Asif Amirguliyev

CS 486 Project

Clustering method for driver telematics analysis.

1. Problem.

My project was a driver telematics analysis competition on Kaggle [1]. The problem was to separate the trips that belonged to main driver from the noise. The unlabelled data was in the form of GPS coordinates of drivers at each second of their trips. For the purposes of anonymity the trips were trimmed and randomly rotated. Each case contained 200 trips with majority belonging to one driver and the rest driven by other people. The output format was a list of classifications of each trip in each test case as “1” or “0”, where “1” meant that the trip belonged to the driver of interest.

2. Approach.

Given the fact that there was not any training data, the most intuitive approach to the problem was to use unsupervised learning methods, clustering in particular. The idea was to extract meaningful features from the trips, build the core cluster out of 101 most compact data points and use its compactness as a benchmark for separating “true” trips from the noise. The assumption was that, with the majority of trips belonging to one person, 101 out of 200 trips that are closest to each other will belong to the main driver.

3. Preprocessing and initial features.

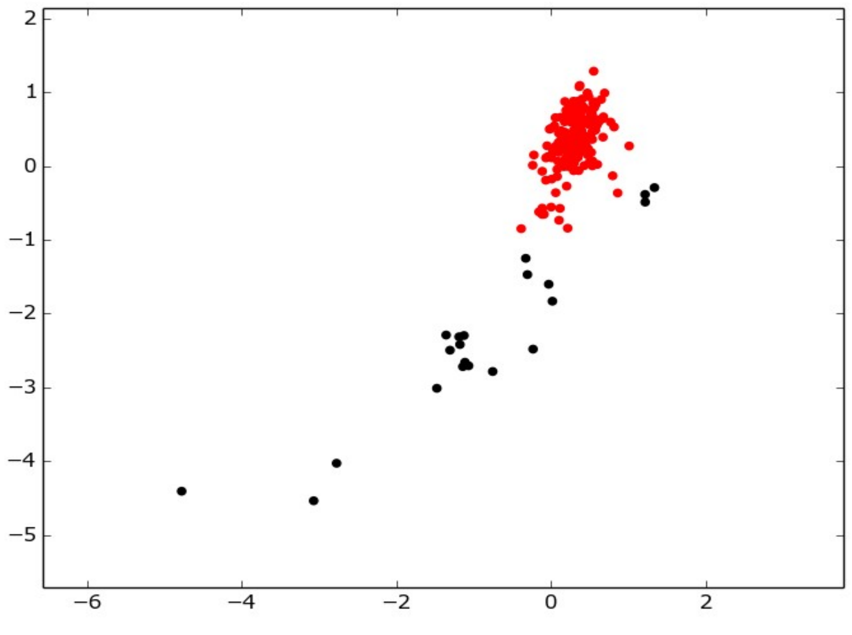
The raw data was in the form of GPS coordinates at each second of the trips. Preprocessing step for the analysis included calculating velocities and accelerations as well as curvature of the road. Ithe curvature was calculated as changes in the direction of the road. Initial features selected for the analysis were means and deviations of accelerations binned according to road curvature. I hoped to detect the driver solely on her behaviour on any kind of road by removing the effect of the road shape. With lack of labelled training data I also needed some measure of feature usefulness. Based on different typical methods of measuring clustering quality [2] I decided to use ratio of the core cluster's compactness to the distance to the farthest point. Adding a useful feature should have made similar points more compact and noise more distant.

4. Core cluster.

I used standard textbook method of agglomerative clustering for building the core cluster [3]. The algorithm was combining the closest points into clusters and the closest clusters into bigger ones until the biggest cluster contained the majority of the points. Leaving 101 closest trips in that cluster I treated it as the core cluster, assumed that it contained mainly true trips and used its compactness for its further expansion.

5. Further classification.

The noise in the data was random, and as expected the visualized data contained only one big cluster with some outliers. That is why methods for dividing data set into several clusters were irrelevant. The problem at this point was to detect the boundary of the main cluster. The closest idea was implemented in DBSCAN algorithm [4]. It builds clusters based on their densities and requires some number of close neighbours of the point for it to belong to the cluster. It also treats points inside the cluster and on the edge differently, allowing the edge points to have fewer neighbours. I implemented a similar method for deciding whether to add a new point to the existing cluster. Using compactness of the core cluster as an epsilon, a candidate point was added to the main cluster if it was within epsilon distance from some core point. Core points were the points that already had at least one other cluster point in their epsilon-neighbourhood.



6. Other considerations.

Apart from cluster quality measure and boundary detection method there were couple of other considerations to be made for this approach. One question was what distance function to use for measuring dissimilarity between points. My choice was Euclidian distance and standardizing the data beforehand so that all features affect the distance equally. Another concern was how to measure distance between clusters. Again my decision was simply choosing the average linkage. One more question was whether to use PCA before clustering for dimensionality and noise reduction. I will discuss the results of removing 20% of variance with PCA in the next section.

7. Results.

The initial result of my approach turned out to be extremely poor with score (0.52) close to random guessing. This hinted to the fact that clustering might not suit the problem at all. Nevertheless, I experimented a bit with the algorithm and got better results after using original features without PCA (0.53), adding more features (0.55) and stopping to build the main cluster after agglomeration step (0.58). Possible explanation may be the fact that the differences between driver behaviours are more subtle than I had initially thought. That's why reducing information with PCA and restricting features based on rough compactness-to-farthest-point ratio made the results worse. It might also be the reason why density-based clustering added a lot of false positives to the result. With these poor scores of clustering approach it did not make any more sense to look for solution in feature selection step. But better features that take road shape and change of acceleration over time into consideration could also be potentially useful.

8. References.

[1] AXA Driver Telematics Analysis, Kaggle, http://www.kaggle.com/c/axa-driver-telematics-analysis.

[2] Y. Liu, X. Li, H. Xiong, X. Gao, J. Wu, “Understanding of Internal Clustering Validation Measures”, IEEE, 2010.

[3] T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning”, 2009.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, AAAI, 1996.