Name: Krishna Chaitanya Naragam

MAV ID: 1001836274

Assignment 1

Using 1 of the late days' quota

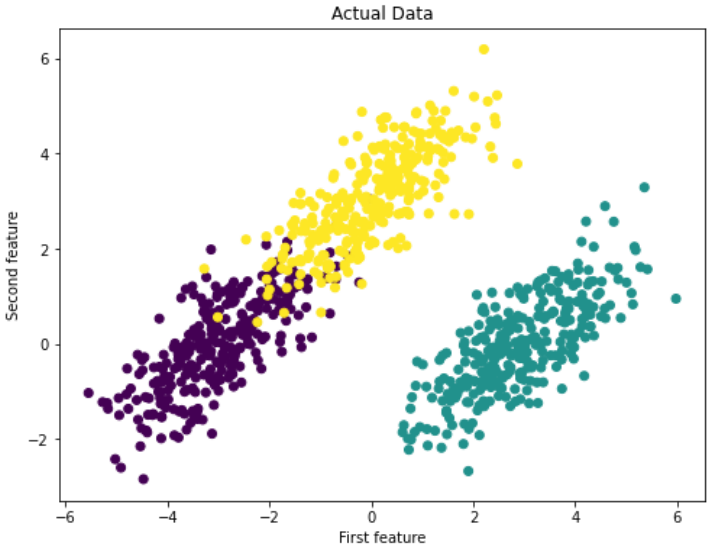
K-Means

The k-means method is a widely used clustering algorithm. One of its distinguished features is its speed in practice. Its worst-case running-time, however, is exponential, leaving a gap between practical and theoretical performance. It is a well-known method for partitioning n points that lie in the d-dimensional space into k clusters. Its main features are simplicity and speed in practice. Theoretically, however, the best known upper bound on its running time (i.e. O(nkd)) is, in general, exponential in the number of points (when kd=Ω(n log n)). (Vattani, 2009)

Problem 1

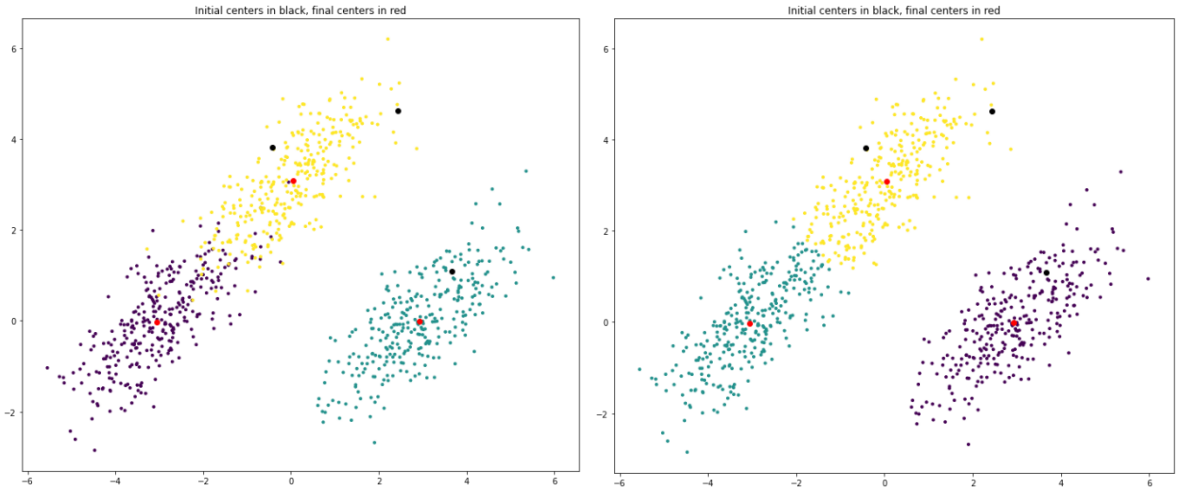
$$\mu_1 = [-3, 0],\ \mu_2 = [3, 0],\ \mu_3 = [0, 3],\ \Sigma = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix}$$

Given the above parameters, the task was to cluster the data into segments. K in k-means, refers to the number of clusters and the above mentioned 2D Gaussian random samples provide 2-dimensional data points on a cartesian plain that looks like the below image.



The algorithm uses L2-Norm or the Euclidian distance to find the distance between the points and assumes K number of fake centers to the data clusters. The algorithm then runs to adjust the centers in a way to

properly cluster the data. Given a new data point, or an existing data point in an unknown space, the cluster center closest to the point would be assigned as class to that data point. The algorithm's task is to adjust the centers until there is no further significant change in the positions of the clusters. Given the data, the algorithm starts off with random points and eventually spaces out to cluster the data.



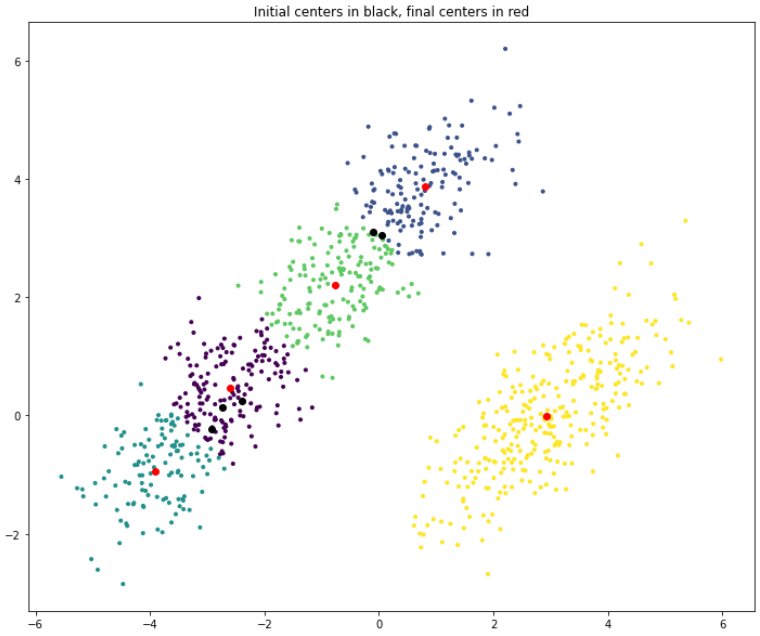
The left represents the data with actual labels and the right picture depicts the data after the algorithm being applied to calculate the clusters. We can see that the centers' perpendicular bisector is used to divide the clusters mutually and the pictures show the clusters when K = 3.

This was a simple example where we know the number of clusters, often in unsupervised learning, we never know the number of clusters. In such situations, the number of clusters is determined using the elbow method.

The elbow method looks at the percentage of variance explained as a function of the number of clusters: One should choose several clusters so that adding another cluster does not give much better modeling of the data.
More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion".
This "elbow" cannot always be unambiguously identified, making this method very subjective and unreliable.
Percentage of variance explained is the ratio of the between-group variance to the total variance, also known as an F-test. A slight variation of this method plots the curvature of the within group variance.

The method can be traced to speculation by Robert L. Thorndike in 1953. (Determining the Number of Clusters in a Data Set, n.d.)

The problem with K-means in an unsupervised scenario is that it could end up doing something like below for the same data when k=5.



And since there is now good way of knowing the number of clusters, in an unsupervised case, we might have to live that fact. Many techniques like the elbow method do exist but are limited when it comes to implementation and reliability.

Performing the task, the observation was that the number of clusters can never be fixed upon unless it is a supervised data and we have the class labels already. Even in a supervised situation, clustering is not always the best option to predict the class when the data overlaps. Like in the above case, the actual data has yellow and purple cluster overlapping and the final clustered output not giving any consideration to that. This decreases the accuracy in that region leading to wrong prediction and being wrongly classified. When K=3, the clusters almost classified to the actual data, but the overlap region would be misclassified.

When K was changed to be other values, most classes would be misclassified as there exists either less or a greater number of clusters. K=3 is an ideal situation where the number of clusters was equal to the actual cluster. The cluster in this data has good accuracy but the same might not be the case with different data that overlaps too much or has reasonable boundaries. With reasonably separated data, it is easier to cluster that and visual cues always help if the data is visualizable, which is not always the case. When the data overlaps too much, the k-means might not be the best option to classify there. This is the situation where we can use KNN classifier.

K-nearest Neighbors

In pattern recognition, the k-nearest neighbors(KNN) algorithm is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. (K-nearest Neighbors Algorithm, n.d.)

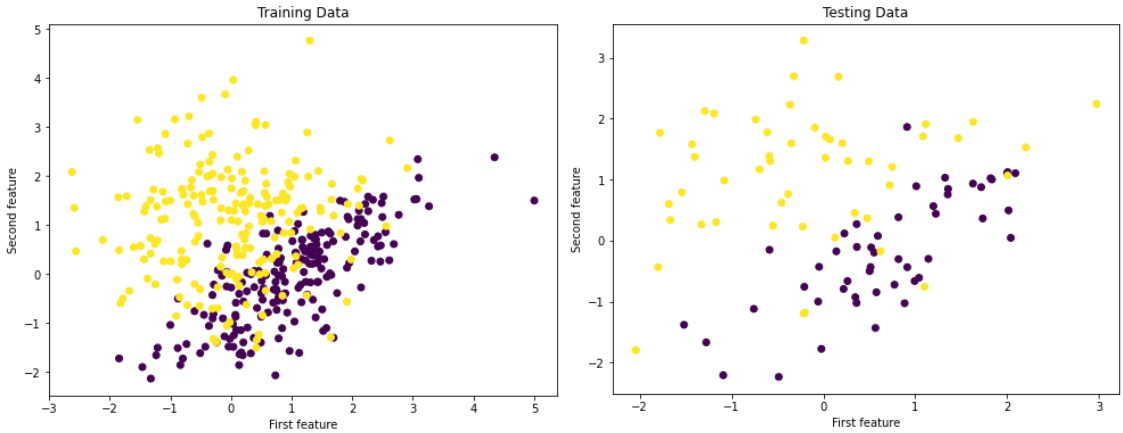
In KNN, the classification occurs with K nearest numbers. The maximum of the neighbors is selected as the class of the current given new element. It is a lazy algorithm that do not start the training process until the sample for classification is given. There are methods to skip few nodes that help speed up the process of finding the neighbors in a large space like, KD trees. In worst case all the nodes are traversed but it does improve the speed of the algorithm in other cases. KNN can also be used for regression. In case of regression, the mean of all the K neighbors is used to calculate the current regressed value. Improvements have been made on this as well to use weights that says the nearest neighbors weigh more to regression than that are far away. This is supposed to improve the accuracy of regression. Other methods have been proposed like using K nearest neighbors to fit a regression line and use least squares method or gradient descent methods to adjust the best fit in that data and use it to regress the point locally. This is also called as locally weighted regression.

The given data was,

$$\mu_0 = [1, 0],\ \mu_1 = [0, 1], \Sigma_0 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix},\ \Sigma_1 = \begin{bmatrix} 1 & -0.5 \\ 0.5 & 1 \end{bmatrix}$$

to generate the training and the test data. Initially we generated 200 training and 50 test samples form the defined variables.

The training samples(left) and testing samples(right) look as follows,



We can see that there is significant overlap in the training data. This is a classic situation where KNN could come in handy. Our task is now to predict the testing data using KNN, assuming we do not have the label. The algorithm uses K as the number of nearest neighbors to find and uses Euclidian distance to measure the distance between the points. Once we find the K nearest numbers for a classification problem, we would calculate maximum class that appears in the neighbors and assign the same class to the new sample. K is usually preferred to be odd to break ties.

The accuracy would be good in case of k=2 as the number of clusters is equal to the number of actual clusters. As the K value changes, the neighbors increase and the accuracy along with it. But increase it just too much and it is useless to do so. There is a sweet spot technique like elbow methods are useful here.

The observed accuracies for given data for different K value is,

K= 1; Accuracy: 81.00%

K= 2; Accuracy: 74.00%

K= 3; Accuracy: 84.00%

K= 4; Accuracy: 83.00%

K= 5; Accuracy: 88.00%

K= 10; Accuracy: 88.00%

K= 20; Accuracy: 88.00%

KNN can be used for regression also. The K still refers to the number of neighbors and the regressed value would be calculated by calculating the mean of the target value. The task given was to regress on the given 2D data.

$$\mu_0 = [1, 0], \Sigma_0 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix}$$

$$y = 2x_1 + x_2 + \epsilon$$

Where **ε** is gaussian random noise with mean as 0.5. 300 training samples were generated and 100 testing samples to do regression and root mean square error method was used to calculate the accuracy. The accuracies observed were as follows,

K= 1; RMSE is: 0.685278473891992

K= 2; RMSE is: 0.574095606766331

K= 3; RMSE is: 0.5353630970638048

K= 5; RMSE is: 0.5101459535541647

K= 10; RMSE is: 0.5080170973675647

K= 20; RMSE is: 0.5785607880374504

K= 50; RMSE is: 0.8865958814573042

K= 100; RMSE is: 1.254794904740767

The root mean square is defined as the square root of the mean square. The RMS is also known as the quadratic mean and is a particular case of the generalized mean with exponent 2. RMS can also be defined for a continuously varying function in terms of an integral of the squares of the instantaneous values during a cycle. (Root Mean Square, n.d.)

Lower RMS values are preferred over larger to reduce the error that occurred in the regression. There is no good way to decide the best K value here just the lowest RMS value is supposed to give good results.

The next task is to use weighted KNN to give weightage to nearest neighbors first and then use least squares method to fit a line and use that line to predict the new value and then calculate the RMS value average that out and use the K value with lower RMS value to get a good prediction.

The predictions of

K= 1; RMSE is: 2.610546619435418

K= 2; RMSE is: 10.27444432088117

K= 3; RMSE is: 8.664271510016356

K= 5; RMSE is: 7.716013576069756

K= 10; RMSE is: 5.8333233196571515

K= 20; RMSE is: 5.007624772788164

K= 50; RMSE is: 4.1662019622684765

K= 100; RMSE is: 3.841883542733365

We usually choose the k value with lowest RMSE. This was the predictions of the code that is attached with this document.

References

Determining the Number of Clusters in a Data Set. (n.d.). Retrieved 10 1, 2020, from Wikipedia: The Free Encyclopedia: http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

K-nearest Neighbors Algorithm. (n.d.). Retrieved 10 1, 2020, from Wikipedia: The Free Encyclopedia: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Manthey, B., & Röglin, H. (2009). *Improved smoothed analysis of the k -means method*. Retrieved 10 1, 2020, from https://dl.acm.org/citation.cfm?id=1496770.1496821

Root Mean Square. (n.d.). Retrieved 10 1, 2020, from Wikipedia: The Free Encyclopedia: http://en.wikipedia.org/wiki/Root_mean_square

Vattani, A. (2009). k-means requires exponentially many iterations even in the plane. *Discrete and Computational Geometry, 45*(4), 324-332. Retrieved 10 1, 2020, from http://cseweb.ucsd.edu/~avattani/papers/kmeans-journal.pdf