Detecting Transportation Modes Using Deep Neural Network

Hao WANG† (a), GaoJun LIU† (b), Jianyong DUAN† (c), Nonmembers, and Lei ZHANG†† (d), Member

SUMMARY Existing studies on transportation mode detection from global positioning system (GPS) trajectories mainly adopt handcrafted features. These features require researchers with a professional background and do not always work well because of the complexity of traffic behavior. To address these issues, we propose a model using a sparse autoencoder to extract point-level deep features from point-level handcrafted features. A convolution neural network then aggregates the point-level deep features and generates a trajectory-level deep feature. A deep neural network incorporates the trajectory-level handcrafted features and the trajectory-level deep feature for detecting the users’ transportation modes. Experiments conducted on Microsoft’s GeoLife data show that our model can automatically extract the effective features and improve the accuracy of transportation mode detection. Compared with the model using only handcrafted features and shallow classifiers, the proposed model increases the maximum accuracy by 6%.

key words: transportation mode detection, deep feature, trajectory mining

1. Introduction

The understanding of user behavior plays an important role in pervasive computing and context awareness. Detection of the transportation mode, such as walking, cycling, or riding a train, is one of the most important research topics in trajectory mining. A user’s lifestyle, for example, can be inferred through transportation modes [1] and is an important reference for applications in location-based services and traffic management [2].

As the popularity of mobile devices embedded with the global positioning system (GPS) has grown substantially in recent years, it has become convenient to collect users’ GPS data; this also makes it possible to detect transportation modes from GPS trajectories. Currently, transportation mode detection mainly relies on handcrafted effective features designed by researchers, such as speed, acceleration, head change rate, and stop rate. This requires researchers with a professional background because the researchers’ experience affects the effectiveness of the features. However, handcrafted features do not always work well owing to the complexity of traffic behavior. For example, walking and driving have the same speed when traffic congestion occurs.

This paper utilizes deep learning technology to compensate for the above-mentioned shortcomings. First, a sparse autoencoder extracts point-level deep features from point-level handcrafted features. A convolution neural network then aggregates the point-level deep features and generates a trajectory-level deep feature. Finally, a deep neural network is designed to detect the transportation mode using trajectory-level handcrafted features and the trajectory-level deep feature. The experiments show that our model can automatically extract the effective features and improve the accuracy of transportation mode detection. Compared with the model using only handcrafted features and shallow classifiers, the proposed model increases the maximum accuracy of transportation mode detection by 6%.

2. Related Work

Based on data resources, we divide transportation mode detection into two types: trajectory data combined with external information and only trajectory data [3]. In our paper, we focus on studies using only trajectory data. Yu [4] investigates the feature definition and classification algorithm. Yu [5] proposes a traffic model automatic estimation system. In the studies mentioned above, models based on handcrafted features and shallow classifiers are used.


3. Deep Feature Learning

3.1 Basic Concepts and Handcrafted Features

We begin by first introducing the definitions of a feature level containing either a point-level feature or a trajectory-level feature. A point-level feature (PF), obtained directly through the sensor or from the calculation over adjacent points, describes the characteristics of a single GPS point. A trajectory-level feature (TF) requires the overall analysis of a trajectory and describes the characteristics of the trajectory.

We select velocity ($V$), head change ($head$), time interval ($int$), and distance ($dist$) as the point-level handcrafted
features (PHF) and average velocity (AV), variance of velocity (DV), head change rate (HCR), stop rate (SR), and velocity change rate (VCR) as the trajectory-level handcrafted features (THF). All of these features are defined in the same manner as that used in paper [4].

3.2 Deep Features

3.2.1 Deep Feature Learning Model

The overall structure of our deep feature learning model is shown in Fig. 1. The PHF are simply calculated using the GPS data collected from mobile devices. The sparse autoencoder (SAE) further transforms the PHF into point-level deep features (PDF). PDF and THF belong to different levels and cannot be mixed. Therefore, we utilize a convolution neural network (CNN) to aggregate all PDF of a trajectory to a trajectory-level deep feature (TDF). Finally, we feed both features into a deep neural network (DNN) classifier. The algorithm for learning TDF is shown in Algorithm 1.

Algorithm 1: TDF learning algorithm

```
input : points of a trajectory segment P=[p1, p2, ..., pn],
         P=[lat, long, time],
         transonation mode label of segment S,
         a threshold \alpha controlling the rounds
output: P.TDF
begin
  1. P.PHF = [p1.PHF, p2.PHF, ..., pn.PHF] ← Calculate([p1, p2, ..., pn]);
  2. \text{while } i < \alpha \text{ do}
  3. \text{P.PDF} ← SAE.Learning(P.PHF);
  4. \text{P.TDF} ← CNN.Learning(P.PDF);
  5. i++;
return P.TDF;
```

3.2.2 Point-Level Deep Feature Learning

A sparse autoencoder (SAE) is proposed to transform the PHF to PDF, as detailed in Fig. 2. First, we calculate PHF (velocity, head change, interval, and distance) from raw data and use PHF as the input to the autoencoder, which works in an over-complete manner and generates a point-level deep feature, a 16-d vector.

This encoder is formulated as follows:

\[
\begin{align*}
\bar{u} &= f(w_f F_v + w_d F_d + b) \\
(F_v, F_d, \bar{F}_i, \bar{F}_d) &= f(\bar{w}_v u + \bar{w}_d u + \bar{w}_d u + \bar{b})
\end{align*}
\]

Where \( \bar{u} \) is a point-level deep feature; \( w_f, w_d, w_d, w_d \) and \( b \) are parameters of the encoder; \( \bar{w}_v, \bar{w}_d, \bar{w}_d, \bar{w}_d \) and \( \bar{b} \) are parameters of the decoder; \( F_v, F_d, \bar{F}_i \) and \( \bar{F}_d \) are inputs; \( F_v, F_d, \bar{F}_i \) and \( \bar{F}_d \) are reconstructed inputs; \( f() \) is the activation function, and we adopt rectified linear units (ReLU) as the activation function.

The cost function is defined as follows:

\[
\begin{align*}
J(F_v, F_d, \bar{F}_i, \bar{F}_d; \theta) &= \frac{1}{2} \sum_{\sigma \in \{u, d, a, i\}} \|\bar{F}_\sigma - F_\sigma\|^2 + \lambda \left( \sum_{\sigma \in \{u, d, a, i\}} \|\bar{F}_\sigma\|^2 + \|F_\sigma\|^2 \right)
\end{align*}
\]

Where \( J \) is the cost function; \( F_v, F_a, \bar{F}_i \) and \( \bar{F}_d \) represent the input data from the training set; \( \sum_{\sigma \in \{u, d, a, i\}} \|\bar{F}_\sigma - F_\sigma\|^2 \) measures the reconstruction accuracy; \( \lambda \) is the weight decay regularization term that prevents parameters in the model from diverging arbitrarily; \( \lambda \) is the regularization weight.

3.2.3 Trajectory-Level Deep Feature Learning

The point-level deep features from a trajectory in the timeline form a time series. We apply a CNN to model a trajectory using a series of point-level deep features. CNNs have a strong learning capacity with much fewer connections and parameters to learn compared with a similar-sized standard network. They are widely used for learning stationary local features in a series of entities such as images (pixel series), speeches, and other time series. From a time series of point-level deep features, we can learn a trajectory-level deep feature that describes the states of a trajectory. A trajectory-level deep feature is represented as a one-dimensional (1-D) vector.

We obtain the trajectory-level deep feature in two steps. First we perform a sequence compression using a two-dimensional (2-D) convolution network on all the point-level deep features of a trajectory. We then apply a 1-D convolution network on the compressed sequence to obtain the trajectory-level deep feature.

Pooling is another important step. It compresses feature maps into fewer feature instances. Max pooling and mean pooling are two commonly used pooling strategies. In
the max pooling strategy, the pooled feature unit is assigned with the maximum activation among all units in the feature maps. In the mean pooling strategy, the mean of the activations of all units in the feature maps is assigned to the pooled feature unit. We test both pooling strategies in our experiments.

4. Detection Methods

Using the steps described in the previous section, we obtain two types of trajectory-level features: TDF and THF.

We can use both these features for transportation mode detection. This is a classic classification problem in which many state-of-the-art classifiers can be utilized. In this paper, we use a deep neural network for classification; the network structure is shown in Fig. 3.

In addition to the input layer, the network comprises two fully connected hidden layers, each of which contain 256 neurons. We adopt ReLU as the activation function, the regularized least-squares method as the loss function, and the momentum-gradient descent method as the optimization method. The five-class softmax output layer is selected. The network can further extract a deeper relationship from the two trajectory-level features and build a model for transportation mode detection.

5. Experiments

5.1 Dataset

The experiments are performed on the Microsoft GeoLife data defined in the papers [4]. We use the sliding-window approach to segment the trajectory. Each segment has 500 sampling points; moreover, the adjacent segments are independent and no share points exist. The transportation mode of each segment is marked by the label files.

5.2 Experimental Result

We compare the following various feature combinations and classification methods in our experiments, and use accuracy as an evaluation measure:

<table>
<thead>
<tr>
<th>Feature combinations:</th>
<th>THF</th>
<th>TDF</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification methods:</td>
<td>logistic regression (LR), support vector machine (SVM), decision tree (DT) and deep neural network (DNN).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We adopt a five-fold cross validation process over 10 randomized experimental runs. The experiment environment comprises Ubuntu 14.04, CUDA 7.5, scikit-learn, and TensorFlow. The radial basis function (RBF) kernel is used in support vector machine (SVM).

5.2.1 Feature Analysis

The results obtained with various feature combinations show that TDF perform better than THF in both pooling strategies in most situations. This indicates our model can extract effective features. Compared to a single feature, the TDF + THF combination increases the accuracy by 2%–6%, indicating that using both features together significantly increases the accuracy of transportation mode detection.

The results of the pooling strategy show that THF are not affected by the pooling strategy and that the detection accuracy remains the same. Conversely, TDF are significantly affected by the pooling strategy, and the detection accuracy under the max pooling strategy in this case increases by 6% when compared with the mean pooling strategy. In the TDF + THF combination, the accuracy of both strategies are similar because the inclusion of THF reduces the impact of the pooling strategy.

5.2.2 Analysis of the Detection Method

The results obtained using various feature combinations show that when THF are used, the maximum accuracy of 68.3% is obtained with the DNN method and the minimum accuracy of 60.7% with the SVM method. When only TDF are used, the maximum accuracy of 72.2% is obtained with the DNN method (max pooling) and the minimum accuracy of 24.9% is obtained with the SVM method (mean pooling). When both features are used, the maximum accuracy of 74.1% is obtained with the DNN method (max pooling) and the minimum accuracy of 25.1% is obtained with SVM method (mean pooling). Clearly, the DNN method is more capable of extracting the intrinsic relationship within features.

The results for the pooling strategy show that when using the mean pooling strategy, the SVM method is signif-

<table>
<thead>
<tr>
<th>Table 1 Comparison of feature combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max_Pool</td>
</tr>
<tr>
<td>THF</td>
</tr>
<tr>
<td>TDF</td>
</tr>
<tr>
<td>Both</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2 Comparison of the classification methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
</tr>
<tr>
<td>THF</td>
</tr>
<tr>
<td>Max</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>DT</td>
</tr>
<tr>
<td>THF</td>
</tr>
<tr>
<td>Max</td>
</tr>
<tr>
<td>Mean</td>
</tr>
</tbody>
</table>
icantly affected and the detection accuracy decreases substantially. In the other methods, the accuracy increases with the addition of TDF. The maximum accuracy of 74.1% is obtained with the DNN method (THF + TDF) and the minimum accuracy of 24.9% with the SVM method (TDF). Under the max pooling strategy, the maximum accuracy of 74.1% is obtained with the DNN method (THF + TDF) and the minimum accuracy of 51.2% with the SVM method (TDF). Clearly, the DNN method is stable and minimally affected by the pooling strategy, whereas the SVM method is significantly affected by the pooling strategy.

5.2.3 Data Scale Effect Analysis

As shown in Fig. 4, with an increase of data scale in training, the detection accuracy is improved. This shows the advantage of using a larger training set.

5.2.4 Training Time Analysis

As shown in Fig. 5, with an increase in training data, the training time of all the methods increases. The training time for the DT method increases very quickly, much faster than that of the other methods. The increase for the LR method is the most moderate among the three methods. The training time of our DNN method increases in a reasonable way, slightly faster than that of the LR method and significantly slower than that of the DT method. The results show that our method maintains a good balance between the data scale and training time.

6. Conclusion

We propose a deep learning model to detect transportation modes on a GPS trajectory. First, our model extract point-level deep features from point-level handcrafted features. It then use a CNN to aggregate point-level deep features and generate a trajectory-level deep feature. Finally, we select a DNN as the classifier to detect the transportation mode using both trajectory-level features. The experimental results show that our method can automatically extract effective deep features from GPS data and significantly improve the accuracy of transportation mode detection. In the future, we will study how to extract more effective features using the deep learning method.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61672040).

References