 1/(1 - 1/(2max\{l, u\}))$

Proof
This theorem can be proved using a classical bound on the moment-generating function of a random variable, which is as follows. Let X be a real random variable such that $E(X) = 0$ and $a <= X <= b$ for some $a, b \in R$. Then, for all $s \in R$, $log(Ee^{sX}) <= s^2(b - a)^2/8$. On substituting X as $\phi - E[\phi]$, we can calculate the PDF of this function by taking the Inverse Fourier Transform of the Characteristic Function (CF) of X, where the CF can be obtained from MGF by replacing X with iX.

Theorem 2

We define the selection matrix corresponding to a sample of l columns as the matrix $S \in R^{nxl}$ defined by $S_{ii} = 1$ if the i^{th} column of K is among those sampled, $S_{ij} = 0$ otherwise. Thus, C=KS is the matrix formed by the columns sampled. Since K is SPSD, there exists $X \in R^{Nxn}$ such that $K = X^T X$. We shall denote by K_{max} the maximum diagonal entry of K,$K_{max} = max_i K_{ii}$, and by d^K_{max} the distance $max_{ij}\sqrt{K_{ii} + K_{jj} - 2K_{ij}}$.Let $Z = \sqrt{n/l}XS$. Then the following inequality holds: $||K - \tilde{K}||_2 <= ||K - K_k||_2 + 2||XX^T - ZZ^T||_2$ where $\tilde{K}$ denotes the rank-k Nystrom approximation of K based on l columns sampled uniformly at random without replacement from K, and K_k the best rank-k approximation of K.

Proof
This theorem can be proved by combining the following 2 lemmas-

- If $\tilde{G}_k = CW^+_k C^T$ then $||G - \tilde{G}_k||_2 = ||X - U_k U^T_k X||^2_2$.
- Suppose $A \in R^{mxn}$ and let H_k be the mxk matrix whose columns consist of the top k singular vectors of the mxc matrix C. Then, for every k : $0 <= k <= rank(C)$, $||A - H_k H^T_k A||^2_2 <= ||A - A_k||^2_2 + 2||AA^T - CC^T||_2$.

Theorem 3

Let $\tilde{K}$ denote the rank-k Nystrom approximation of K based on l columns sampled uniformly at random without replacement from K, and K_k the best rank-k approximation of K. Then, with probability at least $1 - \delta$, the following inequalities hold for any sample of size l:

$$||K-\tilde{K}||_2 \leq ||K-K_k||_2+2n/\sqrt{l}K_{max}[1+\sqrt{(n-l)/(n-1/2) * log(1/\delta)/\beta(l,n)}d^K_{max}/K^{0.5}{}_{max}]$$

$$||K-\tilde{K}||_F \leq ||K-K_k||_F+(64k/l)^{1/4}nK_{max}[1+\sqrt{(n-l)/(n-1/2)*log(1/\delta)/\beta(l,n)}d^K_{max}/K^{0.5}{}_{max}]^{0.5}$$

Proof
We then apply the McDiarmid-type inequality of Theorem 1 to $\phi(S) = ||XX^T - ZZ^T||_2$. Let S' be a sampling matrix selecting the same columns as S except for one, and let Z denote $\sqrt{n/l}XS$. Let z and z denote the only differing columns of Z and Z', then $|\phi(S') - \phi(S)| <= ||z'z'^T - zz^T||_2 = ||(z'-z)z'^T + z(z'-z)^T||_2 \rightarrow |\phi(S') - \phi(S)| <= 2||z'-z||_2 max(||z||_2, ||z'||_2)$. Columns of Z are those of X scaled by $\sqrt{n/l}$. The norm of the difference of two columns of X can be viewed as the norm of the difference of two feature vectors associated to K and thus can be bounded by d^K_{max}. Similarly, the norm of a single column of X is bounded by $K^{1/2}{}_{max}$. This leads to the following inequality:
$|\phi(S') - \phi(S)| <= (2n/l)d^K_{max}K^{1/2}{}_{max}$....(9).

The expectation of ϕ can be bounded as follows: $E[\phi] = E[||XX^T - ZZ^T||_2] <= E[||XX^T - ZZ^T||_F] <= (n/\sqrt{l})K_{max}$.....(10).

The inequalities (9) and (10) combined with Theorem 1 gives a bound on $||XX^T - ZZ^T||_2$ and yield the statement of the theorem.

8.2 Method 2

Theorem 4

Given the data set $X = (x_i)^n_{i=1}$, and the landmark point set $Z = (z_j)^m_{j=1}$. Then the Nystrom reconstruction of the kernel entry $K(x_i, x_j)$ will be exact if there exist two landmark points such that $z_p = x_i$, and $z_q = x_j$.

Proof
Let $K_{x_k,Z} \in R^{1xm}$ be the similarity between x_k and the landmark points Z. Then the Nystrom reconstruction of the kernel entry will be $K_{x_i,Z}W^{-1}K_{x_j,Z}$, where $W \in R_{mxm}$ is the kernel matrix defined on the landmark set Z. Let W(k) be the k^{th} row of W, then we have $K_{x_i,Z} = W(p)$ and $K_{x_j,Z} = W(q)$ since $x_i = z_p$, and $x_j = z_q$. As a result, the reconstructed entry will be $W^{(p)}W^{-1}(W^{(q)}) = W_{pq} = K(z_p, z_q) = K(x_i, x_j)$.

Theorem 5

We first define a partial approximation error for a sub-kernel Matrix. Suppose that the landmark set is $Z = z^m_{i=1}$, and the whole sample set X is partitioned into m disjoint clusters S_k's. Suppose each cluster has T samples. Repeat the following sampling process T times: at each time t, pick one sample from each cluster and denote the set of samples chosen at time t as X_{I_t}. Then $X = X_{I_1} \cup X_{I_2} \cup ... \cup X_{I_T}$, and the whole kernel matrix will be correspondingly decomposed into T^2 blocks, each of size m × m. Let K_{I_i,I_j}, and $E_{I_i,Z}$ be the m×m similarity matrices defined on (X_{I_i}, X_{I_j}) and (X_{I_i}, Z), respectively, and $W \in R^{mxm}$ the kernel matrix defined on Z. The partial approximation error is the difference between K_{I_i,I_j} and its Nystrom approximation under the Frobenius norm given by $E_{I_i,I_j} = ||K_{I_i,I_j} - E_{I_i,Z}W^{-1}E'_{I_j,Z}||_F$

We assume the kernel k satisfies the following property: $(k(a,b) - k(c,d))^2 <= C^k_X(||a-c||^2 + ||b-d||^2, \forall a,b,c,d$. For any kernel k, the partial approximation error E_{I_i,I_j} is bounded by $E_{I_i,I_j} <= \sqrt{2mC^k_X(e_{I_i} + e_{I_j})} + \sqrt{mC^k_X e_{I_i}} + \sqrt{mC^k_X e_{I_j}} + mC^k_X\sqrt{e_{I_i}e_{I_j}}||W^{-1}||_F$, where $e_{I_i} = \Sigma_{x_i \in X_{I_i}}||x_i - z_{c(i)}||^2$.

Proof
We will first define the following matrices $A_{I_i,I_j} = K_{I_i,I_j}W$; $B_{I_i,Z} = E_{I_i,Z}W$; $C_{I_j,Z} = E_{Ij,Z}W$ where $K_{I_i,I_j}(p,q) = k(x_{I_i(p)}, x_{I_j(q)}); E_{I_i,Z}(p,q) = k(x_{I_i(p)}, z_q); E_{I_j,Z}(p,q) = k(x_{I_j(p)}, z_q)$; and $W(p,q) = k(z_p, z_q)$.
Using the kernel property for kernel k, we get a bound on $||A_{I_i,I_j}||_F^2, ||B_{I_i,Z}||_F^2, ||C_{I_j,Z}||_F^2$. Now $||E_{I_i,I_j}||_F = ||W + A_{I_i,I_j} - (W + B_{I_i,Z})W^{-1}(W + C_{I_j,Z})'||_F = ||W + A_{I_i,I_j} - W' - C'_{I_j,Z} - B_{I_i,Z} - B_{I_i,Z}W^{-1}C'_{I_j,Z}||_F <= ||A_{I_i,I_j}||_F + ||B_{I_i,Z}||_F + ||C_{I_j,Z}||_F + ||B_{I_i,Z}||_F||C_{I_j,Z}||_F||W^{-1}||_F$

Theorem 6

The error of the Nystrom approximation is bounded by $E <= 4T\sqrt{mC_X^k eT} + mC_X^k Te||W^{-1}||_F$ where $T = max|S_k|$, and $e = \Sigma_{i=1}^n ||x_i - z_{c(i)}||^2$.

Proof
$\Sigma_{i,j=1}^T \sqrt{2mC_X^k(e_{I_i} + e_{I_j})} = \sqrt{2mC_X^k}\Sigma_{i=1}^T(\Sigma_{j=1}^T \sqrt{e_{I_i} + e_{I_j}})$
$\leq \sqrt{2mC_X^k}\Sigma_{i=1}^T(\sqrt{T}\sqrt{Te_{I_i} + \Sigma_{j=1}^T e_{I_j}}$
$\leq 2T\sqrt{mC_X^k Te}$.

Similarly the second and the third terms can be simplified as:
$\Sigma_{i,j=1}^T \sqrt{mC_X^k e_{I_i}} = \sqrt{mC_X^k}\Sigma_{i=1}^T(\Sigma_{j=1}^T\sqrt{e_{I_i}}) <= T\sqrt{mC_X^k Te}$.

The last term of the expression can be summarized as:
$\Sigma_{i,j=1}^T mC_X^k\sqrt{e_{I_i}e_{I_j}}||W^{-1}||_F = mC_X^k||W^{-1}||_F(\Sigma_{i=1}^T\sqrt{e_{I_i}})^2 <= mC_X^k||W^{-1}||_F Te$.

The constant C_X^k for the kernel property depends on the choice of kernel taken in the underlying KPCA. Let us take the Gaussian Kernel $G(\alpha) = exp(-\alpha^2)$ By using the mean value theorem and triangular inequality, we have, for any $a, b, c, d \in R$, $(k(a,b)k(c,d))^2 = (G(||a-b||/\sigma)G(||c-d||/\sigma))^2 = (G'(\epsilon)/\sigma)^2(||a-b||-||c-d||)^2$. We also have by $(||ab||||cd||)^2 <= (||ac|| + ||bd||)^2 <= 2(||ac||^2 + ||bd||^2)$. Thus we can choose C_X^k as $max((2G'(\epsilon)/\sigma)^2)$, which is often bounded.

9 Computational Results

We have taken the MNIST dataset, consisting of 1797 images samples of digits from 0 to 9. We have applied the Logistic Regression Classifier with a test-train split of 0.33 on this dataset to deduce the underlying digits in the images. On applying this classifier on the original dataset, we obtain an accuracy of 96.29%. However, this process would not be practical if the size of the dataset exceeds 10^6 since storing all 10^6 dimensions of the input matrix where each image is itself 64 dimensional would take up an enormous amount of space. Thus we applied the methods discussed in our paper to see if we efficiently countered the problem.

	Method	Accuracy	Time Taken
1	PCA	60.44%	0.009 s
2	Kernel PCA	78.96%	15.43 s
3	Nystrom KPCA	76.09%	0.22 s

First we apply the PCA dimensionality reduction with 5 principal components on our original dataset. After achieving a lower-dimensional dataset, we apply the same Logistic Regression Classifier on this new transformed dataset. The PCA algorithm runs in an exceptional 0.009

seconds. However, the accuracy obtained comes out to be 60.44%, far lower than the standard accuracy levels of 80%. So, while PCA takes hardly any additional time in reducing the dimensionality of the dataset, it suffers a substantial dip in its accuracy. This may be due to the fact that the image features do not have linear relationships with each other, which makes it difficult to project the data on a linear manifold. Thus, to tackle this possible issue, we tried out Kernel PCA. We transformed the dataset using Kernel PCA with 5 principal components, and then applied the same Logistic Regression Classifier on it. Now, the accuracy obtained on the classifier was 78.96%, which is a considerable improvement over the empirical PCA. However, the time taken to apply the KPCA algorithm on the original dataset is 15.43 seconds. This is a lot of time for a < 2000 samples dataset and would be computationally impractical for datasets of order 10^6 or more. Lastly, we tried out the Nystrom sampling approach. We transformed the original dataset using Nystrom KPCA with 5 principal components, followed by applying the same Logistic Regression classifier. This time we achieved an accuracy of 76.09% which is comparable to that of Kernel PCA. However, the runtime of the Nystrom KPCA algorithm was only 0.22 seconds, which is far better than the 15 secs. taken by KPCA. Thus, we can see that although the Nystrom approximation causes around 2% decrease in the accuracy of the given classifier, it is a much more time and space-efficient approach and is the way forward for larger and larger datasets. The Github repo containing the code for computational analysis can be found here.

10 References

- Bishop, Christopher M. Pattern Recognition and Machine Learning. New York : Springer, 2006.
- Tripathy, BK., S Anveshrithaa, and Shrusti Ghela. Unsupervised Learning Approaches for Dimensionality Reduction and Data Visualization. 1st ed. CRC Press, 2021
- Alessandro Rudi, Raffaello Camoriano, Lorenzo Rosasco. Less is More: Nystr¨om Computational Regularization, arXiv, 2016.
- Kai Zhang, Ivor W. Tsang, James T. Kwok. Improved Nystrom Low-Rank Approximation and Error Analysis, 2008.
- Sanjiv Kumar, Mehryar Mohri, Ameet Talwalkar. Sampling Methods for the Nystrom Method, 2012.