# ML Assignment 3: Semi-Supervised Learning on Fashion MNIST Dataset

Raktim Dey-MDS202132, Sucheta Jhunjhunwala-MDS202151

May 27, 2022

## 1 Introduction

- Initially we load the Fashion MNIST dataset from the keras library and split the dataset into train and test samples.
- The training dataset has 60000 images, each of dimension 28*28 pixels. We further split our training set into a validation set and a smaller training set. We also scale the pixel intensities to 0-1 range by dividing each pixel with 255.
- Since each image in the training and testing dataset is a two dimensional array, we flatten the array to form a one dimensional vector by reshaping.
- Before proceeding with semi-supervised learning with KMeans, we built a Neural Network with two hidden layers, one consisting of 300 neurons and the other consisting of 100, and an output layer with 10 nodes. Using this model, we obtained an accuracy of 88% on the unlabeled data.
- We initially plot inertia against different values of k and find that the elbow occurs at 8. But we know that the dataset has 10 categories of clothing items we shall perform Kmeans for two values of k, k=10 and k=8.

## 2 Applying KMeans when k=10

- After fitting the model with k=10 we transform the training set into cluster distance space.
- From the cluster-distance space we look at the variables present at the minimum distance and then look at their corresponding images in the training dataset.
- We then hand labelled the 10 images obtained and then performed label propagation on the entire training set on the basis of the hand labelled data.
- Finally we fit the training data with the propagated labels to Logistic Regression and a Neural Network Model to label the test dataset.

| **Model** | **Accuracy** |
|---|---|
| Logistic Regression | 48% |
| Neural Network | 48 % |

## 3 Applying KMeans when k=8

We perform the same steps as above with just k =8.

| **Model** | **Accuracy** |
|---|---|
| Logistic Regression | 47.8% |
| Neural Network | 47% |

# 4 Comparative Evaluation

| Model | k | Time | Memory |
|---|---|---|---|
| Logistic Regression | 10 | 36.3s | 3631.52Mb |
| Neural Network | 10 | 13.1s | 1514.25Mb |
| Logistic Regression | 8 | 36.3s | 3631.52Mb |
| Neural Network | 8 | 13.1s | 1514.25Mb |

# 5 Conclusion

The Fashion MNIST dataset was better trained by a Neural Network with two layers, When we used semi-supervised learning, that is, used clustering to label some clothing items, and then labelled all the items in that cluster the same as the centroid, the accuracy fell to 48% , because a lot of information was lost when we converted 3 dimensional data into 2 dimensional data. In general, Neural Network provided same accuracy as Logistic Regression but in less time and occupied less memory. But we also noticed that when we increased the number of clusters to a large number, say 500, the accuracy increased to around 78%, but it took a long time to apply logistic regression to this clustered data.

```
In [26]: %%memit

         #Applying Logistic Regression to label the entire dataset
         t1=time.perf_counter()
         log_reg = LogisticRegression(multi_class="ovr", solver="saga", max_iter=5000, random_state=42) #train the model
         log_reg.fit(X_train_2d, y_train_propagated) #fitting the model
         y_pred=log_reg.predict(X_new_test) #predicting the values on the test set
         t2=time.perf_counter()

         print("Logistic Regression with 500 clusters took: ", (t2-t1),"seconds")

         log_reg.score(X_new_test, y_test) #accuracy of the prediction

         Logistic Regression with 10 clusters took:  3216.4432616000004 seconds
         peak memory: 873.75 MiB, increment: 0.02 MiB

In [27]: log_reg.score(X_new_test, y_test) #accuracy of the prediction

Out[27]: 0.7851
```

Figure 1:

# 6 Link

https://colab.research.google.com/drive/1ODMd0OnLkpV5RIdKdb2DfbQqPp0ixsNE