Detecting Anomalous Trajectories

Pallavi Gudipati
Aarati K

CS6720: Data Mining

Problem Statement

To create an offline model to mine **anomalous** *trajectories.*

What are *anomalous* trajectories?

Problem Statement

To create an offline model to mine **anomalous** *trajectories.*

What are *anomalous* trajectories?

An anomalous trajectory has one of the three properties:–

Problem Statement

To create an offline model to mine **anomalous** *trajectories.*

What are *anomalous* trajectories?

An anomalous trajectory has one of the three properties:–

- Longer than usual

  A deliberate detour by a taxi driver.

Problem Statement

To create an offline model to mine **anomalous** *trajectories.*

What are *anomalous* trajectories?

An anomalous trajectory has one of the three properties:–

- Longer than usual
  A deliberate detour by a taxi driver.
- Shorter than usual
  A short-cut.

Problem Statement

To create an offline model to mine **anomalous** *trajectories.*

What are *anomalous* trajectories?

An anomalous trajectory has one of the three properties:–

- Longer than usual

  A deliberate detour by a taxi driver.
- Shorter than usual

  A short-cut.
- An unusual route

  A less frequently travelled route.

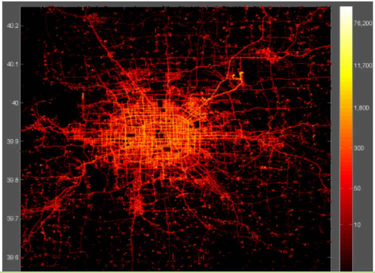
Dataset

This is a sample of T-Drive GPS trajectory dataset.

- GPS traces of about 10,357 taxis from the city of Beijing.
- Total distance exceeds 9 million kilometers.
- Sampling rate once in 20 seconds(approx).
- About 15 million points.

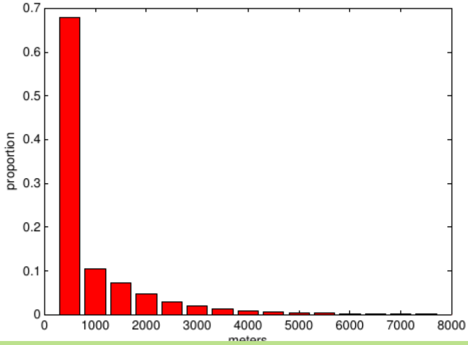
The format of a trajectory point is

$$< \mathit{taxiId}, \mathit{longitude}, \mathit{latitude}, \mathit{timestamp} >$$



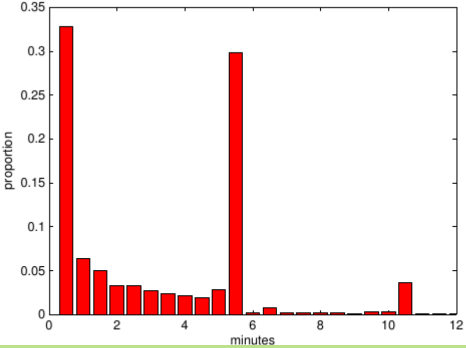
Data Preprocessing

- **Representing Points**
  - We divide the city into grid cells of equal size.
  - Each point is represented as (X, Y) coordinates.
  - The average distance between consequtive points in a trajectory is approximately 633m. We chose the grid cell size as 400m $\approx \dfrac{623}{\sqrt{2}} m$.



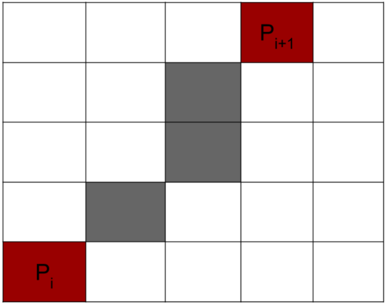
Data Preprocessing

- **Processing GPS traces to obtain trajectories**
  - We have collection of GPS trace points corresponding to each taxi for a week.
  - We divide the collection of points into trajectories.
  - We choose time threshold as 6 min. We chose this value by examining the time statistics of the data.



Data Preprocessing

- **Augmenting trajectories**
  - Trajectories now can contain non-adjacent grid points.
  - We introduce grid points to make the trajectory continuous.
  - As the sampling rate is high, the number of points introduced is not high.



Final Database

- TRAJECTORY DATABASE

  Each trajectory has a unique ID. The database entries are:–
  - *Key*– Trajectory ID
  - *Value*– String containing the *(X,Y)* coordinates of points in the trajectory in order.

- GRID DATABASE

  This is an *inverted index* of Trajectory database. The entries are:–
  - *Key*– String *(X,Y)* corresponding to the grid coordinate
  - *Value*– String containing the tuples ($trajID, position$) where the grid occurs in *trajID* at the specified position.

Different Approaches possible

Different Approaches possible

- *Distance Based*

  Tends to overlook short-cuts.

Different Approaches possible

- *Distance Based*

  Tends to overlook short-cuts.

- *Density Based*

  We need to decide upon a common set of thresholds and parameters for different *source-destination* pairs. Not very efficient.

Different Approaches possible

- *Distance Based*

  Tends to overlook short-cuts.

- *Density Based*

  We need to decide upon a common set of thresholds and parameters for different *source-destination* pairs. Not very efficient.

- **Isolation-based**

  Anomalous trajectories are few and different. They tend to get isolated if the appropriate criterion is used.

Isolation Forest(*iForest*) approach

The salient features of this approach are:–

Isolation Forest(*iForest*) approach

The salient features of this approach are:–

- Builds partial models and exploits sub-sampling

Isolation Forest(*iForest*) approach

The salient features of this approach are:–

- Builds partial models and exploits sub-sampling
- Doesn't use complex distance and density measures and reduces cost of computation

Isolation Forest(*iForest*) approach

The salient features of this approach are:–

- Builds partial models and exploits sub-sampling
- Doesn't use complex distance and density measures and reduces cost of computation
- Highly scalable and can handle large datasets with large number of irrelavant features

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

- We construct an *iTree*. It is a proper BST.

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

- We construct an *iTree*. It is a proper BST.
- Each node of the tree has a set of items. The root node contains all items.

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

- We construct an *iTree*. It is a proper BST.
- Each node of the tree has a set of items. The root node contains all items.
- At each stage, we randomly choose a test attribute q and a corresponding threshold value p. We split the items among the children by comparing with the threshold.

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

- We construct an *iTree*. It is a proper BST.
- Each node of the tree has a set of items. The root node contains all items.
- At each stage, we randomly choose a test attribute q and a corresponding threshold value p. We split the items among the children by comparing with the threshold.
- We proceed till we arrive at a homogeneous node containing the item under consideration.

Isolation Forest(*iForest*) approach

To calculate the anomaly score of an item:–

- We construct an *iTree*. It is a proper BST.
- Each node of the tree has a set of items. The root node contains all items.
- At each stage, we randomly choose a test attribute q and a corresponding threshold value p. We split the items among the children by comparing with the threshold.
- We proceed till we arrive at a homogeneous node containing the item under consideration.
- We construct several such trees and each time we make note of the depth of the node containing the item.

- The anomaly score is calculated as:–

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

where,

- x is the item for which we are calculating the anomaly score
- *d(x)* is the average depth of x over all the trees constructed.
- n is the number of distinct points in the initial set
- $c(n)$ normalizes the average depth. This required because if we start with a bigger set, the tree is likely to be deeper.

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

where $H(i) = ln(i) + 0.5772156649$ (Euler's constant).

- This model is non-parametric.

- This model is non-parametric.
- It can exploit sub-sampling. Also, we need not grow the full tree.

- This model is non-parametric.
- It can exploit sub-sampling. Also, we need not grow the full tree.
- No commmon threshold or parameter is chosen for itemsets with different support. It is homogeneous (in some sense).

The iBAT approach

- This adapts the iForest method to our current problem.

The iBAT approach

- This adapts the iForest method to our current problem.
- *Grids* are used as features for comparision at each stage of building the iTree.

The iBAT approach

- This adapts the iForest method to our current problem.
- *Grids* are used as features for comparision at each stage of building the iTree.
- All the trajectories connecting the same source-destination as the test trajectory are placed in the root. This is called the *support set*.

The iBAT approach

- This adapts the iForest method to our current problem.
- *Grids* are used as features for comparision at each stage of building the iTree.
- All the trajectories connecting the same source-destination as the test trajectory are placed in the root. This is called the *support set*.
- At each stage of the iTree, we randomly choose a grid from the test trajectory and compress our support set to those which contain the grid.

The iBAT approach

- This adapts the iForest method to our current problem.
- *Grids* are used as features for comparision at each stage of building the iTree.
- All the trajectories connecting the same source-destination as the test trajectory are placed in the root. This is called the *support set*.
- At each stage of the iTree, we randomly choose a grid from the test trajectory and compress our support set to those which contain the grid.
- The number of compressions to obtain a homogeneous node correspond to the depth of the test point.

The iBAT approach

- This adapts the iForest method to our current problem.
- *Grids* are used as features for comparision at each stage of building the iTree.
- All the trajectories connecting the same source-destination as the test trajectory are placed in the root. This is called the *support set*.
- At each stage of the iTree, we randomly choose a grid from the test trajectory and compress our support set to those which contain the grid.
- The number of compressions to obtain a homogeneous node correspond to the depth of the test point.

If trajectory corresponds to a not-so-frequently-used path, then it contains grids not visited by other trajectories. It will have lesser depth.

Improvements to iBAT

We propose two improvements to iBAT:

Improvements to iBAT

We propose two improvements to iBAT:

- Modification in the calculation of the depth of the tree.

Improvements to iBAT

We propose two improvements to iBAT:

- Modification in the calculation of the depth of the tree.
- Handling of loops.

Improvements to iBAT: Depth of the tree

- In iBAT, the tree construction terminates when either all the trajectories are isolated or all the remaining trajectories contain all the points of the test trajectory.

Improvements to iBAT: Depth of the tree

- In iBAT, the tree construction terminates when either all the trajectories are isolated or all the remaining trajectories contain all the points of the test trajectory.
- In the second case, the test trajectory is reported to be isolated after as many number of steps as the current depth of the tree.

Improvements to iBAT: Depth of the tree

- In iBAT, the tree construction terminates when either all the trajectories are isolated or all the remaining trajectories contain all the points of the test trajectory.
- In the second case, the test trajectory is reported to be isolated after as many number of steps as the current depth of the tree.
- But, the number of trajectories left in the last node should also play a role in the test trajectory's anomaly score.

Improvements to iBAT: Depth of the tree

- In iBAT, the tree construction terminates when either all the trajectories are isolated or all the remaining trajectories contain all the points of the test trajectory.
- In the second case, the test trajectory is reported to be isolated after as many number of steps as the current depth of the tree.
- But, the number of trajectories left in the last node should also play a role in the test trajectory's anomaly score. As the denominator of the exponent in the formula is of the order $\log(N)$, we report the depth of the tree as:

$$Depth = currDepth + \log(N_{remaining} + 1)$$

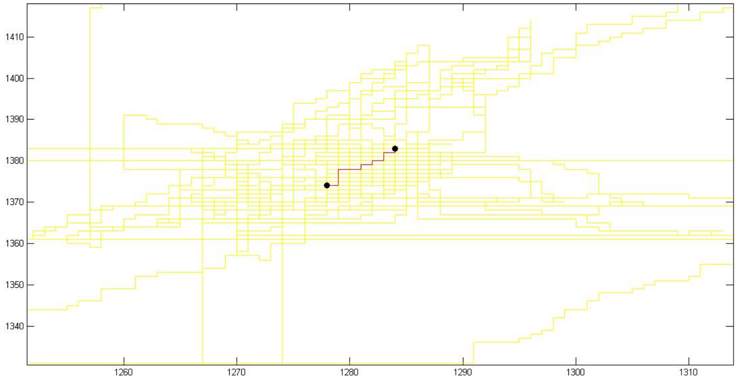
Improvements to iBAT: Depth of the tree

- In iBAT, the tree construction terminates when either all the trajectories are isolated or all the remaining trajectories contain all the points of the test trajectory.
- In the second case, the test trajectory is reported to be isolated after as many number of steps as the current depth of the tree.
- But, the number of trajectories left in the last node should also play a role in the test trajectory's anomaly score. As the denominator of the exponent in the formula is of the order $\log(N)$, we report the depth of the tree as:

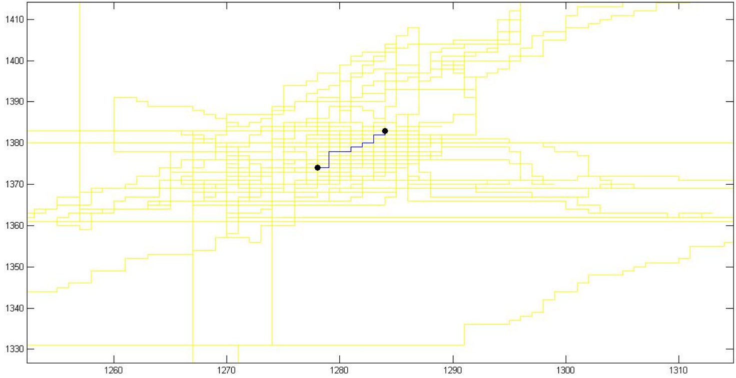
$$Depth = currDepth + \log(N_{remaining} + 1)$$

- If we increase *Depth* with order N, the number of trajectories between the *source* and *destination* would also play a role.

Improvements to iBAT: Depth of the tree



Improvements to iBAT: Depth of the tree



Improvements to iBAT: Handling of loops

To handle multiple loops:

Improvements to iBAT: Handling of loops

To handle multiple loops:

- For each point p in the test trajectory, we take all the trajectories under consideration that contain this point D_p.

Improvements to iBAT: Handling of loops

To handle multiple loops:

- For each point p in the test trajectory, we take all the trajectories under consideration that contain this point D_p.
- We calculate the mean μ and deviation σ of the number of times a trajectory passes through point p.

Improvements to iBAT: Handling of loops

To handle multiple loops:

- For each point p in the test trajectory, we take all the trajectories under consideration that contain this point D_p.
- We calculate the mean μ and deviation σ of the number of times a trajectory passes through point p.
- A trajectory is classified as anomalous if:

$$N_{test,p} \geq \mu + 2\sigma \quad or \quad N_{test,p} \leq \mu - 2\sigma$$

Empirical Evaluation

- Manually labelled trajectories between few source and destination pairs.
- Evaluated our model using these trajectories as test dataset.
- Measures used:

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

Empirical Evaluation

- Manually labelled trajectories between few source and destination pairs.
- Evaluated our model using these trajectories as test dataset.
- Measures used:

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

dr	81%
fpr	12%

Future Work

- Using of sub-trajectories as features instead of grid cells.

Future Work

- Using of sub-trajectories as features instead of grid cells.
- Try to augment paths based on other trajectories or from road networks graphs.

Future Work

- Using of sub-trajectories as features instead of grid cells.
- Try to augment paths based on other trajectories or from road networks graphs.
- Intoduce the concept of 'Landmarks'. Analyze the sub-trajectory between the landmarks instead of th whole.

Future Work

- Using of sub-trajectories as features instead of grid cells.
- Try to augment paths based on other trajectories or from road networks graphs.
- Intoduce the concept of 'Landmarks'. Analyze the sub-trajectory between the landmarks instead of th whole.
- Weight based splitting of nodes instead of binary splitting. This will take the the distance of a point in a trajectory from the point in the test trajectory under consideration.