Machine Learning Report

ZIYUE XU

June 2022

1 Introduction

The objective is to compare the performances of three 3 different algorithms: K-means clustering, logistic regression and MLP on a classification problem. The test uses the MNIST data set, which is a data set containing pictures of handwritten digits from 0 to 9 written by different people all over the world. A comment of the data set is made at the end of the report.

2 K-means Clustering

2.1 Introduction of K-means Clustering

K-means clustering is an unsupervised learning algorithm that uses data without labels during training. The main purpose of K-means clustering is to cluster all the data into k groups and to find the cluster's centroid that can minimize the variance between the centroid and all the data within the cluster. That is, for a given sample $D = \left\{x^1, x^2, ...x^m\right\}$,where $x^i \in \mathbb{R}^n$, K-means clustering will find a collection of k clusters with centroids $\mu_1, \mu_2, ..., \mu_k$, where $\mu_i \in \mathbb{R}^n$. Each data x^i in the sample D is assigned to the cluster c^i where $c^i \in [1, 2, ..., k]$.The optimization objective of K-means clustering is to find $c^1, c^2, ..., c^m, \mu_1, \mu_2, ..., \mu_k$ to minimize J, where J is

$$J\left(c^1, c^2, ..., c^m, \mu_1, \mu_2, ..., \mu_k\right) = \frac{1}{m}\sum_{i=1}^{m}\left\|x^i - \mu_{c^i}\right\|^2$$

2.2 Algorithm of K-means Clustering

K-means clustering is an iterative procedure that starts by initialising the initial centroids, and then repeatedly assigning data to their closest centroids and then recalculating the centroids based on the data assigned.

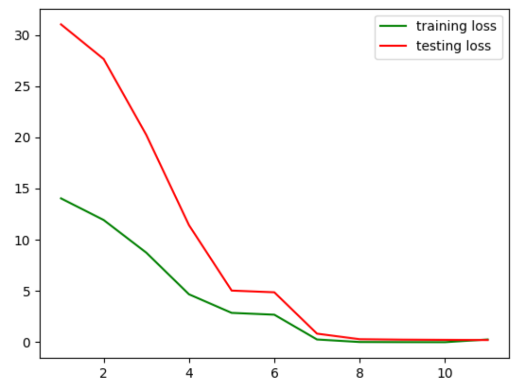


Figure 1: Elbow Method. y-axis: WCSSD, x-axis: K

The K-means algorithm can be represented as follow:

Algorithm 1 K-means Clustering

```
1: Choosing k (Elbow Method / Intuition)
2: Randomly initialise k clusters (centroids)
3: Calculate the distance between the each data and centroids
4: Assign each data to the nearest cluster
5: Calculate new centroids
6: Repeat step 2 - 4 until no further changes in centroids
```

The value k can be determined using intuition if the number of clusters is clear. In this test, as there are only 10 different digits, 10 can be a reasonable value for K. K can also be selected using elbow method. The elbow method will run K-means clustering for many times with different values of K and calculate the corresponding within cluster sum of squared distance. A graph similarly to figure 1 will be plotted. It is clearly from the graph that the cost decreases significantly before K = 7 and there is almost no change in cost after K = 7. So in this example, K = 7 will be a good choice.

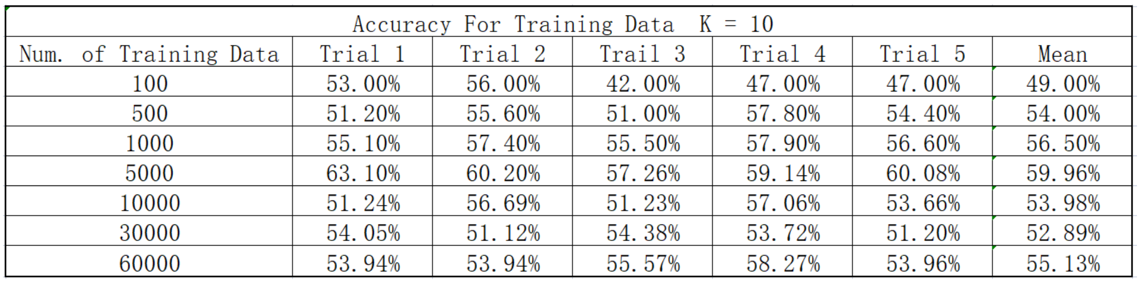
There are varies method to initialise the centroids. The most straightforward one is randomly choose k samples as means. The method is fast but sometimes can lead to a local minimum and results a lower accuracy as the means chosen at the beginning can affect the later performance. Another way is called K-means++ initialization. Each mean is determined based on its distance between other means so that the distances between each mean can be maximized. This can reduce the probability of a local minimum .

2.3 Experiments on Clustering Data and Results

In this section we elaborate our experiments on MNIST data set. Particularly, we investigate the influence of both the value of k and the number of training data on the performance of the algorithm. We describe the experiments on the numbers of training data in section 2.3.1 and that on the value of k in section 2.3.2.

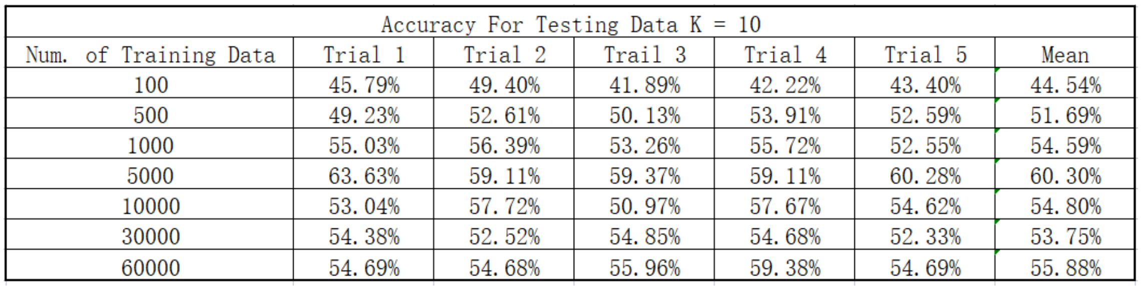
2.3.1 Clustering Data using Same Number of Clusters (Same K)

As the data set contains numbers from 0 - 9, it is conspicuous to set the number of cluster K to 10. The number of training data used varies from 100 to 60000. The results are shown in figure 2 and figure 3.



Accuracy For Training Data K = 10						
Num. of Training Data	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Mean
100	53.00%	56.00%	42.00%	47.00%	47.00%	49.00%
500	51.20%	55.60%	51.00%	57.80%	54.40%	54.00%
1000	55.10%	57.40%	55.50%	57.90%	56.60%	56.50%
5000	63.10%	60.20%	57.26%	59.14%	60.08%	59.96%
10000	51.24%	56.69%	51.23%	57.06%	53.66%	53.98%
30000	54.05%	51.12%	54.38%	53.72%	51.20%	52.89%
60000	53.94%	53.94%	55.57%	58.27%	53.96%	55.13%

Figure 2: The accuracy of model training with same number of clusters (K = 10) but different number of training data.

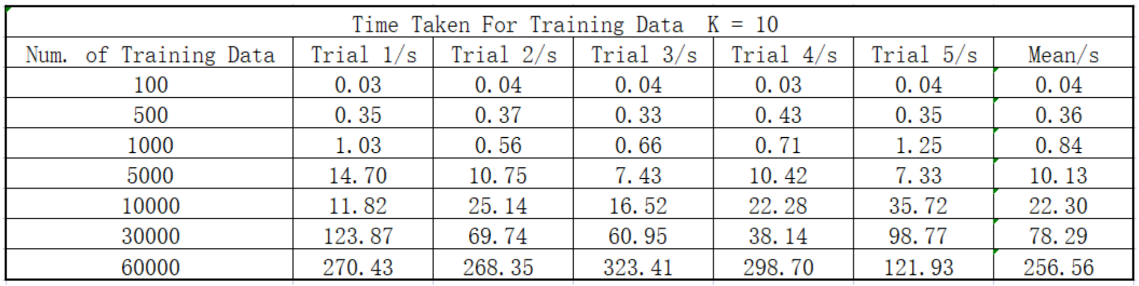


Accuracy For Testing Data K = 10						
Num. of Training Data	Trial 1	Trial 2	Trail 3	Trial 4	Trial 5	Mean
100	45.79%	49.40%	41.89%	42.22%	43.40%	44.54%
500	49.23%	52.61%	50.13%	53.91%	52.59%	51.69%
1000	55.03%	56.39%	53.26%	55.72%	52.55%	54.59%
5000	63.63%	59.11%	59.37%	59.11%	60.28%	60.30%
10000	53.04%	57.72%	50.97%	57.67%	54.62%	54.80%
30000	54.38%	52.52%	54.85%	54.68%	52.33%	53.75%
60000	54.69%	54.68%	55.96%	59.38%	54.69%	55.88%

Figure 3: The result of evaluation on all testing data based on models trained with different number of training data.

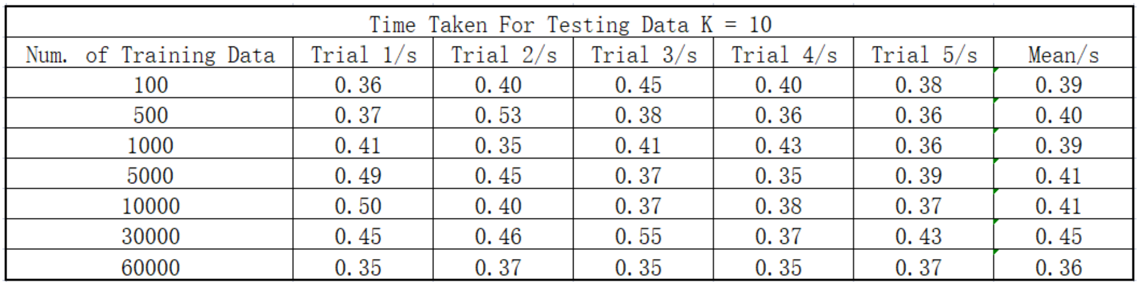
From figure 3, with 5000 images as training data provides the best mean testing score of 60.30%. The accuracy for recognizing handwritten digits is relatively low for small number of training data. The accuracy does increase with the increase in number of training data taken, though for large number of training data like M = 60000, it does not have a significant increase the accuracy.

The time taken for each model to be trained is also recorded as follows.



Time Taken For Training Data K = 10						
Num. of Training Data	Trial 1/s	Trial 2/s	Trial 3/s	Trial 4/s	Trial 5/s	Mean/s
100	0.03	0.04	0.04	0.03	0.04	0.04
500	0.35	0.37	0.33	0.43	0.35	0.36
1000	1.03	0.56	0.66	0.71	1.25	0.84
5000	14.70	10.75	7.43	10.42	7.33	10.13
10000	11.82	25.14	16.52	22.28	35.72	22.30
30000	123.87	69.74	60.95	38.14	98.77	78.29
60000	270.43	268.35	323.41	298.70	121.93	256.56

Figure 4: The time taken for each model with same number of clusters (K = 10) but different number of training data.



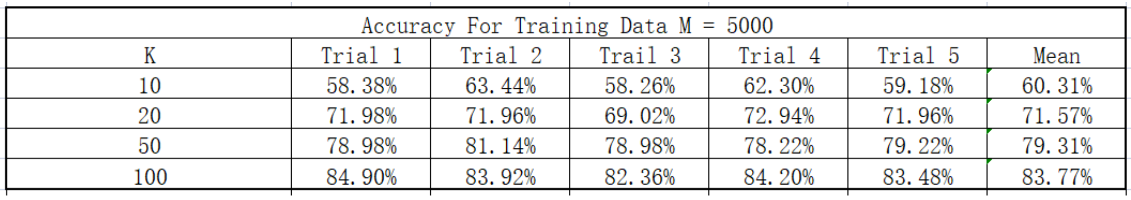
Time Taken For Testing Data K = 10						
Num. of Training Data	Trial 1/s	Trial 2/s	Trial 3/s	Trial 4/s	Trial 5/s	Mean/s
100	0.36	0.40	0.45	0.40	0.38	0.39
500	0.37	0.53	0.38	0.36	0.36	0.40
1000	0.41	0.35	0.41	0.43	0.36	0.39
5000	0.49	0.45	0.37	0.35	0.39	0.41
10000	0.50	0.40	0.37	0.38	0.37	0.41
30000	0.45	0.46	0.55	0.37	0.43	0.45
60000	0.35	0.37	0.35	0.35	0.37	0.36

Figure 5: Time taken for each model to finish test with testing data.

From figure 4, it is obvious that the time taken for training model increases with the number of training data. For large number of M, it takes a longer time to train but its performance of recognizing the digits is worse than the model with M = 5000. In addition, the time taken for testing the data for all models are almost the same. In general, the model with M = 5000 and K = 10 has the best performance in this test.

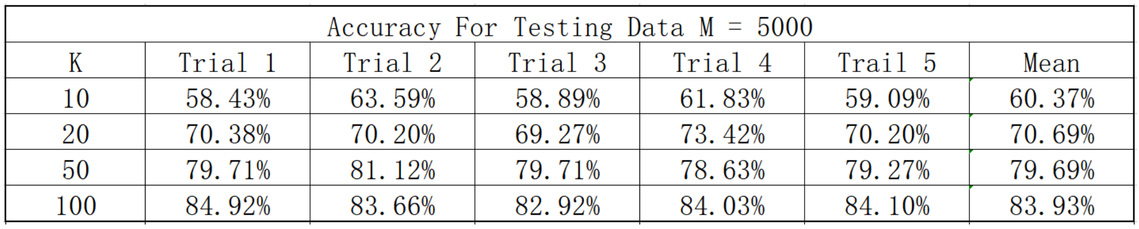
2.3.2 Clustering Data using Same Number of Training Data

As the accuracy of model using K = 10 is only about 60%, it is considerable to increase the number of cluster to fit the data. After viewing the data set, we find that the digits written by different people has different shapes even though they represent the same number. Thus using only 10 clusters may confine the accuracy. In this way more number of clusters K = 10 * n (n is a integer) is tried to fit more types of digits. This means that there may be more than one cluster that corresponds to a single digit. The following training and testing uses M = 5000 training data as the models with M = 5000 has the best performance in the previous experiment.



Accuracy For Training Data M = 5000						
K	Trial 1	Trial 2	Trail 3	Trial 4	Trial 5	Mean
10	58.38%	63.44%	58.26%	62.30%	59.18%	60.31%
20	71.98%	71.96%	69.02%	72.94%	71.96%	71.57%
50	78.98%	81.14%	78.98%	78.22%	79.22%	79.31%
100	84.90%	83.92%	82.36%	84.20%	83.48%	83.77%

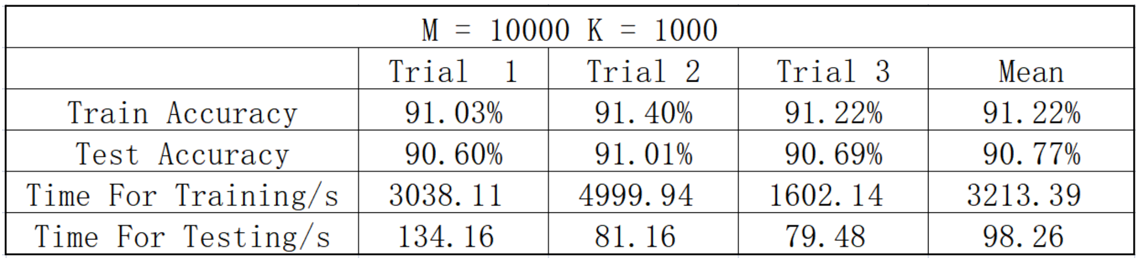
Figure 6: The accuracy of model with same number of training data (M = 5000) but different number of clusters.



Accuracy For Testing Data M = 5000						
K	Trial 1	Trial 2	Trial 3	Trial 4	Trail 5	Mean
10	58.43%	63.59%	58.89%	61.83%	59.09%	60.37%
20	70.38%	70.20%	69.27%	73.42%	70.20%	70.69%
50	79.71%	81.12%	79.71%	78.63%	79.27%	79.69%
100	84.92%	83.66%	82.92%	84.03%	84.10%	83.93%

Figure 7: Accuracy of corresponding model on testing data.

From figure 6 and 7, it is obvious that the accuracy of model increases significantly with the increase of number of cluster. The best model produced using K-means clustering is the model with number of cluster K = 1000 and training data M = 10000. It takes an average time of about 54 minutes to train, which is acceptable, and its accuracy for recognizing both training data and testing data are greater than 90%. The accuracy of the model may increase with larger number of M but the time taken would be too long.



M = 10000 K = 1000				
	Trial 1	Trial 2	Trial 3	Mean
Train Accuracy	91.03%	91.40%	91.22%	91.22%
Test Accuracy	90.60%	91.01%	90.69%	90.77%
Time For Training/s	3038.11	4999.94	1602.14	3213.39
Time For Testing/s	134.16	81.16	79.48	98.26

Figure 8: The model with training data M = 10000 and number of cluster K = 1000.

2.4 Conclusion

The K-means Clustering algorithm has limited, relative low performance on the data set. This may due to the reason that digits are not separable in Euclidean space. The intermingled digits make it difficult to calculate the Euclidean distances between samples and means. The time taken for training the model is moderate and acceptable.

3 Logistic Regression

3.1 Introduction of Logistic Regression

Logistic regression is a supervised learning algorithm, which learns the classification of data. It is usually used for the binary classification. For a given input x^i, the logistic regression will calculate a corresponding output y^i and classify the input according to the value of y^i. As the value of y^i may be unbounded, we need to transform y^i into a probability value, ranging from 0 to 1 to represent its probability belonging to one class. A commonly used function is Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$.

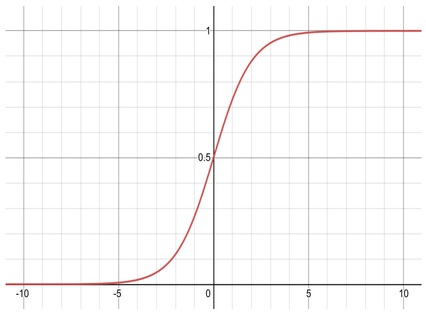


Figure 9: The graph of Sigmoid function.

The Sigmoid function has the features that it can map any value to the range from 0 to 1. The gradient close to x = 0 is very steep. Sigmoid function is also monotonous differentiable, which is useful when training a logistic regression.

After calculating the output $\hat{y^i}$ for the input x^i, the logistic regression will classify it. If the output y^i is greater than 0, the Sigmoid function will have an output greater than 0.5, which will then classify the input x^i class 1. Otherwise the input x^i will be classified as class 2.

$$h_\theta(x) = P(\hat{y} = 1|x; \theta) = \sigma(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Figure 10: $h_\theta(x)$ is the probability that y = 1 given x, parameterized by θ, where $x \in \mathbb{R}^{n+1}$ and $\theta \in \mathbb{R}^{n+1}$

$$\hat{y} = \begin{cases} 1 & h_\theta(x) \geq 0.5 \\ 0 & h_\theta(x) < 0.5 \end{cases}$$

Figure 11: Relationship between $h_\theta(x)$ and prediction $\hat{y}$

In conclusion, logistic regression is a model that can be used to classify the inputs. It has advantages such as predict not only the classification of one input data but also the probability that the data belongs to the classification. However, logistic regression can only classify linear - separable data.

3.2 Mathematical Model

Logistic regression is a linear classifier.

For an input x^i, where $x^i \in \mathbb{R}$, the probability that x^i belongs to the classification y $= 1$ is $\sigma\left(\theta_0 + \theta_1 x\right)$. If $\theta_0 + \theta_1 x \geq 0$, which is $x \geq -\frac{\theta_0}{\theta_1}$, then the prediction $\hat{y} = 1$. In the other case where $\theta_0 + \theta_1 x < 0$, which is $x < -\frac{\theta_0}{\theta_1}$, then prediction $\hat{y}$ $= 0$. Thus the input x^i can be classified by the decision boundary $x = -\frac{\theta_0}{\theta_1}$. This is a straight line in space and is a linear classifier.

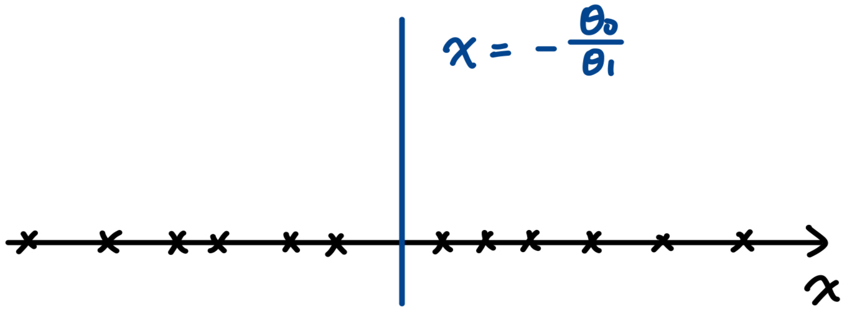


Figure 12: Linear classifier $x = -\frac{\theta_0}{\theta_1}$ for $x^i \in \mathbb{R}$.

Similarly, for an input $x^i \in \mathbb{R}^2$, where $x^i = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, the probability that x^i belongs to the classification y $= 1$ is $\sigma\left(\theta^T x^i\right)$ with $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$. In this situation, we will calculate the dot product of x^i and θ to obtain the angle between the two vectors. If the angle is greater than $90°$, then the prediction $\hat{y} = 0$, or it will be $\hat{y} = 1$. The decision boundary for this case will be the normal of θ.
For the hyperplane with dimension 3, 4 or more, the idea of linear classifier can also be adapted.

3.3 Implementation

As MNIST data set has 10 categories in total (0 - 9), the problem becomes a multi-class classification. The activation function used here is softmax function

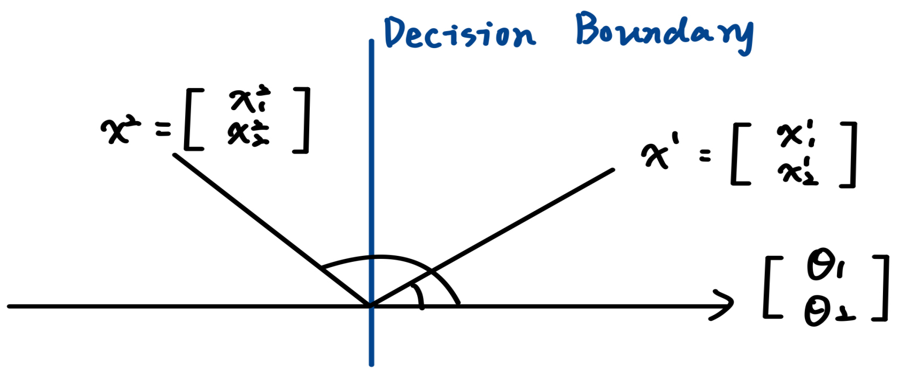

Decision Boundary
$x^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$
$x^1 = \begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix}$
$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

Figure 13: Linear classifier for $x^i \in \mathbb{R}^2$.

instead of Sigmoid. The softmax function can also project the value into a region from 0 - 1, which can be regarded as the probability that this value occurred.

$$S_i(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Figure 14: Softmax function.

The MNIST data set has input $x^i \in \mathbb{R}^{784}$. For each input x^i , the prediction value $\hat{y}$ would be Softmax$\left(\theta^T x^i\right)$, which is an array of shape$(10,)$. Suppose the true value for input x^i is y^i, the expected output should be an array of shape $(10,)$ with 1 in the position y^i and 0 in all other positions. To calculate the difference between the two distributions, the cross entropy function is used. The cross entropy function calculated the additional information needed to transform one distribution into another using the equation:

$$\sum_{i=1}^{n} p\left(x^i\right) log\left(q\left(x^i\right)\right),$$

where P is the target model and q is the current model. Output the additional information needed to transform q to p.

As the expected output has only one 1 and nine 0s, the cross entropy loss would be:

$$loss = -\sum_{i=1}^{n} y^i log\left(\hat{y^i}\right)$$

The model is then trained by gradient descent.

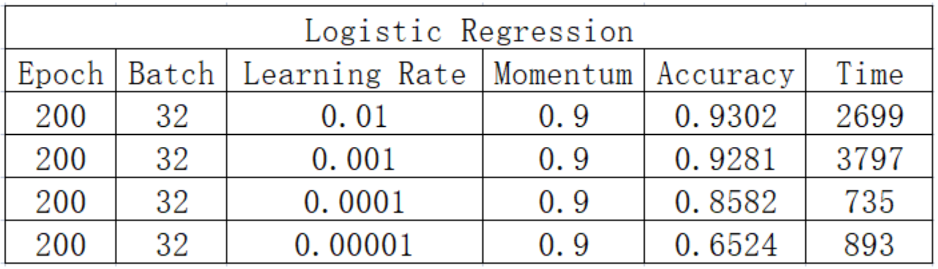
3.4 Results and Conclusion

All the following results are trained using stochastic gradient descent. The hyperparameter adjusted is the learning rate. The hyperparameter momentum is tested but the results are not ideal, so the following results all done with momentum 0.9.

The whole data set is broke down into small number of batches. The gradient is updated with the images in the batches only instead of using the whole 60000 images. This may lead to some inaccuracy in gradient descent, but the overall trend for the gradient descent is correct as each time the gradient is changed to make the expectation of images in the batch better.. The epoch refers to the number of times the whole data set is trained.

The number of epoch and batches are set according to the data set. As the MNIST data set is relative simple and easier to converge, so it is quick for the function to converge to local minimum. Setting the number of epoch and batches to large number will not help too much in this case. Thus we don't evaluate on the performance of model with different number of epoch and batches.

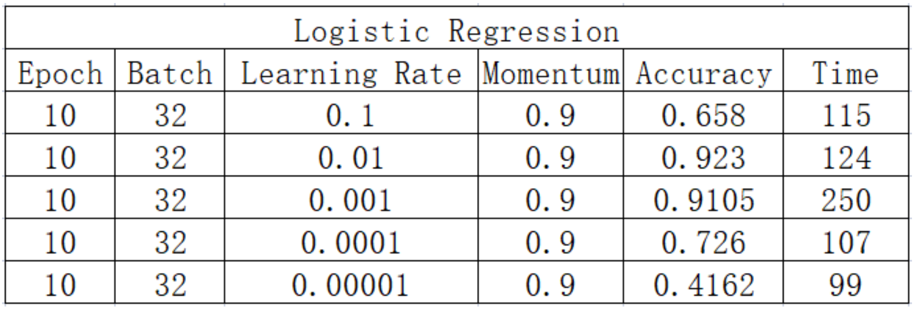
Learning rate is another factor that can affect the accuracy of the model. The learning rate of the model decides the speed of gradient descent. If the learning rate is either too big or too small, there is a high probability of encountering a local minimum. Different learning rates are tested with 32 batches and number of epochs 10 and 200. The results are as follows:



Logistic Regression					
Epoch	Batch	Learning Rate	Momentum	Accuracy	Time
200	32	0. 01	0. 9	0. 9302	2699
200	32	0. 001	0. 9	0. 9281	3797
200	32	0. 0001	0. 9	0. 8582	735
200	32	0. 00001	0. 9	0. 6524	893

Figure 15: Model with varying learning rate with epoch 200 and batches 32.

As we can see from figure 18 and 19, the smaller learning rate may lead to lower accuracy as there is more chance for the local minimum to be encountered, though the time taken is less. There is also an exception in figure 19, where the learning rate is 0.1 but the accuracy is only 0.658. This may be caused by that the gradient is decreasing at a relative quick rate and ends in the local minimum.



Logistic Regression					
Epoch	Batch	Learning Rate	Momentum	Accuracy	Time
10	32	0.1	0.9	0.658	115
10	32	0.01	0.9	0.923	124
10	32	0.001	0.9	0.9105	250
10	32	0.0001	0.9	0.726	107
10	32	0.00001	0.9	0.4162	99

Figure 16: Model with varying learning rate with epoch 10 and batches 32.

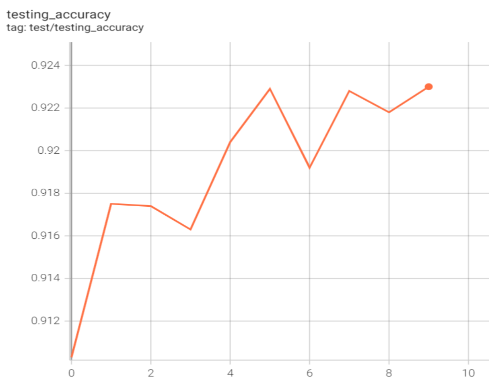


Figure 17: Accuracy of model with number of epoch 10, number of batches 32 and learning rate 0.01.

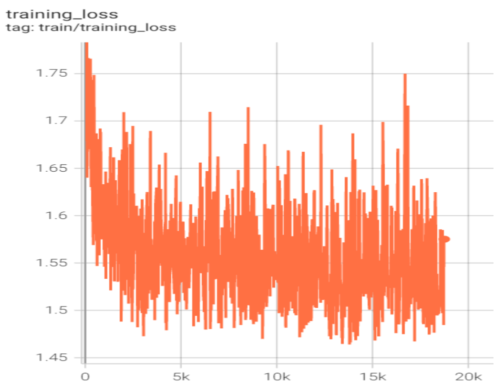


Figure 18: Training loss for model with number of epoch 10, number of batches 32 and learning rate 0.01.

In conclusion, the model of logistic regression has the best performance with number of epoch 10, number of batches 32 and learning rate 0.01. The accuracy of the model is 92.3% and the time requirement for training is acceptable and reasonable, which takes only about 2 minutes.

4 Multi-layer Perceptron(MLP)

4.1 Introduction of Multi-layer Perceptron

A multi-layer perceptron contains one input fully-amended layer, one output layer and some hidden layers. Each layer can have different number of neurons. Each neuron accepts inputs from all the neurons in the previous layer. The output for a neuron is the weighted sum of activations in the previous layer plus the bias.

For the output of each neuron, an activation function is used. The activation function used when implementing the multiple layer perceptron is the Relu function. It is defined as: $f(x) = max(0, x)$. The figure 23 shows the graph for the Relu

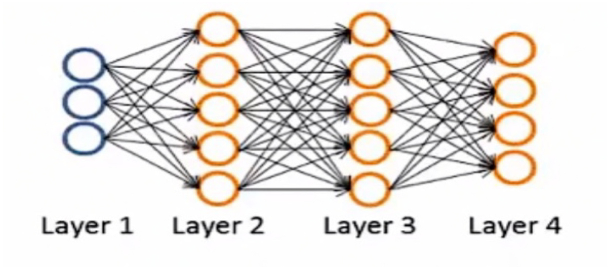


Figure 19: The composition of a MLP with 2 hidden layers.

function.

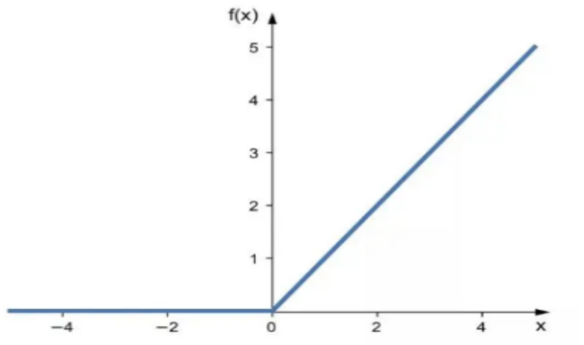


Figure 20: Graph for Relu function.

If there is no activation function in MLP, then the output for each layer is only a linear combination of the input layer, which is equal to a MLP without hidden layer. In this way, a non-linear activation function is introduced to allow the MLP able to fit any types of function instead of the linear combination of input only. This makes the MLP much stronger.

4.2 Mathematical Model

The following explanation of MLP uses a model with one hidden layer. The MLP is shown below:

Let $a_i^{(j)}$ be the activation of unit i in layer j and $\theta^{(j)}$ be the matrix of weights controlling function mapping from layer j to layer j+1. The activation function used is represented by g(x). Then the output unit for the above MLP can be written as follows (x_0 and $a_0^{(2)}$ are bias units):

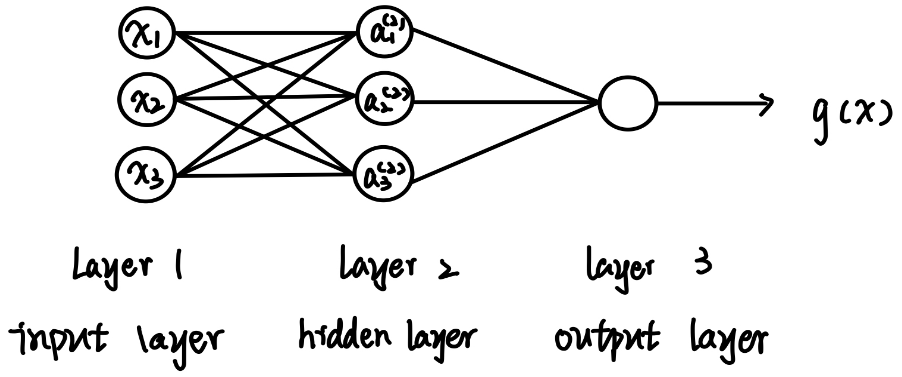


Figure 21: A MLP with one hidden layer. Three neurons in layer 1 and layer 2. One neuron in layer 3 (Output layer)

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$a_1^{(3)} = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

The $\theta^{(j)}$ will be dimension of $S_{j+1} * (S_j + 1)$ if the MLP has S_j units in layer j and S_{j+1} units in layer j + 1.

4.3 Connection with Logistic Regression

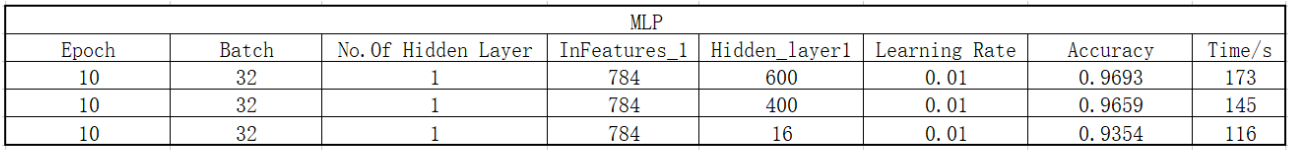
Logistic regression is actually a multi-layer perceptron with no hidden layer. Logistic regression can only classify linearly separable data. In comparison, the hidden layers and activation function allow MLP to classify non-linearly separable data, which is powerful than logistic regression. MLP will firstly transform data to a feature space where they are linearly separable. Then a logistic regression will be applied.

4.4 Implementation and Results

As the model used in logistic regression with number of epoch 10, number of batches 32 and learning rate 0.01 has the best performance and efficiency, so these parameters are kept when training the multi-layer perceptron.

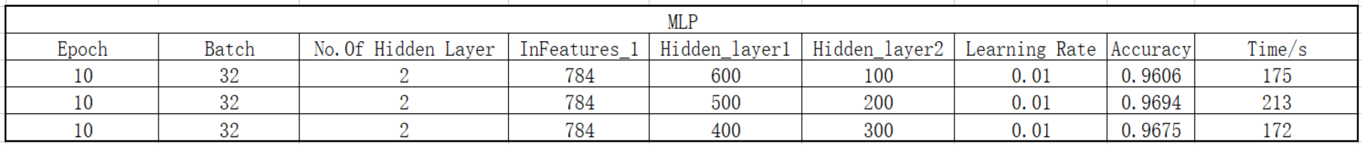
The model is trained with different number of layers. After observing the accuracy and time taken for training, the change in the number of layers do not have a significant impact on the accuracy of the model. Most of the models trained can have a good performance of accuracy 96% on testing data. Time taken is relative short. It can spend less than 4 minutes to get a model with high accuracy. The number of neurons in each hidden layer does have a little impact on the accuracy of output. Since the accuracy doesn't vary a lot, the impact can be negligible.

The overall performance of MLP on MNIST data set is satisfying on both accuracy and training time.



MLP							
Epoch	Batch	No. Of Hidden Layer	InFeatures_1	Hidden_layer1	Learning Rate	Accuracy	Time/s
10	32	1	784	600	0.01	0.9693	173
10	32	1	784	400	0.01	0.9659	145
10	32	1	784	16	0.01	0.9354	116

Figure 22: Model of MLP trained with 1 hidden layer.



MLP								
Epoch	Batch	No. Of Hidden Layer	InFeatures_1	Hidden_layer1	Hidden_layer2	Learning Rate	Accuracy	Time/s
10	32	2	784	600	100	0.01	0.9606	175
10	32	2	784	500	200	0.01	0.9694	213
10	32	2	784	400	300	0.01	0.9675	172

Figure 23: Model of MLP trained with 2 hidden layer.

5 Conclusion on Three Models

Before training the three models, I expected the MLP with the best performance and K-means with the worst. After testing, the results are the same as my expectation.

For K-means clustering, it is an unsupervised learning algorithm. Each iteration may lead to a better performance according to the present means but the performance can be influenced by many factors. Firstly, the initialization of means has a great influence on the later performance of the model. The K-means is also sensitive to the outliers. In case that the handwritten digit is too different from normal digits, it would be hard for K-means to classify it correctly. Secondly, the outlier will influence the mean value and have impacts on the classification of other digits. Moreover, if the number of clusters is set to 10, there is a high probability that the corresponding digit associated with each mean does not contain all the 10 digits. This will results in that the digits not in the output list will not be recognized. Even though this can be implemented so that all the 10 digits have its corresponding mean, the accuracy is still low compare to the others.

For MLP and logistic regression, it is clear that MLP is more powerful than logistic regression as it can classify non-linear data while logistic regression can't. For MLP, data is firstly transformed to a feature space and then logistic regression is used in the feature space. Comparing the performances of algorithms on the MNIST data set, MLP can produce a more accurate model than logistic regression in similar time. Thus the MLP has the best performance among the three algorithm.

6 Conclusion on MNIST Data Set

The evaluation of MNIST data set uses K-means clustering trained with 10000 training data and 1000 clusters. Each of the images below is the composition of all the mean of clusters with the same digit. The general shape of each digit is correct with come exceptions like the digit 1 or 9, which is indistinct at the end of the digit. This may corresponding to the problem that a digit can be written in different ways, which leads to a lower classifying accuracy.

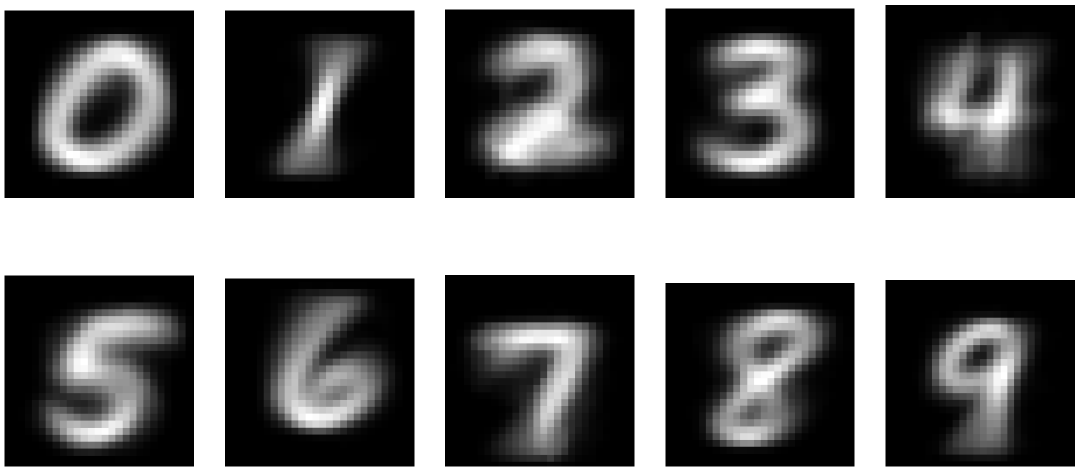
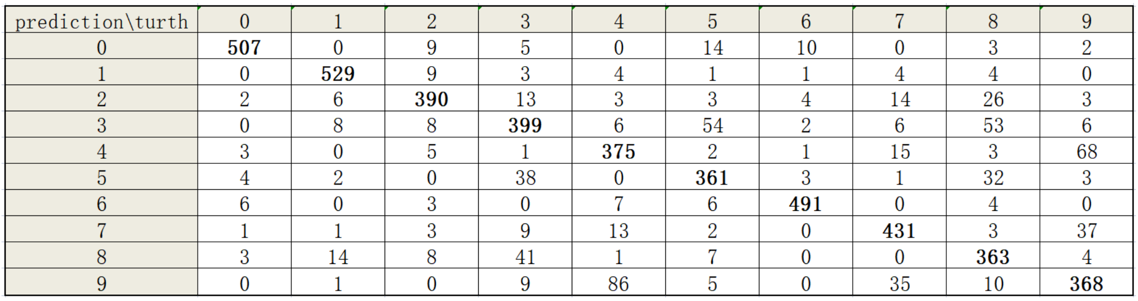


Figure 24: The mean value of digits 0-9 output after training.

After noting down the prediction value with the true value, there are some trends in the recognition of the handwritten digits.



prediction\turth	0	1	2	3	4	5	6	7	8	9
0	**507**	0	9	5	0	14	10	0	3	2
1	0	**529**	9	3	4	1	1	4	4	0
2	2	6	**390**	13	3	3	4	14	26	3
3	0	8	8	**399**	6	54	2	6	53	6
4	3	0	5	1	**375**	2	1	15	3	68
5	4	2	0	38	0	**361**	3	1	32	3
6	6	0	3	0	7	6	**491**	0	4	0
7	1	1	3	9	13	2	0	**431**	3	37
8	3	14	8	41	1	7	0	0	**363**	4
9	0	1	0	9	86	5	0	35	10	**368**

Figure 25: The [i,j] is the number of times that the true value j to be predicted as i. The model is trained with M = 10000 and K = 1000.

Observations:

- 0 may be recognized as 5 or 6.
- 1 is not likely to be recognized as other numbers.
- 2 is more likely to be recognized as 8.
- 3 is more likely to be recognized as 5 and 8.
- 4 is more likely to be recognized as 9.
- 5 is more likely to be recognized as 3.
- 6 is less likely to be recognized as other numbers.
- 7 is more likely to be recognized as 4.
- 8 is more likely to be recognized as 3.
- 9 is more likely to be recognized as 4.

Based on those observations, it is justified to conclude that some of the observations do match our daily experiences like observations 4 and 10. However, there are still some exceptions like observation 8 as it is more probable for people to regard 7 as 1 instead of 4. This means that the model still has some underlying problems which needed to be implemented.

In addition, the MNIST data set contains many digits that are not written formally and these data are more likely to influence the accuracy of the model as this will lead to a wrong classification on the data.

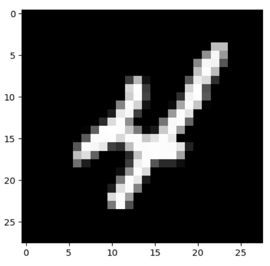


Figure 26: Image with digit 4.

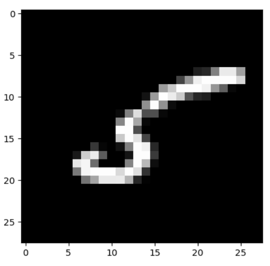
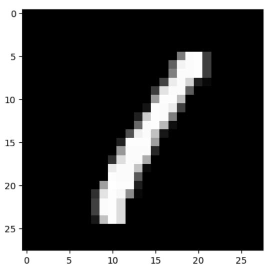
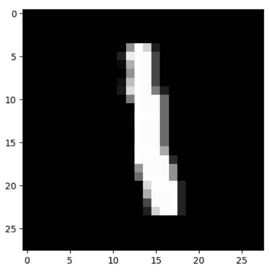


Figure 27: Image with digit 5.

The 2 figures above show digit 4 and 5 respectively. These are examples of digits that are not written formally and have a relative big difference between the normal ones. The means of these digits find by K-means clustering can be quite different, which results a lower accuracy in K-means clustering. Logistic regression and MLP will not be influenced to much.



(a) 11-1



(b) 11-2

Figure 28: Images with digit 1.

The figures above both shown digit 1 but the means of the two figures are different. The left one is more tilted and the direction of each digit is different. This can also lead to problems when classifying using K-means clustering. However, logistic regression and MLP will not be affected a lot as before.

All in all, the MNIST data set contains some abnormal data which can have an impact on K-means clustering as this is an unsupervised learning algorithm. Supervised learning algorithm using labels will be less affected by this.