Project Report*

Amrutha Mandalreddy, Herat Patel, Raakesh Murugaian and Shweta Wahane

1 INTRODUCTION

The project involves the implementation of different types of models to complete the Kaggle Digit Recognizer competition. The aim of the Digit Recognizer competition is to identify hand-drawn digits using machine learning algorithms. One of the machine learning models is a classical model and the other model used is a multi-layered perceptron. The classical model that has been used in this project is K Nearest Neighbors. The dataset provided for the Kaggle competition consists of training data and test data. The models are trained on the training data and the predictions are done on the test data. The KNN classifier resulted in an accuracy of 96.2%, and the Multi Layer Perceptron resulted with an accuracy of 96.5%.

2 METHODS

2.1 KNN

K nearest neighbours is a non-parametric learning algorithm. This means that it has no assumption for the underlying data distribution, as the structure of the model is determined by the dataset. It is also considered a lazy algorithm as it does not need any training data points for model generation. We implement a supervised version of the KNN algorithm for the machine learning model.
Function Parameters:

(1) n_neighbours: It is a parameter that is used for specifying the number of the neighbours that are to be used in the model.
We used n_neighbours = 10.
(2) leaf_size: It is the parameter for determining the leaf size of the tree. It alters the speed of the model construction and the querying.
We used leaf_size = 30
(3) p: Used for mentioning how much the power is for the Minkowski metric.
We used p = 2
(4) metric: Used for specifying the distance metric used for the tree.The default metric is Minkowski.
We used metric = "minkowski"

2.2 Multi-Layered Perceptron

MLP is a feed-forward artificial neural network(2). It consists of a minimum of three layers, with the first being the input layer, then either one or more hidden layers, and finally the output layer. It is used for distinguishing data that is considered to be not linearly separable. It employs a technique called back-propagation for learning. While they were popular in the 80s and used extensively for various applications such as speech recognition, image recognition, etc, they faced more competition later due to simpler models being developed. They again gained popularity after the surge in interest in deep learning.

Function Parameters:

(1) Hidden Layer Sizes: It is a parameter for specifying the number of hidden layers to be used in the model.
We used hidden_layer_sizes = 100
(2) Activation Function: It is used for mapping the input nodes to the output nodes. They help in performing gradient-based optimizations.
We used Activation function = relu
(3) Learning Rate: It is a parameter that determines the step-size in moving towards the least value of the loss function.
We used learning rate = 0.0001
(4) Optimizer: For mentioning the optimizer being used such as SGD or ADAM.
We used Optimizer = adam

3 RESULTS

3.1 KNN

The KNN Classifier from sci-kit learn has been used[1]. The parameters that have been used are n_neighbours = 10 , weights = 'uniform', leaf_size =30 with metric being 'minkowski'. The accuracy score that was obtained was 0.9621. The learning curve for K Nearest Neighbours algorithm was obtained by reducing the dataset size in order to obtain the visualization faster Figure: 1. The method predict_proba was used from sci-kit learn in order to obtain the probability visualization for each of the classes.For each class, images where the correct class probability is very low are given in Figure: 2. These images are gathered where the predicted classes had less probability. For each class, images near the decision boundary, where the probability for the correct class is slightly lower than the other classes is plotted in Figure: 3. The initial model for K Nearest Neighbours involved creating the model with a leaf size of 20 and n_neighbours of 20. The accuracy score obtained was 0.924 which was later increased to the above score using the parameters mentioned. The confusion matrix for KNN is given at Figure: 4

3.2 Multi Layer Perceptron(MLP)

For the Multi Layer Perceptron, the MLPClassifier from sci-kit learn has been used[2]. The parameters used are hidden_layer_sizes=100, activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.0009, power_t=0.5, max_iter=250, shuffle=True, random_state=None, tol=0.0001,verbose = False, warm_start=False, momentum=0.9, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, n_iter_no_change = 10, max_fun=15000. The learning curve for MLP algorithm was is plotted at 5. The method predict_proba was used from sci-kit learn in order to obtain the probability visualization for each of the classes.For each class, images where the correct class probability is very low are given in Figure: 6. These images are gathered where the predicted classes had less probability. For each class, images near the decision boundary, where the probability for the correct

*

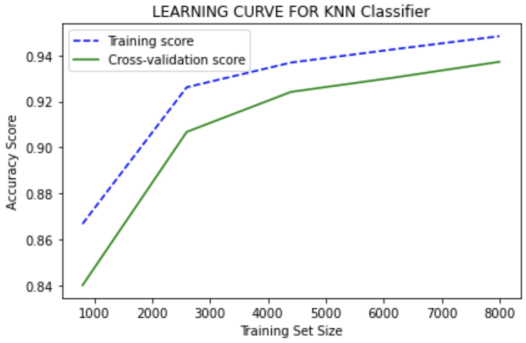

Figure 1: "K Nearest Neighbors:Learning Curve"

Figure 2: K Nearest Neighbors: Images where the correct class probability is very low

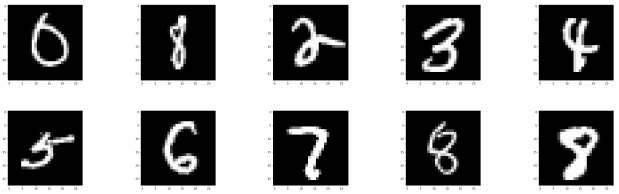

Figure 3: K Nearest Neighbors: Images near the decision boundary, where the probability for the correct class is slightly higher/lower than the other classes

class is slightly lower than the other classes is plotted in Figure: 7.The confusion matrix for KNN is given at Figure: 8

4 DISCUSSION

The results that have been obtained have differences that are very minimal, between the classical K Nearest Neighbour model and the MLPClassifier model. The accuracy score for the K Nearest Neighbour model is 0.962 while the MLP classifier has an accuracy score of 0.96582. The earlier model for the K Nearest Neighbour model had an accuracy score of 0.924 with some of its parameters affecting the score negatively. These were the n_neighbours and leaf_size parameters. After adjusting them, by decreasing the n_neighbours and increasing the leaf_size we get an increased accuracy score of 0.9634.

The learning curves of both the models shows that increasing the data size after a point does not increase the accuracy of the predictions for the test data very much, since the lines for the

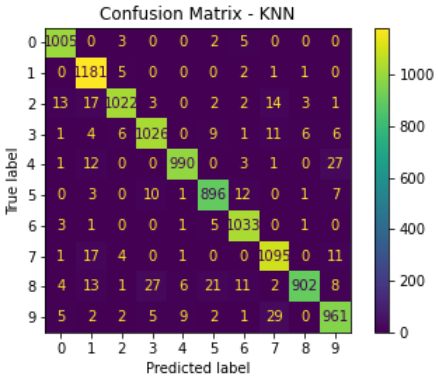

Figure 4: "K Nearest Neighbors:Confusion Matrix"

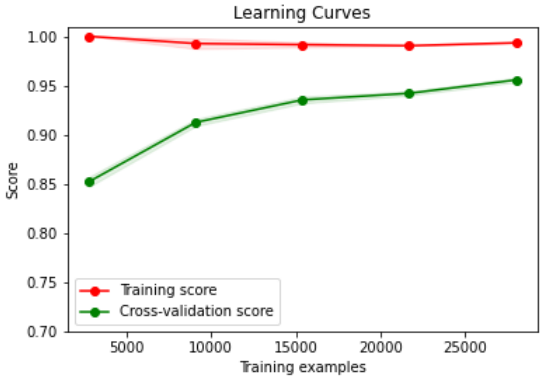

Figure 5: "Multi Layer Perceptron:Learning Curve"

Figure 6: Multi Layer Perceptron: Images where the correct class probability is very low

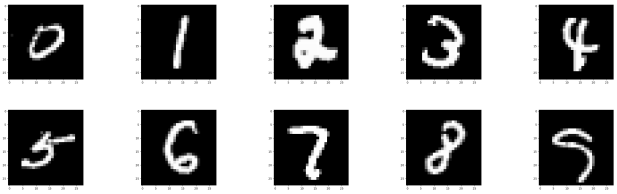

Figure 7: Multi Layer Perceptron: Images near the decision boundary, where the probability for the correct class is slightly higher/lower than the other classes

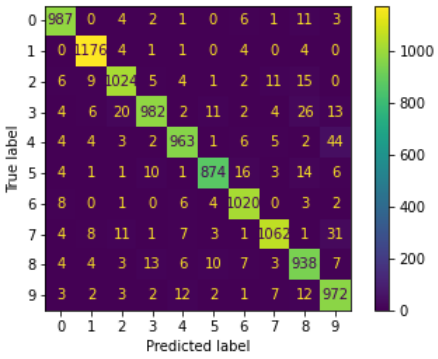

Figure 8: "Multi Layer Perceptron: Confusion Matrix"

training score and test score plotted against the size of data do not meet.

As we can see from the visualization, even when MLP has lower performance, it is not giving much bad output. But that's not the case with knn. When nearest neighbors are increased in KNN and number of hidden layers are increased in mlp, the performance increases. Mlp can give good performance which depends on the tuning of parameters and hyperparamaters. It is better when there is complex data. In our case data is linearly separable, so knn works better.

REFERENCES

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Mlpclassifier. June 2016.