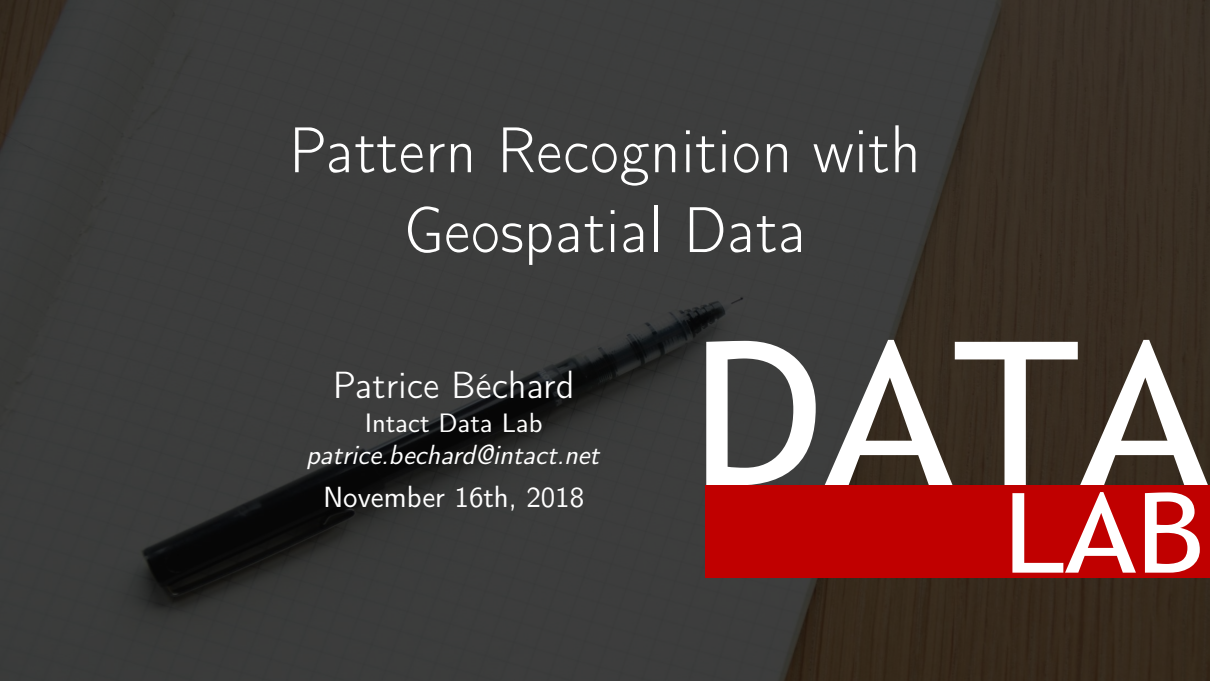
Pattern Recognition with Geospatial Data

Patrice Béchard
Intact Data Lab
patrice.bechard@intact.net

November 16th, 2018



Why is it an interesting problem ?

- Lots of GPS data from users
- May want to do things such as
  - Find patterns in the driving habits of users
  - Detect dangerous road sections
  - Optimize fastest route based on traffic
  - ...

Plan

Open Street Map

Building and visualizing road networks with OSMnx

GeoLife GPS Trajectories Dataset

Finding hotspots in Beijing

Predicting a driver's destination

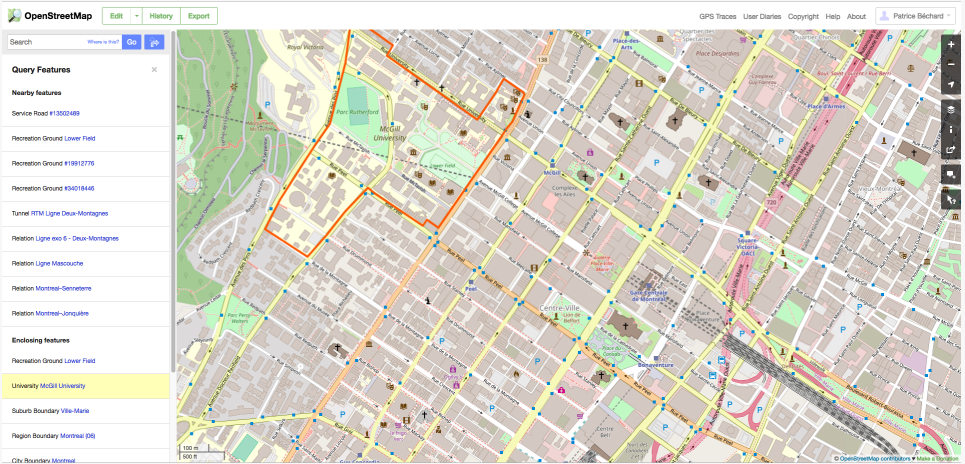
Geospatial Data Science at Intact

Open Street Map

- Open-source map maintained by users
- Contains various informations about :
  - road segments
  - intersections
  - landmarks
  - ...
- Contains a routing engine similar to Google Maps
- https://www.openstreetmap.org/

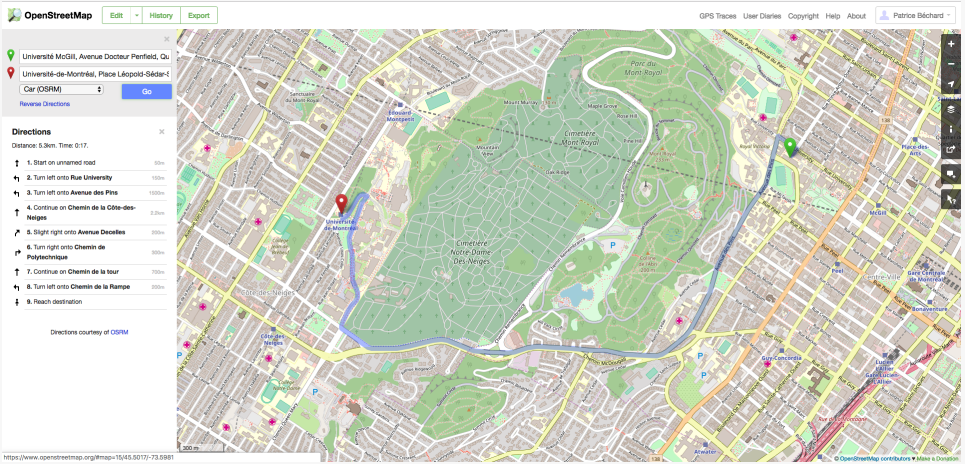
Open Street Map (OSM) [5]

Example : Querying features nearby



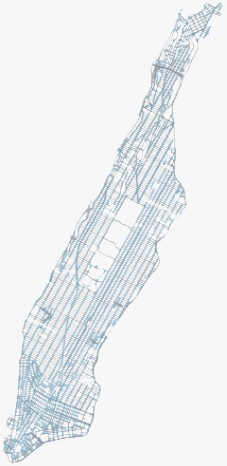
Open Street Map (OSM) [5]

Example : Find optimal route between two points



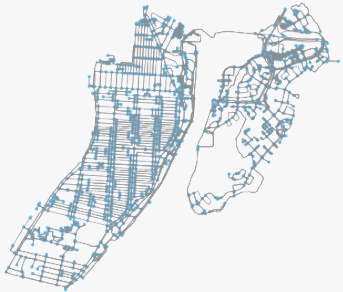
OSMnx Overview

- Open-source Python library
- Represents the road network as a directed
- Allows us to
  - Create the road network of a given location
  - Visualize this network easily
  - Simplify the road network by removing non-intersection nodes
  - Compute statistics about the road network
  - Find the shortest path between two nodes of the graph
  - ...
- https://github.com/gboeing/osmnx



Example : Creating the road network for Verdun

```
import osmnx as ox
G = ox.graph_from_place("Verdun, Montreal, Canada", network_type="all")
ox.plot_graph(G)
```



Example : Creating the shape of the Island of Montreal

```
import osmnx as ox
S = ox.gdf_from_place("Island of Montreal, Canada")
ox.plot_shape(S)
```

OSMnx [2]

Example : Creating a graph from a bounding box

```
import osmnx as ox
bbox = (45.52, 45.49, -73.55, -73.58)
G = ox.graph_from_bbox(bbox)
ox.plot_graph(G)
```

Example : Creating a graph from a single coordinate

```
import osmnx as ox
coord = (48.87378, 2.29504)
G = ox.graph_from_point(coord, distance=1000)
ox.plot_graph(G)
```

Example : Compute statistics about the network

```
import osmnx as ox
G = ox.graph_from_address("Arc de Triomphe, Paris")
stats = ox.basic_stats(G)
```

```
{
    "circuity_avg": 1.0267881322837478,
    "edge_length_avg": 54.606206202850004,
    "edge_length_total": 130290.40800000011,
    "intersection_count": 990,
    "k_avg": 4.156794425087108,
    ...
}
```

OSMnx [2]

Example : Finding the shortest path between two locations

```
import osmnx as ox
import networkx as nx

start_coord = (45.5049756, -73.5736905) # McGill University
end_coord = (45.5035380, -73.6176820) # Universite de Montreal
north, south, east, west = (45.5181450, 45.4854686, -73.5681800, -73.6279802)

G = ox.graph_from_bbox(north, south, east, west, network_type='drive')

start_node = ox.get_nearest_node(G, start_coord)
end_node = ox.get_nearest_node(G, end_coord)

route = nx.shortest_path(G, start_node, end_node)
ox.plot_graph_route(G, route)
```

OSMnx [2]

Example : Finding the shortest path between two locations

For more examples and things to do with OSMnx, check out these links :

- https://geoffboeing.com/2016/11/osmnx-python-street-networks/ (overview)
- https://osmnx.readthedocs.io/en/stable/ (documentation)
- https://github.com/gboeing/osmnx-examples/ (more examples)

The GeoLife GPS Trajectories Dataset

Dataset containing GPS trajectories from 181 users mostly around Beijing, China.

- **Number of unique trips** : 18,670
- **Total distance** : 1,292,951 km
- **Total duration** : 50,176 hours

For a full overview of the dataset :

- https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/User20Guide-1.2.pdf

Finding hotspots in Beijing

We can use trip origins and destinations to find the hotspots in Beijing.

- We use the GeoLife GPS Trajectories Dataset.
- We use the clustering algorithms from the *Scikit-Learn* python library[8].

What is clustering ?

- Type of unsupervised learning problem
- We try to find groups of data with similar properties.
- In our case, we want to find data points that are close to each other.

We decide to use the **DBSCAN**[4] algorithm for many reasons :

- The clusters may have any arbitrary shape
- No need to specify a number of clusters manually

Finding hotspots in Beijing

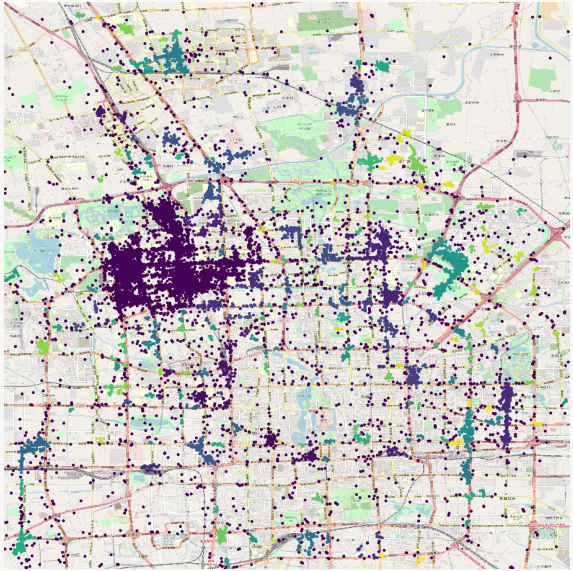
Task : find the 10 largest clusters in our data and identify them

To do :

1. Cluster the data using the DBSCAN algorithm
2. Removing points not in the 10 largest clusters
3. Find the centroid of each cluster
4. Use reverse geocoding to find around what landmark are the clusters positioned

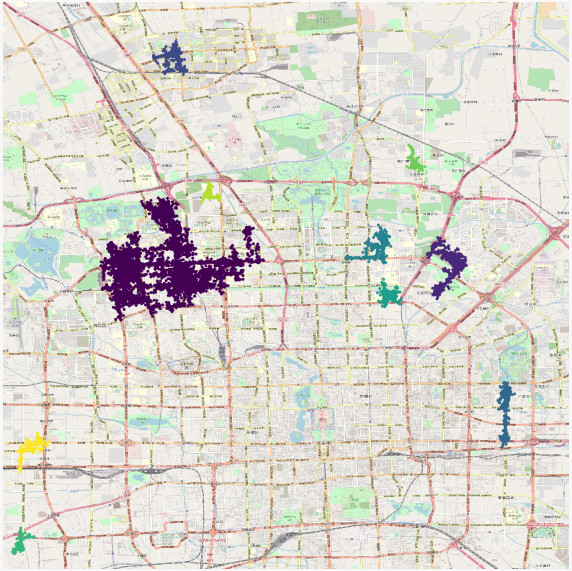
Finding hotspots in Beijing

1. Cluster the data using the DBSCAN algorithm



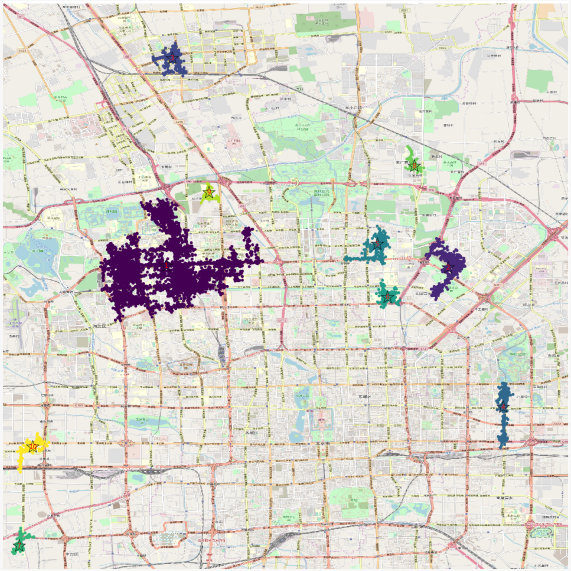
Finding hotspots in Beijing

2. Removing points not in the 10 largest clusters



Finding hotspots in Beijing

3. Find the centroid of each cluster



Finding hotspots in Beijing

4. Use reverse geocoding to find around what landmark are the clusters positioned

```
Centroid 1 : Zhongguancunnaner St, Zhongguancun Haidian District, China, China
Centroid 2 : Xibahe Rd, Chaoyang Qu, China, China
Centroid 3 : Huilongguan West Ave, Changping District, China, China
Centroid 4 : Jintai North St, Hujialou Chaoyang Qu, China, China
Centroid 5 : Beiyuan Rd, Asian Sports Village Chaoyang Qu, China, China
Centroid 6 : N 3rd Ring East Side Rd, Chaoyang Qu, China, China
Centroid 7 : W Wulidian, Fengtai, China, China
Centroid 8 : Hongjunying Rd, Chaoyang Qu, China, China
Centroid 9 : Houbajia East Rd, Haidian District, China, China
Centroid 10 : Beitaiping Rd, Haidian District, China, China
```

Possible improvements

- Make smaller clusters to improve reverse geocoding results
- Find a way to find an actual landmark, not only the name of a street
- Check if the clusters change depending on the hour of the day, the day of the week, ...
- Use clusters to determine frequent locations of individuals

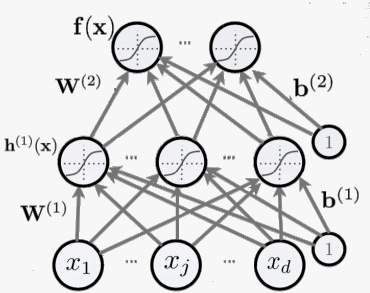
Predicting a driver's destination

We can try to predict a driver's destination based on the beginning of their trip.

- Many people have tried to tackle this problem with various degree of success [3, 6].
- We will use a LSTM network to do the prediction.

Quick introduction to neural networks

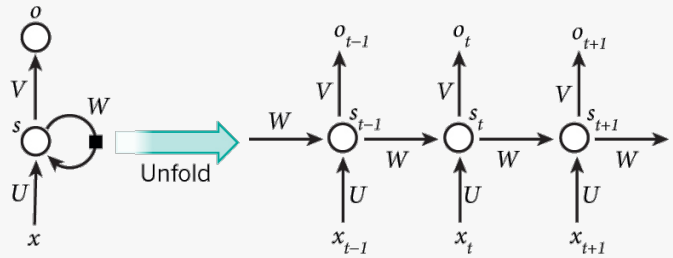
- Machine learning model allowing us to learn non-linear functions
- Can be used for regression (target $\in \mathbb{R}$) and classification (target $\in \{1, \ldots, N\}$).



For more details : http://cs231n.github.io/neural-networks-1/#nn

Quick introduction to neural networks

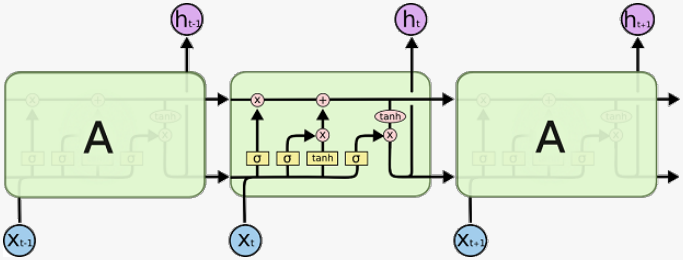
- **Recurrent neural networks** can take variable length sequences as inputs.
- Ideal when the data presents temporal dependencies
- **Limitations** : Can have a hard time learning long-term dependencies



For more details : http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Quick introduction to neural networks

- **Long Short-Term Memory**(LSTM) networks can learn long-term dependencies better.

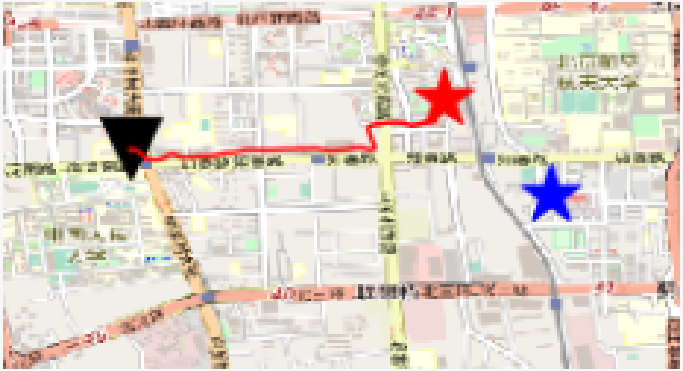


For more details : https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Our approach

- We feed a variable length sequence of *(lat, long)* tuples to the model
- We feed half of the trip in order to make the prediction
- We try to predict the *(lat, long)* tuple of the destination
- This is a regression problem
- We optimize the model based on the distance between the prediction and the actual destination

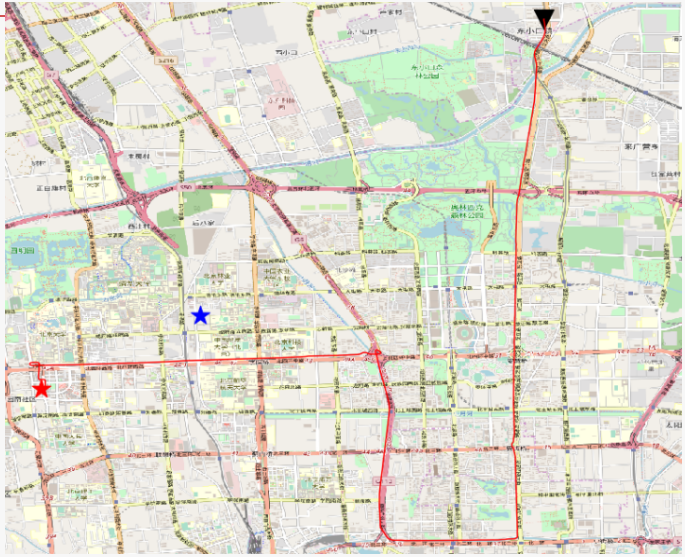
Predicting a driver's destination



Predicting a driver's destination

Predicting a driver's destination

Predicting a driver's destination



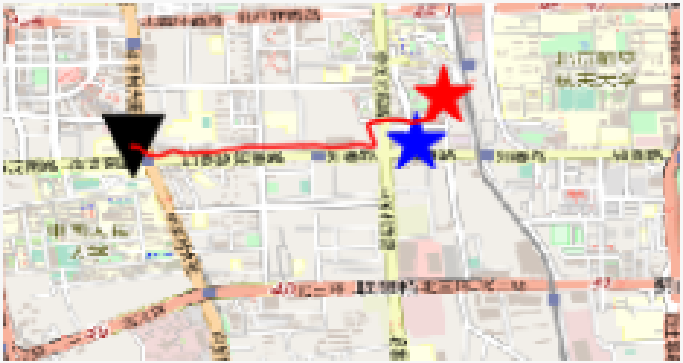
Problems

- The model always predicts points in the largest cluster of destinations seen earlier.
- This is the "Safe bet" in order to reduce the distance between the predicted destination and the actual destination.
- The model has no information about the road network where every trajectory happens.

Our second try

- We feed a variable length sequence of *(lat, long)* tuples to the model
- We feed half of the trip in order to make the prediction
- We try to predict the node of the road network closest to the destination.
- This becomes a classification problem

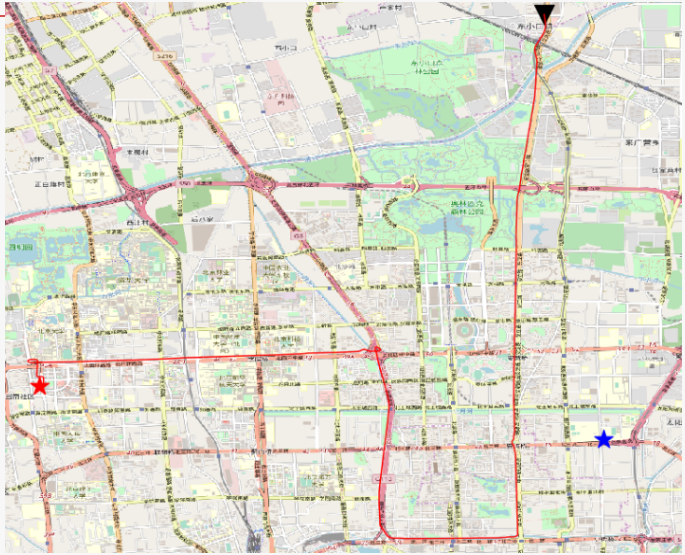
Predicting a driver's destination



Predicting a driver's destination

Predicting a driver's destination

Predicting a driver's destination



Possible improvements

- Add time of the day and day of the week as features for each point.
- Use nodes instead of raw GPS points as inputs (node embeddings)
- Characterize how the error evolves depending on what fraction of the trip we use

Geospatial Data Science at Intact

- The UBI (*Usage Based Insurance*) team at Intact tries to better understand the behavior of users based on their driving habits.
- We deal with huge amounts of geospatial data in order to :
  - Detect dangerous behavior on the road
  - Detect dangerous streets where accidents are more frequent
  - Help users improve their driving habits via a mobile app
- We are always looking for interns !

References I

[1] D. Ashbrook and T. Starner, *Using gps to learn significant locations and predict movement across multiple users*, Personal and Ubiquitous computing, 7 (2003), pp. 275–286.

[2] G. Boeing, *Osmnx : New methods for acquiring, constructing, analyzing, and visualizing complex street networks*, Computers, Environment and Urban Systems, 65 (2017), pp. 126–139.

[3] A. De Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, *Artificial neural networks applied to taxi destination prediction*, arXiv preprint arXiv :1508.00021, (2015).

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., *A density-based algorithm for discovering clusters in large spatial databases with noise.*, in Kdd, vol. 96, 1996, pp. 226–231.

References II

[5] M. Haklay and P. Weber, *Openstreetmap : User-generated street maps*, Ieee Pervas Comput, 7 (2008), pp. 12–18.

[6] J. Krumm and E. Horvitz, *Predestination : Inferring destinations from partial trajectories*, in International Conference on Ubiquitous Computing, Springer, 2006, pp. 243–260.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch*, (2017).

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., *Scikit-learn : Machine learning in python*, Journal of machine learning research, 12 (2011), pp. 2825–2830.

References III

[9] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, *Understanding mobility based on gps data*, in Proceedings of the 10th international conference on Ubiquitous computing, ACM, 2008, pp. 312–321.

[10] Y. Zheng, X. Xie, and W.-Y. Ma, *Geolife : A collaborative social networking service among user, location and trajectory.*, IEEE Data Eng. Bull., 33 (2010), pp. 32–39.

[11] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, *Mining interesting locations and travel sequences from gps trajectories*, in Proceedings of the 18th international conference on World wide web, ACM, 2009, pp. 791–800.