DRFMM: a map-matching algorithm based on distributed random forest multi-classification

GuangLin Zhou and Feng Chen
Department of Automation, School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China
jameslin@mail.ustc.edu.cn

Abstract. In the vehicle navigation system, the vehicle movement trajectory displayed on the electronic map reflects the results of real-time positioning by the GPS measuring device. Map matching is the process of matching a series of GPS coordinates to an electronic map to find the true path of the trajectory. In this study, a distributed random forest map-matching algorithm (DRFMM) is proposed. The point-line matching method is used as the basic feature. The random forest algorithm in the Spark platform is used to train the historical data. The road network data is meshed, and a multi-classification model is trained offline for each grid in the road network to predict the FCD data online. The experimental results show that the DRFMM algorithm proposed in this paper has improved the accuracy of point-line matching by 10%. The multi-classification method keeps the matching accuracy with the increase of data volume. At the same time, with multi-threading and distributed platform, DRFMM's matching speed is nearly 6 times faster than stand-alone matching algorithm.

1. Introduction
With the accelerating process of urbanization, the number of road vehicles has increased sharply and traffic congestion has become increasingly serious. With the gradual deepening of research on traffic, the Intelligent Transportation System (ITS) emerged as the times require.

Floating Car Data has become an advanced collection method for urban road traffic conditions in ITS. FCD data includes GPS positioning data of floating vehicles, driving direction angle, and time information, which can obtain the speed of the road segment and traffic flow parameters.

Map matching is an important step in FCD processing, which directly affects the accuracy of traffic information acquisition. Due to the default error in positioning device, it often leads to a situation where the vehicle trajectory and the actual road are greatly deviated on the electronic map. Therefore, map matching is needed to avoid errors in vehicle positioning.

The rest of the article is organized as follows: Section 2 discusses the related terms and Section 3 discusses related work. The DRFMM model will be introduced in detail in Section 4. We do some experimental verification and analysis in Section 5. Finally, we conclude the paper in Section 6.

2. Related Terms
2.1. Road Segment
A road segment $e$ is a directed edge with a list of number $e.id$, travel speed $e.v$, length $e.l$, start point $e.start$, end point $e.end$, and intermediate points. One road can contain multiple road segments.
2.2. Road Network
The road network is a directed graph $G(V, E)$. $V$ is a set of vertices, representing the intersection or turning points of the road. $E = \{v_i \rightarrow v_j \mid v_i, v_j \in V\}$ that is a series of directed edges, representing the road segments in the road network.

2.3. GPS trajectory
The GPS trajectory is a series of time series data $T: p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$ that are sequentially transmitted by the same car over a period of time.

3. Related Work
According to the application environment, the map matching algorithm can be divided into geometric/incremental matching and global matching algorithms.

The geometric/incremental map matching algorithm [4][7][8] is represented by the geometric map matching method. The point-to-line matching algorithm can find the correct road segment more accurately by calculating the distance from the point to the road segment. This method ensures a certain accuracy and matching speed. The global matching algorithm [1][3][9] attempts to find a path in the road network to be closest to the floating car trajectory. Yu Zheng et al [2] proposed a new global matching algorithm ST-Matching for low sampling rate GPS trajectories.

In general, the incremental algorithm has better performance than the global algorithm in terms of algorithm running time, and the global algorithm can obtain more accurate matching results.

In order to obtain a faster matching speed, Renchu Song [5] achieved map matching through multicore CPU to greatly speed up the matching speed. Antonio [6] proposed a distributed map matching algorithm using MapReduce computing framework to improve the speed of map matching.

4. Map Matching Method Based On Distributed Machine Learning

4.1. Road Network Mesh
The road network data used in this paper is about Hefei city downloaded from OpenStreetMap. We use the xml.etree library of Python to extract the required road segment and node information.

In the classification algorithm, the more training categories, the longer the training and predicting cost. And the extracted road network contains 11981 road segments. Therefore, we use meshing to process road network data, and operations within each grid can be processed in parallel.

We assume that the road network is $L \times H$ and the study divides the entire map into $M \times N$ grids of the same size. The coordinates of the lower left corner of the road network map is $p_0(\text{lon}_0, \text{lat}_0)$. The coordinate of any point in the map is $p(\text{lon}, \text{lat})$, then the grid ID to which the point belongs can be calculated by the following formula:

$$ ID = \text{int} \left( \frac{N \times (\text{lon} - \text{lon}_0)}{L} \right) + \text{int} \left( \frac{M \times (\text{lat} - \text{lat}_0)}{H} \right) \times N \quad (1) $$

4.2. Constructing train/test dataset

4.2.1. Distance from point to road segment. In order to measure the distance from the GPS point $p_i$ to a road segment $e$, $p'_i$ is the projection point of the point $p_i$ to the road segment $e$. We calculate the distance using the following formula:

$$ d(p_i, e) = \begin{cases} d(p_i, p'_i), & \text{if } p'_i \in E_i \\ \min\{d(p'_i, e.\text{start}), d(p'_i, e.\text{end})\}, & \text{otherwise} \end{cases} \quad (2) $$
4.2.2. **Preprocess.** The original data set used in the train/test set is the full-day taxi driving data of Hefei City in March, 7th, 2017. The data includes taxi number, license plate number, driving time, passenger status, latitude, longitude, and road name, totally 3,460,564 travel data.

The driving data that is not in the range of the road network will be removed, establishing the mapping relationship between the road name in FCD and the road segments in the road data.

For the GPS point $p_i$ of the sample data, we calculate the corresponding grid number using formula (1), and the distance of the point $p_i$ with all the segments in the grid by using formula (2). The distance is used as a feature of the training/test set, using the real road segment ID as the training set label.

Algorithm 1 Data Preprocess

**Input:** original road data hefei.osm and trajectory dataset hefei_taxi.sql  
**Output:** roadmap data-roadDataByGrid and train/test dataset

1. **function** preprocess()  
   - roadSectionNodeTable ← create road,section and node table using python xml.tree library  
   - for each record in roadSectionNodeTable do  
     - gridIndex ← calculate gridIndex corresponding to record using formula (1)  
     - roadDataByGrid[gridIndex].append(record)  
   - end for  
2. end function
3. for data in dataset do  
   - filter data if not in map range or not match map road name  
   - gridIndex ← calculate gridIndex using formula (1)  
   - for secID in gridMapInfo[gridIndex] do  
     - train/test ← distance feature from trajectory point to secID using formula (2)  
   - end for  
4. end for  
5. **end for**  
6. **return** roadDataByGrid, train, test  
7. **end function**

4.2.3. **Random forest multi-classification algorithm.** This paper hopes to improve the matching accuracy and speed up the matching process. The random forest algorithm is more suitable for multi-classification problems. The training and prediction speed is fast and it is easy to achieve parallelization. Therefore, this paper chooses random forest algorithm to train multi-classification model in each grid.

Algorithm 2 Random Forest Multi-Classification

**Input:** train dataset  
**Output:** multi-classification model in each grid

1. **function** randomForest()  
   - for train of each grid do  
     - for $i = 1, 2, \ldots, N_{tree}$  
     - sample original train data using bootstrap and generate train‘  
     - randomly select M features from train‘ and use gini index to select optimal feature to split tree  
     - generate random forest model $\{h_i, i = 1, 2, \ldots, N_{tree}\}$  
   - end for  
   - end for  
2. end function
4.3. Distributed implementation details
The Spark platform has many ecological environment tools and integrates many machine learning libraries. We choose Spark as the distributed computing platform.

The overall flow chart of this article is as follows:

![Flow Chart](image)

Figure 1. Overall architecture of the DRFMM model

The offline training phase is model-based parallel processing that multiple nodes simultaneously train some parameters of a single model. After training, we save the corresponding model of the grid in the HDFS distributed file system. The online prediction phase is Data-based parallelism, based on the Python multi-threaded programming and Spark's parallel framework, each sample of the test set is allocated in parallel to its grid, and the offline trained model is used to predict the segment where it is located.

5. Experimental verification and analysis
The pre-processed test set format example is as follows. The example is the corresponding test set in the NO.74645 grid, and the label of the training set is the real Section ID.

| latitude  | longitude | Sec9870 | Sec9871 | ... | Sec11080 | Sec11081 | Sec11082 | label |
|-----------|-----------|---------|---------|------|----------|----------|----------|-------|
| 31.980185 | 117.251238| 8.241   | 12.978  | ...  | 18.71    | 18.73    | 124.70   |       |
| 31.980162 | 117.251320| 9.917   | 14.647  | ...  | 20.38    | 20.40    | 123.03   |       |
| ...       | ...       | ...     | ...     | ...  | ...      | ...      | ...      | ...   |
| 31.980667 | 117.251517| 20.157  | 15.209  | ...  | 16.354   | 16.371   | 158.962  |       |

5.1. Experimental setup and result
We used eight LENOVO machines to build a Hadoop/Spark cluster. All code is written in Python and runs on a Spark cluster in the Yarn mode.

5.1.1. Accuracy. Firstly, we compare with the point-line to verify the accuracy of the distributed multi-classification map matching algorithm. Using three sets of experiments, we test the accuracy of the
algorithm using 10000, 30000, 50000 trajectory points, and the average of each set of experiments was taken as the final accuracy.

It is found from the experimental results that the accuracy of DRFMM is significantly higher than that of point-line matching, and as the matching trajectory points increase, the accuracy of point-line matching decreases, but the accuracy of DRFMM almost remains stable.

In order to verify that the multi-classification method maintains a stable accuracy when the trajectory points number increase, we add a neural network NN as the matching algorithm of the multi-classification model. It can be seen from the accuracy that when the trajectory increases, the NN method still maintains stable accuracy.

| Trajectories | Times | Node-Line Match | NN   | DRFMM |
|--------------|-------|-----------------|------|-------|
| 10000        | 1     | 72.19%          | 77.26% | 80.33%|
| 10000        | 2     | 71.87%          | 78.10% | 80.2% |
| 10000        | 3     | 72.08%          | 77.75% | 80.56%|
| **10000**    | **Mean** | **72.04%**     | **77.70%** | **80.36%**|
| 30000        | 1     | 70.38%          | 78.83% | 80.41%|
| 30000        | 2     | 70.6%           | 77.89% | 80.41%|
| 30000        | 3     | 70.41%          | 77.95% | 80.49%|
| **30000**    | **Mean** | **70.46%**     | **77.89%** | **80.43%**|
| 50000        | 1     | 69.26%          | 77.82% | 80.42%|
| 50000        | 2     | 69.17%          | 77.78% | 80.38%|
| 50000        | 3     | 68.57%          | 77.78% | 80.39%|
| **50000**    | **Mean** | **69%**        | **77.79%** | **80.40%**|

5.1.2. Distributed Map-Matching Speed. Based on Python multi-thread programming and Spark distributed computing framework, an 8-node Hadoop/Spark cluster is established. The experiment records the matching speed when the number of nodes is 1 (stand-alone), 3, 4, 5, 6, 7, and 8 respectively. We calculate the acceleration ratio of the system, and draw the following line chart in Figure 2.

It can be seen that the 8-node distributed matching algorithm only needs 12.08 seconds to match 51,545 trajectory points, which is nearly 6 times faster than the stand-alone matching speed.

6. Conclusion

Two main contributions of this paper is the distributed map matching algorithm based on random forest multi-classification model and using Spark paradigm to speed up matching speed.

The experimental results show that the DRFMM algorithm improves the matching accuracy by 10% compared with the traditional point-line matching algorithm. With the help of Python multi-threading and Spark distributed computing framework, the matching speed is improved by nearly 6 times compared with the stand-alone matching algorithm.

We will continue to expand our work in the following two directions. Firstly, the Spark distributed computing paradigm greatly reduces the matching time, but the system's acceleration ratio is still not good enough. It is necessary to explore the Spark parallel mechanism and improve the ratio. Secondly, we use the map data source—OpenStreetMap itself has some errors, and the error of the road network data will directly affect the matching result. We consider updating the road network data from the trajectory data to correct the error of the map data itself.
Figure 2. The run time and acceleration ratio of the system

References

[1] Alt H, Wenk. Matching planar maps. *In Journal of Algorithms* 49, 2, 262-283, 2003.

[2] Yu Zheng, Xing Xie. Map-Matching for Low-Sampling-Rate GPS Trajectories. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 352-361, 2009.

[3] Brakatsoulas, Pfoser. On map-matching vehicle tracking data. In Proceedings of the 31st international conference on very large data bases, 853-864, 2005.

[4] Sudarshan S. Chawathe. Segment-based Map Matching. In IEEE Symposium on Intelligent Vehicles Symposium, 2007.

[5] Renchu Song, Wei Lu. Quick Map Matching Using Multi-Core CPUs. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 605-608, 2012.

[6] Antonio, Maria. DMM: A Distributed Map-matching algorithm using the MapReduce Paradigm. IEEE 19th International Conference on Intelligent Transportation Systems(ITSC), 2016.

[7] J.S.Creenfeld. Matching GPS observations to locations on a digital map. Transportation Research Board 81st Annual Meeting, 2002.

[8] Wenk, Pfoser. Addressing the need for map-matching speed: localizing global curve-matching algorithms. In Proceedings of the 18th international Conference on Scientific and Statistical Database Management, 2006.

[9] Yin and Wolfson. A Weight-based map matching method in moving objects databases. In Proceedings of the 16th international Conference on Scientific and Statistical Database Management, 437, 2004.