Machine Learning Classifiers

Thao Cap
Department of Electrical and Computer Engineering, Temple University
Thao.cap@temple.edu

Introduction: In this research, there is a dataset of two-dimensional data that will be used to analyzed and classified based on two different types of machine learning algorithms. The first algorithm uses a non-neural approach, and the second algorithm uses a neural network-based approach. The dataset consists of training data, development data, and evaluation data.

We choose K-nearest neighbor (kNN) algorithm for the non-neural approach and Multilayer perceptron (MLP) for the neural network approach because we have relatively small amount of data (around 100000 samples) and our training data has provided labels.

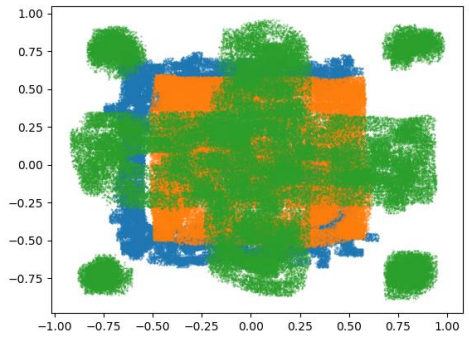

Figure 1: Scatter plot of a training set

Algorithm No. 1 Description: kNN – This is a simple and supervised machine learning algorithm that classifies the new data based on the surrounding training data. The kNN algorithm finds K samples, where K is the number of samples, that have the nearest distance compared to the new data. It then uses the majority vote to assign the class that has most votes to the new data.

Knn uses a few parameters which can be used for tuning the algorithm:

- **n_neighbors** determines number of neighbors that we want to use.
- **Weights** determines the weight function used in prediction.
- **Algorithm** used to compute the nearest neighbors.
- **Leaf_seize** determines number of points at which to switch to brute-force.
- **P** determines the power parameter for the Minkowski metric. P =1 is using Manhattan_distance, and p=2 is using the standard Euclidean metric.

I choose an initial k value of 2 then run the classifier on the training and development data to test if feature scaling is necessary for our data. After testing the classifier with Normalization scaler and Standardization scaler, and without scaler, I have the result on Table 1

Feature Scaler	Error rate
Normalization	29.85%
Standardization	29.78%
None	29.78%

Table 1. Error rate when apply feature scaling on knn, k=2

Based on the table above, we decide to choose Standardization for feature scaling our data.

To find the optimal parameters to tune the algorithm, Sklearn has provided sklearn.model_selection.GridSearchCV that we can do a bruce-force search to find the optimal parameters among all parameters that we want to test. GridSearchCV basically applies cross-validation to split and test the data with provided parameters.

After running the Knn model with GridSearchCV, we found the optimal parameters of k value is 108, power parameter for the Minkowski metric is Euclidean.

Applying the optimal parameters gives us the result in table 2 below.

Algorithm No. 2 Description: MLP - A multilayer perceptron is a class of feedforward artificial neural network and often used to supervised learning problem. The MLP consists of one input layer, one output layer, and an arbitrary number of hidden layers. We use mlpclassifer module provided by sklearn to implement. With similar tools used in Knn, GridSearchCV is used to find the optimal parameters for the model.

After running GridSearchCV, we have selected the activation function to be the rectified linear unit function, which returns $f(x) = \max(0, x)$, and the hidden layer sizes to be (100, 100, 50).

Results:

Below are error rates from each model for different data. We can see that the performance of the two methods are very similar. Generally, KNN performs slightly better on our training data and development data, whereas MLP gives a slightly smaller error rate on the eval data. However, the difference between the error rates is not statistically significant. To improve my KNN model, I can use cross-validation on the training set find the optimal k. To improve my MLP model, I can test with more hidden layer sizes and learning rate to find the best optimal parameters.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
Knn	25.62%	25.39%	67.32%
MLP	26.29%	26.35%	67.07%

Table *2. Error rates for knn and mlp*

Conclusions:

Because time is limited but Sklearn GridSearchCV requires a longer computing time, I was not able to find the best parameters for both methods. This project provides me a lot of insight on how to tune a model and how to pick with algorithms used for a given task.