Driver Telematics Analysis

Use telematic data to identify a driver signature

Harel Lustiger

Thursday, April 30, 2015

The Challenge:

Data

- 5.5GB of raw data; 547200 csv files organized in 2736 folders such that each folder contains 200 files.
- Each cvs file has two columns X and Y which are Cartesian coordinate of a driven trip (GPS positions).
- Each row i in the csv file is the position of the vehicle at second $i, i \in \{1, 2, ..., m\}$ where m variate between the different files.

Goal

- For every trips' batch, assign a probability for the 200 trips within the batch, such that the main driver was driven it.
- Measure evaluation criteria: AUC.

Prior knowledge

- **No labels for any of the driven trips' driver**
- The majority of the trips $t \in \{1, 2, ..., 200\}$ in folder $r \in \{1, 2, ..., 2736\}$ are driven by the same driver. The other trips were driven by different arbitrary drivers.
- All trajectories:
  - Starts at $(X, Y) = (r, \varphi) = (0, 0)$. (r, φ) are the polar coordinate system of the trip.
  - Each trip's trajectory had been rotated, that is an arbitrary angle θ was added to φ.
  - Each trip's trajectory ends have been trimmed (for privacy reasons)

Unsupervised/Supervised Learning

- Since the data set is unlabeled, the natural approach is to treat the problem as **graph partitioning problem** $G = (V, E)$:
    - The nodes of the graph V are the trips.
    - An edge E is formed between each pair of nodes.
    - The weight on each edge is a function of the similarity between trips i and j.
- Using prior knowledge we could utilize the underlying principles of **multiple-instance learning** and thus shifting to the classification regime:
    - Take 1 trip batch (of the driver of interest) and assign for all the trips positive labels.
    - Draw 1,000 (arbitrary number) trips randomly, and assign these trips negative labels.

Local Evaluation

- Essentially, utilizing **multiple-instance learning** we've got 2,736 different classification models to build and evaluate.
- We can use k-fold cross-validation to evaluate the AUC performance of a model.
- We pick an arbitrary batch, build a random forest with our engineered features and evaluate the model performance.

Why Random Forest?

- In theory, (Cortes and Mohri 2004) discuss algorithms that directly maximize AUC, particularly **RankBoost**[1].
- In practice, using set of 20 speed quantiles we get:

	Logistic Regression	AdaBoost	Random Forest
LB Score	0.64	0.72	0.80

[1] This is similar to the choice of the bestweak learner for boosted stumps in AdaBoost.

Methodology

The Statistician Approach:

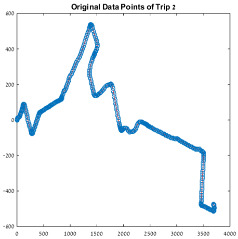
- Simultaneous inference (a.k.a. selective inference)
- Use statistical methods to analyze the data
  - Time series approach to depict the driver's behavior
  - Descriptive statistics (e.g. Mean, Variance, Quantiles)
- Gently handcraft the features

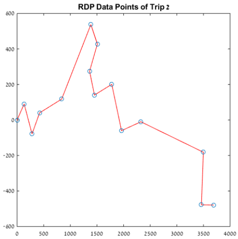
The Computer Science guy Approach:

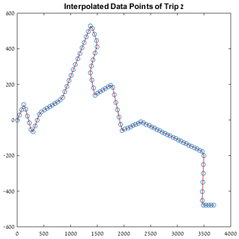
- Use technique utilized in digital image processing
  - Image descriptors to depict the trip's trajectory (e.g. HoG descriptor)
  - Shape matching
  - Template matching
- Brutally extract features from the data

Data Representations

- As movement patterns - trajectories with time dimension of moving vehicles. The original representation.
- As spatial shapes - trajectories without time dimension of their trajectories. Acquired by Ramer–Douglas–Peucker (RDP) algorithm, and (linear) curve interpolation at fixed step size [e.g. 50 meters].







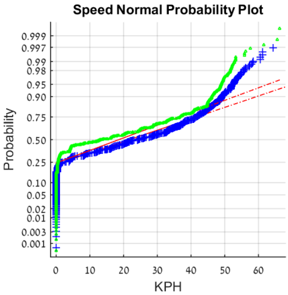
773 time points $\overset{RDP}{\to}$ 15 RPD points $\overset{Inter}{\to}$ 99 Equally spaced points

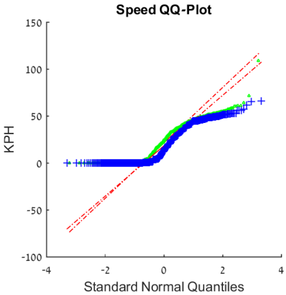
Feature Engineering; Quantiles of Derivatives (QoD)

- QoD trip descriptor takes a telemetric signal (i.e. speed), differenced it, and extract quantiles as a feature vector.
- **Important parameters**:
  - `V`; measurement vector containing the values to be differenced
  - `Q`; the number of quantiles to extract
  - `lag`; an integer indicating which lag to use
  - `rz`; a logical whether to ignore zeros when calculating the quantiles
- Feasible temporal signals to feed into QoD feature extractor:
  - *Distance* $\underset{\frac{\mathrm{d}}{\mathrm{d}t}}{\rightarrow}$ Speed
  - *Speed* $\underset{\frac{\mathrm{d}}{\mathrm{d}t}}{\rightarrow}$ Acceleration
  - *Acceleration* $\underset{\frac{\mathrm{d}}{\mathrm{d}t}}{\rightarrow}$ Jerk
  - *Angle* $\underset{\frac{\mathrm{d}}{\mathrm{d}t}}{\rightarrow}$ Angular velocity

Setting QoD Parameters

- To set Q and rz we could look on QQ-Plots
  - BLUE: Trip 2, GREEN: Trip 1

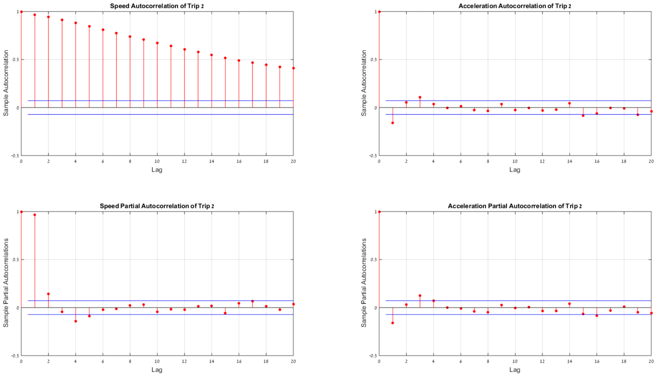




- For example (Trip 2), if we set $Q = 100$ (Percentiles), then:
  - $\{1\%, 2\%, ..., 25\%\} \approx 0 \rightarrow$ *Redundant*
  - $\{25\%, ..., p\%, ..., 75\%\} \approx p/2 \rightarrow$ *Good*
  - $\{76\%, 77\%, ..., 100\%\} \approx 50 \rightarrow$ *Redundant*

Setting QoD Parameters (cont.)

- To set the lag-difference[2] we analyze the autocorrelation and partial-autocorrelation functions for each signal $S(t)$



- The 1st,2nd,4th lag-difference of the **Speed** is statistically significant.
- The 1st,2nd,3rd lag-difference of the **Acceleration** is statistically significant.

[2] The l lag-difference of $S(t)$ is $S_{t+l} - S_t$

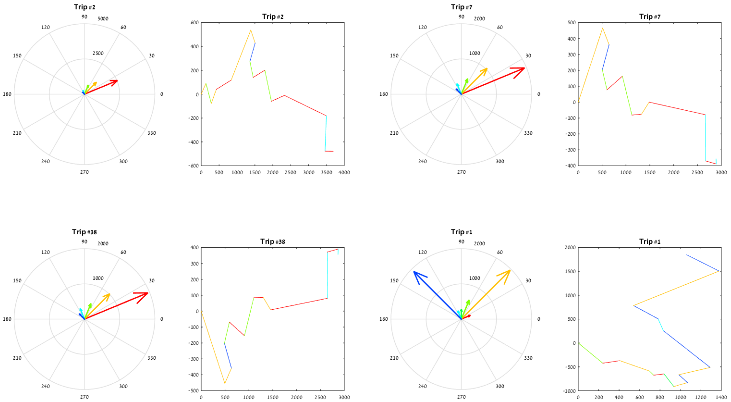
Feature Engineering; HoG Features

- HoG trip descriptor takes a trip's trajectory and described the distribution of path orientation.

- **Important parameters**:
  - `NumBind`; Number of orientation histogram bins
  - `UseSignedOrientation`; Should the orientation be in $[-\pi, \pi]$ or $[0, \pi]$?
  - `scale_feature` Normalize the feature?
- Input signals:
  - Angles (in rad) between pairs of successive points
  - Distances between pairs of successive points
- **Prerequisite**: The arbitrary rotation should be removed from the angle vector first.
- Note: The inputs can be acquired both from the temporal data points or the spatial data points (after RDP).

HoG Visualization

- 8 Angle bins without sign orientation



- Trips 2,7 are the same (with some edge trimming)
- Trip 8 is a horizontal mirroring of trips 2,7
- Trip's 1 HoG features show the trip is not similar to 2,7 and 8

Feature Engineering; Miscellaneous Features

- In addition to using trip spatial descriptor and temporal quantiles, extracting miscellaneous features could be useful. Here are some examples of types of features introduced to the model (the ones ended up in the model are in bold):
  - Moment: Mean, **Variance**, Skewness, Kurtosis for the different temporal signals.
  - Number of rows (representing seconds) for each trip.
  - **Total length of a trip**.
  - **The ratio between Number of rows to total trip length**
  - **Sum of Angles Shift of the Trip**
  - **Trip centroid mean**
  - The total area of the trip
  - Max speed (we assume outliers are not arbitrary)

Ablative Analysis

- How the system breaks for the best model submitted:

Components	**AUC**
Overall system 0.902	0.902
Quantiles of Derivatives 0.877	0.877
Histogram of Oriented Gradients 0.771	0.771
Miscellaneous Features 0.781	0.781
Baseline:1st Place	0.979

Winning Solutions[3]

Two stage solution:

1. Trip matching; identify frequently taken trips as they were likely to be trips from the respective driver.
2. Apply supervised models to telematic features for unmatched trips.

> *Our final telematics model consisted of an ensemble of 6 different models. We mainly used Random Forests, in combination with Gradient Boosting and Logistic Regression.*

[3] by Team Driving It (2nd rank)

Trip Matching; How to?

- First, we define each trip as a node of a graph, and an edge is formed between each pair of nodes ($400^2 = 160000$ pairs).
- Second, we extract *spatial features*; there are 2 different approaches:
  - HoG descriptor; requires either temporal/spatial data points and choice for:
    - *number of angle bins*
    - whether to use sign orientation
  - Trip K-Shingles (resemble to document matching); requires equally spatial intervals and choice for:
    - *stride* (number of successive intervals)
    - K the number of angle tokens (somewhat like in HoG)
- Third, define the distance metric; *Cosine dist.* and *Euclidean dist.* could be a good choice.
- Fourth, find the similarity between each pair of trips; Ncuts/SVD are appropriate.
- Fifth, match the trips.

Trip Matching; Case Study

- Here we take the HoG features[4], find the cosine distance, and create the weight matrix

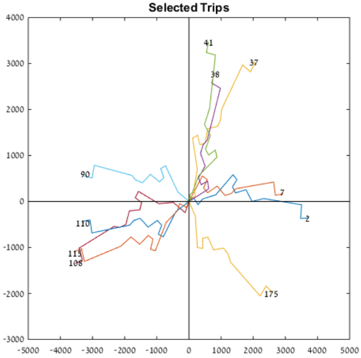
Trip #	***1***	***2***	***3***	***4***	***7***	***38***	***101***	***370***
1	**1.00**	0.59	0.54	0.54	0.62	0.61	0.53	0.57
2	0.59	**1.00**	0.85	0.87	**0.98**	**0.98**	0.93	0.93
3	0.54	0.85	**1.00**	**0.95**	0.87	0.87	0.93	0.91
4	0.54	0.87	**0.95**	**1.00**	0.89	0.89	0.94	**0.96**
7	0.62	**0.98**	0.87	0.89	**1.00**	1.00	0.90	0.93
38	0.61	**0.98**	0.87	0.89	**1.00**	**1.00**	0.90	0.93
101	0.53	0.93	0.93	0.94	0.90	0.90	**1.00**	**0.95**
370	0.57	0.93	0.91	**0.96**	0.93	0.93	**0.95**	**1.00**

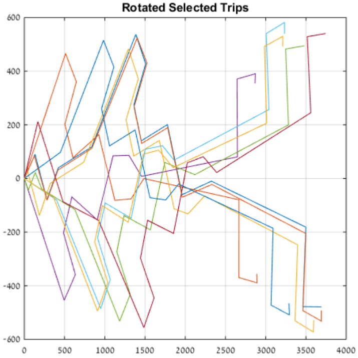
- Similarities ≥ 0.95 are in bold

[4] The parameters were found with grid search and 5 fold-CV

Trip Matching; Case Study (cont.)

- Using 0.95 as a threshold we found 9 similar trips to trip 2



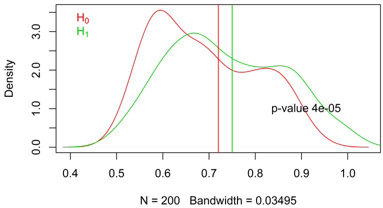


Trip Matching; Case Study (cont.)

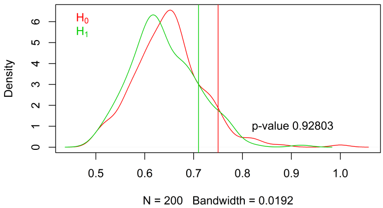
- We can find a threshold in non arbitrary way. Since we have two different populations, trips 201:400 are unknown to be random, therefore the *similarity mean* between trip 2 and 201:400 should be lower than the *similarity mean* between trip 2 and 1:200. We can apply t-test (for independent samples).

$$\begin{cases} H_0 : & \text{the trip is random} \\ H_1 : & \text{the trip is not random} \end{cases} \rightarrow \begin{cases} H_0 : & \mu_{\text{trip}_2} \leq \mu_{\text{negative examples}} \\ H_1 : & \mu_{\text{trip}_2} > \mu_{\text{negative examples}} \end{cases}$$

- Now we have 200 p-values, one for each trip, this let us control the false negatives using p-values threshold.



Two Sample t test for Comparing Two Means for trip 2



Two Sample t test for Comparing Two Means for trip 259

Trip Matching; Implementation

- How to implement the trip matching in our algorithms?

1. Use as extraneous variable a count variable for how many similar trips had been detected.
2. Use as extraneous variable a factor variable is/isn't in the random group.
3. Use the first p eigenvectors from the weight matrix SVD factorization.

Further Resources & References

The code is fully available at: `https://github.com/harell`

Cortes, Corinna, and Mehryar Mohri. 2004. “AUC Optimization Vs. Error Rate Minimization.” *Advances in Neural Information Processing Systems* 16 (16): 313–20.