Predicting Globally and Locally: A Comparison of Methods for Vehicle Trajectory Prediction

William Groves, Ernesto Nunes, and Maria Gini
Department of Computer Science and Engineering, University of Minnesota
{groves, enunes, gini}@cs.umn.edu

Abstract

We propose eigen-based and Markov-based methods to explore the global and local structure of patterns in real-world GPS taxi trajectories. Our primary goal is to predict the subsequent path of an in-progress taxi trajectory. The exploration of global and local structure in the data differentiates this work from the state-of-the-art literature in trajectory prediction methods, which mostly focuses on local structures and feature selection. We propose four algorithms: a frequency based algorithm FreqCount, which we use as a benchmark, two eigen-based (EigenStrat, LapStrat), and a Markov-based algorithm (MCStrat). Pairwise performance analysis on a large real-world data set reveals that LapStrat is the best performer, followed by MCStrat.

1 Introduction

In order to discover *characteristic* patterns in large spatio-temporal data sets, mining algorithms have to take into account spatial relations, such as topology and direction, as well as temporal relations. The increased use of devices that are capable of storing driving-related spatio-temporal information helps researchers and practitioners gather the necessary data to understand driving patterns in cities, and to design location-based services for drivers. To the urban planner, the work can help to aggregate driver habits and can uncover alternative routes that could help alleviate traffic. Additionally, it also helps prioritize the maintenance of roads.

Our work combines data mining techniques that discover global structure in the data, and local probabilistic methods that predict short-term routes for drivers, based on past driving trajectories through the road network of a city.

The literature on prediction has offered Markov-based and other probabilistic methods that predict paths accurately. However, most methods rely on local structure of data, and use many extra features to improve prediction accuracy. In this paper we use only the basic spatio-temporal data stream. We advance the state-of-the-art by proposing the LapStrat algorithm. This algorithm reduces dimensionality and clusters data using spectral clustering to then predict a subsequent path using a Bayesian network. Our algorithm supports global analysis of the data, via clustering, as well as local inference using the Bayesian framework. In addition, since our algorithm only uses location and time data, it can be easily generalized to other domains with spatio-temporal information. Our contributions are summarized as follows:

1. We offer a systematic way of extracting common behavioral characteristics from a large set of observations using an algorithm inspired by principal component analysis (EigenStrat) and our LapStrat algorithm.
2. We compare the effectiveness of methods that explore global structure only (FreqCount and EigenStrat), local structure only (MCStrat), and mixed global and local structure (LapStrat). We show experimentally that LapStrat offers competitive prediction power compared to the more local structure-reliant MCStrat algorithm.

2 Related Work

Eigendecomposition has been used extensively to analyze and summarize the characteristic structure of data sets. The structure of network flows is analyzed in [Lakhina *et al.*, 2004], principal component analysis (PCA) is used to summarize the characteristics of the flows that pass through an internet service provider. [Zhang *et al.*, 2009] identify two weaknesses that make PCA less effective on real-world data. i.e. sensitivity to outliers in the data, and concerns about its interpretation, and present an alternative, Laplacian eigenanalysis. The difference between these methods is due to the set of relationships each method considers: the Laplacian matrix only considers similarity between close neighbors, while PCA considers relationships between all pairs of points. These studies focus on the clustering power of the eigen-based methods to find structures in the data. Our work goes beyond summarizing the structure of the taxi routes, and uses the eigenanalysis clusters to predict the subsequent path of an in-progress taxi trajectory.

Research in travel prediction based on driver behavior has enjoyed some recent popularity. [Krumm, 2010] predicts the next turn a driver will take by choosing with higher likelihood a turn that links more destinations or is more time efficient. [Ziebart *et al.*, 2008] offer algorithms for turn prediction, route prediction, and destination prediction. The study uses a Markov model representation and inverse reinforcement learning coupled with maximum entropy to provide accurate predictions for each of their prediction tasks. [Veloso *et al.*, 2011] proposes a Naive Bayes model to predict that a taxi will visit an area, using time of the day, day of the week, weather, and land use as features. In [Fiosina and Fiosins, 2012], travel time prediction in a decentralized setting is investigated. The work uses kernel density estimation to predict the travel time of a vehicle based on features including length of the route, average speed in the system, congestion level, number of traffic lights, and number of left turns in the route.

All these studies use features beyond location to improve prediction accuracy, but they do not offer a comprehensive analysis of the structure of traffic data alone. Our work addresses this shortcoming by providing both an analysis of commuting patterns, using eigenanalysis, and route prediction based on partial trajectories.

3 Data Preparation

The GPS trajectories we use for our experiments are taken from the publicly available Beijing Taxi data set which includes 1 to 5-minute resolution location data for over ten-thousand taxis for one week in 2009 [Yuan *et al.*, 2010]. Beijing, China is reported to have seventy-thousand registered taxis, so this data set represents a large cross-section of all taxi traffic for the one-week period [Zhu *et al.*, 2012].

Because the data set contains only location and time information of each taxi, preprocessing the data into segments based on individual taxi fares is useful. The data has sufficient detail to facilitate inference on when a taxi ride is completed: for example, a taxi waiting for a fare will be stopped at a taxi stand for many minutes [Zhu *et al.*, 2012]. Using these inferences, the data is separated into taxi rides.

To facilitate analysis, the taxi trajectories are discretized into transitions on a region grid with cells of size 1.5 km × 1.5 km square. $V =< \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_w >$ is a collection of trajectories. We divide it into V_{TR}, V_{TE}, V_{VA} which are the training, test, and validation sets, respectively. A trajectory $\boldsymbol{v_i}$ is a sequence of N time-ordered GPS coordinates: $\boldsymbol{v_i} =< c_1^{\boldsymbol{v_i}}, \ldots c_j^{\boldsymbol{v_i}}, \ldots, c_N^{\boldsymbol{v_i}} >$. Each coordinate contains a GPS latitude and longitude value, $c_j^{\boldsymbol{v_i}} = (x_j, y_j)$. Given a complete trajectory ($\boldsymbol{v_i}$), a *partial trajectory* (50% of a full trajectory) can be generated as $\boldsymbol{v}_i^{partial} =< c_1^{\boldsymbol{v}_i}, c_2^{\boldsymbol{v}_i}, \ldots, c_{N/2}^{\boldsymbol{v}_i} >$. The last location of a partial trajectory $\boldsymbol{v}_i^{last} =< c_{N/2}^{\boldsymbol{v}_i} >$ is used to begin the prediction task.

The relevant portion of the city's area containing the majority of the city's taxi trips, called a *city grid*, is enclosed in a matrix of dimension 17 × 20. Each s_i corresponds to the center of a grid square in the euclidean xy-space. The city graph is encoded as a rectilinear grid with directed edges ($e_{s_i s_j}$) between adjacent grid squares. $\mathbb{I}(c_j, s_i)$ is an indicator function that returns 1 if GPS coordinate c_j is closer to grid center s_i than to any other grid center and otherwise returns 0. Equation 1 shows an indicator function to determine if two GPS coordinates indicate traversal in the graph.

$$\Phi(c_j^{\boldsymbol{v}_i}, c_k^{\boldsymbol{v}_i}, e_{s_l s_m}) = \begin{cases} 1, & \text{if } \mathbb{I}(c_j^{\boldsymbol{v}_i}, s_l) * \mathbb{I}(c_k^{\boldsymbol{v}_i}, s_m) = 1 \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

From trajectory $\boldsymbol{v}_i$, a policy vector $\boldsymbol{\pi}_i$ is created having one

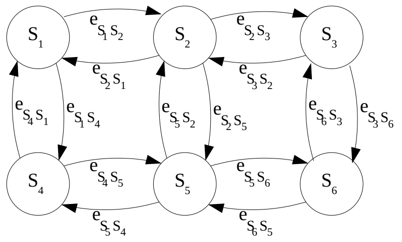

Figure 1: City grid transitions are all rectilinear.

value for each edge in the city grid. Each δ_{s_l,s_m} is a directed edge coefficient indicating that a transition occurred between s_l and s_m in the trajectory. The policy vectors for this data set graph have length ($|\boldsymbol{\pi}|$) of 1286, based on the number of edges in the graph. A small sample city grid is in Figure 1. A collection of policies $\Pi =< \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_w >$ is computed from a collection of trajectories V:

$$\boldsymbol{\pi}^{\boldsymbol{v}_i} =< \delta_{s_1,s_2}^{\boldsymbol{v}_i}, \ldots, \delta_{s_l,s_m}^{\boldsymbol{v}_i}, \ldots > \tag{2}$$

$$\delta_{s_l,s_m}^{\boldsymbol{v}_i} = \begin{cases} 1, & \text{if } \sum_{j=1}^{N-1} \Phi(c_j^{\boldsymbol{v}_i}, c_{j+1}^{\boldsymbol{v}_i}, e_{s_l,s_m}) \geq 1 \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

A graphical example showing a trajectory converted into a policy is shown in Figure 2. All visited locations for trajec-

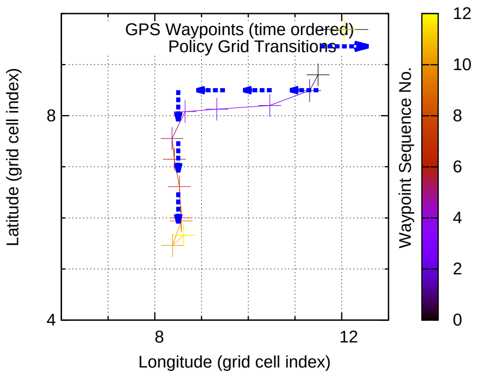

Figure 2: A trajectory converted to a policy in the city grid.

tory $\boldsymbol{v}_i^{partial}$ are given by $\boldsymbol{\theta}^{\boldsymbol{v}_i^{partial}}$:

$$\boldsymbol{\theta}^{\boldsymbol{v}_i^{partial}} =< \omega_{s_1}, \omega_{s_2}, \ldots, \omega_{s_m} >, \text{with} \tag{4}$$

$$\omega_{s_i} = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} \mathbb{I}(c_j^{\boldsymbol{v}_i^{partial}}, s_i) \geq 1 \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

A baseline approach for prediction, FreqCount, uses observed probabilities of each outgoing transition from each node in the graph. Figure 3 shows the relative frequencies of transitions between grid squares in the training set. This city grid discretization is similar to methods used by others in this domain [Krumm and Horvitz, 2006; Veloso *et al.*, 2011].

4 Methods

This work proposes four methods that explore either the local or the global structure or a mix of both to predict short-term trajectories for drivers, based on past trajectories.

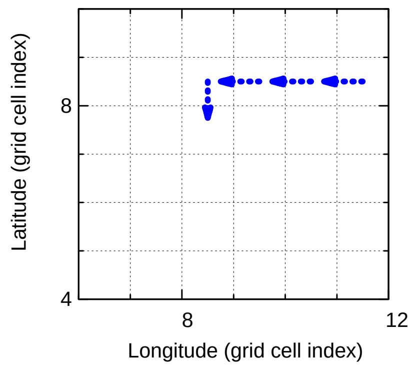

Figure 4: A sample partial policy $\boldsymbol{\pi}^{\boldsymbol{v}_i^{partial}}$

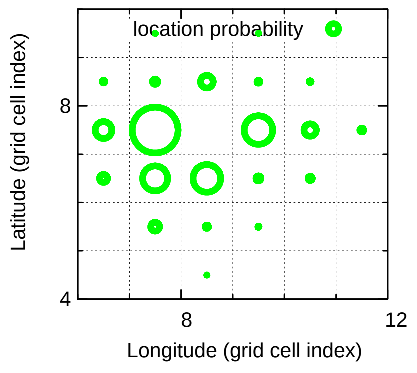

Figure 5: $\hat{\boldsymbol{\theta}}$, location probabilities from FreqCount method with horizon of 3

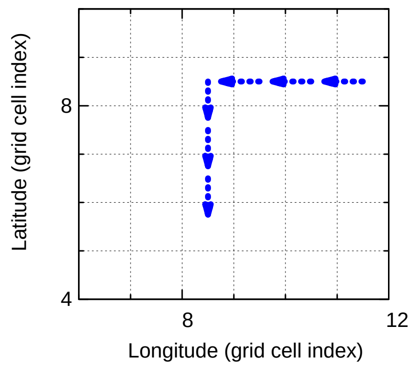

Figure 6: Actual complete trajectory corresponding to Fig. 4, trajectory $\boldsymbol{\pi}^{\boldsymbol{v}_i}$

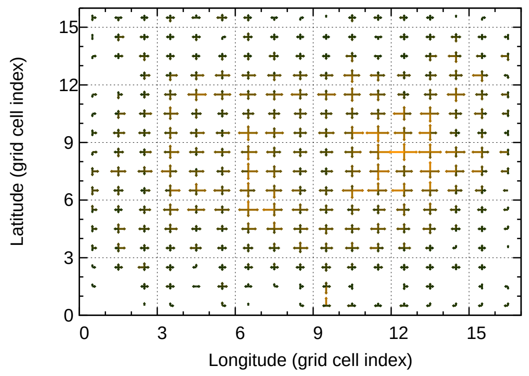

Figure 3: Visualization of frequency counts for edge transitions in the training set. Longer arrows indicate more occurrences.

Benchmark: Frequency Count Method. A benchmark prediction measure, FreqCount, uses frequency counts for transitions in the training set to predict future actions. The relative frequency of each rectilinear transition from each location in the grid is computed and is normalized based on the number of trajectories involving the grid cell. The resulting policy matrix is a Markov chain that determines the next predicted action based on the current location of the vehicle.

The FreqCount method computes a policy vector based on all trajectories in the training set V_{TR}. $\boldsymbol{\pi}^{\text{FreqCount}}$ contains first order Markov transition probabilities computed from all trajectories as in Equation 6.

$$\delta_{s_i,s_j}^{\boldsymbol{\pi}^{\text{FreqCount}}} = \frac{\sum_{\boldsymbol{v}\in V_{TR}} \delta_{s_i,s_j}^{\boldsymbol{v}}}{\sum_{\boldsymbol{v}\in V_{TR}} \sum_{k=1}^{M} \delta_{s_i,s_k}^{\boldsymbol{v}}} \tag{6}$$

The probability of a transition ($s_i \rightarrow s_j$) is computed as the count of the transition $s_i \rightarrow s_j$ in V_{TR} divided by the count of all transitions exiting s_i in V_{TR}.

Policy iteration (Algorithm 1) is applied to the last location of a partial trajectory using the frequency count policy set $\Pi^{\text{FreqCount}} =< \boldsymbol{\pi}^{\text{FreqCount}} >$ to determine a basic prediction of future actions. This method only considers frequency of occurrence for each transition in the training set, so it is expected to perform poorly in areas where trajectories intersect.

```
Algorithm 1: Policy Iteration
   Input: Location vector with last location of taxi $\boldsymbol{\theta}^{last}$, a
          policy list $\Pi$, prediction horizon $niter$
   Output: A location vector containing visit probabilities
           for future locations $\hat{\boldsymbol{\theta}}$
1  $\boldsymbol{\theta}^{accum} \leftarrow \boldsymbol{\theta}^{last}$
2  for $\boldsymbol{\pi} \in \Pi$ do
3      $t \leftarrow 1$
4      $\boldsymbol{\theta}^0 \leftarrow \boldsymbol{\theta}^{last}$
5      while $t \leq niter$ do
6          $\boldsymbol{\theta}^t =< \omega_{s_1}^t, \omega_{s_2}^t, \ldots, \omega_{s_i}^t, \ldots, \omega_{s_M}^t >$
7          , where $\omega_{s_i}^t = \max_{s_j \in S}(\omega_{s_j}^{t-1} * \delta_{s_j,s_i}^{\boldsymbol{\pi}})$
8          $t \leftarrow t + 1$
9      for $S_i \in S$ do
10         $\omega_{s_i}^{\boldsymbol{\theta}^{accum}} = \max(\omega_{s_i}^{\boldsymbol{\theta}^{accum}}, \omega_{s_i}^{\boldsymbol{\theta}^t})$
11 $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{accum}$
```

EigenStrat: Eigen Analysis of Covariance. This method exploits linear relationships between transitions in the grid which 1) can be matched to partial trajectories for purposes of prediction and 2) can be used to study behaviors in the system. The first part of the algorithm focuses on model generation. For each pair of edges, the covariance is computed using the training set observations. The n largest eigenvectors are computed from the covariance matrix. These form a collection of characteristic *eigen-strategies* from training data.

When predicting for an in-progress trajectory, the algorithm takes the policy generated from a partial taxi trajectory $\boldsymbol{\pi}^{\boldsymbol{v}_{predict}}$, a maximum angle to use as the relevancy threshold α, and the eigen-strategies as Π. Eigen-strategies having an angular distance less than α to $\boldsymbol{\pi}^{\boldsymbol{v}_{predict}}$ are added to Π_{rel}. This collection is then used for policy iteration. Optimal values for α and *dims* are learned experimentally.

Eigenpolicies also facilitate exploration of strategic decisions. Figure 7 shows an eigenpolicy plot with a distinct pattern in the training data. Taxis were strongly confined to trajectories either the inside circle or the perimeter of the circle,

Algorithm 2: EigenStrat

```
Input: $\Pi_{\text{TR}}$, number of principal components (*dims*),
       minimum angle between policies ($\alpha$), prediction
       horizon (*horizon*), partial policy ($\boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}}$)
Output: Inferred location vector $\hat{\boldsymbol{\theta}}$
1 Generate covariance matrix $C_{|\boldsymbol{\pi}_i| \times |\boldsymbol{\pi}_i|}$ (where $\boldsymbol{\pi}_i \in \Pi_{\text{TR}}$)
  between transitions on the grid
2 Get the *dims* eigenvectors of C with largest eigenvalues
3 Compute cosine similarity between $\boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}}$ and the
  principal components ($\boldsymbol{\pi}_j, j = 1 \dots dims$):
  $\boldsymbol{\Pi}_{rel} = \{\boldsymbol{\pi}_j \| cos(\boldsymbol{\pi}_j, \boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}})| > \alpha\}$
4 If the $cos(\boldsymbol{\pi}_j, \boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}}) < 0$, then flip the sign of the
  coefficients for this eigenpolicy. Use Algorithm 1 with
  $\Pi_{rel}$ on $\boldsymbol{v}_i^{partial}$ for *horizon* iterations to compute $\hat{\boldsymbol{\theta}}$
```

but rarely between these regions. The two series (positive and negative) indicate the sign and magnitude of the grid coefficients for this eigenvector. We believe analysis of this type has great promise for large spatio-temporal data sets.

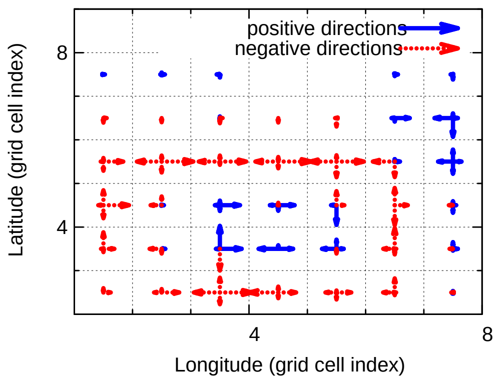

Figure 7: An eigenpolicy showing a strategic pattern.

LapStrat: Spectral Clustering-Inspired Algorithm. LapStrat (Algorithm 3) combines spectral clustering and Bayesian-based policy iteration to cluster policies and infer driver next turns. Spectral clustering operates upon a similarity graph and its respective Laplacian operator. This work follows the approach of [Shi and Malik, 2000] using an unnormalized graph Laplacian. We use Jaccard index to compute the similarity graph between policies. We chose the Jaccard index, because it finds similarities between policies that are almost parallel. This is important in cases such as two highways that only have one meeting point; in this case, if the highways are alternative routes to the same intersection, they should be similar with respect to the intersection point. The input to the Jaccard index are two vectors representing policies generated in Section 3. $J(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j)$ is the Jaccard similarity for pair $\boldsymbol{\pi_i}$ and $\boldsymbol{\pi_j}$. The unnormalized Laplacian is computed by subtracting the *degree matrix* from the similarity matrix in the same fashion as [Shi and Malik, 2000]. We choose the *dims* eigenvectors with smallest eigenvalues, and

Algorithm 3: LapStrat

```
Input: $\Pi_{\text{TR}}$, dimension (*dims*), number of clusters ($k$),
       similarity threshold ($\epsilon$), prediction (*horizon*),
       partial policy ($\boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}}$)
Output: Inferred location vector $\hat{\boldsymbol{\theta}}$
1 Generate similarity matrix $W_{|\Pi_{\text{TR}}| \times |\Pi_{\text{TR}}|}$ where
  $w_{ij} = \begin{cases} J(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j), & \text{if } J(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j) \geq \epsilon \\ 0 & \text{Otherwise} \end{cases}$
2 Generate Laplacian (L): $L = D - W$ and $\forall d_{ij} \in D$
  $d_{ij} = \begin{cases} \sum_{z=1}^{|\Pi_{TR}|} w_{iz}, & \text{if } i = z \\ 0 & \text{Otherwise} \end{cases}$
3 Get the *dims* eigenvectors with smallest eigenvalues
4 Use $k$-means to find the mean centroids ($\boldsymbol{\pi}_j, j = 1 \dots k$)
  of $k$ policy clusters
5 Find all centroids similar to $\boldsymbol{\pi}^{\boldsymbol{v}_i{}^{partial}}$:
  $\Pi_{rel} = \{\boldsymbol{\pi}_j | J(\boldsymbol{\pi}_j, \boldsymbol{\pi}^{v_i{}^{partial}}) > \epsilon\}$
6 Use Algorithm 1 with $\Pi_{rel}$ on $\boldsymbol{v}_i^{partial}$ for *horizon*
  iterations to compute $\hat{\boldsymbol{\theta}}$
```

perform k-means to find clusters in the reduced dimension. The optimal value for *dims* is learned experimentally.

MCStrat: Markov Chain-Based Algorithm. The Markov chain approach uses local, recent information from $\boldsymbol{v}_{predict}^{partial}$, the partial trajectory to predict from. Given the last k edges traversed by the vehicle, the algorithm finds all complete trajectories in the training set containing the same k edges to build a set of relevant policies V_{rel} using the match function. match$(k, \boldsymbol{a}, \boldsymbol{b})$ returns 1 only if at least the last k transitions in the policy generated by trajectory $\boldsymbol{a}$ are also found in $\boldsymbol{b}$. Using Equation 9, V_{rel} is used to build a composite single relevant policy $\boldsymbol{\pi}_{rel}$, that obeys the Markov assumption, so the resulting policy preserves the probability mass.

$$V_{rel} = \{\boldsymbol{v}_i | \text{match}(k, \boldsymbol{\pi}^{\boldsymbol{v}_{predict}^{partial}}, \boldsymbol{\pi}^{\boldsymbol{v}_i}) = 1, \boldsymbol{v}_i \in V_{\text{TR}}\} \quad (7)$$

$$\boldsymbol{\pi}_{\text{rel}} = < \delta_{s_1,s_2}^{\boldsymbol{\pi}^{rel}}, \dots, \delta_{s_i,s_j}^{\boldsymbol{\pi}^{rel}}, \dots > \quad (8)$$

$$\delta_{s_i,s_j}^{\boldsymbol{\pi}^{rel}} = \frac{\sum_{\boldsymbol{v} \in V_{rel}} \delta_{s_i,s_j}^{\boldsymbol{v}}}{\sum_{\boldsymbol{v} \in V_{rel}} \sum_{k=1}^{M} \delta_{s_i,s_k}^{\boldsymbol{v}}} \quad (9)$$

Using the composite $\boldsymbol{\pi}_{rel}$, policy iteration is then performed on the last location vector computed from $\boldsymbol{v}_{predict}$.

Method Complexity Comparison. A comparison of the storage complexity of the methods appears in Table 1.

Model	Model Construction	Model Storage
FreqCount	$O(\lvert\boldsymbol{\pi}\rvert)$	$O(\lvert\boldsymbol{\pi}\rvert)$
EigenStrat	$O((\lvert\boldsymbol{\pi}\rvert)^2)$	$O(\text{dims} \times \lvert\boldsymbol{\pi}\rvert)$
LapStrat	$O((\lvert\Pi_{TR}\rvert)^2)$	$O(k \times \lvert\boldsymbol{\pi}\rvert)$
MCStrat	$O(1)$	$O(\lvert\Pi_{TR}\rvert \times \lvert\boldsymbol{\pi}\rvert)$

Table 1: Space complexity of methods.

5 Results

Given an in-progress taxi trajectory, the methods presented facilitate predictions about the future movement of the vehicle. To simulate this task, a collection of partial trajectories (e.g. Figure 4) is generated from complete trajectories in the test set. A set of relevant policy vectors is generated using one of the four methods described, and policy iteration is performed to generate the future location predictions. The inferred future location matrix (e.g. Figure 5) is compared against the actual complete taxi trajectory (e.g. Figure 6). Prediction results are scored by comparing the inferred visited location vector $\hat{\boldsymbol{\theta}}$ against the full location vector $\boldsymbol{\theta}^{\boldsymbol{v}_i}$. The scores are computed using Pearson's correlation: $score = Cor(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^{\boldsymbol{v}_i})$. The scores reported are the aggregate mean of scores from examples in the validation set.

The data set contains 100,000 subtrajectories (of approximately 1 hour in length) from 10,000 taxis. The data set is split randomly into 3 disjoint collections to facilitate experimentation: 90% in the training set, and 5% in both the test and validation sets. For each model type, the training set is used to generate the model. Model parameters are optimized using the test set. Scores are computed using predictions made on partial trajectories from the validation set.

Results of each method for 4 prediction horizons are shown in Table 2. The methods leveraging more local information near the last location of the vehicle (LapStrat, MCStrat) perform better than the methods relying only on global patterns (FreqCount, EigenStrat). This is true for all prediction horizons, but the more local methods have an even greater performance advantage for larger prediction horizons.

	Prediction Horizon			
Method	1	2	4	6
FreqCount	.579 (.141)	.593 (.127)	.583 (.123)	.573 (.122)
EigenStrat	.563 (.143)	.576 (.134)	.574 (.140)	.574 (.140)
LapStrat	.590 (.144)	**.618** (.139)	**.626** (.137)	**.626** (.137)
MCStrat	**.600** (.146)	.616 (.149)	.621 (.149)	.621 (.149)

Table 2: Correlation (std. dev.) by method and prediction horizon. The best score is in **bold**.

Statistical significance testing was performed on the validation set results, as shown in Table 3. The best performing methods (LapStrat and MCStrat) achieve a statistically significant performance improvement over the other methods. However, the relative performance difference between the local methods is not significantly different.

6 Conclusions

The methods presented can be applied to many other spatio-temporal domains where only basic location and time information is collected from portable devices, such as sensor networks as well as mobile phone networks. These predictions assume the action space is large but fixed and observations implicitly are clustered into distinct but repeated goals. In this domain, each observation is a set of actions a driver takes in fulfillment of a specific goal: for example, to take a passenger from the airport to his/her home. In future work, we propose to extend this work using a hierarchical approach which simultaneously incorporates global and local predictions to provide more robust results.

Method	FreqCount	EigenStrat	LapStrat	MCStrat
FreqCount	■	**n/a**	**n/a**	**n/a**
EigenStrat	0.431	■	**n/a**	**n/a**
LapStrat	*0.000211	*0.000218	■	0.462
MCStrat	*0.00149	*0.000243	**n/a**	■

Table 3: p-values of Wilcoxon signed-rank test pairs. Starred (*) values indicate the row method achieves statistically significant (0.1% significance level) improvement over the column method for a prediction horizon of 6. If **n/a**, the row method's mean is not better than the column method.

References

[Fiosina and Fiosins, 2012] J. Fiosina and M. Fiosins. Cooperative kernel-based forecasting in decentralized multi-agent systems for urban traffic networks. In *Ubiquitous Data Mining*, pages 3–7. ECAI, 2012.

[Krumm and Horvitz, 2006] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. *UbiComp 2006*, pages 243–260, 2006.

[Krumm, 2010] J. Krumm. Where will they turn: predicting turn proportions at intersections. *Personal and Ubiquitous Computing*, 14(7):591–599, 2010.

[Lakhina *et al.*, 2004] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. *Perform. Eval. Rev.*, 32(1):61–72, 2004.

[Shi and Malik, 2000] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Veloso *et al.*, 2011] M. Veloso, S. Phithakkitnukoon, and C. Bento. Urban mobility study using taxi traces. In *Proc. of the 2011 Int'l Workshop on Trajectory Data Mining and Analysis*, pages 23–30, 2011.

[Yuan *et al.*, 2010] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proc. of the 18th SIGSPATIAL Int'l Conf. on Advances in GIS*, pages 99–108, 2010.

[Zhang *et al.*, 2009] J. Zhang, P. Niyogi, and Mary S. McPeek. Laplacian eigenfunctions learn population structure. *PLoS ONE*, 4(12):e7928, 12 2009.

[Zhu *et al.*, 2012] Y. Zhu, Y. Zheng, L. Zhang, D. Santani, X. Xie, and Q. Yang. Inferring Taxi Status Using GPS Trajectories. *ArXiv e-prints*, May 2012.

[Ziebart *et al.*, 2008] B. Ziebart, A. Maas, A. Dey, and J. Bagnell. Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior. In *Proc. of the 10th Int'l Conf. on Ubiquitous computing*, UbiComp '08, pages 322–331, 2008.