COMP6245 Foundations of Machine Learning – Assessed Work

Supritha Konaje
Email: ssk1n21@soton.ac.uk
Student ID: 32864477

1. Lab 1 to Lab 5

All the 5 Labs were attempted and completed. After completing the code and understanding the concept, uploaded the reports for the respective labs within the mentioned deadline. Feedback for the first 3 labs were given, and the suggestions mentioned for few of the labs were implemented and cleared the concept associated with it.

2. Non-Negative Matrix Factorization

2.1 Implementation for Non-Negative Matrix Factorization

As the name suggests, Non-Negative Matrix Factorization (NMF) is a factorization of a matrix which has values that are greater than 0. When we are given with a multivariant data, we can perform statistical analysis using NMF. Consider a vector $\boldsymbol{X}$ which has a dimension $\boldsymbol{n*m}$. According to the rule, X needs to be factorized into two matrix $\boldsymbol{W}$ and $\boldsymbol{H}$ with dimensions $\boldsymbol{n*r}$ and $\boldsymbol{r*m}$ where $\boldsymbol{r}$ is the rank into which the matrix had to be reduced for better analysis of data**.** $\boldsymbol{r}$ should always be less than the $\boldsymbol{n}$ and $\boldsymbol{m}$. This gives us a compressed matrix that of the original matrix. The NMF equation is $\boldsymbol{V \approx WH}$. Here, $\boldsymbol{W}$ is the basis vector and $\boldsymbol{H}$ is weighted by its component.

In the algorithm that is used to perform NMF, we are updating the value of $\boldsymbol{W}$ and $\boldsymbol{H}$ for a particular number of iterations. Hence, we will be going with the multiplicative update technique of Lee and Seung. This way it improves the approximation and converge to the locally optimal matrix factorization. The equation given is using the circle dot product also known as the Hadamard product and element wise elimination.

$$\mathrm{W} \leftarrow \mathrm{W} \odot \frac{XH^T}{WHH^T}, \quad \mathrm{H} \leftarrow \mathrm{H} \odot \frac{XW^T}{HWW^T}$$

Figure 1 shows the convergence of NMF algorithm using synthetic data. We are randomly, generating data for **X**. Initially, we are assigning a random value to $\boldsymbol{W}$ and $\boldsymbol{H}$ with dimensions as $\boldsymbol{n*r}$ and $\boldsymbol{r*m.}$ Value of $\boldsymbol{r}$ is 6 in the figure given below. For 40000 iterations it will update the value of W and H. Error is calculated by calculating the difference between each the original data $\boldsymbol{X}$ and the value of $\boldsymbol{W.H}$.

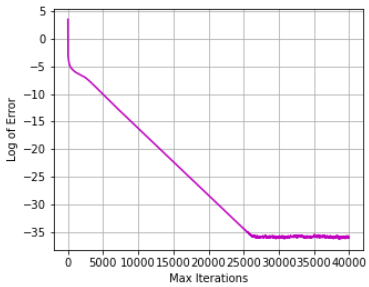

Figure 1: Convergence of NMF algorithm on synthetic data

2.2 Compare the sklearn.decomposition with self-developed NMF

Next task was to compare the results that were obtained by using the inbuilt *NMF* provided by *sklearn.decomposition.* Parameters used to perform NMF are ***n_components=6*** (rank of matrix), ***init='random'*** (random initialization of *W* and *H* initially), ***max_iter=MaxIter*** (Maximum number of times the value of W and H must be calculated).

The reconstruction error when calculated for both the methods are given in the Table 1. The self-developed NMF algorithm has a better result as compared to the inbuilt algorithm. The performance might improve if we give appropriate parameters to the inbuilt method.

Method Used	Reconstruction Error
Using self-developed Algorithm	2.4793e-16
Using sklearn.decomposition	0.00015

Table 1: Reconstruction error for both the ways of finding NMF

2.3 Using NMF to decide factor-trading in stock market

- Next task was to apply self-developed algorithm to the dataset *Equities.xlsx*. The dataset has a record of stock analysis for FTSE100 and its constituent assets from the year 2011 to 2019. We need to invest in combination of assets that might track the index of the dominant determinants of the market. The dataset was passed to the algorithm with ***r=10*** for plotting the convergence graph of NMF as seen in Figure 2. It is seen that more the data, the smoother the curve and it is converging smoothly.

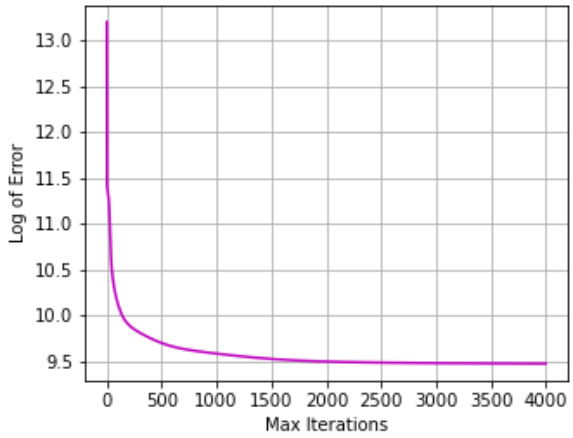

Figure 2: Convergence of NMF algorithm on the Equities.xlsx data

- To find the correlation with the FTSE index, ***pandas.corr()*** function was used by passing the ***W*** to correlate the data and columns in it to FTSE100 column. In Figure 3, the heatmap shows the correlation values of the 10 columns against FTSE100. The value that has negative is not correlated to the FTSE100. 1 being the highest value for FTSE100 indicates that it very well correlated with FTSE100 but in our case, it was FTSE100 itself correlating to itself. So, the value closer to 1 is chosen and it is seen from Figure 2 that BAE

SYSTEMS has the maximum value of correlation. Hence, BAE SYSTEMS correlates the most.

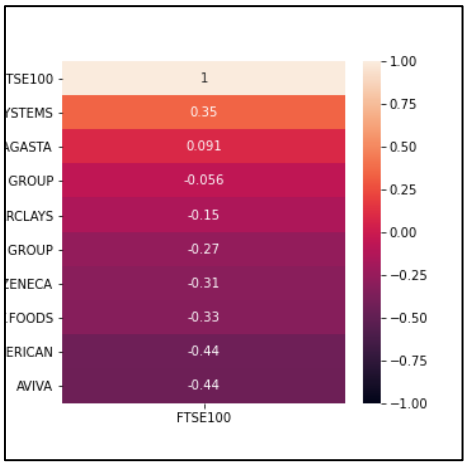

Figure 2: Heatmap for the finding the correlation between FTSE100 and the other 10 assets (BAE SYSTEMS, ANTOFAGASTA, ASHTEAD GROUP, BARCLAYS, ADMIRAL GROUP, ASTRAZENECA, ASSOCIATED BRIT. FOODS, ANGLO AMERICAN, AVIVA)

- To compare the cumulative return for FTSE100 and BAE SYSTEMS, we first calculate the difference between the weights of the first day weight and last day weight and divide it by the first day weight. The graph is given in Figure 3 for the FTSE100 and BAE SYSTEMS.

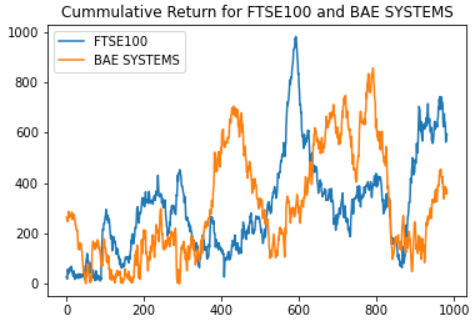

Figure 3: Cumulative Return graph for FTSE100 and BAE SYSTEMS

- When the assets are chosen in random, the performance is not that great as seen in Figure 4. The is not easy to interpret and it doesn't summarize the what the graph indicates. Hence, factorizing first and then calculating the W and H will make it better for easy interpretation of data.

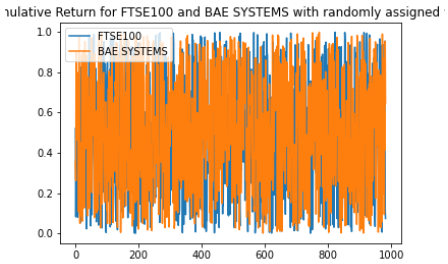

Figure 4: Return for FTSE100 and BAE SYSTEMS with randomly assigned weights

3. K-Means Clustering

3.1 Implementing K-Means Clustering Algorithm

When a dataset is divided into multiple groups based on a pattern it is termed as clustering. When there are target variables to compare our outputs with it is termed as Supervised Learning. In some cases, we do not have the target variables, that is termed as Unsupervised Learning. One of the famous algorithms of unsupervised learning is the K-Means algorithm.

Algorithm for K-Means clustering is as follow:

i. Assign a random value for the initial variable around which our datasets will get clustered.
ii. Calculate the Euclidean distance from each data point to the centroid that was previously assigned.
iii. Form a data cluster around the minimum distance between the centroid and data point from dataset.
iv. Now, calculate the new centroid by calculating the average/mean of the grouped dataset.
v. Repeat step ii. to iv. until we get the same centroids

In Figure 5, using the K-Means algorithm, scatter plot is plotted.

Figure 5: Clustering using K-Means with the centroids

3.2 Contours on Probability Density:

Next task was to draw contours using the probability density. In Figure 6, we can see the contour plots are surrounded around the centroid. And the contours are spread over covering each cluster.

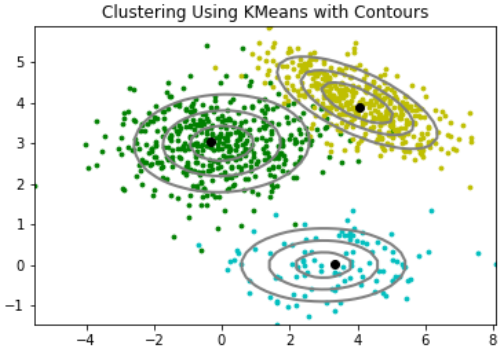

Figure 6: Clustering using K-Means with Contours around the centroid

3.3 Implement K-Means using sklearn

Next task was to perform K-Means using sklearn package. In Figure 7, the scatter plot after performing K-Means is shown. It is seen that both the regions are similar that we got from the sklearn package and from the algorithm. Accuracy of both the plots are also almost similar.

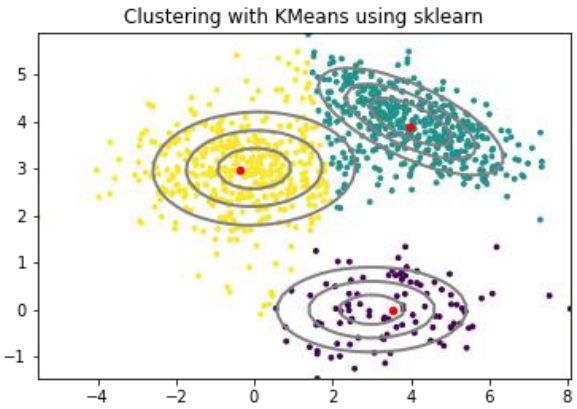

Figure 7: Clustering using K-Means using sklearn.KMeans

3.4 Sensitivity of Initial cluster centers and value of k

In Figure 8, the scatter plot for the same synthetic data is plotted after applying K-Means algorithm but with plotting the initial and final centroid. Initially the centroid is chosen randomly. In the graph it is seen that the initial centroids belong to same cluster. Hence, this fails the algorithm.

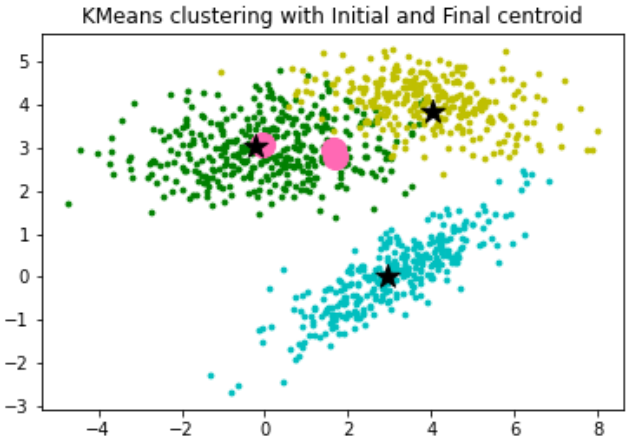

Figure 8: KMeans clustering with Initial and Final Centroid

The value if k is not determined before we perform K-Means. If we knew the value for number of clusters, then we can at first initialize the value of centroid randomly. Next step would be instead of calculating the average of the newly formed cluster, we can choose a random data point from the assigned cluster as our new centroid. Repeat the process until we get same centroid. Here, instead of calculating the average of the cluster points, we take a cluster point from the assigned cluster. This way, we can get minimum inertia between choosing the initial and final cluster points.

We can calculate the value of k before performing K-Means using Elbow Curve to determine the number of clusters before performing the K-Means.

3.5 KMeans on K-Class classification dataset

The K Classification chosen for implementing K-Means was **wine.data.** It has 178 samples and 13 features based on the content of a wine. There are 3 variety of wine available. We will perform K-Means with cluster size as 3. In **wine.data**, the variance between two features is different. This will hamper the performance of the algorithm. To use this data, the data needs to be transferred and this is achieved by *StandardScalar()* from *sklearn*. It transforms every feature to have mean 0 and variance as 1.
Now we will use K-Means to assign each data point to its respective type of wine. Since, there are three types of wine, number of clusters(k) will be 3.

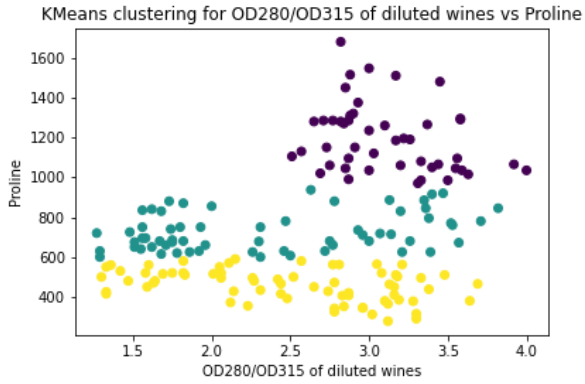

Figure 9: Wine dataset clustered into 3 clusters where they represent 3 types of wine without standardizing the features

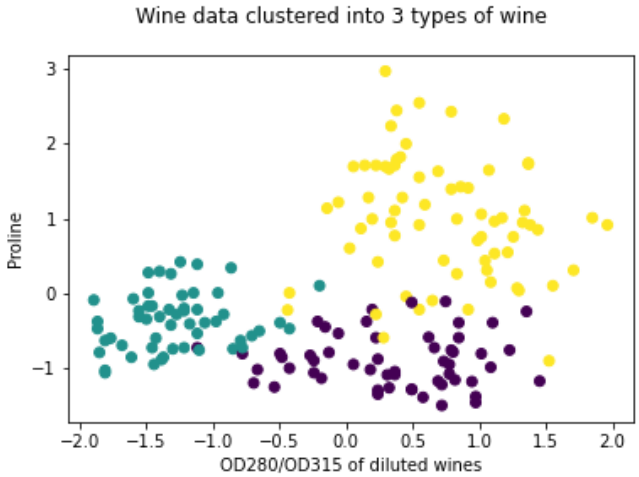

Figure 10: Wine dataset clustered into 3 clusters where they represent 3 types of wine with standardizing the features

In Figure 9 and Figure 10, we can see the difference when we use K-Means without standardizing and when we use it after standardizing.