EARIN Lab 4 Variant 4 Report

Dawid Wypych

Aayush Gupta

Implementation of Solution

In this lab, a data set describing the traits of three different iris species was analyzed. The data set was split randomly into a training set and a validation set, and two classification algorithms were applied. Validation sets ranging from 30% to 70% of the data set were tested. The two algorithms tested were the k nearest neighbors algorithm and the decision tree algorithm. The results were then analyzed with the help of a confusion matrix, as well as t-distributed stochastic embedding.

The k nearest neighbors algorithm decides which class a tested data point belongs to by calculating the distances from the tested data point to all of the points in the training set, and looking at the k points that are closest to the tested data point. The class that has the most points within those k neighbors becomes the prediction. In the case of ties, the class that the closest point belongs to becomes the prediction.

The decision tree algorithm splits the training data set into subsets based on some criterion. In this case, the Gini impurity, which measures the probability of incorrect classification, was used to split the data set. The algorithm splits the data set into nodes in a way that minimizes the Gini impurity in the resulting subsets. This splitting is repeated recursively until either the maximum depth is reached, there are a minimum number of samples in a node, or there is a node with a low impurity value. When a stopping criterion is met, the node is classified based on the majority class label of the samples in the node. When a prediction is made on the validation set, the tree is traversed based on the tested data point's values. The prediction becomes the label of the node that the test data point ends up on.

The confusion matrix is a matrix which compares predicted values to the actual values. It shows which predictions are true positives, true negatives, false positives, and false negatives. These values can then be used to determine the accuracy of each algorithm tested.

T-distributed stochastic embedding enables the reduction of a high dimensional data set into just two dimensions. Since the data set has four different traits that are used to describe the different iris species, the data points, if graphed, would require four dimensions. By flattening the graph into just two dimensions, the data set can much more easily be visualized.

The TSNE method was used to plot both the training set and the validation set simultaneously. Points belonging to the training set have solid colors. Whereas points belonging to the validation are colored with outlines. The inside color represents the predicted value. The outside color represents the actual value if the prediction is incorrect. If the prediction is correct, the outside color is black.

Analysis of Results

Both algorithms were able to perform reasonably well (at least 90% accuracy) in nearly all cases. Reducing the parameters to very low values (less than 3) in both algorithms (k in KNN and max depth in decision tree) had a significant impact on their effectiveness. However, beyond these very low values, raising these parameters did seem to have a huge impact on the accuracy. This is likely because, for the most part, the data sets (particularly their borders) were reasonably well defined. Additionally, both algorithms had very similar computation times of about three seconds.

The KNN algorithm (averaging 95% accuracy) consistently performed better than the decision tree algorithm (averaging 92% accuracy). The difference between the two was larger when the training data set was small and smaller when the training data set was large. Additionally, the decision tree algorithm occasionally made predictions that seemingly made no sense, based on the split in the data on the TSNE graph.

For these reasons, it would seem that the KNN algorithm is the best (out of the two investigated) at classification problems. However, it does have a significant limitation that was not seen in this relatively small data set. For every point in the validation set, the algorithm must calculate the distances to every point in the training set. For large data sets numbering in the thousands or even tens of thousands, this can be very computationally intensive. As such, while it may be a good algorithm for this data set, there may be much better options for larger data sets.

Reflections

Both algorithms applied were reasonably accurate in almost all cases. Additionally, the plotted graphs proved invaluable at troubleshooting issues that were found in the algorithms, and, later, analysis of each individual prediction.

One possible improvement would be to test the algorithms on other larger data sets to see how well they perform on huge sets. Additionally, testing other algorithms would have been nice as well.