Project Final Report
Title: Inferring transportation modes from GPS trajectories

Sole Team Member: Hasan Pourmahmoodaghababa
uID: u1255635

Abstract

In this project, using a conditional Markov model (CMM), also called maximum entropy Markov model (MEMM) and recurrent neural networks (RNN, specially LSTM) we try to infer transportation modes from GPS trajectories. We employ the feature mapping defined in [6], length, velocity and acceleration of trajectories to infer transportation modes. We show that CMM outperforms other inferring methods available in the literature that the author is aware of.

1 Introduction

Transportation modes analysis is one of the hot research areas in transportation and traffic research area. In fact, travel mode choice is a typical behavior attribute of people and so is of interest for researchers. Nowadays with the advent of Global Positioning System (GPS), it is easy to collect travel data, for instance through mobile phones, in contrast with the old days where people had to obtain data through a survey for example. This has led to more interest in research in this area and absorbed many researchers.

Understanding mobility patterns in mining GPS datasets has many applications like learning significant locations and identification of transportation modes (see [4, 5, 1, 2] and references therein). Authors in [1, 2, 3, 7] utilize convolutional neural networks to infer transportation modes and a hidden Markov model is employed in [5] for this purpose. All these papers focus on Geolife dataset (see Subsection 3.1) which we also would like to analyze it here.

In this project we will apply two models with several feature mappings. The first model is a *conditional Markov model* (CMM) also called *maximum entropy Markov model* (MEMM), which is a next-state model. In this model we will use some machine learning classical classifiers such as SVM, Decision tree and Random Forest as local models to get local scores for each state. Then for inference step we will apply Viterbi algorithm. The next model is a *recurrent neural network* (RNN), which is appropriate for sequential data analysis. Indeed, for we will leverage the special RNN termed *long-short-term-memory* (LSTM) where the previous states' information is transferred as a memory to the next state and so preserves the global dependencies in data, and hence suitable for structured predictions.

2 Definitions and Feature Mappings

In this section we give some definitions and introduce the feature mappings we will use in this project.

Curve. By a *curve* we mean the image of a continuous map $\gamma : [0, 1] \to \mathbb{R}^2$. The set of curves is denoted by Γ.

So, note that we are only considering the geometric shape of curves and so two different maps can have a same curve. For instance, if γ is a curve, then γ and $\overline{\gamma}$, where $\overline{\gamma}(t) = \gamma(1 - t)$, are considered the same curves as they are the same geometrically. However, their direction is not the same and thus they are not the same as two maps.

Trajectory. By a *trajectory* we mean a finite sequence of triples (a, b, t), where a and b denote latitude and longitude of a moving object and t is the timestamp showing the time where the object was at (a, b) position in x, y-coordinate at that time. Assuming $\{(a_i, b_i, t_i) : 0 \leq i \leq m\}$ is a trajectory, connecting consecutive (a_i, b_i)'s by a line segment we get a piece-wise linear curve, which we call it trajectory as well. Therefore, the set of trajectories is a subset of Γ.

Feature Mapping. Let $\gamma \in \Gamma$ be a curve and let $q \in \mathbb{R}^2$ be a landmark. According to [6] we define $v_q : \Gamma \to [0, \infty)$ by

$$v_q(\gamma) = \text{dist}(q, \gamma) = \min_{p \in \gamma} \|q - p\|,$$

where $\|\cdot\|$ is the l^2-norm on $\mathbb{R}^2$. Now let $Q = \{q_1, q_2, \ldots, q_n\} \subseteq \mathbb{R}^2$ be a set of landmarks. Then, as it is introduced in [6], we can define a feature mapping $v_Q : \Gamma \to \mathbb{R}^n$ by

$$v_Q(\gamma) = (v_{q_1}(\gamma), v_{q_2}(\gamma), \ldots, v_{q_n}(\gamma)).$$

This also give a distance on Γ provided that Q is sufficiently dense in a bounded region containing all curves under consideration [6]. The distance is given by $d_Q(\gamma_1, \gamma_2) = \|v_Q(\gamma_1) - v_Q(\gamma_2)\|$.

It is worth noting that it is commented that in practice 20 landmarks are usually enough for trajectory data analysis tasks.

We will use a trajectory dataset and so we can define three more features, namely *average length*, *velocity* and *acceleration*. More formally, for a trajectory $\gamma = \{(a_0, b_0, t_0), \ldots, (a_m, b_m, t_m)\}$ we consider

$$\text{length} = \frac{1}{m} \sum_{j=1}^{m} \|(a_j, b_j) - (a_{j-1}, b_{j-1})\|,$$

$$\text{velocity} = \frac{1}{m} \sum_{j=1}^{m} \frac{\|(a_j, b_j) - (a_{j-1}, b_{j-1})\|}{t_j},$$

$$\text{acceleration} = \frac{1}{m} \sum_{j=1}^{m} \frac{\|(a_j, b_j) - (a_{j-1}, b_{j-1})\|}{t_j^2}.$$

Using these features, we can have a bunch of feature mappings:

1) $\phi_1(\gamma) = (\text{length}, \text{velocity}, \text{acceleration}) \in \mathbb{R}^4$;
2) $\phi_2(\gamma) = (a_0, b_0, a_m, b_m, \text{length}, \text{velocity}, \text{acceleration}) \in \mathbb{R}^8$;
3) $\phi_3(\gamma) = v_Q(\gamma) \in \mathbb{R}^{|Q|}$;
4) $\phi_4(\gamma) = (v_Q(\gamma), \text{length}, \text{velocity}, \text{acceleration}) \in \mathbb{R}^{|Q|+4}$;
5) $\phi_5(\gamma) = (v_Q(\gamma), a_0, b_0, a_m, b_m, \text{length}, \text{velocity}, \text{acceleration}) \in \mathbb{R}^{|Q|+8}$;

3 Experiments

In this section we run some experiments to examine our selected models' performance.

3.1 Data

The dataset we employ here is Geolife GPS Trajectory dataset [9], available for public[1], which is released by Microsoft. It consists of trajectories of 182 users from 2007 to 2012. The dataset consists of 17,621 trajectories which are mostly recorded in Bejing, China. To have a pictorial sense of data, see Figure 1, where one is taken from the user guide of the data and the other is generated by the author.

Among 182 users 69 of them have labeled their trajectories with transportation modes such as walk, bike, bus, car, taxi, subway, railway, train, airplane, motorcycle, run. As other modes are in extreme minority in the labeled data, we will only consider walk, bike, bus, car, taxi, subway, railway and train. Moreover, as it is recommended in the user guide of data, we regard car and taxi as one transportation mode, namely car, and similarly, we regard all three labels train, railway and subway as train. Therefore, we end up with 5 labels (walk, bike, bus, car and train).

[1] https://msropendata.com/datasets/d19b353b-7483-4db7-a828-b130f6d1f035

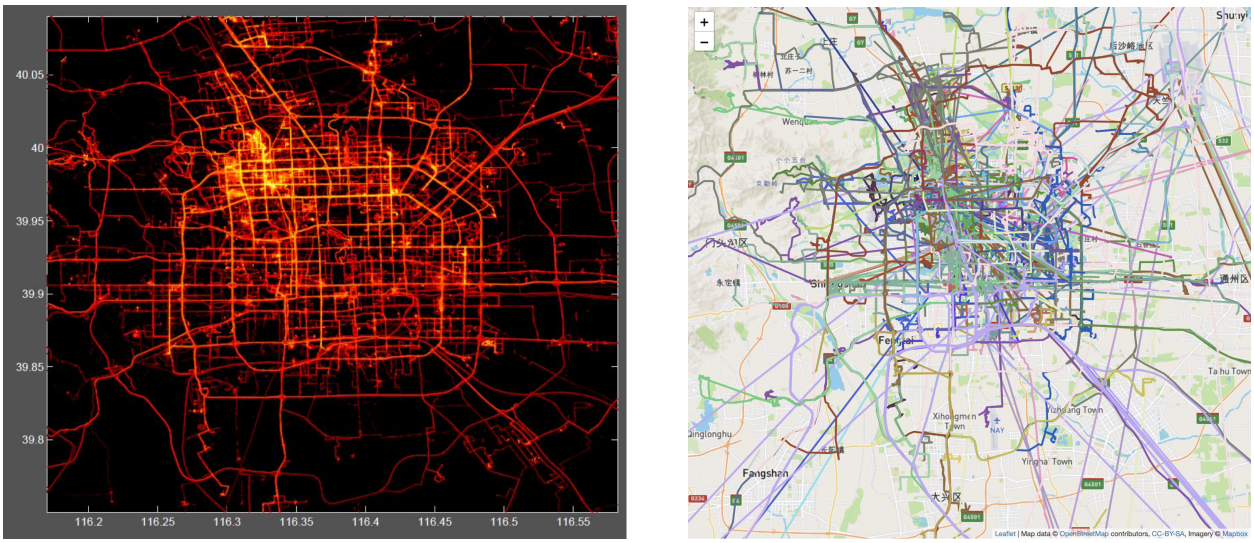

Figure 1: Left: Distribution of the dataset in Beijing city (taken from the user guide of the data). Right: Visualizing Geolife GPS Trajectory Dataset on the map (about 18,000 trajectories, created by the author).

3.2 Preprocessing

First we remove trajectories without a label as well as each part of a trajectory having no matchable label. Then we partition each labeled GPS trajectory of each user into some sub-trajectories (trips) according to the time interval. In fact, for partitioning, we use the threshold 20 minutes recommended in [10] and is applied in several studies like [1, 2]. In this way, we obtain about 10,000 trajectories with only 1 label, 1,700 trajectories with 2 labels and less than 100 trajectories with 3 or 4 labels. Considering a trajectory as a sequence of trips (remember that each trip consists of only one label), we get 10,000 sequences of length 1, 1,700 sequences of length 2 and less than 100 sequences of length 3 or 4. However, we need more sequences with longer lengths. Thus we decided to generate more sequences with lengths 3, 4 and 5. We do this utilizing 10,000 sequences of length 1, by randomly concatenating some of them together. At the end, we get about 5,400 trajectories such that lengths (belonging to $\{1, 2, 3, 4, 5\}$) are evenly distributed.

We remark that generating sequence elements by random will possibly not enforce an essential structure on previous states of the sequence. This means that if we have a real data having such sequences, the performance of our models will definitely be better than what we will get here.

3.3 Experimental Setup

We can consider our problem as a part-of-speach tagging problem, where we would like to assign our 5 labels to these 5,000 trajectories. The first model we use is CMM and the next one is LSTM. In both models, the preprocessed data is randomly split to train and test datasets by 2/3 of data as train data and 1/3 as test data.

3.3.1 Conditional Markov Model

To use CMM, we apply 5 feature mappings associated with ϕ_i in Section 2. Indeed, let $x = (x_1, x_2, \ldots, x_n)$ be a trajectory with label $y = (y_1, y_2, \ldots, y_n) \in \{0, 1, 2, 3, 4\}^n$ $(1 \leq n \leq 5)$, where each x_i is a sub-trajectory having label y_i. Then for each $j \in \{1, 2, 3, 4, 5\}$, using the next-state model $p(y_i|y_{i-1}, \phi(x_i))$, we

define

$$\Phi_j(x_0) = \phi_j(x_0), \quad \Phi_j(x_i, y_{i-1}) = (\phi_j(x_i), y_{i-1}),$$

and so, $\Phi_j(x, y) = (\Phi_j(x_0), \Phi_j(x_1, y_0), \ldots, \Phi_j(x_n, y_{n-1}))$.

Applying these feature mappings, we use 4 ML classifiers Linear SVM, Gaussian SVM, Decision Tree and Random Forest as local scoring functions. Last but not least is inferring step which we employ the **Viterbi** algorithm which is implemented by the author from scratch. Accuracy results are given in Table 1. As it can be seen, the conditional Markov model with feature mapping Φ_4 and Random Forest as a local classifier outperforms others and we get 83.2% test accuracy.

Feature Mapping	Local Classifier	Train Accuracy	Test Accuracy
Φ_1	Linear SVM, $C = 1$	0.5111	0.5043
	Gaussian SVM, $C = 1e5$, $\gamma = 10$	0.5729	0.4820
	Decision Tree	1.0000	0.6744
	Random Forest with 100 estimators	0.9918	**0.7310**
Φ_2	Linear SVM, $C = 0.01$	0.5162	0.5102
	Gaussian SVM, $C = 1e4$, $\gamma = 0.1$	0.6226	0.5475
	Decision Tree	1.0000	0.7374
	Random Forest with 100 estimators	1.0000	**0.8208**
Φ_3	Linear SVM, $C = 1e8$	0.4276	0.4290
	Gaussian SVM, $C = 1e4$, $\gamma = 1$	0.5467	0.4882
	Decision Tree	1.0000	0.6825
	Random Forest with 100 estimators	1.0000	**0.7294**
$\mathbf{\Phi_4}$	Linear SVM, $C = 1$	0.5218	0.5282
	Gaussian SVM, $C = 1e4$, $\gamma = 1$	0.7336	0.5287
	Decision Tree	1.0000	0.7343
	Random Forest with 100 estimators	0.9999	**0.8316**
Φ_5	Linear SVM, $C = 1$	0.5279	0.5144
	Gaussian SVM, $C = 1e4$, $\gamma = 1$	0.7554	0.5252
	Decision Tree	1.0000	0.7452
	Random Forest with 100 estimators	1.0000	**0.8276**

Table 1: CMM-based transportation modes' train and test accuracies (66/33 train/test split) for Beijing dataset using 5 feature mappings. Notice 20 landmarks are chosen randomly around curves to get the feature mappings Φ_3, Φ_4, Φ_5.

3.3.2 Recurrent Neural Networks

In this experiment we apply LSTM to predict transportation modes of GPS trajectories. We apply above five feature mappings with two different LSTM architectures and 66/33 train-test split. These neural networks are implemented in pytorch on a 16 GB Ram and 2 GHz Quad-Core Intel Core i5 system.

First Architecture. The first one is a simple 1-layer LSTM with 1 fully connected layer neural network. The negative log likelihood function (torch.nn.NLLLoss()) is chosen as loss function. We also tried with cross entropy loss (torch.nn.CrossEntropyLoss()) and got roughly the same results. Different number of hidden layers and learning rates are applied to tune hyperparameters and get a robust and high performance model. Train and test accuracies are presented in Table 2. As it can be viewed, the feature mapping Φ_4 outperforms others with 64.3% test accuracy.

Feature Mapping	Hidden Dimension	Learning rate	Train Accuracy	Test Accuracy
Φ_1	20	0.005	0.6310	0.6294
Φ_2	200	0.0008	0.6001	0.6022
Φ_3	100	0.001	0.3732	0.3733
$\mathbf{\Phi_4}$	20	0.01	0.63.34	**0.6427**
Φ_5	200	0.001	0.6329	0.6357

Table 2: Transportation modes' train and test accuracies for Beijing dataset based on LSTM with 1 fully connected layer, using 5 feature mappings. 20 landmarks are chosen randomly around curves to get the feature mappings Φ_3, Φ_4, Φ_5.

Second Architecture. The next one is a multi-layer LSTM with 2 fully connected layers neural network. The loss function is again negative log likelihood function. We tried this architecture with more LSTM layers but we found the more layers the less test accuracies. Therefore, we used only 1 layer of LSTM. Different number of hidden layers and learning rates are applied to tune hyperparameters and get a robust and high performance model. In this architecture, similarly, we can observe that the feature mapping Φ_4 outperforms others with 65% test accuracy.

Feature Mapping	Hid_dim_1	Hid_dim_2	Learning rate	Train Accuracy	Test Accuracy
Φ_1	10	20	0.02	0.6382	0.6276
Φ_2	100	200	0.005	0.5089	0.5223
Φ_3	20	20	0.01	0.3676	0.3753
$\mathbf{\Phi_4}$	100	50	0.001	0.6560	**0.6495**
Φ_5	200	100	0.0015	0.6085	0.5966

Table 3: Transportation modes' train and test accuracies for Beijing dataset based on LSTM with 2 fully connected layers, using 5 feature mappings. 20 landmarks are chosen randomly around curves.

3.4 Baseline Case

As a baseline approach, we try to predict transportation modes of trajectories independent from each other, i.e., considering the problem as a multi-class classification task. We apply K-nearest neighbor search (KNN), Gaussian kernel SVM, Decision Tree and Random Forest with feature mappings ϕ_i ($1 \leq i \leq 5$). Nevertheless, we have included the best performed feature mapping (ϕ_4) results. The setup is clear as it is a multi-class classification and we utilize sklearn classifiers with 66/33 train-test split. The chosen hyperparameters and train and test accuracies are included in Table 4. We have done each classification method for 100 times and reported the average train and test accuracies along with their standard deviations. As it can be viewed, Random Forest outperforms other classifiers with 82.3% accuracy.

Classifier	Train Accuracy	Test Accuracy	Standard Deviation
KNN with 5 centers	0.8117	0.7449	0.0102
Gaussian kernel SVM, $C = \gamma = 1e4$	0.0100	0.7274	0.0112
Decision Tree	1.0000	0.7562	0.0109
Random Forest with 100 estimators	1.0000	**0.8230**	0.0089

Table 4: Multi-class classification train and test accuracies for Beijing dataset using ϕ_4 vectorization. 20 landmarks are chosen randomly around curves.

4 Comparison With Previous Studies

Although the experimental setup in [2] is somehow different and makes comparison a bit hard, our 83.2% test accuracy outperforms the best test accuracy 76.8% reported there. The test accuracy 84.8% reported in [1] is achieved by convolutional neural network with 80/20 train/test split. For the sake of comparison we did our CMM-based experiments with Random Forest as a local classifier and 80/20 train/test split and got 85% accuracy on test data using feature mapping Φ_4 and 50 randomly chosen landmarks; see Table 5 for train and test accuracies. However, it is worth noting that we could play with some hyperparameters, like the number of estimators or entropy type selection in Random Forest or increasing the number of landmarks or choosing them in an optimal way, to increase the accuracy a bit more.

Feature Mapping	Local Classifier	# Estimators	Train Accuracy	Test Accuracy
Φ_1	Random Forest	100	0.9895	0.7481
Φ_2	Random Forest	100	1.0000	0.8356
Φ_3	Random Forest	100	1.0000	0.7833
$\mathbf{\Phi_4}$	Random Forest	80	1.0000	**0.8499**
Φ_5	Random Forest	120	1.0000	0.8480

Table 5: CMM-based transportation modes' train and test accuracies (80/20 train/test split) for Beijing dataset using 5 feature mappings. 50 landmarks are chosen randomly around curves.

To compare more, our CMM method outperforms accuracies reported in many other studies like [3] with 67.9%, [7] with 74.1% and [8] with 76.2% accuracies. A summary of these accuracies is given in Table 6. Note that the experimental setup of each study is a bit different but all are trying to infer transportation modes.

Study	Accuracy
Using CNN [3]	67.9%
Using CNN [7]	74.1%
Inference plus Decision Tree [8]	76.2%
Using CNN [2]	76.8%
Using CNN (80/20 split) [1]	84.8%
Our CMM (66/33 split)	83.2%
Our CMM (80/20 split)	85.0%

Table 6: Accuracy comparison with previous studies

5 Conclusion

According to Tables 1, 2 and 3, we can conclude that CMM-based models had much more better performances than LSTM models on our dataset. The biggest advantage of our conditional Markov model is using Random Forest as a local classifier and extraction of important features. However, looking at Table 4, our multi-class classification results as a baseline method show that there is a small dependency of a transportation mode on previous mode of a trip, i.e., there is no substantial sequential structure in data using our feature mappings. This can be because of the fact that we have randomly concatenated length 1 sequences to get longer sequences.

References

[1] Sina Dabiri and Kevin Heaslip. Inferring transportation modes from gps trajectories using a convolutional neural network. *Transportation Research Part C*, 86:360–371, 2018.

[2] Sina Dabiri, Chang-Tien Lu, Kevin Heaslip, and Chandan K. Reddy. Semi-supervised deep learning approach for transportation mode identification using gps trajectory data. *IEEE Transactions on Knowledge and Data Engineering*, 32:1010–1023, 2020.

[3] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Akihisa Kawanobe. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 6:54–66, 2016.

[4] Miao Lin and Wen-Jing Hsu. Mining gps data for mobility patterns. *Pervasive Mob. Comput.*, 12:1 – 16, 2014.

[5] Li Meng and Shen Zhang. Inferring travel modes from trajectory data based on hidden markov model. *International Conference on Transportation and Development 2020*, 7:95 – 103, 2020.

[6] Jeff M. Phillips and Pingfan Tang. Simple distances for trajectories via landmarks. In *ACM GIS SIGSPATIAL*, 2019.

[7] Hao Wang, GaoJun Liu, Jianyong Duan, and Lei Zhang. Detecting transportation modes using deep neural network. *IEICE Trans. Inf. & Syst.*, 100:1132–1135, 2017.

[8] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. *Proceedings of the 10th International Conference on Ubiquitous Computing. ACM*, 100:312–321, 2008.

[9] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset - User Guide*, July 2011.

[10] Yu. Zheng, Like. Liu, Longhao. Wang, and Xing. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th International Conference on World Wide Web. ACM*, pages 247–256, 2008.