Data Classification Using kNN Classifier and Deep Learning Based Method

Md Waqeeb Tahmeed Sayeed Chowdhury
Department of Electrical and Computer Engineering, Temple University
tun62948@temple.edu

Introduction: In this project an effort has been made to classify the datasets provided using a non-neural network approach, which is the k-Nearest Neighbor for my case and a deep learning based approach based on multi-layer perceptron. Three sets of data were provided which where 2D training data, development data and the testing data sets. The classifiers were being initialized and trained with the training data, and then they were updated and parameter weights were regularized by performing on the development data. Finally, a blind data set was introduced to the classifier. The performance of both the classifiers on this blind data set was evaluated and then compared. The detailed analysis of the two algorithm has been provided below in details.

Algorithm No. 1 (kNN classifier): The kNN is a supervised machine learning algorithm which depends on the class labels of the training data, and adjusts its weight and parameters accordingly. It is a non-parametric method used for classification and regression. The reason why I chose kNN because it typically has a single tuning parameter and therefore is much more convenient to regularize and implement compared to other non-NN methods.

In my classifier model, I trained the model using the training data sets for different values of k , from 1 to 100. It was seen that, for k = 1 the error rate on the training data set was 0%, and as k increased to 100, the error rate increased up to 25.33% as shown in Fig 1. On the other hand, the development set has a maximum error rate of 32.09% when k = 1 , and a minimum error rate of 25.71% was recorded on the development data when k = 79. The classifier was then tuned for k = 79, and then the blind evaluation data set was introduced to the classifier. The distance metric used for classification was the euclidean distance and the criterion for choosing the class for a data point was based on majority votes. This algorithm was implemented in the MATLAB environment.

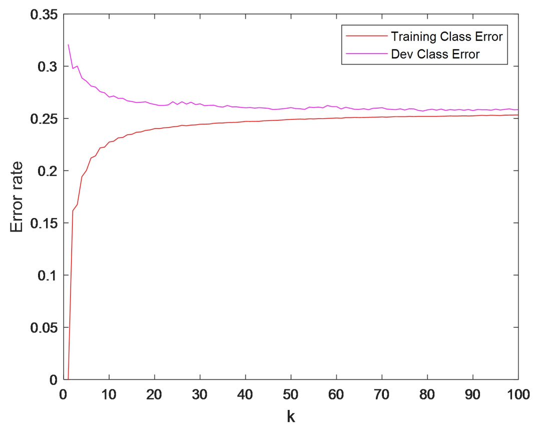

Fig. 1 Error rate of kNN classifier for train and dev data

Algorithm No. 2 (MLP): A multi-layer perceptron is a class of neural network formed of numerous neurons stacked in layers and each neuron is multiplied by a weight. Such a neural network is called a multi-layer perceptron because the neurons are stacked into layers and they adjust the their weights accordingly when the classifier is fed with a training data set. A neural network has three fundamental layers called the input layer, the hidden layer and the output layer. The hidden layer can consist of a single layer or multiple layers. In a MLP, the tune-able parameters are the number of perceptron in each layer, the number of hidden layers , the epoch number. The performance of the model depends on the optimal combination set of the parameter values, and they were found by trial and error basis. Initial weight and bias are set for the hidden layers and they are adjusted by back-propagating the calculated error rates and adjusting accordingly. From my experimentation experience it was observed that as the number of layers increased the model tend to over-fit on the training data. And finally the number of neurons used per perceptron and the activation function used also plays a significant role in determining how well the model would perform. As stated before, a perfect combination is hard to find since there are a lot of parameter on which the performance depends. So the best combination is chosen by trial and error basis.

The parameter that were used in my model were two hidden layers with 70 perceptron in each layer. The Adam optimizer was used with ReLU as the activation function. The categorical cross entropy function was used as the loss function The learning rate, batch size and epoch were chosen as 0.001, 64 and 200 respectively.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
kNN ; k = 79	25.71 %	25.19 %	64.48 %
MLP	27.12 %	27.07 %	62.06 %

Table 1. Error rate for different datasets and algorithms

Results: The table below shows the performance of the two classification model on the data set that has been provided. It has been shown that as expected the MLP model has a much better performance than the kNN method. One thing to notice is that kNN has lower error rates than MLP on the training and development set because it tends to overfit on these two sets of data, and therefore the performance on the evaluation set is poor compared to MLP.

Using the concept of statistical significance, using the provided result on the data set as the baseline, the evaluation data set has a confidence level of 70.54 % when evaluated with kNN , and has a confidence level of 99%.

Conclusions: From the above results it seems apparent that the MLP performs much better than kNN in terms of error rate and the statistical significance of the value, further more kNN seems to overfit the model on the seen data sets thus making it much more difficult to perform better on unseen data, for example the evaluation data set. But it is to be noticed that regularizing a MLP is much more difficult than a kNN since the model depends on a lot of parameters. Also the MLP also overfit on the training and development data set as the number of hidden layers are being increased, and shows poor performance when number of layers is less.

Therefore, it is difficult to choose one as the absolute better option another. While kNN offers a model with much less computational complexities and slightly poor performance, the MLP shows better performance at the cost of a much computationally expensive process.