A Fast Trajectory Outlier Detection Approach via Driving Behavior Modeling

Hao Wu
Fudan University, Shanghai, China
Shanghai Key Laboratory of Data Science
wuhao5688@fudan.edu.com

Weiwei Sun
Fudan University, Shanghai, China
Shanghai Key Laboratory of Data Science
wwsun@fudan.edu.com

Baihua Zheng
Singapore Management University, Singapore
bhzheng@smu.edu.sg

ABSTRACT

Trajectory outlier detection is a fundamental building block for many location-based service (LBS) applications, with a large application base. We dedicate this paper on detecting the outliers from vehicle trajectories efficiently and effectively. In addition, we want our solution to be able to issue an alarm early when an outlier trajectory is only partially observed (i.e., the trajectory has not yet reached the destination). Most existing works study the problem on general Euclidean trajectories and require accesses to the historical trajectory database or computations on the distance metric that are very expensive. Furthermore, few of existing works consider some specific characteristics of vehicles trajectories (e.g., their movements are constrained by the underlying road networks), and majority of them require the input of complete trajectories. Motivated by this, we propose a vehicle outlier detection approach namely DB-TOD which is based on probabilistic model via modeling the driving behavior/preferences from the set of historical trajectories. We design outlier detection algorithms on both complete trajectory and partial one. Our probabilistic model-based approach makes detecting trajectory outlier extremely efficient while preserving the effectiveness, contributed by the relatively accurate model on driving behavior. We conduct comprehensive experiments using real datasets and the results justify both effectiveness and efficiency of our approach.

CCS CONCEPTS

• **Information systems** → **Location based services**; • **Computing methodologies** → **Anomaly detection**;

KEYWORDS

Outlier detection; trajectory data processing; driving behavior; inverse reinforcement learning

1 INTRODUCTION

Different from traditional services that are static, *location-based services* (LBSs) consider *location*, the key for providing *valuable* services to customers. LBS has the great potential to tap into the pulse of a city and to improve the quality of citizens' life, in various areas such as transport, safety and security, entertainment, emergency management and urban planning. It is expected to add a new dimension to smart cities. There are many building blocks to support LBSs, including spatial data indexing, clustering, and pattern mining. In this paper, we focus on *trajectory outlier detection*, a common ingredient required by many LBS applications. In the data mining literature, an outlier refers to a data object that is different from or inconsistent with the remaining objects [16]. In our context, trajectory data capture the movements of vehicles (e.g., cars, taxis) on road networks, and a trajectory outlier (or "anomalous trajectory", an alternative term used in some papers) represents a trajectory that is significantly different from other trajectories [4, 25], which reflects the "few" and "different" characteristic w.r.t. the normal/majority trajectories [3, 4]. Note that for the issue of trajectory outlier detection, each trajectory is corresponding to a given source and destination pair (SD-pair), and an outlier trajectory of a given SD pair is expected to be very different from the normal trajectories that correspond to the same SD-pair. Take Fig. 1 as an example. It defines two SD-pairs, denoted as $\langle S_1, D_1\rangle$ and $\langle S_2, D_2\rangle$, and plots many trajectories moving from S_1 (S_2) to D_1 (D_2). Note trajectories of the same color take the same route. Trajectory R_1 is an outlier w.r.t. $\langle S_1, D_1\rangle$ as it is very different from other trajectories moving from S_1 to D_1. Similarly, trajectory R_2 is considered as an outlier w.r.t. $\langle S_2, D_2\rangle$. In addition, trajectories corresponding to $\langle S_1, D_1\rangle$ will have *zero* impact on the detection of outlier w.r.t. $\langle S_2, D_2\rangle$.

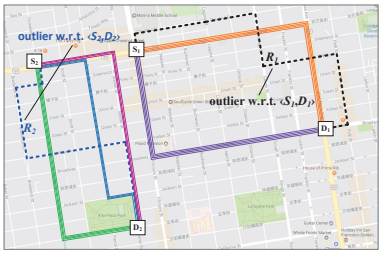


Figure 1: Example trajectory outliers.

Trajectory outlier detection has a large application base. Take taxi fraud as an example. Complaint about dishonest drivers who overcharge passengers by taking routes longer than necessary is one of the most common complaints passengers make. Trajectory outlier detection enables the passengers to find out whether the route taken by the taxi driver is normal or not. In addition, it allows the taxi company to monitor the movements of all the taxis and to identify the dishonest drivers who tend to take routes longer than usual. Take event detection as another example. When many trajectory outliers with similar patterns are reported within a short

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132933

time window, it might be an alter of an abnormal event such as traffic accidents, a temporal close of certain road segments and so on [23]. Last but not least, in terms of data pre-processing, detecting and eliminating outlier is a necessary step for cleaning the trajectory data to generate the data with better quality to support LBS applications, such as pattern mining [7], route recommendation [5], etc.

The rich application base inspires lots of research works on trajectory outlier detection. Most existing works study a more general problem of detecting the trajectory outliers under the Euclidean space [6, 9, 10, 21] by distance or density measure. These approaches do not consider the special characteristics of vehicle trajectory such as the restriction of underlying road networks and the driving preference of experienced drivers. Other works solve the problem by supervised/semi-supervised learning [12, 13, 19]. They all need massive manual labeling which is impractical in real applications. Among these outlier detection approaches, [4, 25] are two typical approaches designed for the vehicle trajectories. [4] proposes an isolated trajectory outlier detecting approach named *iBOAT*, which needs to build an inverted index for historical trajectory data, and to query the database. [25] determines whether an input trajectory is an outlier by matching the input trajectory to the road network and computing the edit distance between the historical data and the input trajectory. However, it is only able to label a *complete* trajectory as normal or abnormal based on the derived anomaly score. In other words, it is *not* able to locate the sub-trajectory that is actually abnormal, or to derive the anomaly score for an incomplete trajectory. We claim that the capability of inferring whether a trajectory is normal or abnormal from a partial trajectory is actually very desirable. Consider the example of taxi rides. Given the pick-up location and the destination a customer wants to go, if we can detect the current path taken by the taxi driver is a potential outlier, customers would like to receive alters asap to improve the quality of service and to minimize customers' lost and risk.

As a conclusion, we claim that an ideal vehicle trajectory outlier detection approach should have following desirable properties.

(1) *Efficiency.* The computation speed should be fast to support the throughput of the day-by-day-growing volume of trajectory data.
(2) *Effectiveness.* It should precisely detect the outliers with low false positive rate as well as correctly quantifying the degree of abnormality.
(3) *Ability of early warning.* It should support the task of early detecting a potential outlier with a partial trajectory.

However, to the best of our knowledge, none of the existing works is able to achieve all three properties. Motivated by this, we try to solve the problem via a novel aspect, i.e., modeling the human driving behavior and proposing a fast vehicle trajectory outlier detection approach called DB-TOD. We adopt and extend the maximum entropy inverse reinforcement learning model with automatic feature correction mechanic to model the driving behavior. The effectiveness is guaranteed by the capability of our model in correctly inferring the latent cost of each road segment via the routing preferences learned from historical trajectory data generated by experienced drivers. The efficiency is achieved by the model-based framework of our approach. That is to say, after training the probabilistic model, there is no need to access the trajectory database any more. For a query trajectory, the only task required is to compute the anomalous score from the model which is very efficient. Moreover, our approach can also handle the problem of early warning of potential outliers with partial trajectory observed. In summary, we mainly make three-fold contribution in this paper.

- We propose a fast online trajectory outlier detection approach. To the best of our knowledge, this is the first work on the outlier detection problem through the view of modeling human driving behavior. The computation cost of our approach is linear to the size of road segments passed by the trajectory which makes it extremely fast for handling tremendous trajectory data streams.
- We extend the MEIRL model by introducing the automatic feature correction mechanic to improve the capability of modeling driving behaviors.
- We conduct comprehensive experiments based on two real datasets. The results demonstrate that DB-TOD outperforms the existing approaches significantly, in terms of both efficiency and effectiveness.

2 RELATED WORK

In the literature, the trajectory outlier/anomaly detection approaches can be mainly divided into three categories. The first category is based on *distance and density metrics*. [9] studies the problem of detecting distance-based outliers in multidimensional datasets. [10] partitions a trajectory into a set of line segments, and then detects outlying line segments through a distance-and-density-hybrid approach. [6] also proposes an approach combining distance and density to detect the fraud taxi trajectories. [21] defines the trajectory outliers by two density-based definitions, i.e., point outlier and trajectory outlier, and aims at finding outliers from the trajectory stream data. However, it can not quantize the outlier degree of a trajectory and its parameter highly depends on the dataset. These metric-based approaches often study general trajectories but ignore the unique characteristic of vehicle trajectories. Moreover, these approaches require accesses to the historical database for computing the density or distances of the other trajectories which further relate to the research area of querying trajectory data in database system [8, 17, 20].

The second category is based on *supervised/semi-supervised learning*. [12] proposes an outlier detection framework on identifying suspicious movements. The trajectories are expressed using discrete pattern fragments and features are extracted to further support the classification algorithm. [13] proposes a conditional random field-based approach for anomaly detection of GPS traces. [19] proposes a semi-supervised learning framework for detecting trajectory with anomalous behavior in a video surveillance scenario. However, these supervised/semi-supervised learning-based approaches require massive manual labeling of dataset which has many restrictions and is impractical for real applications.

The third category is based on detecting outliers by the *pattern*. These approaches are mostly designed for vehicle trajectories, because of the strong tendency of forming patterns and restriction on the road network. Based on the "few" and "different" characteristic of trajectory outliers, [22] proposes an isolation-based approach called *iBAT* which first maps the raw trajectory into grid series and

builds an index for historical grid trajectories, then adopts an iForest algorithm to detect isolated trajectories. However, this approach can only support complete trajectories. To fix this disadvantage of iBAT, [3, 4] propose an enhanced version namely *iBOAT*, which can support online detection with a partial trajectory. [11] also transfers the trajectory into grid trajectories by considering the spatial, temporal and behavior characteristics. However, it is designed for detecting anomalous maritime behavior. *TPRO*, proposed in [24], also focuses on detecting vehicle outliers, by finding top-k popular trajectories of each SD-pair and deriving the outlier scores based on the edit distances between the querying trajectory and the popular trajectories. *TPRRO*, as an enhanced version of TPRO proposed in [25], is a real-time outlier detection algorithm as it can detect trajectories that are not in the historical trajectory dataset. Note TPRO can only detect the outliers in the historical data. However, both TPRO and TPRRO are able to derive the outlier score *only* when the complete trajectory has been fully observed.

DB-TOD, the approach we are going to propose in this paper, is based on a probabilistic model which models the distribution of driving behavior in an unsupervised manner and it does not involve any distance or density metric to avoid the access to database which helps to improve the performance. Moreover, our approach can also be effectively adapted to detect outliers from partial trajectories.

3 PRELIMINARY

Before presenting our approach, we first introduce some basic definitions.

Definition 3.1 (Road Network). A road network is modeled as a directed graph $G(V, E)$, where V refers to the vertex set representing crossroads and E refers to the edge set representing road segments. Each edge $r \in E$ is from a vertex $v \in V$ to another vertex $v'(\neq v) \in V$, where $r.s = v$ and $r.e = v'$ represent the starting vertex and the ending vertex of the edge respectively.

Definition 3.2 (Raw Trajectory). A raw trajectory $\mathcal{T} = \{p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_N\}$ is a list of point locations with time stamps, obtained by localization techniques such as GPS.

In this paper, we focus on vehicle trajectories generated by taxis and cars. As the movements of taxis and cars are constrained by the underlying road networks, we adopt the map matching algorithm [15] to map the trajectory onto the road network to get the edge trajectory defined as follows. To simplify our discussion, we call an edge trajectory as a trajectory in the following and our work is based on edge trajectories instead of raw trajectories.

Definition 3.3 (Edge Trajectory). An edge trajectory is a route $R = \{r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_M\}$ which consists of list of adjacent road segments recording the path of a raw trajectory $\mathcal{T}$, i.e., $\forall r_i, r_{i+1} \in R, r_i, r_{i+1} \in E$ and $r_i.e = r_{i+1}.s$.

Definition 3.4 (Routing Decision). A routing decision $a_i = (r_u \rightarrow r_v) \in E \times E$, representing the transition from road r_u to r_v, where $r_u.e = r_v.s$. We denote the complete set of routing decisions as A. Note that given an edge trajectory $R = \{r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_M\}$, it can also be represented by the series of routing decisions, i.e., $R = \{a_1 \rightarrow a_2 \rightarrow \cdots a_{M-1}\}$, where $a_i = (r_i \rightarrow r_{i+1})$. In the following sections, we will use both notations, e.g., "$a \in R$" and "$r \in R$".

Definition 3.5 (Outlier Trajectory). Given a SD pair $\langle r_s, r_d \rangle$, a trajectory R is an outlier if it rarely occurs and is different from other trajectories w.r.t. $\langle r_s, r_d \rangle$.

4 DB-TOD

In this section, we present our Driving Behavior-based Trajectory Outlier Detection approach, in short DB-TOD. Inspired by the probabilistic/statistic model utilized by point object outlier detection [1], we want to detect the trajectory outlier in a probabilistic way as well. In other words, we plan to build a probabilistic model to model the distribution of trajectory data and to return the trajectories with low probability as outliers. To be more specific, given a SD-pair $\langle r_s, r_d \rangle$, we report a trajectory R as an outlier if $P(R|r_s, r_d) < \xi_{sd}$, where ξ_{sd} is a pre-defined threshold.

Leveraging the probabilistic model to detect outliers is promising because the model can be regarded as a compressed representation and a summarization of the historical data. Thus, for a new trajectory, it only needs to go through the probabilistic model to get the likelihood without the need of scanning the historical trajectories, which can help boost the detection speed/throughput and hence address the efficiency issue. Moreover, modeling the driving behavior provides a better way to understand the vehicle trajectories and the cause of a trajectory outlier.

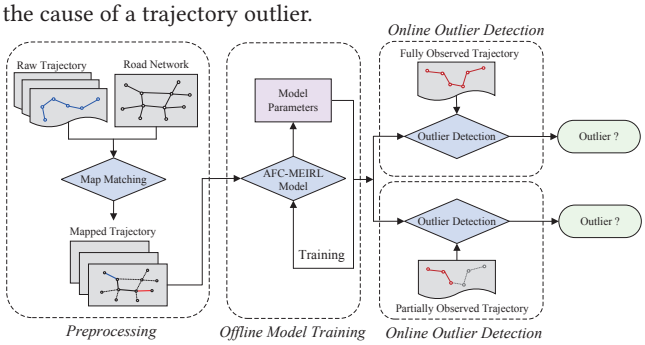


Figure 2: The system overview of DB-TOD.

Fig. 2 summarizes the overview of DB-TOD. The *preprocessing* phase is processed offline to generate the mapped trajectories as defined in Definition 3.3 and to gather the trajectories w.r.t. the same SD-pair. The *offline model training* phase, serving as the key part of our system, adopts a model called AFC-MEIRL which is a probabilistic model for modeling the vehicle trajectories via exploiting the driving preference as discussed above. The model parameters are trained offline based on the historical trajectories obtained in the preprocessing phase. After the model parameters are inferred, two *online outlier detection* tasks are performed. The first task is to perform traditional outlier detection, i.e., judging whether a fully observed trajectory is an outlier; while the second task is designed to issue an alert on the potential outlier as early as possible when the trajectory has not yet reached the destination and only a partial trajectory has been observed.

4.1 Modeling Driving Behavior with MEIRL

Unlike the modeling the distribution of traditional point data, the trajectories are sequential data with topological constraints which we decided to adopt the the *maximum entropy inverse reinforcement learning* (MEIRL) model [26]. It focuses on modeling sequential decision policies via a set of historical action trajectories.

In reinforcement learning/sequential decision theory, there are three key elements, i.e., state ς, action α and reward ρ. State $\varsigma \in \mathbb{S}$ represents the state of an agent. Action $\alpha \in \mathbb{A}$ refers to a mapping

from $\mathbb{S}$ to $\mathbb{S}$. Action $\alpha_1 = \varsigma_1 \rightarrow \varsigma_2$ represents that if an agent is in state ς_1, after performing the action α_1, the agent will be in state ς_2.[1] Reward ρ refers to the reward that will be received after performing an action, which is often a function of states or actions. These three elements enable the modeling of the decision process, i.e., an agent in a certain state needs to decide which action to perform by considering the reward it will gain, and it will be in a new state after performing certain action. It continues doing the decision which transits the agent from one state to another state until the terminal state is reached.

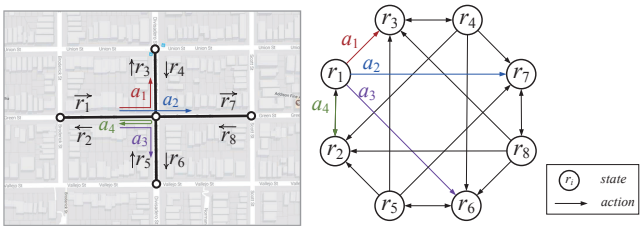
To adopt the MEIRL on modeling the vehicle trajectory for detecting vehicle trajectory outliers, we can regard the road segments as the states, regard the routing decision at a certain cross road (e.g., turning left, turning right or moving straight forward) as an action, and regard drivers as agents. Each driver moves along a road segment, makes a routing decision at the crossroad, and then moves along another road segment. She/he continues this process until the destination (i.e., the terminal road segment) has been reached.

Fig. 3(a) shows a part of road network in the real world. There are 8 road segments $r_1, r_2, \cdots, r_8$, forming 4 bi-directional roads. We can model the decision process by transforming the road network to its dual graph as shown in Fig. 3(b). Then we can regard the state ς as the road segment r_i and the transition between adjacent road segments, i.e., the routing decision a_i, as the action α. For the driving scenario, when someone making a decision, there always has a certain routing cost in his/her mind, e.g., the length of the next road, the turning angle, the road level, etc. These routing cost is hard to quantized explicitly, so we regard them as the *latent cost* w.r.t. an action. In such a way, we then can regard the *negative* latent cost as the reward since the lower the cost, the better, which is equivalent to a higher reward. Given a complete vehicle trajectory $R = \{r_a \rightarrow r_b \rightarrow \cdots\}$, it can be represented as an action series continuously performed by the driver with the reward continuously collected respectively. According to MEIRL [26], it models the action series obeying the maximum entropy principle, i.e.,

$$P(R|sd) = \frac{\exp\left(-cost(R)\right)}{\sum_{R' \in \mathfrak{R}_{sd}} \exp(-cost(R'))} = \frac{1}{Z_{sd}} \exp\left(-cost(R)\right) \quad (1)$$

Here, $\mathfrak{R}_{sd}$ captures *all possible* trajectories corresponding to SD-pair $\langle r_s, r_d \rangle$. Z_{sd} is the normalization coefficient to make sure it is a probability. $cost(R)$ is a function modeling the latent costs gathered according to the actions performed in the trajectory R, and $-cost(R)$ refers to the reward collected by the trajectory. We can infer that the maximum entropy principle tends to assign a very high probability to a trajectory with low latent cost and the model will equally prefer trips with similar costs. Considering a trajectory outlier which has the "few" and "different" characteristics, as discussed in the introduction section, this model implies that the "difference" between the outlier and other normal trajectories, which results in the larger latent cost, in turn has a very low probability according to Eq. (1), reflected by the "few" property. Note that in the following sections, we will use the terms "state" and "road segment" interchangeable, and terms "action" and "routing decision" interchangeably.

[1]Note that we here only discuss the deterministic case, i.e., the state transition w.r.t. a given action is deterministic.



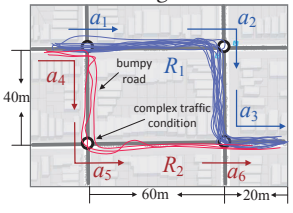
(a) The example of the road network (b) The dual graph representation

Figure 3: Example showing how to model the routing decision with decision process.

According to Eq. (1), we can find that the latent cost function is still undetermined. The cost function can be assumed to be a linear combination of the explicit features of the actions, e.g., $\mathbf{f}_a \in \mathbb{R}^d$, by some latent cost weights $\boldsymbol{\theta} \in \mathbb{R}^d$, where d refers to the dimension of the explicit features. The explicit features can be obtained from the road network such as the type of turning (e.g., hard-turn, soft-turn or reverse-turn), the road type of the new state (new road segment) that the agent will be in after performing this action and the speed-limit, etc. Then, the latent cost of an action is the weighted sum of features i.e., $\boldsymbol{\theta}^\top \mathbf{f}_a$. For the detail of what features we have used, please refer to Table 2 in Section 5.6.

Moreover, for an action trajectory $R = \{a_1 \rightarrow a_2 \cdots\}$, the aggregated latent cost of R is modeled as $cost(R) = \sum_{a \in R} \boldsymbol{\theta}^\top \mathbf{f}_a = \theta^\top \mathbf{f}_R$, where $\mathbf{f}_R = \sum_{a \in R} \mathbf{f}_a$ is the *feature counts*, the sum of the state features along the path R. The purpose of MEIRL is to learn the latent cost weights $\boldsymbol{\theta}$ from some trajectories demonstrated by some experts (e.g., experienced taxi drivers). Finally, we can compute the likelihood of any trajectory w.r.t. an SD-pair according to Eq. (1).

Discussion. Although MEIRL seems to be already able to handle the task of modeling driving behavior, it still has room for improvement. Considering trajectories R_1 and R_2 with similar feature counts in Fig. 4. According to MEIRL, these two trajectories will definitely be assigned the same probabilities since the latent cost is related to the feature counts of a trajectory. However, in the dataset, the occurrence of R_1 might be actually much larger than that of R_2, which means more drivers prefer R_1 than R_2. This may be caused by other factors that cannot be explicitly encoded into the feature vector, e.g., the roads in R_2 may be bumpy or the traffic condition in the crossroads in R_2 may be more complex. The reason that MEIRL is not able to model such tiny differences is that the likelihood of a trajectory only depends on the explicit features which might not be sufficient to correctly describe an action without a failure. This example demonstrates some missing factors in the explicit features while MEIRL is not capable of capturing such factors. Thus, in the next section, we will enhance and extend the MEIRL model to tackle this shortcoming.



Features	$\mathbf{f}_{a_1}$	$\mathbf{f}_{a_2}$	$\mathbf{f}_{a_3}$	$\mathbf{f}_{R_1}$	$\mathbf{f}_{a_4}$	$\mathbf{f}_{a_5}$	$\mathbf{f}_{a_6}$	$\mathbf{f}_{R_2}$
Straight	1	0	0	1	0	0	1	1
Left turn	0	0	1	1	0	1	0	1
Right turn	0	1	0	1	1	0	0	1
U turn	0	0	0	0	0	0	0	0
level: Primary	60	0	20	80	0	60	20	80
level: Secondary	0	40	0	40	40	0	0	40

Figure 4: Example showing the failure of modeling such difference in MEIRL model

4.2 MEIRL with Automatic Feature Correction

4.2.1 Model Introduction. As discussed above, the explicit features might not be sufficient to correctly describe a certain action and MEIRL is not able to capture such factors. We name such factors as *latent feature bias*, something that cannot be easily observed but will affect the routing decisions. Thus, we extend the original MEIRL by the idea of *correcting the features automatically*, yielding the name *Automatic Feature Correction MEIRL* (AFC-MEIRL). In detail, we introduce a learnable feature bias $\Delta_a \in \mathbb{R}$ into the features of each action. Consequently, for an explicit feature of an action a, i.e., $\mathbf{f}_a = \left[f_a^{(1)}, f_a^{(2)}, \cdots, f_a^{(d)}\right]^\top$, the new feature vector $\mathbf{f}_a^+$ with feature correction can be represented as

$$\mathbf{f}_a^+ = \left[f_a^{(1)}, f_a^{(2)}, \cdots, f_a^{(d)}, \Delta_a\right]^\top$$

Then, it gives the model the flexibility to correct the feature by adjusting the additional feature bias Δ_a according to the historical trajectories to allow the model to fit the data better. Again taking Fig. 4 as an example. We assume the road segment (the one if performing a_4) is very bumpy and the crossroad condition is very complex. AFC-MEIRL, after observing the fact that the number of occurrences of action a_4 is very small from the historical dataset, will automatically increase Δ_{a_4} to increase the latent cost and fit the data more reasonably.

4.2.2 Parameter Learning. In this section, we will introduce how to learn the parameter from a set of historical trajectories which can be regarded as the action series demonstrated by experts. We maximize the likelihood of the set of historical trajectories denoted by $\mathcal{R}$ under AFC-MEIRL model which is equivalent to minimize the negative log-likelihood, i.e.,

$$\boldsymbol{\theta}, \Delta = \arg\min_{\boldsymbol{\theta},\Delta} \ell(\mathcal{R}|\boldsymbol{\theta}, \Delta) = \arg\min_{\boldsymbol{\theta},\Delta} -\sum_{R\in\mathcal{R}} \log P(R|\boldsymbol{\theta}, \Delta) \tag{2}$$

$$= -\sum_{R\in\mathcal{R}} \log \frac{\exp\left[-\left(\boldsymbol{\theta}^\top \sum_{a\in R}\mathbf{f}_a + \sum_{a\in R}\Delta_a\right)\right]}{\sum_{R'\in\mathfrak{R}} \exp\left[-\left(\boldsymbol{\theta}^\top \sum_{a'\in R'}\mathbf{f}_{a'} + \sum_{a'\in R}\Delta_{a'}\right)\right]} \tag{3}$$

where $\Delta = \left[\Delta_{a_1}, \Delta_{a_2}, \cdots, \Delta_{a_{|\mathbb{A}|}}\right]^\top$. Moreover, we will have the following convexity theorem.

Theorem 4.1. *The negative likelihood loss function $\ell(\mathcal{R}|\boldsymbol{\theta}, \Delta)$ of AFC-MEIRL model with summarized feature bias, i.e., Eq. (3), is convex w.r.t. $\boldsymbol{\theta}$ and Δ.*

Proof. Constructing an auxiliary vector, w.r.t. a trajectory R, $\mathbf{h}_R \in \mathbb{R}^{|\mathbb{A}|}$ where $\mathbf{h}_R[i] = 0$ if $a_i \notin R$ and $\mathbf{h}_R[i] = 1$ if $a_i \in R$. Denoting $\boldsymbol{\chi} = \begin{bmatrix}\boldsymbol{\theta}\\ \Delta\end{bmatrix}$ by concatenating the parameters $\boldsymbol{\theta}$ and Δ to be one vector and $\mathbf{F}_R = \begin{bmatrix}\mathbf{f}_R\\ \mathbf{h}_R\end{bmatrix}$. The latent cost of the trajectory R, i.e., $cost(R) = \boldsymbol{\theta}^\top \sum_{a\in R}\mathbf{f}_a + \sum_{a\in R}\Delta_a$, can be rewritten as $cost(R) = \boldsymbol{\chi}^\top \mathbf{F}_R$. Given any $\boldsymbol{\chi} \in \mathbf{dom}\ \ell$, we denote the unnormalized probability of R w.r.t. $\boldsymbol{\chi}$ as $\tilde{P}_{\boldsymbol{\chi}}(R) = \exp(-\boldsymbol{\chi}^\top \mathbf{F}_R)$. Accordingly, the loss function of a trajectory R w.r.t. $\boldsymbol{\chi} \in \mathbf{dom}\ \ell$ can be represented as $\ell(R|\boldsymbol{\chi}) = \boldsymbol{\chi}^\top \mathbf{F}_R + \log \sum_{R'\in\mathfrak{R}} \tilde{P}_{\boldsymbol{\chi}}(R')$. The gradient can be derived by $\nabla\ell(R|\boldsymbol{\chi}) = \mathbf{F}_R - \sum_{R'} P_{\boldsymbol{\chi}}(R')\mathbf{F}_{R'}$. $\forall \mathbf{x}, \mathbf{y} \in \mathbf{dom}\ \ell$, according to Jensen's inequality,

$$\log \frac{\sum_{R'} \tilde{P}_{\mathbf{y}}(R')}{\sum_{R'} \tilde{P}_{\mathbf{x}}(R')} = \log \frac{\sum_{R'} \left(\tilde{P}_{\mathbf{x}}(R') \cdot \frac{\tilde{P}_{\mathbf{y}}(R')}{\tilde{P}_{\mathbf{x}}(R')}\right)}{\sum_{R'} \tilde{P}_{\mathbf{x}}(R')} \geq \frac{\sum_{R'} \left(\tilde{P}_{\mathbf{x}}(R') \cdot \mathbf{F}_{R'}^\top(\mathbf{x}-\mathbf{y})\right)}{\sum_{R'} \tilde{P}_{\mathbf{x}}(R')}$$

Thus we have

$$\log \sum_{R'} \tilde{P}_{\mathbf{y}}(R') \geq -\sum_{R'} \log \tilde{P}_{\mathbf{x}}(R') - \sum_{R'} P_{\mathbf{x}}(R')\mathbf{F}_{R'}^\top(\mathbf{y}-\mathbf{x})$$

Appending $\mathbf{y}^\top\mathbf{F}_R$ to both sides, we can get

$$\begin{aligned}
&\underline{\ell(R|\mathbf{y})} = \mathbf{y}^\top\mathbf{F}_R + \log \sum_{R'} \tilde{P}_{\mathbf{y}}(R')\\
&\geq \left(\mathbf{x}^\top\mathbf{F}_R + \sum_{R'} \log \tilde{P}_{\mathbf{x}}(R')\right) + \left(\mathbf{y}^\top\mathbf{F}_R - \mathbf{x}^\top\mathbf{F}_R - \sum_{R'} P_{\mathbf{x}}(R')\mathbf{F}_{R'}^\top(\mathbf{y}-\mathbf{x})\right)\\
&= \ell(R|\mathbf{x}) + \left(\mathbf{F}_R - \sum_{R'} P_{\mathbf{x}}(R')\mathbf{F}_{R'}\right)^\top (\mathbf{y}-\mathbf{x}) = \underline{\ell(R|\mathbf{x}) + \nabla\ell(R|\mathbf{x})^\top(\mathbf{y}-\mathbf{x})}
\end{aligned}$$

which holds the first-order condition of convexity [2], i.e., $\ell(R|\boldsymbol{\chi})$ is convex. Since $\ell(\mathcal{R}) = \sum_{R\in\mathcal{R}} \ell(R)$, we can get $\ell(\mathcal{R})$ is also convex. □

Theorem 4.1 shows a desirable property of the AFC-MEIRL model, i.e, we can achieve the global optima by optimizing the loss function. Similar as [26], we adopt the exponential stochastic gradient descent algorithm to train the model. Note that the trajectory set is composed of many trajectories corresponding to different SD-pairs. To reduce the computation cost, we first partition the whole trajectory dataset $\mathcal{R}$ according to SD-pairs into several clusters sufficing that the trajectories in each cluster correspond to the same SD-pair. Given a historical trajectory $R \in \mathcal{R}_{sd}$ w.r.t. an SD-pair $\langle r_s, r_d\rangle$, the gradient of the parameter $\boldsymbol{\theta}$ and the summarized feature biases $\boldsymbol{\Delta}$ are derived as follows,

$$\frac{\partial\ell}{\partial\boldsymbol{\theta}} = \mathbf{f}_R - \sum_{R'\in\mathfrak{R}_{sd}} P(R')\cdot\mathbf{f}_{R'} = \mathbf{f}_R - \sum_{a\in\mathbb{A}} \tilde{D}_a\mathbf{f}_a \tag{4}$$

$$\frac{\partial\ell}{\partial\Delta_a} = \sum_{a\in R} \mathbf{1}\{a\in R\} - \sum_{R'\in\mathfrak{R}_{sd}} P(R')\cdot\mathbf{1}\{a\in R'\} = D_a - \tilde{D}_a \tag{5}$$

where $\tilde{D}_a$ is the expected action performing frequencies computed from the AFC-MEIRL model w.r.t. action a and $\mathbf{1}\{conditon\}$ is an indicator function returning 1 when *condition* is true and 0 otherwise. Please refer to Algorithm 1 for the details of the algorithm.

Discussion. We can see that at the minima, i.e., the gradient equals 0, the expected feature counts estimated by the model ($\sum_{a\in\mathbb{A}} \tilde{D}_a\mathbf{f}_a$) match the true observed feature counts of the training trajectory ($\mathbf{f}_R$). Meanwhile, for each action a, its expected action performing frequencies $\tilde{D}_a$ will match the observed action performing frequency D_a. In other words, AFC-MEIRL tries to model the distribution of the dataset by approximating the expected feature counts to the real feature counts collected from the dataset and approximating the performing frequency of each action. Note that for traditional MEIRL, the gradient only includes $\frac{\partial\ell}{\partial\boldsymbol{\theta}}$ [26], hence MEIRL only allows the model to approximate the feature counts of a trajectory while matching between the expected action performing frequencies and the true ones is not considered. This explains the reason that MEIRL loses its generality on unobserved trajectory.

Computing $\tilde{\mathbf{D}}_\mathbf{a}$. From Eq. (4) and Eq. (5), we can see that the gradient computation requires the computation on expected action performing frequency $\tilde{D}_a$. Directly enumerating all possible trajectories corresponding to a given SD-pair $\langle r_s, r_d\rangle$ is impractical for its exponential time complexity. We solve this problem by dynamic programming. Let $Z_{si}^{(=t)}$ be the summation of all exponential negative costs of all trajectories w.r.t. a given SD-pair $\langle r_s, r_d\rangle$ in

exactly t steps, i.e., $Z_{si}^{(=t)} = \sum_{R \in \mathfrak{R}_{si} \& |R|=t} \exp(-cost(R))$. In addition, let $Z_{jd}^{(\leq t)}$ refer to the ones w.r.t. a given SD-pair $\langle r_s, r_d \rangle$ with *at most* t steps, i.e., $Z_{jd}^{(\leq t)} = \sum_{R \in \mathfrak{R}_{jd} \& |R| \leq t} \exp(-cost(R))$. Through dynamic programming which is actually similar to the famous forward-backward algorithm [18], we can derive the recursion equation as follows, where the computation of $Z_{si}^{(=t)}$ is conducted in the forward pass and $Z_{jd}^{(\leq t)}$ is computed in the backward pass.

$$Z_{si}^{(=t)} = \mathbf{1}\{r_i = r_s\} + \sum_{r_{i'} \in in(r_i)} Z_{si'}^{(=t-1)} \exp(-cost(a_{i' \to i}))$$

$$Z_{jd}^{(=t)} = \mathbf{1}\{r_j = r_d\} + \sum_{r_{j'} \in out(r_j)} Z_{j'd}^{(=t-1)} \exp(-cost(a_{j \to j'}))$$

$$Z_{jd}^{(\leq t)} = Z_{jd}^{(=t)} + Z_{jd}^{(\leq t-1)}$$

As the number of possible trajectories in $\mathfrak{R}_{sd}$ is infinite, we only consider those no longer than T steps, and thus we will get the approximated action performing frequencies as

$$\tilde{D}_{a_{i \to j}} = \frac{1}{Z_{sd}^{\leq T}} \sum_{t=1}^{T-1} \left(Z_{si}^{(=t)} \exp(-cost(a_{i \to j})) Z_{jd}^{(\leq T-t-1)} \right)$$

ALGORITHM 1: Parameter Learning for AFC-MEIRL

```
Randomly initialize θ and Δ, t ← 0;
while not convergence do
    for random R_sd do
        Compute D̃_a for each a ∈ 𝔸 ;
        for trajectory R ∈ R_sd do
            f_R ← Collect feature counts of each action;
            for action a ∈ 𝔸 do
                Compute ∂ℓ/∂Δ_a according to Eq. (5);
                Δ_a ← Δ_a exp(γ/t · ∂ℓ/∂Δ_a);
            Compute ∂ℓ/∂θ according to Eq. (4);
            θ ← θ exp(γ/t · ∂ℓ/∂θ), t ← t + 1;
return θ and Δ;
```

4.3 Outlier Detection with Full Observation

As mentioned in the beginning of Section 4, we label a trajectory R as an outlier if $P(R|sd) < \xi_{sd}$. Since $P(R|sd) = \frac{1}{Z_{sd}} \tilde{P}(R|sd)$, we can change the criteria of detecting the outlier to "judge whether the *unnormalized negative log-likelihood (U-NLL)*, i.e., $\tilde{\ell}(R) = -\log \tilde{P}(R|sd)$, is larger than a given threshold $\zeta_{sd} = \log(Z_{sd}\xi_{sd})$". According to Eq. (1), U-NLL of a trajectory R can be computed by

$$\tilde{\ell}(R) = \boldsymbol{\theta}^\top \sum_{a:(r_i \to r_{i+1}) \in R} \mathbf{f}_a + \sum_{a:(r_i \to r_{i+1}) \in R} \Delta_a \tag{6}$$

Hence, when the parameters of AFC-MEIRL model have been inferred from historical data, given a complete trajectory R which has been fully observed, the computation cost for computing Eq. (6) is $O(|R|)$ since we only need to sum up the features and the biases along the trajectories. Notice that ξ_{sd} is a pre-defined parameter which can be easily decided via the historical data and Z_{sd} can be cached when training the model. Hence, given a complete trajectory, our approach has the linear computation complexity w.r.t. the number of road segments passed by the trajectory and does not require any database access [2], which almost reaches the minimum complexity theoretically since at least we should spare $O(|R|)$ to read in the trajectory.

4.4 Outlier Detection with Partial Observation

As mentioned in Section 1, it's desirable for an outlier detection system to be able to identify a potential outlier when the trajectory is only partially observed. More formally, suppose we have known the destination the driver wants to go (since the outlier is defined w.r.t. an SD-pair), and the partial trajectory R_{st} from r_s to r_t (but not yet r_d) has been observed, we want to judge whether R_{st} is a potential outlier or not, corresponding to $\langle r_s, r_d \rangle$. Note that with the elapse of the time, more proportion of the trajectory is revealed and we want an outlier detection approach to report the outlier as early as possible.

To achieve such a goal, we first introduce the minimum U-NLL for all the sub-trajectories corresponding to $\langle r_t, r_d \rangle$ (where r_t refers to the current state of the partial trajectory), i.e., $\tilde{\ell}(R_{td}^*) = \min_{R' \in \mathfrak{R}_{td}} \tilde{\ell}(R')$. Thanks to the summation property of U-NLL w.r.t. feature counts of actions according to Eq. (6), the $\tilde{\ell}(R_{td}^*)$ can be computed by constructing the dual graph G' of the original road network G according to Fig. 4, and assign each edge (i.e., routing decision a) the weight of $\boldsymbol{\theta}^\top \mathbf{f}_a + \Delta_a$. Leveraging shortest path algorithms (e.g. dijkstra algorithm), the value can be easily achieved. Realizing the fact that

$$\begin{aligned}\tilde{\ell}(R_{sd}) &= \boldsymbol{\theta}^\top \left(\sum_{a \in R_{st}} \mathbf{f}_a + \sum_{a' \in R_{td}} \mathbf{f}_{a'} \right) + \left(\sum_{a \in R_{st}} \Delta_{a'} + \sum_{a' \in R_{td}} \Delta_{a'} \right) \\ &= \tilde{\ell}(R_{st}) + \tilde{\ell}(R_{td}) \geq \tilde{\ell}(R_{st}) + \tilde{\ell}(R_{td}^*) = LB(\ell(R_{sd}), r_t)\end{aligned}$$

we can get the lower bound of the U-NLL, i.e., $LB(\ell(R_{sd}), r_t)$, of the complete trajectory R_{sd} when it is observed till r_t. Thus, when $LB(\ell(R_{sd}), r_t)$ exceeds the outlier threshold ζ_{sd}, it can ensure that the U-NLL of the complete trajectory will definitely exceed ζ_{sd}, thus an outer alert will be reported by the algorithm.

Note that if we run the shortest path algorithm to compute $\tilde{\ell}(R_{td}^*)$ whenever a partial trajectory moves to a new road segment, it will be time consuming when the action size is very large. Here, we introduce a strategy to reduce the computation cost. Since the destination is fixed when judging a certain trajectory, we can run a single-sink shortest path algorithm once to locate the shortest path from all nodes to the destination, at the beginning of the trajectory. Then, $\tilde{\ell}(R_{td}^*)$ can be easily obtained by looking up from the pre-computed results. Note that there is no need to compute $\tilde{\ell}(R_{td}^*)$ for all states. When we run the single-sink shortest path algorithm, we should in advance stop the algorithm when the current smallest U-NLL popped by the heap is already larger than ζ_{sd}.

4.5 Complexity analysis

For the training phase, computing the $\tilde{D}(a)$ for all actions takes $O(AT)$ time. The complexity for learning parameters by seeing each sample once is $O(ATP + \sum_{R \in \mathcal{R}} |R|)$, where the former $O(ATP)$ includes the computation of $\tilde{D}(a)$ w.r.t. all SD-pairs (i.e., P refers to the number of different SD-pairs in the training set) and the latter

[2] If the threshold ζ_{sd} is stored in the database, one database access is required to get the value of ζ_{sd}.

$O(\sum_{R\in\mathcal{R}}|R|)$ is the cost for collecting feature counts of each trajectory. For the online outlier detection with full observation, as analyzed in Section 4.3, the complexity is only $O(|R|)$ which the number of edges $|R|$ is often a small number (smaller than 100). For the case with partial observation, when a new edge is observed, we should compute the $\tilde{\ell}(R^*_{td})$ which introduces a single-sink shortest path algorithm whose complexity depends on the shortest path algorithm, e.g., $O(E+V\log V)$ for the dijkstra algorithm with a Fibonacci heap. Note that actual time cost is far smaller than $O(E+V\log V)$ since it can be in advance stopped as introduced in Section 4.4.

5 EXPERIMENT

Dataset. We conduct our experimental study on two real-world taxi trajectory datasets. The first one is generated by 442 taxis in Porto from Jan. 07, 2013 to Jun. 30, 2014 and the average sampling rate is 15s/point. The other one is collected in Shanghai from Apr. 01 to Apr. 10 in 2015 by 13,650 taxis with the sampling rate at 10s/point. We select the area with the highest trajectory density to ensure there are enough trajectories w.r.t. SD-pairs to conduct the experiment. The Porto dataset contains 486,268 trajectories while Shanghai dataset contains 757,032 trajectories. Notice that we only extract the trips occupied by passengers.

Ground Truth. Existing works tend to label the outlier *manually*. However, we claim that this may not be an ideal solution, because i) it is hard to manually label a large amount of data thus the testing samples may not be sufficient to show the average performance of an algorithm; ii) manual labeling requires domain expert, i.e., experienced drivers who are very familiar with the road network, which disqualifies many volunteers; and iii) the label might not be reliable if we can not find sufficient volunteers. As a solution, our experiment requires *zero* manual labeling. We leverage the drivers from the dataset itself to be the "volunteers". To be more specific, we partition the dataset according to SD-pairs, and we pick up those SD-pairs with sufficient historical trajectories, i.e., $|\mathcal{R}_{sd}| > N_{support}$. For the trajectories in $\mathcal{R}_{sd}$, we adopt complete-linkage clustering algorithm to hierarchically cluster the trajectories with similar paths. The distance metric of two mapped trajectories R_1 and R_2 is defined as the length-weighted Jaccard similarity, i.e., $1-\frac{len(R_1\cap R_2)}{len(R_1\cup R_2)}$. Note that if the maximum distances of each pair of clusters all exceed a threshold η, the clustering algorithm terminates which ensures the distance between every two trajectories in the same cluster is bounded by η. We name each cluster formed by trajectories with similar paths a *pattern* P. Note that outliers are "few" and "different". Since we have clustered the trajectories, trajectories belonging to different patterns are expected to be different. Hence, the ratio $\frac{|P|}{|R_{sd}|}$ can be regarded as the popularity of a pattern, and all the patterns with the ratio smaller than a given threshold ψ will be labeled as outliers. We empirically set $N_{support}$ to 100, η to 0.2 and ψ to 3% by visualizing several testing samples and judging whether this setting is reasonable or not. Under such settings, we obtain 114 valid SD-pairs and in total 20, 920 trajectories from Porto dataset for testing, where 5% trajectories are reported as outliers. In Shanghai dataset 88 SD-pairs and 14, 740 trajectories are extracted for testing. About 7% trajectories appear to be outliers in the dataset.

Baselines. We include two typical works, i.e., TPRRO [4] and iBOAT [25], as the competitors. To our best knowledge, they are the start-of-the-art trajectory outlier detecting approaches specifically designed for *vehicle trajectories* which also consider efficiency. TPRRO summaries the historical trajectories w.r.t. an SD-pair by k most popular ones and computes the outlier score by calculating the edit distances from the querying trajectory to these k trajectories. iBOAT maps the trajectory to sequence of grids and detects the outliers by how the querying trajectory, in the form of grid-sequence, differs from the patterns of historical grid sequences. Note iBOAT supports the mapping from raw trajectory to any discrete sequences; while DB-TOD and TPRRO both take the edge trajectories as input. For fairness, we let iBOAT in our implementation also take in edge trajectories. Thus, we rename original iBOAT that takes in grid trajectories as iBOAT-grid and the one using the edge trajectories as iBOAT-edge. All the parameters of different models have been tuned to achieve the best performance.

5.1 Ranking Evaluation

Outlier detection algorithms rely on the outlier scores derived by the algorithms. They use a threshold to "give a cut" to the ranked trajectories by reporting the half with outlier score exceeding the threshold as outliers and the remaining half as the normalcies. Our first set of experiments evaluates the ranking accuracy of different algorithms. Note that if the ranking is accurate, we have the flexibility to set thresholds to cater for different application scenarios.

Note we have clustered trajectories into patterns and the outlier detecting approach is expected to rank these patterns according to their outlier scores. The ground truth order is ranked by their pattern frequencies in the ascending order. We define $rank^*(i)$ to be the ith pattern in the ground truth order, i.e., $|rank^*(1)| \le |rank^*(2)| \le \cdots \le |rank^*(k)|$. For example, suppose there are four patterns generated within a given SD-pair, $|P_1| = 40$, $|P_2| = 1$, $|P_3| = 80$, and $|P_4| = 3$. Accordingly, $rank^*(1) = P_2$, $rank^*(2) = P_4$, $rank^*(3) = P_1$ and $rank^*(4) = P_3$. In addition, $rank(i)$ refers to the ith pattern in the order reported by an outlier detection approach.

In information retrieval, the discounted cumulative gain (DCG) is often used to measure the effectiveness of a ranking algorithm w.r.t. the relevance [14], with $DCG = \sum_{i=1}^{k}\frac{2^{rel(rank(i))}-1}{\log_2(i+1)}$. Here, $rel(rank(k))$ is a relevance scoring function defined in $[0,1]$ representing the degree of relevance of the object $rank(i)$. In the case of outlier detection, the larger the degree of an outlier, the larger the value of relevance. Thus, we define it to be $rel(rank(i)) = 1-\frac{\log|rank(j)|}{\log\sum_{j=1}^{k}|rank^*(j)|}$. If the pattern has fewer occurrences, the rel will be larger, meaning the relevance to an outlier is large. From DCG we can see that when a normal trajectory is ranked in the front of the list, the denominator $log_2(i+1)$ will be large and the relevance will be small, i.e., it will penalize the situation of wrongly ranking a normal trajectory as an outlier, which satisfies our need. By denoting the DCG value of the reverse order of the ground truth as $wDCG$ (worst DCG) and the one w.r.t. the order of ground truth as $iDCG$ (ideal DCG), we can get the normalized DCG value as $NDCG = \frac{DCG-wDCG}{iDCG-wDCG} \in [0,1]$. The NDCG value achieves 1 if the order sorted by the algorithm is exactly the same as the ground truth and reaches 0 if the order is the reverse of the ground truth.

Table 1 lists the NDCG values of all approaches. "Random" refers to the NDCG value under a random order. First, we can find in

Table 1: The result of NDCG

	DB-TOD	iBOAT-grid	iBOAT-edge	TPRRO	Random
Porto	**0.953**	0.879	0.899	0.851	0.697
Shanghai	**0.903**	0.816	0.877	0.875	0.718

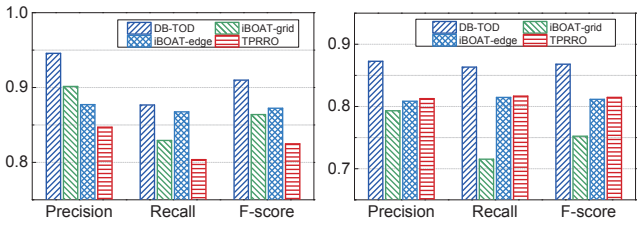
both datasets, iBOAT-edge outperforms iBOAT-grid. This is because when a trajectory is transferred into a grid sequence, it loses some information and the grid representation is fuzzy and coarse-grained. On the other hand, edge representations do not have such problems since each edge passed by a trajectory is accurate. Next, TPRRO is slightly worse than iBOAT-edge. This might be because that TPRRO is not a historical database-based approach. Thus, it should summarize the information of historical trajectory data. Although non-database based approaches are more efficient (like DB-TOD), they have to bear the risk of not summarizing the historical information probably. If the information loss is significant, we have to balance the trade-off between effectiveness and efficiency. Here, TPRRO obviously sacrifices its effectiveness for efficiency, which will be analyzed further in Section 5.3. Surprisingly, the random sorting achieves a not too low value. We explain the reason is that, given an SD-pair, after clustering corresponding trajectories to patterns, there will be some outlier patterns sharing the same count (such as only 1 trajectory passed by in the historical dataset). Therefore, randomly sorting these patterns with the same historical frequencies will not reduce the NDCG. Last but not least, DB-TOD achieves the highest NDCG score, indicating it can more accurately rank the trajectories w.r.t. their popularity than other competitors, which justifies the effectiveness of modeling the driving behavior.

5.2 Outlier Detection with Complete Trajectories

The second set of experiments evaluates the performance of detecting the outliers of fully observed trajectories. We collect the counts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN). TP refers to the number of outlier trajectories reported as outliers, FP refers to the number of normal trajectories that are reported as outliers, TN refers to the number of outlier trajectories reported as normal, and FN refers to the number of normal trajectories reported as normal. Then, we report the common metrics precision p, recall γ, and F-score F, with $p = \frac{TP}{TP+FP}$, $\gamma = \frac{TP}{TP+FN}$, and $F = \frac{2 \cdot p \cdot \gamma}{p+\gamma}$. The thresholds to separate normalcies and outliers for outlier detection approaches are selected to have the best F-score performance. The results are shown in Fig. 5. We can find in both datasets, DB-TOD outperforms the competitors under all metrics. This is because, as compared with other approaches, DB-TOD incurs a much higher TP but a much lower FP, which further demonstrates the effectiveness of our approach. iBOAT-edge performs slightly better than iBOAT-grid, remaining a similar trend as reported in previous ranking evaluation. Moreover, TPRRO is inferior to iBOAT-grid in Porto dataset and superior to iBOAT-grid in Shanghai dataset, which can be also inferred from the performance of ranking. This phenomena proves that the performance of outlier detection is strongly correlated to ranking accuracy.

5.3 Efficiency Evaluation

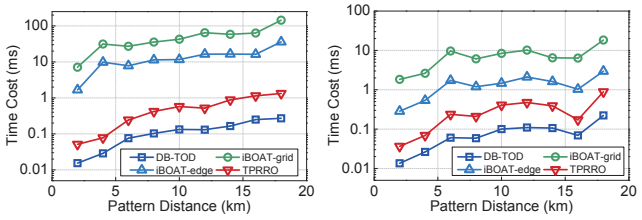
The third set of experiments evaluates the efficiency of detecting outliers. Fig. 6 plots the results w.r.t. SD-pairs of different distances. We can observe that two iBOAT approaches are much slower than DB-TOD and TPRRO. This is because although iBOAT has reduced



(a) Porto (b) Shanghai

Figure 5: Performance of detecting outliers with fully observed trajectories

the computation cost by inverted indexing, it still requires accesses to the historical data and checks how many trajectories containing the current windows being scanned. iBOAT-edge is several times faster than iBOAT-grid. The reason is that for each road segment, it may across several grids which makes the number of grids in a grid trajectory much larger than the number of edges in an edge trajectory. Accordingly, iBOAT-edge is much more efficient than iBOAT-grid. TPRRO is much faster than iBOAT as it requires zero access to *all* data w.r.t. the given SD-pair. It only needs to fetch k most popular trajectories in the historical dataset, and computes the edit distances between the querying trajectory and the popular trajectories online. However, even compared with TPRRO which summarizes the historical dataset via top-k typical trajectories, DB-TOD still runs about 3~5 times faster, demonstrate superior performance of our approach. Moreover, we can observe that with the increase of the distance of SD-pairs, the computation cost of all approaches increases. This is because as trajectories become longer, they include more edges/grids and more index accessing, edit distance computation as well as U-NLL computation are involved which increases the computation cost. Our approach is linear to the number of road segments passed by the trajectory which furthermore proves the capability of our approach on supporting the throughput of tremendous trajectory stream data.



(a) Porto (b) Shanghai

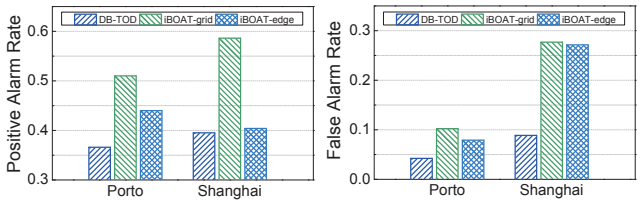
Figure 6: Detection time cost vs. distances of SD-pairs

5.4 Outlier Detection with Partial Trajectories

The fourth set of experiments evaluates the performance on detecting potential outliers when the trajectory R_{st} has not yet reached the destination (i.e., $r_t \neq r_d$). We simulate the process as follows. For a complete trajectory $R = \{r_1, r_2, \cdots, r_n\}$ w.r.t. $\langle r_1, r_n \rangle$, we set r_t to r_1 first, r_2 second, and so on until $r_t = r_d = r_n$. Assume $len(R)$ stands for the length of a trajectory R, and R_{st} stands for the partial trajectory that triggers an alarm issued by the respective detection algorithm. We split the trajectories into two disjoint sub-sets, $\mathcal{R}^o$ capturing all the outlier trajectories and $\mathcal{R}^n$ preserving all the normal ones. For trajectories in $\mathcal{R}^o$, we prefer R_{st} to be shorter for early warning, and hence report *positive alarm rate* $(= \frac{len(R_{st})}{len(R)})$ which means a smaller positive alarm rate indicates the ability to

alarm the outlier earlier; if an alarm is triggered only when the destination r_n is reached or no alarm is triggered at all, its positive alarm rate is set to 1, the worst value. For trajectories in $\mathcal{R}^n$, no alarm shall be issued at all, and hence we report *false alarm rate* $(= 1 - \frac{len(R_{st})}{len(R)})$; if no alarm is issued, its false alarm rate is set to 0. Fig. 7 reports the results. Note TPRRO is excluded from this set of experiments as its input must be complete trajectories.

For outlier trajectories, DB-TOD has the smallest positive alarm rate value, as shown in Fig. 7(a). It means DB-TOD can detect an outlier trajectory much earlier than two iBOAT-based approaches, mainly contributed by that fact that DB-TOD is able to model the road latent costs accurately. Between two iBOAT-based approaches, iBOAT-edge performs better because a grid may overlap with multiple road segments, including both popular ones and unpopular ones. Consequently, the pattern represented by grids is less accurate than that represented by edges. For normal trajectories, DB-TOD again performs the best as it has the smallest false alarm rate.



(a) True alarm rate vs. outliers (b) False alarm rate vs. normalcies

Figure 7: Performance of detecting potential outliers of partially observed trajectories (the lower the values, the better)

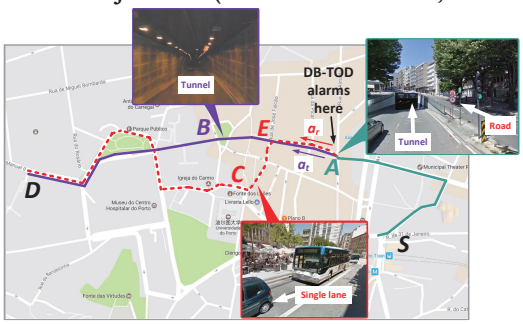
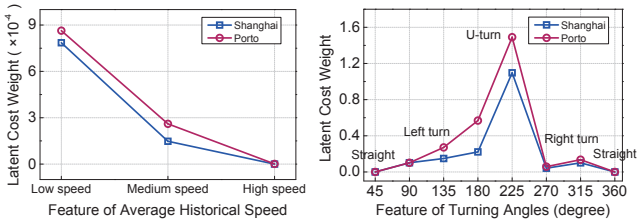


Figure 8: Case study. The trajectory in red dash line is an outlier.

Moreover, we report an interesting case study in Fig. 8. Two trajectories w.r.t. $\langle S, D\rangle$ are plotted, where the solid line trajectory is a normal one that has occurred 107 times in the historical dataset, and the dash line trajectory is an outlier occurring only once. According to the road network, there are two routing decisions available in location A, i.e., driving into the tunnel or continuing moving on the road, denoted as actions a_t and a_r respectively. Suppose the partial trajectory R_{SA} from S to A is known, as plotted in Fig. 8. DB-TOD estimates the lower bound of the whole trajectory to be 13.17, smaller than the preset threshold $\zeta_{SD} = 14.625$. If the driver performs action a_r, the lower bound of the partial trajectory will be increased to 26.12 and an alarm will be issued. To explain why DB-TOD is able to issue an alarm early, we plot the street views w.r.t. several locations in Fig. 8. We can observe the solid line route from A to D via B is a very smooth route inside the tunnel, while the dash line route from A to D via E and C involves many turns and passes mainly uni-directional road segments with one narrow lane. Once the driver performs action a_r, she will miss the tunnel and be forced to move along a very uncomfortable route which in turn increases the lower bound of the U-NLL of the whole trajectory.

5.5 Visualization of Latent Cost Weights

Recall that in AFC-MEIRL model, the latent cost weights $\boldsymbol{\theta}$ will be learned from the historical data to model the preference of drivers w.r.t. different features. Fig. 9 plots the weights w.r.t. different speed features and turning angle features under two datasets. We can observe that with the increase of the historical average speed of a road, the weight largely reduces, which is consistent with our expectation as drivers tend to take the route on expressway or major roads with higher speed limit. Fig. 9(b) shows the weights w.r.t. the turning angle formed by two consecutive road segments. Assuming a turning is from r_1 to r_2, the turning angle is $\arccos(vec(r_1.s \rightarrow r_1.e) \cdot vec(r_2.s \rightarrow r_2.e))$. We can observe the weight is nearly 0 when the angle is smaller than 45° which can be reflected by the fact that drivers tend to go straight forward. With the increase of the turning angle, the weights also increase which is also reasonable, since it gradually moves from "going straight" to "turning left". The cost weight drastically increases when angle feature is between 180° and 225°, which implies that drivers are very reluctant to perform a U-turn. Afterwards, the weight again drops significantly because angle feature in the range of 270° ~ 315° indicates a right turn and 360° (i.e., 0°) again indicates moving straight forward with a weight close to 0. Note that the weights trained by AFC-MEIRL from two datasets are very similar, which is also consistent with common knowledge, as drivers in difference places do share similar driving preferences of turning and road speed. This phenomena implies that the latent cost weights $\boldsymbol{\theta}$ learned from one dataset provides a good initial value for the latent cost weights of another dataset to reduce the iterations of gradient decent algorithm.



(a) vs. average historical speed (b) vs. turning angles

Figure 9: The learned latent cost weights w.r.t. speed and angle features. A smaller weight indicates a lower cost and a higher preference.

5.6 Comparison Between MEIRL and AFC-MEIRL

Note that our AFC-MEIRL is extended from MEIRL by adding the new capability of modeling trajectories with unobserved features. In our last set of experiments, we study the performance of AFC-MEIRL and MEIRL on modeling trajectory. For a probabilistic model, the negative log likelihood (NLL) of the dataset assigned by the model is a common metric which is computed as Eq. (2). A smaller NLL means the model can assign a larger likelihood to the dataset which implies a better capability of modeling the data. Table 3 lists the result. We can observe that AFC-MEIRL has a much lower NLL than MEIRL, which demonstrates the superior capability of AFC-MEIRL for modeling trajectories.

Table 2: The features of the action $a : r_1 \rightarrow r_2$

	Historical Average Speed			Road Level								Turning Angle								Feature Bias
Feature	low	medium	high	0	1	2	3	4	5	6	7	45	90	135	180	215	270	315	360	Δ_a
Weight	9E-4	3E-4	1E-6	0.1	5E-3	4E-3	3E-3	2E-3	2E-3	7E-4	2E-7	2E-06	0.10	0.27	0.57	1.49	0.06	0.13	3E-6	1.0
Value	504.5	0	0	0	504.5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3.64

Table 3: The NLL of two models (the lower the better)

Dataset	Porto	Shanghai
AFC-MEIRL	**4.75**	**6.23**
MEIRL	7.84	7.28

Moreover, to further demonstrate the superior modeling capability of AFC-MEIRL model, we visualize a real case which is shown in Fig. 10. This case shows an action $a : r_1 \rightarrow r_2$ with the feature showing in the text box in the figure. Table 2 lists all the features used in our experiment. Specifically, the more major the road is, the higher the level will be, which can be obtained from the OpenStreetMap data. We also list the trained weights corresponding to each feature in the table. For the action a in Fig. 10, its corresponding feature vector can be constructed as the "value" row in Table 2. If we do not include the additional feature bias Δ_a, which is what the tradition MEIRL model does, its latent cost is $\boldsymbol{\theta}^\top \mathbf{f}_a = 3.22$. However, if we add the automatic feature correction mechanic, the AFC-MEIRL model learns the latent feature bias Δ_a as 3.68 which makes the final latent cost of action a being 6.9. From the figure we can find that if we perform the action, it will take a round to r_1 which is obviously a meaningless action. Moreover, the road of r_2 is also very narrow with many cars parking on both sides which makes people reluctant to drive on. However, the feature defined in Table 2 can not capture such information thus MEIRL fail to correctly model this situation. Supposing there exists another straight road r_3 connecting to r_1 with the same length, level, average speed and turning angle as r_2. By traditional MEIRL, the latent cost of the action $a' : r_1 \rightarrow r_3$ will be the same as a, which is obviously unreasonable as a' seems much natural than a. Fortunately, such information can be successfully captured through historical data by AFC-MEIRL model which further justifies its superiority.

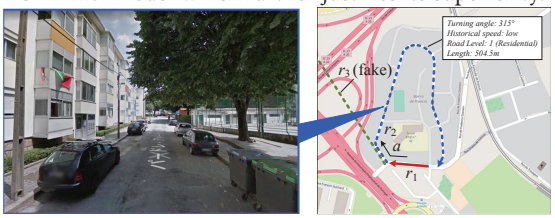


Figure 10: Case study. The left photo is the street view of the corresponding place in r_2.

6 CONCLUSIONS

In this paper, we propose DB-TOD, a probabilistic-model based vehicle trajectory outlier detection approach. It requires *zero* access to the database storing the historical trajectories when detecting outliers online. The detecting complexity is linear to the number of road segments passed by a trajectory which guarantees the efficiency of our algorithm. We model the driving behavior by AFC-MEIRL model which is extended from MEIRL model. It models the latent cost of routing decision via the explicit feature counts and the latent feature biases. Our model has the convex objective function which ensures the global optima can be achieved. We conduct comprehensive experiments using two real datasets. The results show that DB-TOD significantly outperforms the start-of-art approaches on both effectiveness and efficiency.

ACKNOWLEDGMENTS

This research is supported in part by National Natural Science Foundation of China (NSFC) under grant 61772138 and the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative. Special thanks to Jiangyun Mao (jymao14@fudan.edu.cn), Ziyang Chen (ziyangchen13@fudan.edu.cn) and Qize Jiang (jiangqz14@fudan.edu.cn) for the hard coding work of this paper.

REFERENCES

[1] V. Barnett, T. Lewis, and Francine Abeles. 1994. *Outliers in statistical data.* Wiley & Sons. 117–118 pages.
[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization.* Vol. 1. Cambridge University Press.
[3] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, and Shijian Li. Real-time detection of anomalous taxi trajectories from GPS traces. In *ICST'11.* 63–74.
[4] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, Shijian Li, and Zonghui Wang. 2013. iBOAT: isolation-based online anomalous trajectory detection. *IEEE Trans. Intelligent Transportation Systems* 14, 2 (2013), 806–818.
[5] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *ICDE'11.*
[6] Yong Ge, Hui Xiong, Chuanren Liu, and Zhi Hua Zhou. A taxi driving fraud detection system. In *ICDM'12.* 181–190.
[7] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *KDD'07.* 330–339.
[8] Ralf Hartmut Güting, Victor Teixeira de Almeida, and Zhiming Ding. 2006. Modeling and querying moving objects in networks. *VLDB J.* 15, 2 (2006), 165–190.
[9] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. 2000. Distance-based outliers: algorithms and applications. *The VLDB Journal* 8, 3 (2000), 237–253.
[10] Jae Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory outlier detection: a partition-and-detect framework. In *ICDE'08.* 140–149.
[11] Po-Ruey Lei. 2016. A framework for anomaly detection in maritime trajectory behavior. *Knowl. Inf. Syst.* 47, 1 (2016), 189–214.
[12] Xiaolei Li, Jiawei Han, Sangkyum Kim, and Hector Gonzalez. ROAM: rule- and motif-based anomaly detection in massive moving object data sets. In *SDM'07.* 273–284.
[13] Zicheng Liao, Yizhou Yu, and Baoquan Chen. Anomaly detection in GPS data based on visual analytics. In *VAST'10.* 51–58.
[14] Christopher D. Manning, Prabhakar Raghavan, and Hinrich SchÃijtze. 2008. An introduction to information retrieval. *Journal of the American Society for Information Science & Technology* 43, 3 (2008), 824–825.
[15] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *SIGSPATIAL'09.* 336–343.
[16] Tan Pang-Ning, Michael Steinbach, and Vipin Kumar. 2006. *Introduction to data mining.* Vol. 1. Pearson Addison Wesley Boston.
[17] Lin Qi and Markus Schneider. MONET: modeling and querying moving objects in spatial networks. In *SIGSPATIAL.*
[18] L. R Rabiner and B. H. Juang. 1986. An introduction to hidden Markov models. *IEEE Assp Magazine* 3, 1 (1986), 4–16.
[19] Rowland R. Sillito and Robert B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *BMVC'08.* 1–10.
[20] Michalis Vazirgiannis and Ouri Wolfson. A Spatiotemporal model and language for moving objects on road networks. In *SSTD'01.* 20–35.
[21] Yanwei Yu, Lei Cao, Elke A. Rundensteiner, and Qin Wang. Detecting moving object outliers in massive-scale trajectory streams. In *SIGKDD'14.* 422–431.
[22] Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li. iBAT: detecting anomalous taxi trajectories from GPS traces. In *UbiComp'11.* 99–108.
[23] Yu Zheng, Huichu Zhang, and Yong Yu. Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In *SIGSPATIAL'15.* 2:1–2:10.
[24] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. Time-dependent popular routes based trajectory outlier detection. In *WISE'15.* 16–30.
[25] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. 2017. Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web* 20, 1 (2017), 111–134.
[26] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI'08.* 1433–1438.