Algorithmic Description Of Principle Component Analysis (PCA)

The general idea of what PCA does is that it takes in a large dimensional dataset where the dimension refers to the number of features or variables in the dataset and flattens it into a smaller dimensional dataset while keeping the important information of the original dataset as much as possible such that it makes the analysis and computation of the dataset much easier and faster.

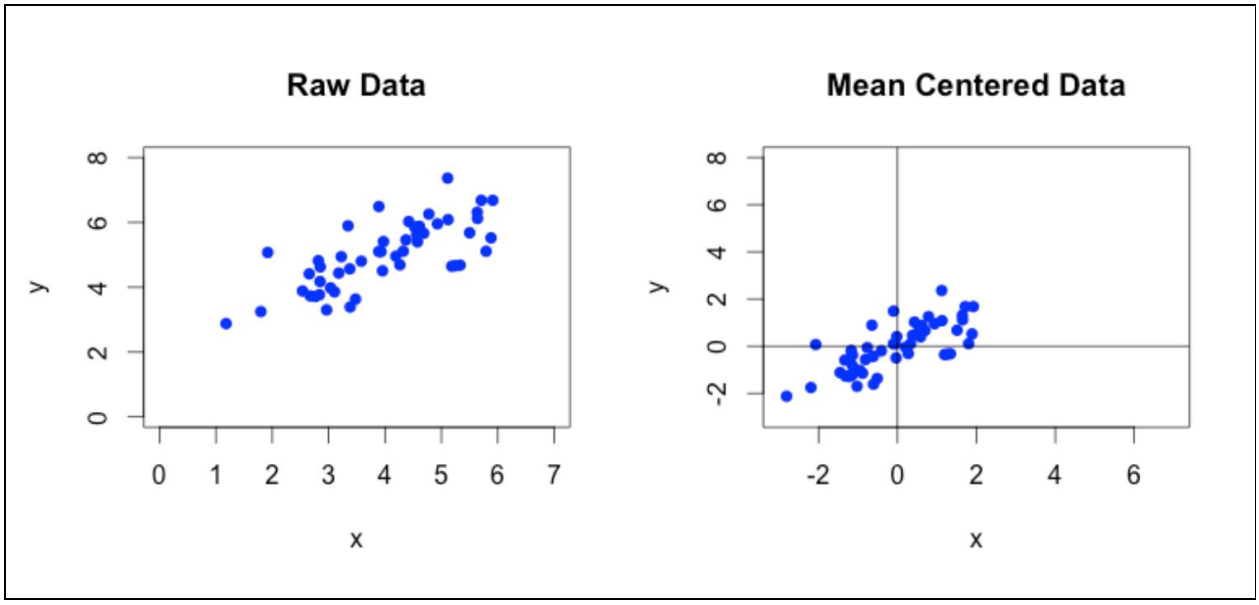


Figure 1: Step 1 (Source = https://cybernetist.com/2016/04/12/principal-component-analysis-part-2/)

- Step 1: Calculate the Mean of the given dataset x in order to calculate the center of the data.
- Step 2: Computing the Covariance Matrix to identify the relationship between k features of data in a given dataset. What it does is that it tells you about how much a particular feature is correlated to another where the lower the correlation the higher the variance. We are looking for features of the data with the highest amount of variance because more variance equal to more information.

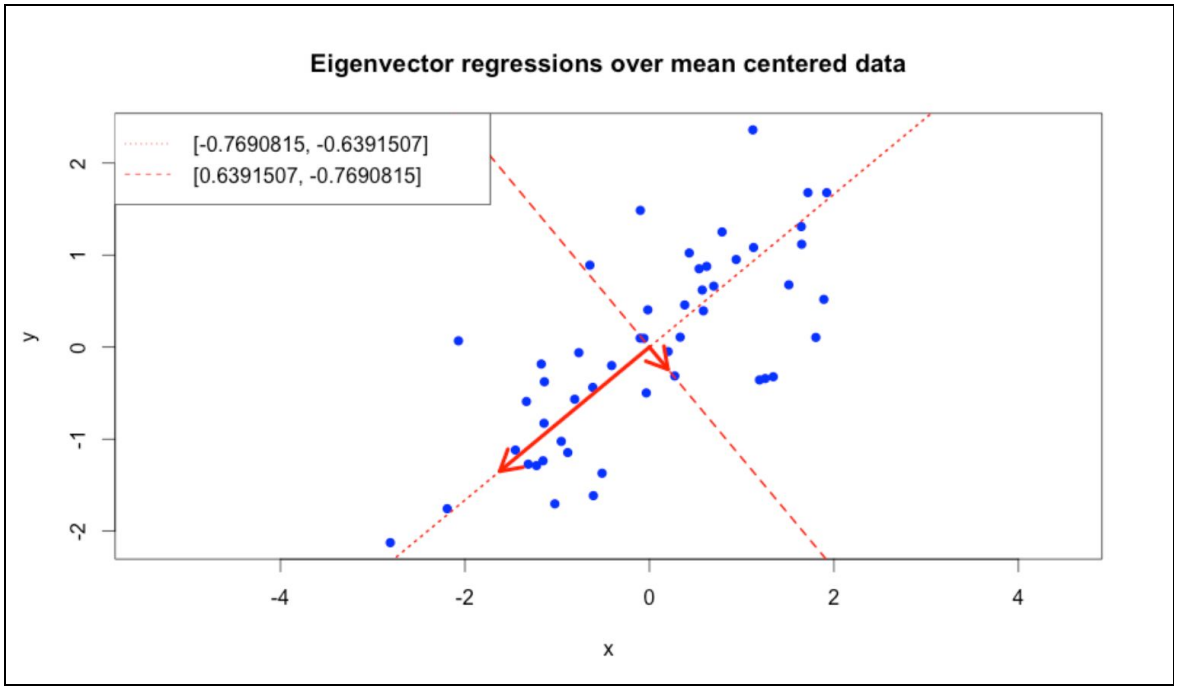


Figure 2: Step 3 (Source = https://cybernetist.com/2016/04/12/principal-component-analysis-part-2/)

- Step 3: Calculating the Eigenvectors and Eigenvalues where the Eigenvectors will be the principal components and Eigenvalues will be the measure of variance of the data in relation to the Eigenvectors/Principal Compoent. Seeing that this is the case, the higher the magnitude of the Eigenvalue the higher variance.
- Step 4: Order the Eigenvalues in the descending order such that the highest Eigenvalue will directly correlates to the Eigenvector/Principal Component with the highest amount of variance.

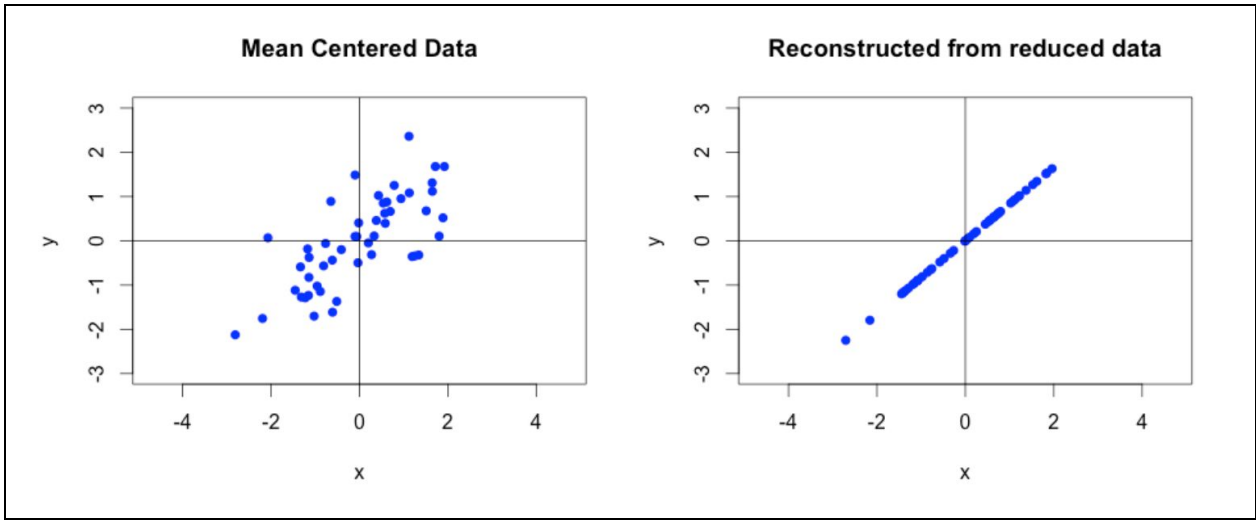


Figure 3: Step 5 (Source = https://cybernetist.com/2016/04/12/principal-component-analysis-part-2/)

- Step 5: Project the scaled version of the original dataset X onto Principal Component with the highest Eigenvalue to achieve the reduced dimensional dataset of X.

Algorithmic Description Of K-Means Clustering

K-means clustering algorithm is an unsupervised machine learning algorithm that is used on unlabelled data. The algorithm basically tries to group data points into K number of groups/clusters where data points that are in the same K group/cluster have the same features. This basically allows us to analyse on the organically labelled data points.

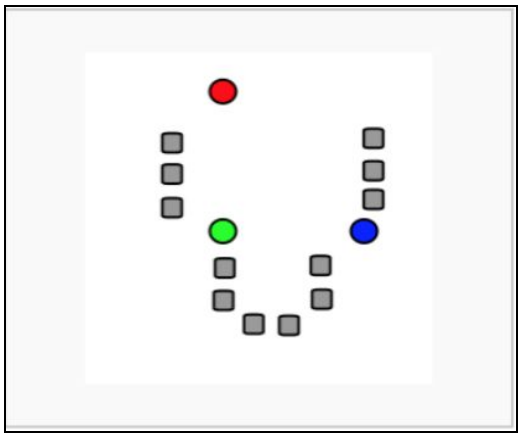


Figure 1: Step 1 (Source = https://brilliant.org/wiki/k-means-clustering/#)

- Step 1: Initialise K number of centroids randomly. The centroids here refer to the mean but initially the centroids are initialised as the data points of a given dataset, X.

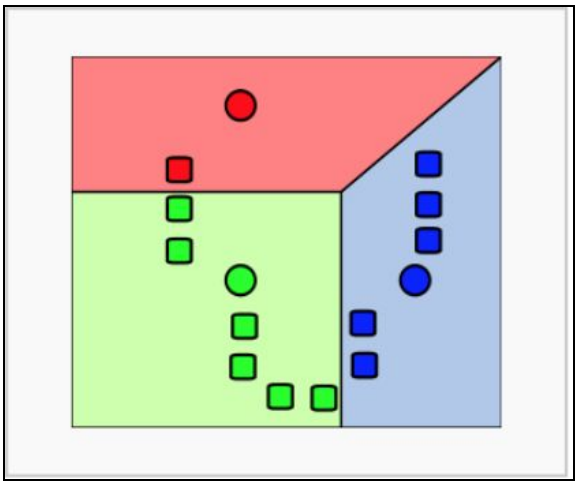


Figure 2: Step 2 (Source = https://brilliant.org/wiki/k-means-clustering/#)

- Step 2: Calculate the Euclidean distance of each data points in the given dataset, X with the K number of centroids. Then assign each of the data points to the K number of centroids it is closest to.
    - Mathematics behind it is as follows :
        - $ArgMin\ [\ D(x_{ij},\ C_k)\ =\ \sqrt{\sum_{j=1}^{j}(x_{ij} - C_k)^2}\ ]$
        - $D\ =\ Euclidean\ Distance\ Formula$
        - $x_{ij} = The\ data\ points\ of\ row\ i\ and\ feature\ j$
        - $C_k = The\ k\ centroid$

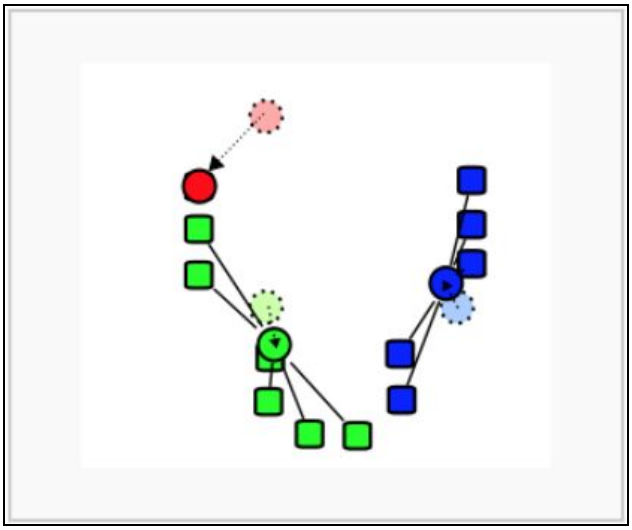


Figure 4: Step 3 (Source = https://brilliant.org/wiki/k-means-clustering/#)

- Step 3: Recompute the centroids based off of the mean of the data points in each group/cluster.
    - Mathematics behind it is as follows:
        - $C_k = \frac{1}{|S_k|} * \sum_{x_k \subseteq S_k} x_k$
        - $C_k = The\ k\ centroid$
        - $S_i = The\ Number\ of\ Sample\ data\ in\ the\ k\ Centroid/Cluster$
        - $x_k = The\ data\ k\ in\ the\ Sample\ data\ k$

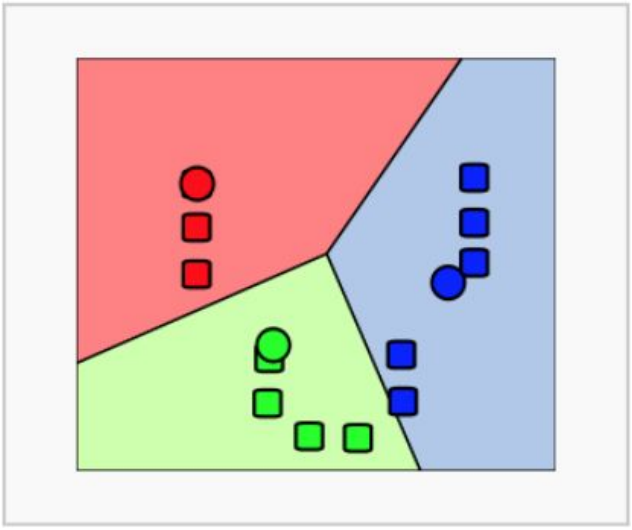


Figure 5: Step 4 (Source = https://brilliant.org/wiki/k-means-clustering/#)

- Step 4: Repeat step 2 and step 3 until the centroids don't change. (Convergence) Meaning we have found the most optimal centroids and cluster.

It is trying to minimise the squared error objective function : such that it minimises the distance of all points belonging to each of their respective cluster so that the centroids are more representative of the data points that it groups with.

$$J = \sum_{i=1}^{n} \sum_{j=1}^{K} \left| x_{ij} - \mu_j \right|^2$$

- n = *the number of samples or data x*
- K = *the number of centroids/clusters*
- x_{ij} = *the ith data point in the jth cluster*
- μ_j = *the jth cluster's mean*

Task 1 & 2

Figure 1: The Plot for the 1-Nearest-Neighbours Testing Error against the Number of Principal Components

Based on the plot above that was generated from my code, the testing error of the 1-Nearest Neighbours classifier seems to be decreasing at an exponential rate from 0 number of principal components to around 25 number of principal components. However, the testing error seems to be increasing right after the 25 number of principal components mark. This is well to be expected because the higher the number of principal components or features in this case can lead to a higher chance of overfitting meaning that it learns the features and noises of the training dataset too well that it adversely impacts the performance of the 1-Nearest Neighbours Classifier on the testing data. Another reason is that the higher the number of principal components the higher the amount of redundant information which would lead to a higher correlation between the data which in turns lead to a lower measure of variance between the data, in other words less important information.

The lowest testing error seems to be hovering around 20 to 50 number of principal components which is to be expected as explained above.

Task 3

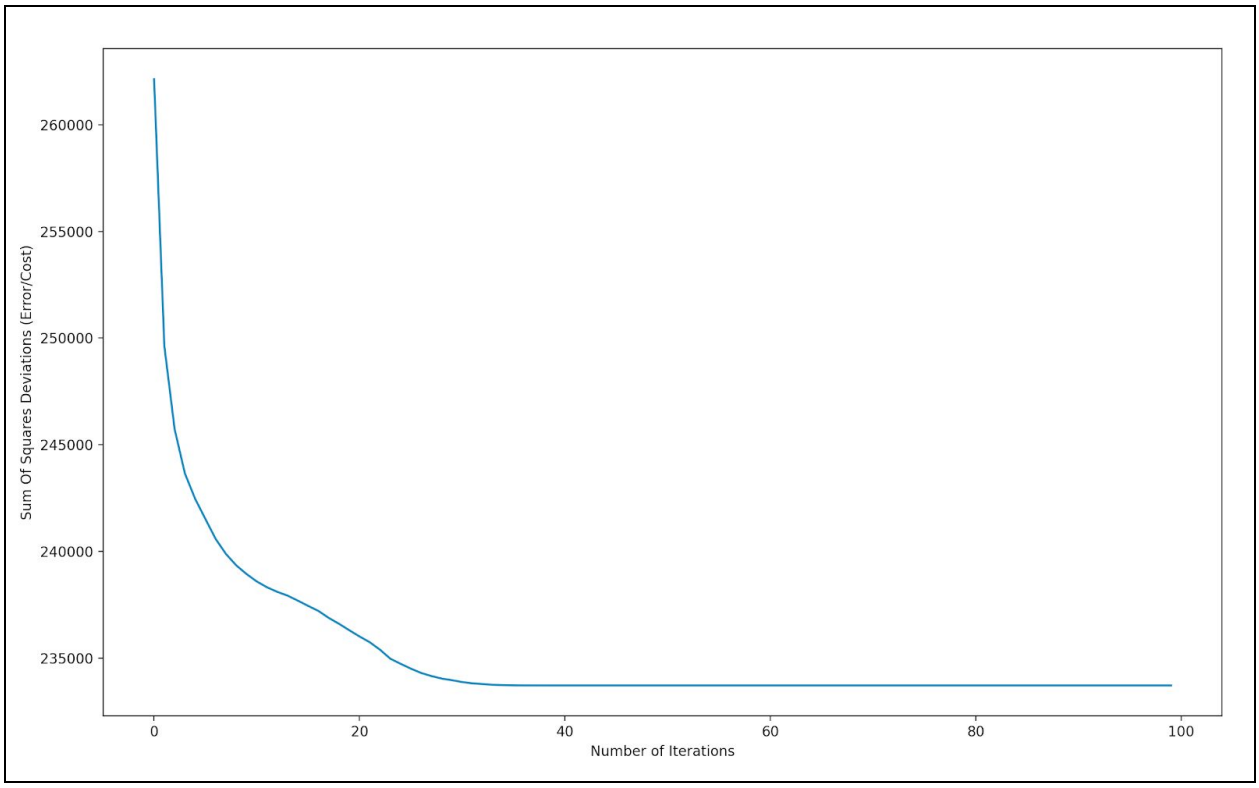
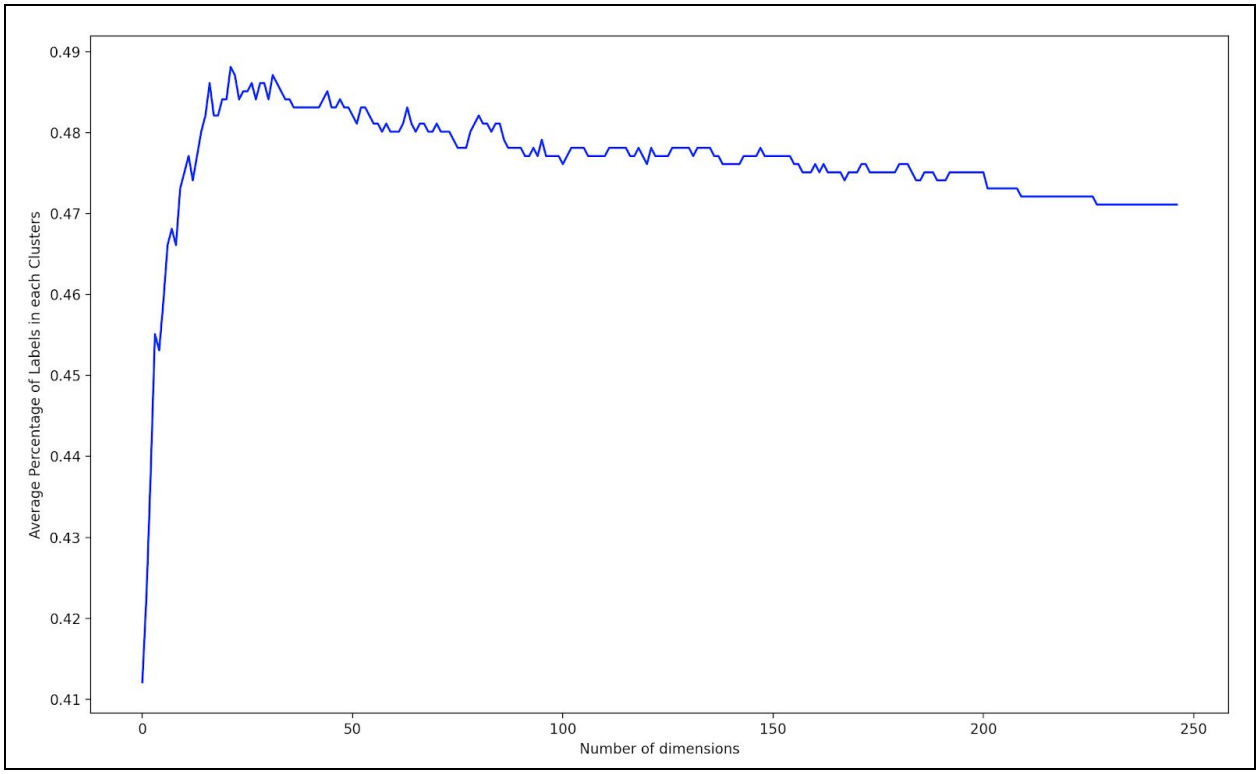


Figure 2: The Plot for the Sum of Squares of Deviations within each cluster(Cost/Error) against the number of iterations

Based on the plot above generated from my code, the sum of squares of deviation or variance between the data and the centroid of each cluster seems to be decreasing at an exponential rate until the number of iterations reaches 10. Then after this point, the sum of squares of deviation or variance seems to be decreasing at a much lower rate until it reaches around 27 number of iterations where the sum of square of deviation or variance seems to have converged. This is again to be expected. As the number of iterations starts increasing, the centroids will become more and more representative of the data points in each of the cluster meaning the variance between the data point and their respective cluster's mean gets smaller and smaller where it eventually converges to the local minimum of where the variances and centroids no longer changes. This also means that we have found the optimal centroids and clusters.

Task 4



Based on the plot above, the average of the combined percentage of the same initially set labels that are in each of the cluster seems to be increasing from 0 number of iterations to around 30 number iterations, after this point the percentage seems to be decreasing. This is again to be expected due to the fact that the higher the number of dimensions of dataset, the higher the chance of overfitting as the presence of noisy and redundant information are more prevalent which can sway the percentage of the same initially set labels that are in each of the cluster in a negative way.

The highest average of the combined percentage of the same initial labels seems to be around 20 to 50 number of dimensions. This makes sense as the goal of pca is to find a reduced dimensional dataset that contains the least amount of noise and the highest amount of important features that can lead to the best learning outcome.