MINING ANOMALOUS TRAJECTORIES IN A ROAD NETWORK

PALLAVI GUDIPATI & AARATI K

30 April, 2014

Abstract

The aim of our project was to develop an offline model to identify anomalous trajectories. This targets applications like fraud detection or road network change in modern cities. We were given GPS traces of about 10,000 taxis from the city of Beijing. Our approach was *Isolation based*. We calculate an anomaly score for a test trajectory which indicates how *different* it is from the support set. If the score is above a threshold, the trajectory is categorized as anomalous. We have adapted iBAT's approach which is based on Isolation Forest(iForest) method and suggested improvements to it. We tested our model by manually labelling trajectories as anomalous or not and compared it with our model's result. We achieved a detection rate of 81% and a false alarm rate of 12%.

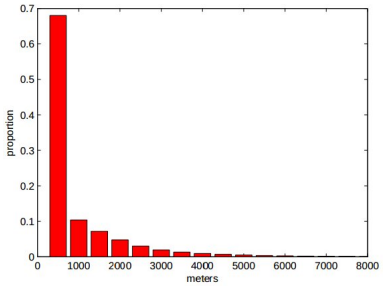
CONTENTS

1 INTRODUCTION

In modern cities, more and more vehicles, such as cabs, have been equipped with GPS devices for localization and navigation. Gathering and analyzing the large-scale GPS traces have provided us a great opportunity to reveal the hidden facts about the city dynamics and human behaviours, enabling diverse innovative applications.

We aim to identify anomalous driving patterns given a cabs GPS traces. A trajectory is anomalous if it contains a sub-trajectory that deviates from the expected path. More specifically, the route used to connect the first and the last node in the sub-trajectory is not the expected route. The more the given path deviates from the expected, the more anomalous it is.

Such scenarios are encountered in real life. Many a times, tourists are tricked by fraudulent taxi drivers who deliberately take longer routes in order to overcharge the passenger. But such routes are usually very different from



(b) distance interval

Figure 1: Histogram of distance between consecutive points in the dataset

the more frequented routes and can be automatically detected by mining existing trajectories. Given the GPS-traces of a trajectory, we can determine how different it is from the expected path by comparing them.

Another example where is can be used is changing road networks. Urban road networks often change over time in developing cities, it is important to update these changes in the digital map. If this is done manually by digital map providers, it would be expensive and also difficult to cap- ture the changes in time. If GPS-equipped taxis are viewed as moving sensors probing the real-time information about urban road network, then the taxi traces accumulated in a new and different area might indicate a sudden road network change, i.e. a newly-built or blocked road segment nearby.

2 DATA PREPROCESSING

The dataset being is the T-drive sample dataset published my *Microsoft*. It contains the GPS traces of routes of 10,357 taxis from the city of Beijing during a period of 7 days. The total number of points in this dataset is about 15 million and the total distance of the trajectories reaches to 9 million kilometers. The sampling rate is about once in 20 seconds during a trip. Each GPS trace has the format

$$< taxiID, timestamp, longitude, latitude >$$

2.1 *Representing Points*

We divide the city into grids of equal size. A point in the trajectory is now relaxed to a grid. We choose the size of the grids by considering the average distance between consecutive points in the dataset. Figure 1 is a visualization of distance between consecutive points in the dataset.

The average distance between consecutive points is $633m$. We have chosen our grid cell size to be 400m $\approx \frac{633}{\sqrt{2}}m$. This has been done keeping in mind that the taxi might travel along the diagonal of the grid, in which case the two points should not belong to the same grid. The city has been divided into grids, each of which is labelled using an (x, y) pair.

2.2 *Dividing taxi GPS route into trajectories*

The GPS dataset contains the taxis' traces. There is no distinction between when the taxi was unhired, idle or occupied. We want to divide the taxis' traces into *trajectories*, when the taxi was actually travelling from a source to a destination. We regard a continuous stretch of taxi-travel as a trajectory. If

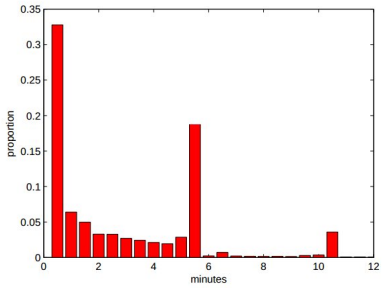


Figure 2: Histogram of time difference between consecutive points in the dataset

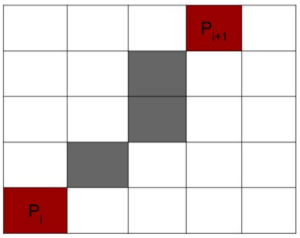


Figure 3: Augmentation of Trajectory

we encounter a difference in timestamps between consecutive points to be greater than the threshold, we terminate the trajectory at that point and start a new one.

Figure 2 is a histogram showing the time between consecutive points in the dataset. For points of our use, we see that the sampling rate is high (around once in 20s). We also see a peak around 5 min. We have chosen 5 min to be our threshold time. We have not chosen a lesser value because, from the histogram, we infer that the taxi is still in motion (not idle) and by dividing it further, we may be losing valid sub-trajectories.

2.3 *Augmenting Trajectories*

We now seek to represent taxi trajectories as a continuous sequence of adjacent grid cells rather than that of points. For each point in the trajectory, we represent it as a grid. Now, a given trajectory has a sequence of disconnected points (typically). This requires us to *augment* the path, i.e, artificially add a sequence of cells to obtain a contiguous path between the disconnected cells. Figure 3 is a representation of the augmentation process. While augmenting, we are constrained to move within the rectangle with source and destination as the corners.

3 DATABASE

We have created two databases as follows:

- TRAJECTORY: This database's key is the trajectory ID(`trajID`) and value a string containing the coordinates of the grids in the trajectory in order.

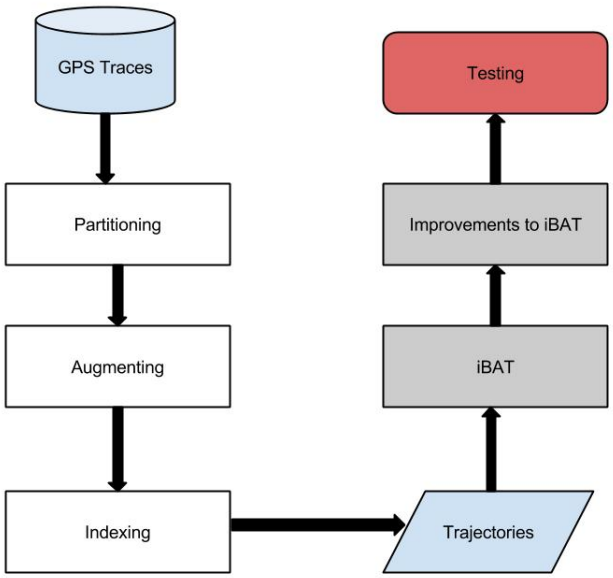


Figure 4: Overview of our approach

- GRID: This database's key is a string (x, y) denoting the coordinates of the grid and the value is a string containing the tuples (`trajID`,*position*) where the grid occurs in `trajID` at the corresponding *position*.

We have created an inverted index database(GRID) for the ease of obtaining trajectories between a given source and destination grid cells.

4 I-FOREST METHOD OF DETECTING ANOMALOUS ITEMS

iForest (*Isolation Forest*) is an isolation based approach towards anomaly detection. Each item has a set of attributes and associated values. For each test item, we calculate an anomaly score. If the score is greater than a given threshold, the item is declared anomalous.

First, we describe what an iTree is. It is Binary Search Tree where each node contains a set of items. We start with the root node containing all items. At each node, an attribute and a threshold value are chosen at random such that it lies between the minimum and maximum values of that attribute. Then, the node is split into two children depending on this attribute and threshold. We now try growing the tree for the children and stop when all the leaf nodes have exactly one element. The original iTree construction algorithm assumes that the original set of items has no duplicates.

To assign an anomaly score to an item, we construct many iTrees, i.e, we construct an iForest. For each tree, we note down the depth at which we obtained a leaf node containing the test item. As anomalous items are different, they tend to get isolated at lesser depths. Now, we calculate the anomaly score for the test item as

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where

- x is the test item
- n is the total number of items considered during tree construction

- $E(h(x))$ is the average depth of the test item and
- $c(n)$ is the normalization term associated with n as greater the number of elements considered, deeper the tree grows.

The value of $c(n)$ is given by

$$c(n) = 2H(n-1) - 2(\frac{n-1}{n})$$

where $H(i) = ln(i) + 0.5772156649$ (Euler's constant). This normalization term has been derived from the analysis of average path length of binary unsuccessful searches in a BST.

Some attributes of this method are

- *Subsampling*: We can use a subsample of the whole dataset for constructing a tree.
- *Partial Model Generation*: In each tree, we need to find the depth of the test point. Hence, at every stage, we need to grow only the branch that contains the test item. We need not grow the whole tree.
- *Scalability*: It scales to perform well on datasets with lot of irrelavant features.
- *Cost of computation*: It does not require us to calculate complicated distance measures and is hence computationally light-weight.

5 i-BAT method to detect anomalous trajectories

We would first like to give a precise definition of what an anomalous trajectory is. An anomalous trajectory has one of the three properties

- *Very long*: Unusually longer than trajectories connecting the same source destination pair. For example, when a fraudulent taxi driver takes a deliberate detour.
- *Very short*: A less-frequently used short-cut.
- *A less-frequented route*: Any less frequently used route.

For this problem, there are three popular approaches:

1 *Distance based approach* Using this approach we can find unusually longer or shorter routes. But this is not effective in the case of trajectories which take some other non-frequent route of the same length between the source and destination.

2 *Density based approach* In the case of density based approaches, we need to choose a common set of parameters and many times, this is not effective enough to identify all the trajectories. This basically happens because our assumptions and invariants need hold across itemsets will different supports.

3 *Isolation based approach* Under this approach, we see how *different* a test item is from its support set. This approach proves to be more useful for our case. In some sense, we are deriving the parameters for a support set from the support set itself instead of assuming some global invariants.

The iBAT algorithm adapts the iForest algorithm for our current problem in the following way:–

- For a given test trajectory, the itemset is the set of trajectories connecting the same source and destination grid. Also, we consider subtrajectories of our current set of trajectories that contain the source and destination in the same order, that is, the source should occur before the destination.
- The attributes in this case are the grids in the *test trajectory*. At each stage of tree construction, we choose a grid from the test trajectory and divide our support set depending upon whether or not the trajectories contain that particular grid.

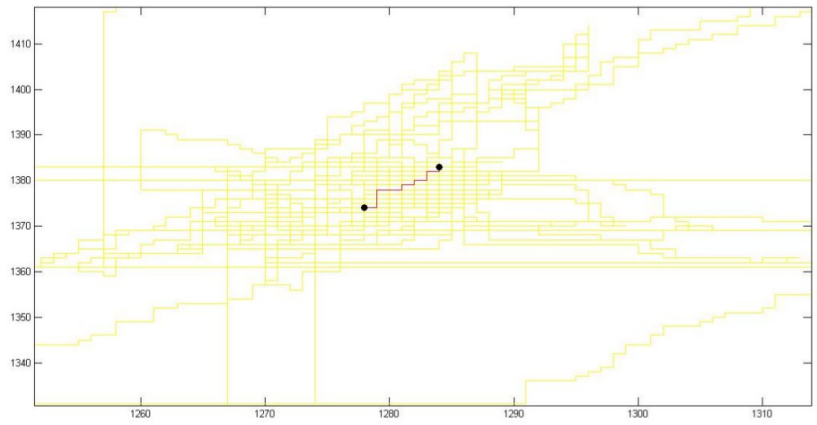
The calculation of anomaly score remains the same. Using grids as attributes is effective in the case of longer trajectories as they tend to have many unvisited cells. In the case of shorter trajectories, we see that the number of attributes (number of grids) is low and hence shorter trajetories tend to separate out quickly. For the case of less-frequently used trajectories, this approach is again seen to be effective because, intuitively, we can define a less-frequently used trajectory as one containing an unusal combination/sequence of points, which is captured by this model.

However, one difference between the iBAT approach and iForest approach is the present of duplicates in the support set. Also, this approach is *lazy*, in the sense we donot start constructing iTrees until given a source-destination pair and a test trajectory.

6 IMPROVEMENTS SUGGESTED TO IBAT

We have suggested two improvements to iBAT.

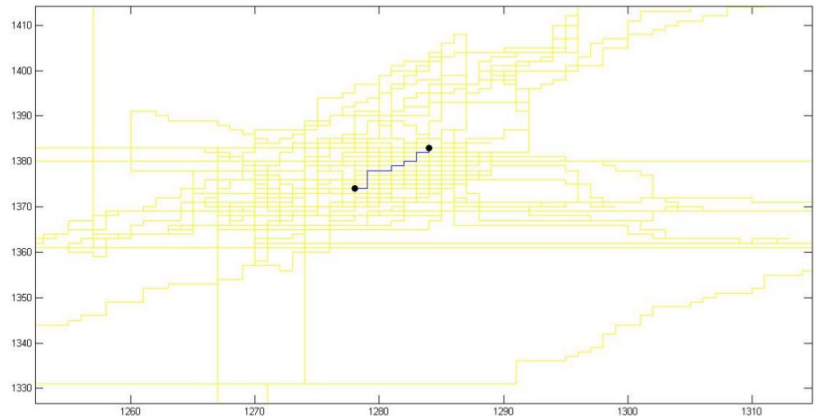
1 *Handling duplicate points*: Consider the case given in the figure below. This trajectory was classified as anomalous initially. On growing the iTree, we ended up with many trajectories in the leaf node all of which had the grids contained in the test trajectory. Because support in the leaf node was not being accounted for, we initially classified this trajectory as anomalous.



iBAT reports height of the test trajectory when it reaches a homogeneous node. Say the node has x trajectories left. In our model, we report the height as $h + log(i+1)$ instead of x. We chose to increase the height by a factor of $log(i)$ and not linearly in i by examining the formula for anomaly score. In the exponent, the denominator term which normalizes the height increases at a rate of $log(n)$. If the numerator increases linearly in i, then the magnitude of support for the test trajectories will start playing a role. Greater the support, lesser

will be the value of exponent (exponent is negative) and the trajectory will be classified as non-anomalous. As the denominator increases at a rate of $log(n)$, we chose to increase the numerator at the rate of $log(i)$.

After incorporating this change, the trajectory misclassified previous was correctly classified. This is illustrated in the figure below.



2 *Detecting loops*: The above model does not account for loops in the trajectory. Given a direct trajectory and another with the same grids but containing loops, the attribute set will not change and hence both will attain the same height under this strategy.

The method we suggest examines the number of times a grid has occurred in the trajectories in the support set. For every grid in the test trajectory, we first examine trajectories in the support set which contain this grid. For such trajectory in the support set, we make note of how many times this grid has been encountered. We calculate the mean and standard deviation of this evidence. A trajectory is classified as anomalous if $|N(p) - \mu| > 2\sigma$, where μ is the mean and σ is the standard deviation.

The interval of 2σ is the 95% confidence interval in the case of Gaussian distribution. Hence, if the frequency lies beyond this interval for any grid, one can classify the grid as anomalous. The value 2 chosen can be tweaked so as to increase or decrease flexibility in the model. We perform this check by randomly sampling grids from the trajectory (we want to keep our model light-weight). If an anomalous grid is found, then we examine the length of the loop. If it is beyond a threshold, we classify the trajectory as anomalous.

Choosing threshold based on σ makes sense against choosing a common threshold value (an interval of same length around mean) for all trajectories. If for a given support, the standard deviation for a grid is low, then it reflects that we are highly confident that this grid should be visited μ times. On the other hand, if the standard deviation is high, it means that our support shows has a wide range of values and is not very localized. Again, we focus on our aim to make the model such that the effect of the size of the support set is not much.

7 EMPIRICAL EVALUATION

As we were not able obtain any golden set for T-Drive dataset, we decided to empirically evaluate our model.
We manually labelled the trajectories between few source-destination pairs. We used the trajectories' visual plot and Beijing's road network to label them. We then used these trajectories as our test dataset.
We used two evaluation measures:

1. $Detection\ rate(dr) = \frac{TP}{TP+FN}$
2. $False\ alarm\ rate(fpr) = \frac{FP}{FP+TN}$

where TP stands for *true positives*, FP for *false positives*, TN for *true negatives* and FN for *false negatives*
We achieved the following results:

Measures	*Our model*	*Paper*
dr	81%	90%
fpr	12%	2%

One of the reasons that our results don't match with that given in the paper is that the choice of the trajectories taken as test set can have a major influence on the results obtained.

8 POSSIBLE FUTURE WORK

The following are the possible approaches that can build upon this model:–

1 *Augmenting Trajectories using the Dataset*: Currently, to augment trajectories, we are taking a random walk between the two disconnected grids. Instead of this, we can mine trajectories from the road-network graph. This will be effective as we will know the actual route taken. Also, if two taxis use the same route (for some part of the journey), their trajectory will be exactly the same which is not the case with our current approach.

2 *Using sub-trajectories as features*: Instead of using grids as features, we can use subtrajectories as features. This will be, in some sense, similar to asking the question "Which taxis have taken this route?", which is more intuitive in our setting.

3 *The concept of Landmark Graphs*: In our current approach, we give equal importance to all the grids. However, in practice, we know that a city has *landmarks*. Not all places are equally important. One different approach towards representing the trajectories could be to represent them as a sequence of landmarks and then analyze the trajectory piecewise, from one landmark to the next.

4 *Weighing the nodes*: In our current setting, we discard a trajectory from the support set if it does not contain the grid under consideration. However, this trajectory might contain a grid close to the grid under consideration, in which case, discarding this trajectory entirely is not really correct. What one could do is to give some weight to the trajectory depending upon how close the grid is to the grid under consideration.

[1] *iBAT: Detecting Anomalous Taxi Trajectories from GPS Traces*, by Daqing Zhang , Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun and Shijian Li

[2] *Isolation Forest*, by Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou

[3] *iBOAT: Isolation-Based Online Anomalous Trajectory Detection*, by Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, Shijian Li and Zonghui Wang

[4] *A Taxi Driving Fraud Detection System*, by Yong Ge, Hui Xiong1, Chuanren Liu and Zhi-Hua Zhou

[5] *T-Drive: Enhancing Driving Directions with Taxi Drivers Intelligence*, by Jing Yuan and Xing Xie

[6] *T-Drive: Driving Directions Based on Taxi Trajectories*, by Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie Xing Xie , Guangzhong Sun and Yan Huang