A Report for Foundations in Machine Learning Assignment Work

Ziyi Guo

zg2u21@soton.ac.uk

1 Introduction

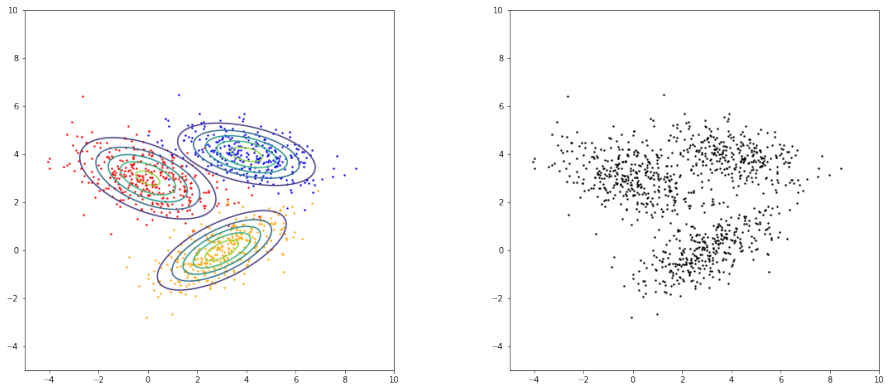
In the past experiments of Foundations in Machine Learning, works have been carried out on the basic matrix calculation, various Gaussian densities generation, linear and Bayesian classifiers implementation and the class boundaries description, linear regression implementation and its regularization, as well as data projection and transformation like Fisher LDA and RBF. All the experiments achieved considerable results.
In this report, researches on the K-means clustering algorithm and multi-layer perceptron classifier are illustrated to study the principle, performance and application for both algorithms. The details of the experiment data, the algorithm operation and the experiment results are in the following parts as below.

2 K-Means Clustering

In the first part, it is aimed to implement K-means clustering algorithm, study the algorithm principles, and apply the algorithm to K-class classification problems on public data sets.

2.1 Data Samples

Above all, data are sampled from a mixture Gaussian density consisting of 3 multivariate Gaussian densities with distinct means of $\begin{pmatrix}0\\3\end{pmatrix}$, $\begin{pmatrix}3\\0\end{pmatrix}$ and $\begin{pmatrix}4\\4\end{pmatrix}$, and randomly generated covariance matrices. The scatters of the distributions of data samples in each Gaussian density and all the data samples and the contours of the probability densities of each density are generated as below:



(a) 3 Multivariate Gaussian Density (b) Gaussian Density Mixture

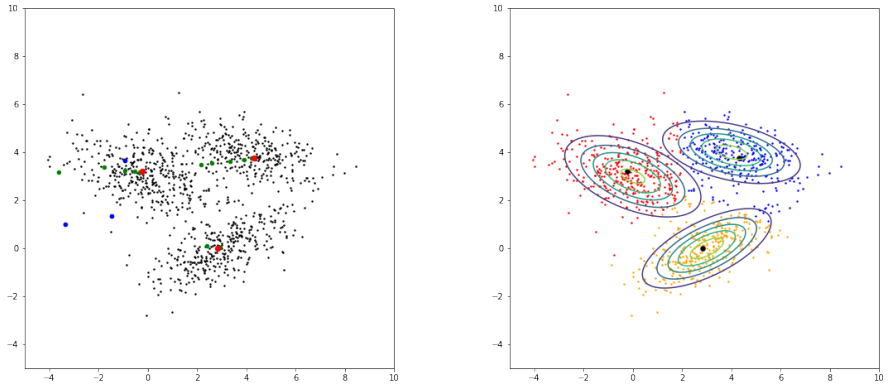
Figure 1: Sample Distributions and Probability Density

It can be observed that the samples generated from the 3 Gaussian densities are well separated and likely to be greatly divided by three clusters. From the perspective of the probability density contours, the data points of each Gaussian density are largely located in the corresponding regions and the 3 distinct regions have little common part, indicating the possible effectiveness of K-means clustering algorithm($K = 3$).

2.2 K-Means Clustering Algorithm

Acquiring the data samples above, the second step of the experiment is to implement the K-means clustering algorithm to classify the data points into 3 independent groups.
The implementation of K-means clustering is the process of iterations and contains several essential settings and functions. Firstly, define the distance between the data points by the Euclidean distance of vectors. Secondly, define the initialization function of cluster centers as a set of 3 random data points between the maximum and minimum data, ensuring the cluster

centers to be in the data space. Then, build the main body of the algorithm to partition the data based on the similarity index of objects and cluster centers and set the convergence condition as judging the change of cluster centers and the objective function value in the iterations. Finally, update the cluster centers with the mean values of the data points in each class. With the implementation of a converged K-means cluster, cluster centers that minimize the square sum of each clusters will be obtained. Here, the K-means cluster algorithm as well as the cluster tool in *sklearn* are both implemented. The results of the algorithms and the visualizations of the algorithm operating process are as below:



(a) Convergence of K-Means Clustering (b) K-Means Clustering in Sklearn

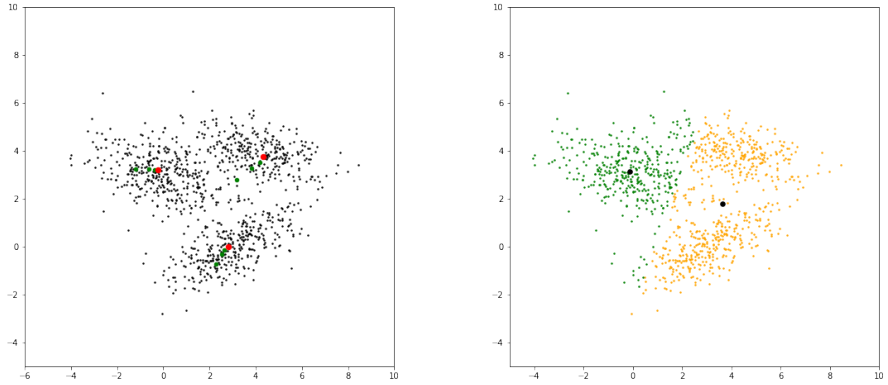
Figure 2: Scatters of Data Distribution and Cluster Center Convergence

According to the convergence figure and the results of the algorithm, it can be observed that the K-means cluster algorithm begins with an initialized cluster means of $\begin{pmatrix}-0.94\\3.66\end{pmatrix}$, $\begin{pmatrix}-3.37\\1.01\end{pmatrix}$ and $\begin{pmatrix}-1.47\\1.34\end{pmatrix}$, converges after 8 iterations, and outputs the final cluster means of $\begin{pmatrix}4.31\\3.74\end{pmatrix}$, $\begin{pmatrix}-0.22\\3.20\end{pmatrix}$ and $\begin{pmatrix}2.83\\-0.02\end{pmatrix}$. The cluster means of initialization, iterations and convergence are marked differently in blue, green and red. Comparatively, the K-means clustering algorithm in *sklearn* outputs the same results of the cluster means as above. The square sum of distances from data points to the means is 2454.43. The data visualization figure shows the cluster means and the samples in each cluster, as well as the contours on the probability densities of each class.

2.3 Experiment Analysis

Numerically analysing the result of the experiment, both algorithms converge at the same group of means rapidly within 10 iterations, showing the reliability of the result to some extent. From the perspective of the visualization figures, not only the data points in each cluster are largely falling in the contours of probability densities of the corresponding Gaussian distributions, but the cluster means of the algorithm outputs are quite close to the center of the probability densities, further indicating the validity of the cluster algorithms.

Researches proved that the K-means algorithm is sensitive to the initialization of the cluster means and the choice of K value[1]. Unreasonable choices make the experiment results unsatisfactory. In the algorithm operations in this report, data points that are far from the data space are chosen as bad samples to test the sensibility of the algorithm to initial cluster means. As a result, the algorithm still converges in the ideal cluster means, indicating the insensitivity of the algorithm to the initialization of cluster means in simple classification problems. Moreover, when setting the K value to 2, the result of the experiment is apparently failed for the algorithm can not figure out the proper cluster means and the square sum of distances from data points to the means becomes 5002.17 in this case, showing the high inaccuracy of the cluster algorithm. The experiment results are in Figure 4 as below.



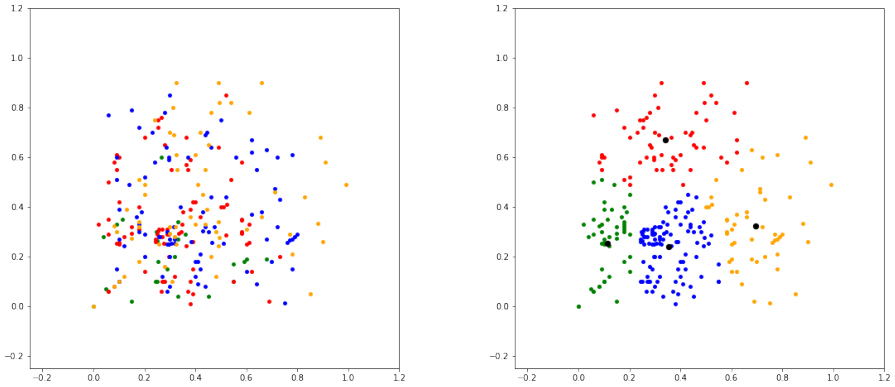
(a) Convergence of Altering Initialization (b) K-Means Clustering Algorithm with K=2

Figure 3: Samples of Altering Algorithm Parameters

It can be inferred that successful K-means clustering need reasonable choice of K value above all. After that, the initialization of cluster means need to be taken into consideration and can be important in complex cases.

2.4 K-Means Clustering Application

Finally, practical classification problem on the data set are illustrated to put the K-means clustering algorithm into application. The data set is concerning $User\ Knowledge\ Modelling$ and contains five input values from different aspects of study time and exam performances and a target value of $very\ low$, low, $Middle$, and $High$, representing the knowledge level of the users. In the experiment, features named STG, SCG representing the degree of study time for goal object and the degree of repetition number of users for goal object are chosen as the value of the feature vector. Given the categories of the data target, the K value of the clustering algorithm is set to be 4 aiming to classify the data points into 4 corresponding clusters. The original data distribution with targets marked by four colors and the results of the clustering algorithm are in the figures as below:



(a) Data Distribution (b) Clustering Convergence

Figure 4: Applications of K-Means Clustering on User Knowledge Dataset

It can be observed that the data are divided into 4 groups through clustering and each cluster includes the nearest data points. However, the four cluster have very weak correlation with the data targets, and can not indicate the true origins of the data, showing the inapplicability of applying K-means clustering with only two features on the data set. To improve this, it is necessary to choose more features as the input value of the algorithm or apply other models on the data set to make reliable prediction. Kahraman proposed the 'Intuitive Knowledge Classifier' method to model the domain dependent data of users in which the user modeling approach and the weight-tuning method are combined with instance-based classification algorithm to improve classification performances of well-known the Bayes and the k-nearest neighbor-based methods[2].

3 Multi-Layer Perceptron

In the second part, it is aimed to implement the multi-layer perceptron classifier and figure out its performance in approximating the posterior probabilities of the Baysian classifier. The experiment consists of pilot experiment and formal experiment.

3.1 Data Samples

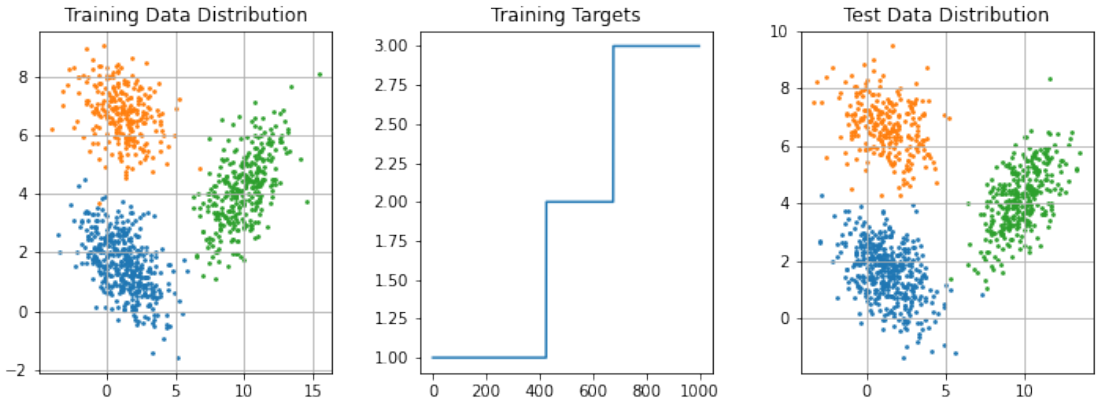
Above all, data samples are generated from mixture Gaussian densities combined with 3 distinct Gaussian densities corresponding to 3 classes. Illustrating two circumstances in one of which the classes are far apart and relatively easy to classify while overlapping and difficult to classify in another and considering the data for both model-training and testing, the data are generated and the distributions and training targets are plotted in Figure 5 as below.

In the pilot experiment for both problems, the data for training and testing are generated from two similar mixture Gaussian densities by controlling some data points the same in each class, which is mainly aimed to make the training data and test data either similarly far apart or similarly overlap and figure out an estimated result for the following experiment. In the formal experiment, *only* the data from the training data in the pilot experiment above are used as experiment data. The data in Figure 3(a) are for the relatively easily learning problem while the data in 3(b) are overlapped corresponding the difficult learning problem. Specifically, data targets for MLP classifier are encoded with One-Hot coding to transfer category information and reduce function error in neural network models.

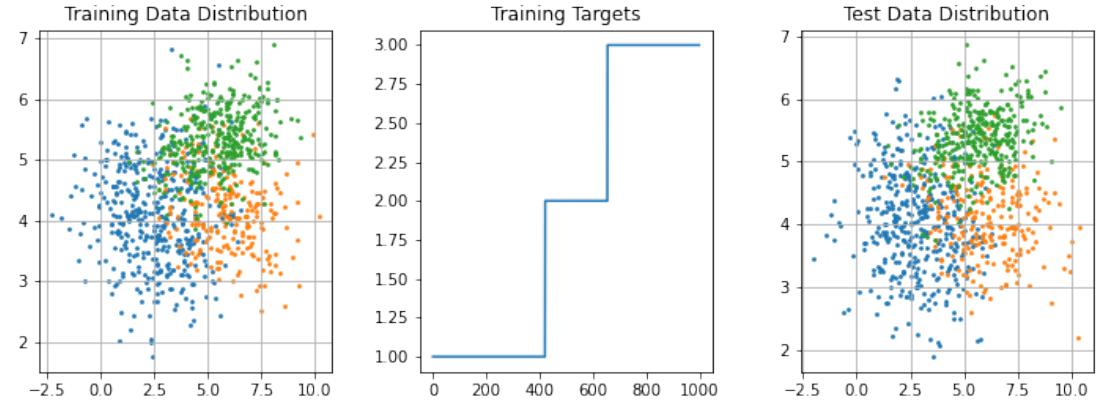
3.2 Multi-Layer Perceptron Algorithm

For both experiments, the Bayesian and MLP classifier are imported from *naive_bayes* and *neural_network* in *sklearn*. The Baysian classifier uses default settings while the MLP classifier sets the maximum iteration to 10000.

In the pilot experiment of the first problem, the MLP classifier outputs a confusion matrix of $\begin{bmatrix} 425 & 0 & 0 \\ 0 & 251 & 0 \\ 1 & 0 & 321 \end{bmatrix}$ for training data, indicating there to be only one misclassified sample of the total 998 data points, and acquires an testing accuracy of 0.997995992 after 340 iterations, approximately equal to 1; meanwhile, the Bayesian classifier outputs the prior probabilities



(a) An Easy Classification Problem



(b) A Difficult Classification Problem

Figure 5: Data Samples for Multi-Layer Percceptron Algorithm

corresponding to each class as $[0.426, 0.252, 0.323]$ and acquires an accuracy of 0.994 for test data. On the second problem, the MLP classifier outputs a confusion matrix of $\begin{bmatrix} 378 & 9 & 34 \\ 91 & 110 & 33 \\ 53 & 16 & 275 \end{bmatrix}$ for training data, showing 236 samples of misclassification, and acquires an testing accuracy of 0.76 after 772 iterations while the Bayesian classifier outputs the prior probabilities as $[0.421, 0.234, 0.344]$ and acquires an accuracy of 0.797 for test data. From the results of the pilot experiment, it can be initially inferred that the MLP classifier performances better in simple classification problem while the Bayesian classifier performances better in classifying overlapped data.

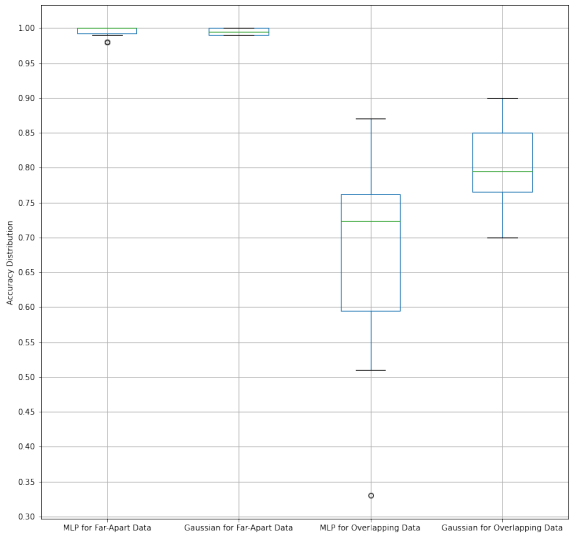


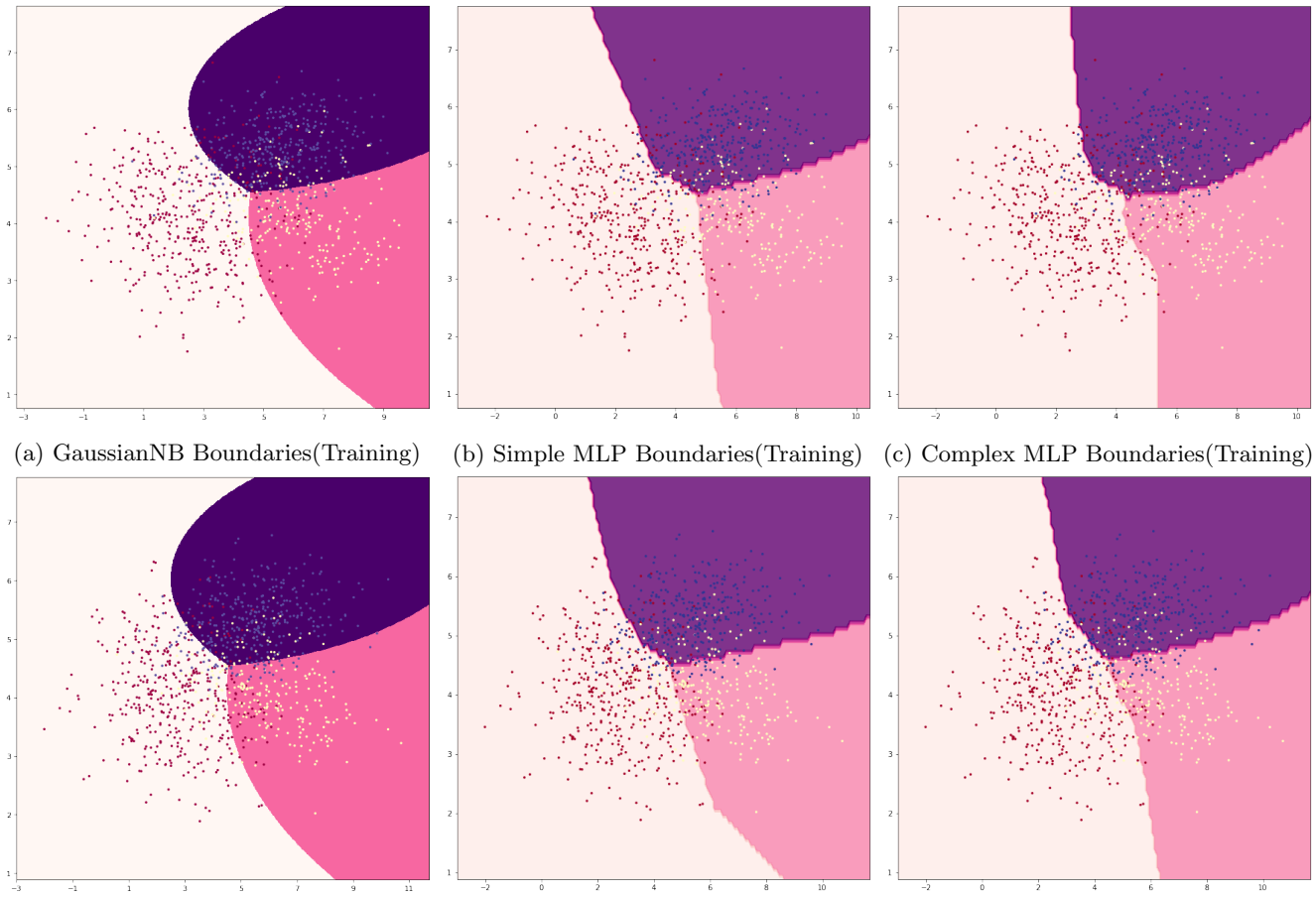
Figure 6: Accuracy Distribution.png

In the formal experiment, the experiment data are entirely from the training data in the two problems above and split into training and test sets with a proportion of 0.1 to implement ten-fold cross validation. Based on the *cross_val_score* function

from *sklearn*, the ten-fold cross validation for both the Bayesian classifier and the MLP classifier on the far-apart data and the overlapped data are implemented. The distributions of validation scores in four circumstances are plotted as below:
From the observation of the boxplots and the algorithm outputs, the results on the first problem for the MLP classifier are a series of accuracy values of $[1, 1, 0.98, 1, 1, 0.99, 1, 0.98, 1, 1]$ with a mean value of 0.995 and for the Bayesian classifier are $[1, 0.99, 0.99, 1, 0.99, 0.99, 0.99, 1, 1, 1]$ with the same mean value of 0.995; the results on the second problem are $[0.79, 0.73, 0.77, 0.87, 0.51, 0.33, 0.56, 0.74, 0.7, 0.72]$ with a mean value of 0.672 for the MLP classifier and $[0.9, 0.7, 0.76, 0.85, 0.76, 0.78, 0.81, 0.87, 0.78, 0.85]$ with a mean value of 0.806. It is indicated that the MLP classifier and the Bayesian classifier performance similarly well on the simple classification problem in which data are far apart and easy to classify, while on the complex classification problem in which data are overlapped and difficult to classify, the Bayesian classifier shows not only a higher mean value of classification accuracy but a higher and stabler overall distribution as well.

3.3 Experiment Analysis

The experiment above shows the difference and similarity between the Gaussian classifier and the multi-layer perceptron classifier on the classification problems. For further study on the correlation between them, figures describing the class boundaries of the two classifiers are generated. The visualization of results Figure 7 shows the class boundaries for the test data in one case, corresponding a Gaussian classifier, a simple MLP classifier with hidden layer size of $[1000, 1000]$ and a complex MLP classifier with hidden layer size of $[10, 10]$.



(a) GaussianNB Boundaries(Training) (b) Simple MLP Boundaries(Training) (c) Complex MLP Boundaries(Training)

(d) GaussianNB Boundaries(Test) (e) Simple MLP Boundaries(Test) (f) Complex MLP Boundaries(Test)

Figure 7: Class Boundaries for GaussianNB and MLP Classifiers

From the observation of the figures, it can be indicated that both the two MLP classifiers and the the GaussianNB classifier outputs non-linear boundaries which appear not to be flat. The GaussianNB classifier is well classifying the data while the MLP classifiers do not performance as well as it and has higher rate of misclassification. Specifically, the comlex MLP with hidden layer size of $[1000, 1000]$ has more similar class boundaries on both training and test data to the GaussianNB classifier. It is known that a Bayesian classifier is choosing the class with the highest posterior probability, optimizing the probability of misclassification. And thus the more accurate posterior probability is , the better performance of the classifier will have. Though the training process of the MLP is actually altering classification hyperplane, it can be regarded as implied learning of the posterior probability. The GaussianNB classifier assumes the posterior probability to be of Gaussian distributions and is likely to have good performance in the cases like above, but may become unreliable when the actual distribution is not Gaussian distributed. The MLP classifier need the training data to be linearly separable but the GaussianNB classifier could solve non-linear problems. Generally, the MLP classifier is minimizing the sum function of all the misclassified data points with a hyperlane geometrically classifying the space and has similar effects to the GaussianNB classifier optimizing posterior probability. The more complex the hidden layers of the MLP classifier is, the more likely the class boundaries are to approximate posterior probability.

3.4 Parameter Initialization of MLP Classifier

Parameters are controlling the operation of the algorithm in different ways and are thus essential to an MLP classifier setting. In the last section of the experiments, works are carried out to test the influence of the parameters *learning_rate_init* and the *solver* to the MLP classifier.

Firstly, the value of the *learning_rate_init* presents the step length of updating weights. It is set to be six values from 0.0001 to 10. Working on the same data set for ten times, the corresponding average values of the training accuracy, test accuracy and the iterations are in the Table 1 as below.

Learning Rate	Training Accuracy	Test Accuracy	Iterations
0.0001	0.71	0.69	1529
0.001	0.76	0.75	785
0.01	0.77	0.76	147
0.1	0.75	0.72	22
1	0.56	0.53	52
10	0.005	0.004	39

Table 1: MLP Performances at Different Learning Rates

It can be observed that the overlong updating step will make the algorithm accuracy much lower, but the learning rate that is too small will make the convergence difficult and cause waste. And, the best learning rate of the case is approximately falling in the intervals of $[0.001, 0.1]$. Further, fine-grained experiment is carried out to figure out the proper learning rate for the problem. The results of the similar experiment are as Table 2.

Learning Rate	Training Accuracy	Test Accuracy	Iterations
0.001	0.76	0.75	785
0.005	0.77	0.76	236
0.01	0.77	0.76	147
0.05	0.74	0.72	46
0.1	0.75	0.72	22

Table 2: MLP Performances at Detailed Learning Rates

Considering the accuracy and the convergence rate, it can be inferred that the best learning rate of the case is at around $[0.005, 0.01]$, also indicating that the algorithm is not easy to converge or output considerable results when the learning rate is either too high or too low, and there can be a most suitable learning rate for each case.

As for parameter *solver*, it defines the weights optimizer for the classifier and have mainly 3 values of $lbfgs$, sgd and $adam$, corresponding to the optimization methods of the Quasi-Newton, the Stochastic Gradient Descent and the Adam. Similarly setting the *solver* to different values and working on the same data sets above for ten times, the results of average values are as below:

Solver	Training Accuracy	Test Accuracy	Iterations
$lbfgs$	0.78	0.76	1120
sgd	0.71	0.69	1130
$adam$	0.74	0.74	445

Table 3: MLP Performances with Various Opitimizers

It can be observed that the $lbfgs$ optimizer shows the best performance of training and test accuracy in this case while the $adam$ optimizer shows the most rapid convergence. Specifically, the performances of the $adam$ optimizer varies greatly in each case and is not stable in accuracy and convergence on this data set. Generally, the $lbfgs$ optimizer is the suitable choice for the given data set of small data scale and the $adam$ optimizer is likely to performance better on large data sets[3].

References

[1] Zhu, M., Wang, W. and Huang, J. (2014), "Improved initial cluster center selection in K-means clustering", Engineering Computations, Vol. 31 No. 8, pp. 1661-1667. https://doi.org/10.1108/EC-11-2012-0288

[2] H. Tolga Kahraman, Seref Sagiroglu, Ilhami Colak, The development of intuitive knowledge classifier and the modeling of domain dependent data, Knowledge-Based Systems, Volume 37, 2013, Pages 283-295, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2012.08.009.

[3] E.A. Zanaty, Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification, Egyptian Informatics Journal, Volume 13, Issue 3, 2012, Pages 177-183, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2012.08.002.