A K-Nearest Neighbor Approach to Classifying New York City Taxi Ride Payment Types

Harshil Jain, Alexander Kahn, Steven Lymperis, Henry Warzecha

Introduction

It may not be immediately obvious what factors determine whether a taxi rider pays with cash, credit, or other local rewards programs. As a result, a simple non-parametric classification algorithm, such as K-Nearest Neighbors, makes sense as it makes no strong assumptions as to how the structure of how model features may affect outcomes. To demonstrate this, we used a data set comprising of 46.2 million rows of New York taxi data, where each row represents a ride and includes information such as the pickup and drop off locations, fare amount, tip amount, and of course the payment type.

We used Scikit-Learn's KNN classifier implementation, which simply calculates the closest neighbors to a row based on a uniformly weighted distance function that factors in all 13 different features found in each row of data. Although "fitting" the model is computationally trivial, as there is really nothing to fit, using the model to make predictions is much more taxing. For each row we want to predict, the model must loop through all the training observations and find the k most similar observations. As a result, scoring the model on large data sets can become impractical without the use of multiprocessing techniques, especially when the size of the training set is large.

Data

To preprocess the data, we first needed to select the training set. We randomly selected one fifth of the data, representing about 9.25 million rows, as our training data by scrambling the order of our rows and selecting the top 20%. Each feature was then normalized, to avoid assigning too much or too little importance to any feature based on it's scale. Without this, the model might assign too much weight to features measured in larger units as the distance function has no sense of scale. The models were then "fit" on this training set, which is implemented in SKLearn by storing the training samples within a model object. The model objects were then saved to disk. We used four different models: k=5, k=10, k=15, and k=20 in order to judge how predictive power varies with different values of k.

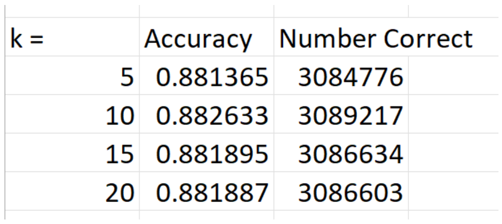
Implementation

In order to implement such a slow predictive algorithm on large amounts of data, we combined SKLearn's functionality with the MapReduce framework using Python's mrjob library. For each line in the CSV file, we mapped a 1 to those models that correctly scored it, and a 0 for those that did not. By summing across these we then were able to efficiently compare how the models compared to one another. Despite this, the runtime was still understandably slow given the vast amount of training data, and the fact that four models needed to be scored for every ride we wanted to predict. Running on a single machine, it took about a minute to score all four models across 1000 rows of data. Since the dimensionality of the data and the size of the training set is fixed, we would expect runtime to increase linearly with n. As a result, we would expect scoring one million rows would take almost 17 hours.

Because of this, it made sense to run our code using Google's cloud computing infrastructure. Thanks to multiprocessing, we were able to score 3.5 million randomly selected rows in an afternoon.

Results

The model predicts very well. Surprisingly, the value of k does not seem to matter very much.



k =	Accuracy	Number Correct
5	0.881365	3084776
10	0.882633	3089217
15	0.881895	3086634
20	0.881887	3086603

We might expect that moving from k=5 to k=20 might result in an improvement in predictive accuracy, but the difference is insignificant. All the models seem to correctly predict around 88% of ride payment types. It seems that KNN is a good way to approach such a classification problem.

Conclusion

By using MapReduce and Google's cloud computing platform, we were able to predict millions of data points using four different models. Ultimately, all the models performed very well. KNN is clearly well suited for such a relatively simple, non-parametric classification problem. Perhaps we would have found more of a difference between models if we had allowed k to vary more significantly. For instance, if we compared k=3 with k=300 we may have seen more of a difference than we did comparing k=5 and k=20. Further research might look to see if such a model predicts certain types of rows better than others. It might also be interesting to compare different weights on various features, to see which features predict better than others. Of course, this would involve scoring the data many more times which is extremely time intensive. All in all, KNN proved to be a successful approach for this problem.