

Pattern recognition avec des données géospatiales

Patrice Béchard
Intact Data Lab
patrice.becharde@intact.net
19 janvier 2019



Pourquoi est-ce un problème intéressant ?

- ▶ Beaucoup de données GPS provenant des usagers
- ▶ On voudrait les utiliser pour
 - ▶ Trouver des *patterns* dans les habitudes de conduite des usagers
 - ▶ Déetecter des sections de routes dangereuses
 - ▶ Optimiser le trajet le plus vide en fonction du traffic
 - ▶ ...

Plan

Open Street Map

Construire et visualiser le réseau routier avec OSMnx

GeoLife GPS Trajectories Dataset

Trouver les points d'intérêt à Beijing

Prédire la destination d'un conducteur

Science des données géospatiales chez Intact

Open Street Map

- ▶ Carte *Open-Source* maintenue par les utilisateurs
- ▶ Contient des informations à propos des :
 - ▶ segments de route
 - ▶ intersections
 - ▶ monuments
 - ▶ ...
- ▶ Contient un moteur de routage similaire à Google Maps
- ▶ <https://www.openstreetmap.org/>



Open Street Map (OSM) [5]

Exemple : requête des objets à proximité

The screenshot shows the OpenStreetMap interface with a map of Montreal. Several features are highlighted with colored outlines:

- A large red outline encloses a rectangular area in the upper left, covering parts of the Royal Victoria Hospital and McGill University.
- An orange outline encloses a larger area in the center, spanning from the Lower Field recreation ground to the Gare Centrale de Montréal.
- A yellow outline encloses a vertical strip along the right side of the map, roughly between Rue Peel and Rue Sainte-Catherine.
- A pink outline encloses a smaller area near the Place des Arts.
- A grey outline encloses a curved area along the bottom right, near the Bonaventure station.

The map also displays street names, landmarks, and other geographical features. A sidebar on the left lists "Query Features" such as "Service Road #13002489", "Recreation Ground Lower Field", and "University McGill University". A top navigation bar includes links for "Edit", "History", "Export", "GRS Traces", "User Diaries", "Copyright", "Help", "About", and a user profile for "Patrice Béchard".

Open Street Map (OSM) [5]

Exemple : Trouver le trajet optimal entre deux points

The screenshot shows a search interface for finding the optimal route between two locations on the OpenStreetMap website. The starting point is "Université McGill, Avenue Docteur Penfield, Québec" and the destination is "Université de Montréal, Place Léopold-Sédar-Senghor". The route calculated by OSRM consists of 9 segments, with a total distance of 5.3km and a time of 0:17. The route starts on an unnamed road, turns onto Rue University, then Avenue des Pins, then Chemin de la Côte-des-Neiges, then Avenue Decelles, then Chemin de la tour, then Chemin de la Rampe, and finally reaches the destination. The map also highlights the Parc du Mont-Royal and the Cimetière Notre-Dame-des-Vérités.

OpenStreetMap Edit History Export

Université McGill, Avenue Docteur Penfield, Québec
Université de Montréal, Place Léopold-Sédar-Senghor

Car (OSRM)

Go

Reverse Directions

Directions

Distance: 5.3km. Time: 0:17.

- Start on unnamed road
- Turn left onto Rue University
- Turn left onto Avenue des Pins
- Continue on Chemin de la Côte-des-Neiges
- Slight right onto Avenue Decelles
- Turn right onto Chemin de la tour
- Continue on Chemin de la Rampe
- Reach destination

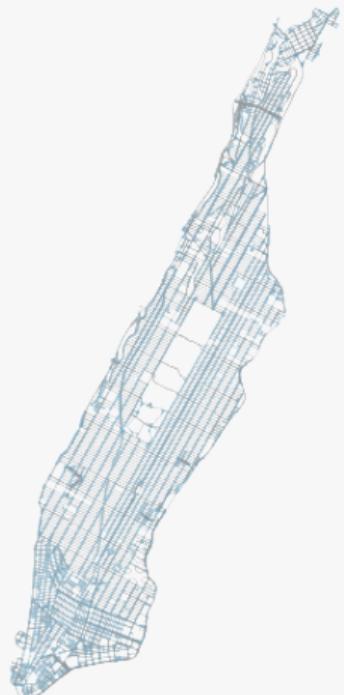
Directions courtesy of OSRM

<https://www.openstreetmap.org/#map=15/45.5017/-73.6981>

GPS Traces User Diaries Copyright Help About Patrice Béchard

OSMnx

- ▶ Librairie Python *open-source*
- ▶ Représente le réseau routier comme un graphe orienté
- ▶ Nous permet de :
 - ▶ Créer le réseau routier d'un endroit défini
 - ▶ Visualiser le réseau facilement
 - ▶ Simplifier le réseau routier en supprimant les noeuds n'étant pas des intersections
 - ▶ Calculer des statistiques à propos du réseau routier construit
 - ▶ Trouver le chemin le plus cours entre deux noeuds du réseau
 - ▶ ...
- ▶ <https://github.com/gboeing/osmnx>



OSMnx [2]

Exemple : Créer le réseau routier de Verdun

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



OSMnx [2]

Exemple : Créer la forme de l'Île de Montréal

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



OSMnx [2]

Exemple : Créer un graphe à partir d'un *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)
```

Exemple : Créer un graphe à partir d'un seul point GPS

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

OSMnx [2]

Exemple : Calculer les statistiques d'un réseau routier

```
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]

Exemple : Trouver le chemin le plus court entre deux endroits

```
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)
```

Exemple : Trouver le chemin le plus court entre deux endroits



OSMnx [2]

Pour plus d'exemples et de fonctionnalités de OSMnx, consultez ces liens :

- ▶ <https://geoffboeing.com/2016/11/osmnx-python-street-networks/> (aperçu)
- ▶ <https://osmnx.readthedocs.io/en/stable/> (documentation)
- ▶ <https://github.com/gboeing/osmnx-examples/> (plus d'examples)

GeoLife GPS Trajectories Dataset

Ensemble de données contenant les trajectoires GPS de 191 usagers, la plupart aux alentours de Beijing, Chine.

- ▶ **Nombre de trajets** : 18,670
- ▶ **Distance totale parcourue** : 1,292,951 km
- ▶ **Temps total** : 50,176 heures

Pour un aperçu complet de l'ensemble de données :

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

Trouver les points d'intérêt à Beijing

Trouver les points d'intérêt à Beijing

On peut utiliser l'origine et la destination des trajet pour trouver les points d'intérêt à Beijing.

- ▶ On utilise le GeoLife GPS Trajectories Dataset.
- ▶ On utilise les algorithmes de *clustering* de la librairie python *Scikit-Learn*[8].

Trouver les points d'intérêt à Beijing

Qu'est-ce que le *clustering* ?

- ▶ Type de problème d'apprentissage non-supervisé
- ▶ On essaie de trouver des groupes de données avec des propriétés similaires.
- ▶ Dans notre cas, on veut trouver les points GPS étant proche de chacun.

On décide d'utiliser l'algorithme **DBSCAN**[4] pour plusieurs raisons :

- ▶ Les *clusters* peuvent être de forme arbitraire
- ▶ Pas besoin de spécifier un nombre de *clusters* manuellement

Trouver les points d'intérêt à Beijing

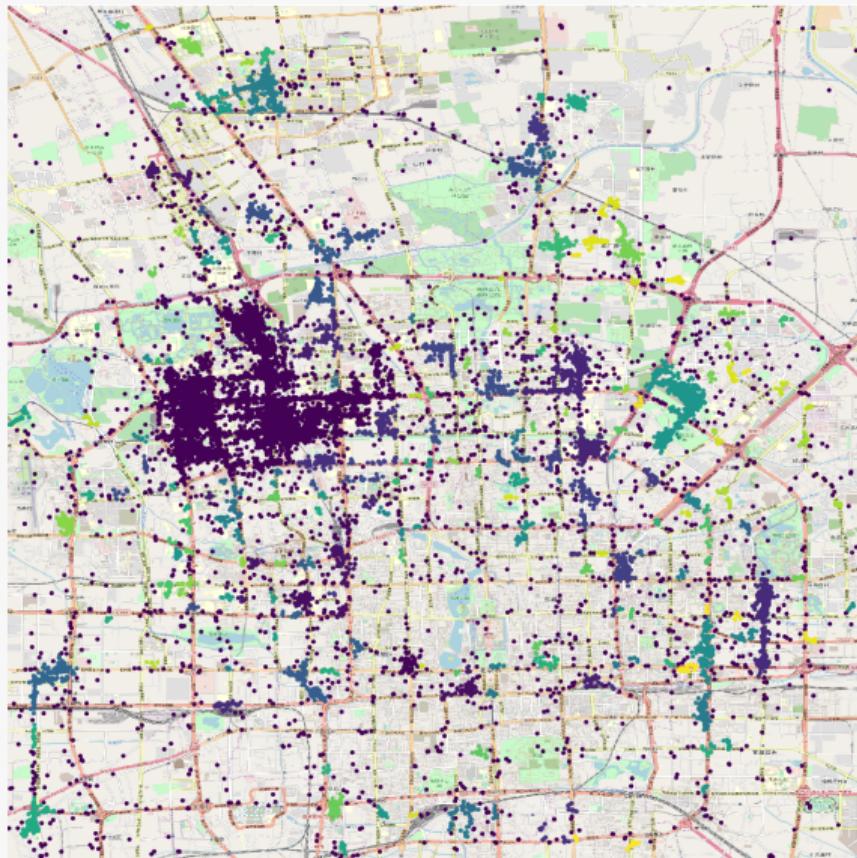
Tâche : trouver les 10 plus grands *clusters* dans l'ensemble de données et les identifier

À faire :

1. Regrouper les données avec l'algorithme DBSCAN
2. Enlever les points ne faisant pas partie des 10 plus grands *clusters*
3. Trouver le centroïde de chaque *cluster*
4. Utiliser le géocodage inversé (reverse geocoding) pour trouver les points d'intérêt proche des centroïdes

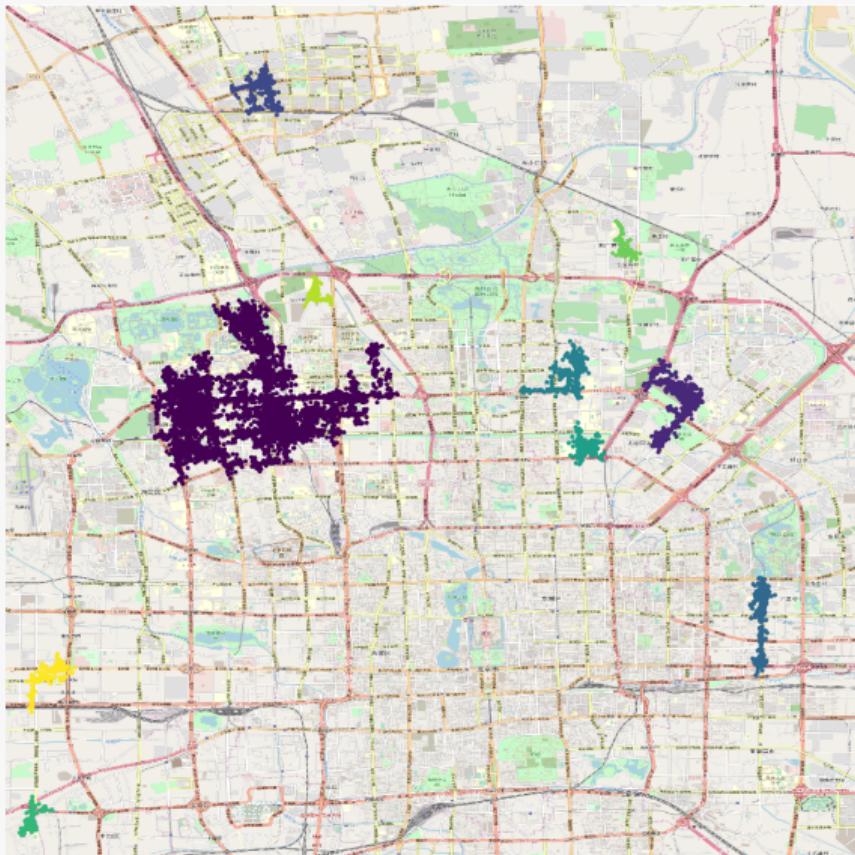
Trouver les points d'intérêt à Beijing

1. Regrouper les données avec l'algorithme DBSCAN



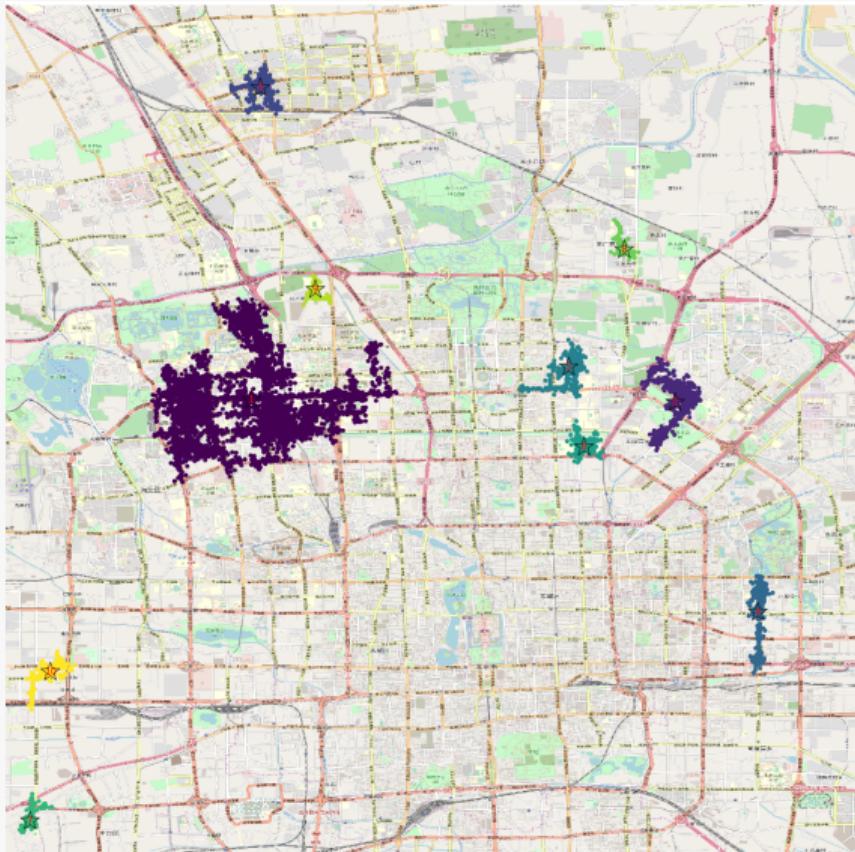
Trouver les points d'intérêt à Beijing

2. Enlever les points ne faisant pas partie des 10 plus grands *clusters*



Trouver les points d'intérêt à Beijing

3. Trouver le centroïde de chaque cluster



Trouver les points d'intérêt à Beijing

4. Utiliser le géocodage inversé (reverse geocoding) pour trouver les points d'intérêt proche des centroïdes

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

Améliorations possibles

- ▶ Faire des *clusters* plus petits pour améliorer les résultats du géocodage inversé
- ▶ Trouver une manière de trouver le point d'intérêt associé au *clusters* et pas seulement le nom de la rue
- ▶ Regarder si les *clusters* changent en fonction de l'heure de la journée, de la journée dans la semaine, ...
- ▶ Trouver les *clusters* associés à un seul usager pour trouver ses endroits souvent fréquentés (ex. sa maison, son travail, etc.)

Prédire la destination d'un conducteur

Prédire la destination d'un conducteur

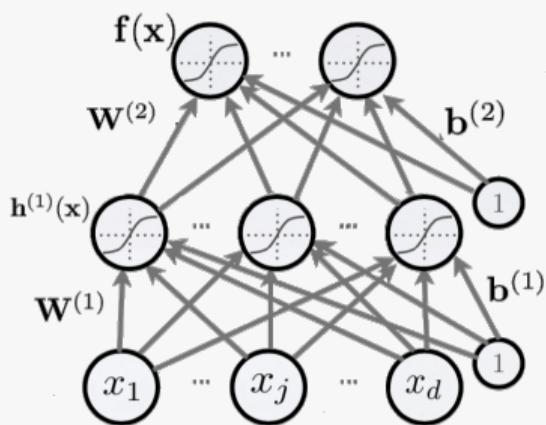
On peut essayer de prédire la destination d'un conducteur en fonction du début de son trajet.

- ▶ Plusieurs personnes ont essayé de s'attaquer à ce problème avec divers degré de succès. [3, 6].
- ▶ On va utiliser un réseau de neurones LSTM pour faire la prédiction.

Prédire la destination d'un conducteur

~~Introduction rapide aux réseaux de neurones~~

- ▶ Modèle d'apprentissage automatique nous permettant d'apprendre des fonctions non-linéaires
- ▶ Peut être utilisé pour la régression (cible $\in \mathbb{R}$) et la classification (cible $\in \{1, \dots, N\}$).

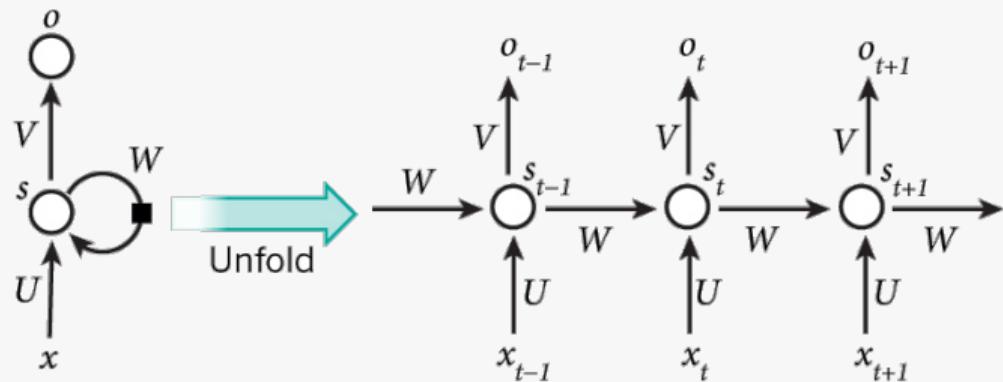


Pour plus de détails : <http://cs231n.github.io/neural-networks-1/#nn>

Prédire la destination d'un conducteur

~~Introduction rapide aux réseaux de neurones~~

- ▶ Les **réseaux de neurones récurrents (RNN)** peuvent prendre en entrée des séquences de longueur variable.
- ▶ Idéal lorsque les données présentent des dépendances temporelles
- ▶ **Limitations** : Peuvent avoir de la difficulté à apprendre des dépendances à long-terme

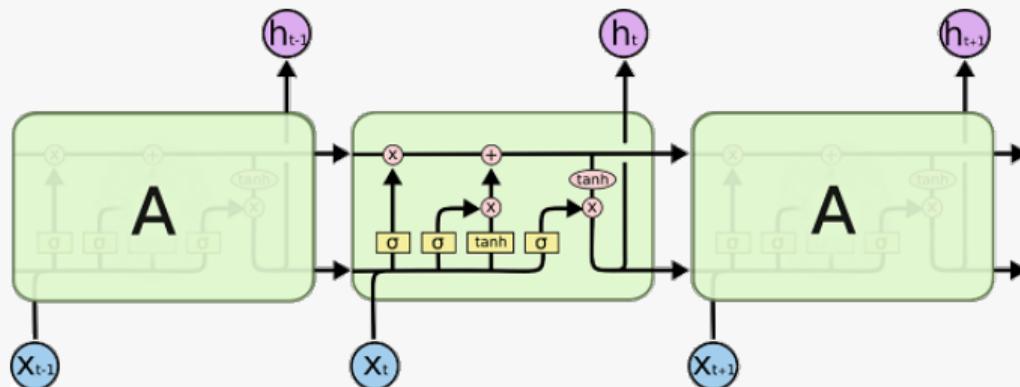


Pour plus de détails : <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Prédire la destination d'un conducteur

Introduction rapide aux réseaux de neurones

- ▶ Les réseaux **Long Short-Term Memory(LSTM)** peuvent apprendre ces dépendances à long-terme plus facilement.



Pour plus de détails :

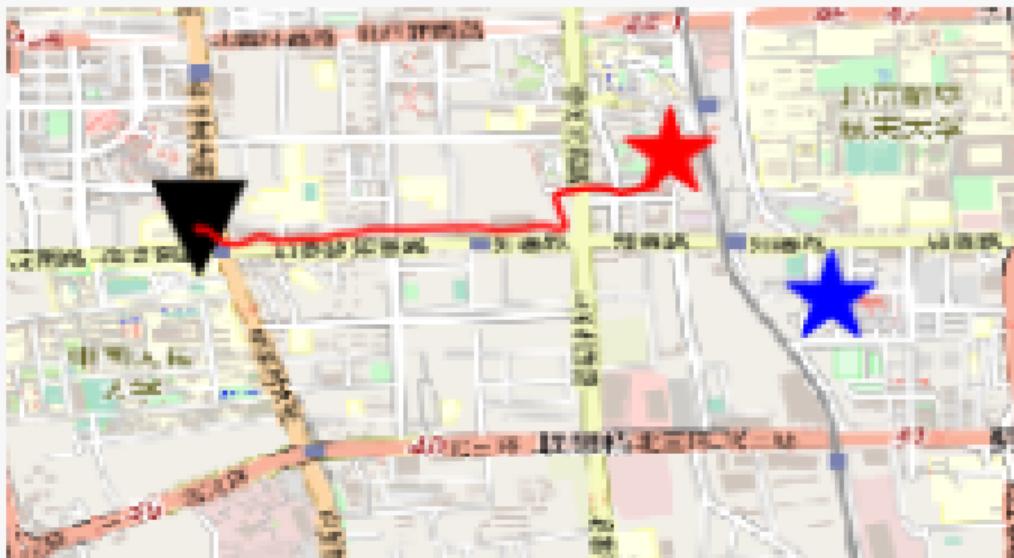
<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Prédire la destination d'un conducteur

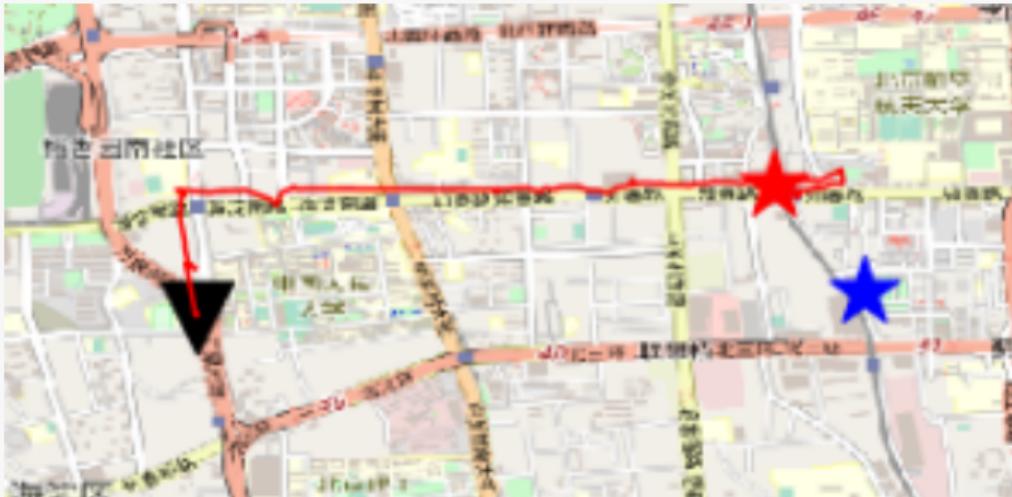
Notre approche

- ▶ On fournit en entrée une séquence de longueur variable de tuples (*lat, long*) au modèle.
- ▶ On fournit la moitié du trajet pour faire la prédiction.
- ▶ On essaie de prédire la position géographique (*lat, long*) de la destination.
- ▶ C'est un problème de régression.
- ▶ On optimise le modèle en essayant de minimiser la distance entre la destination prédite ainsi que la vraie destination.

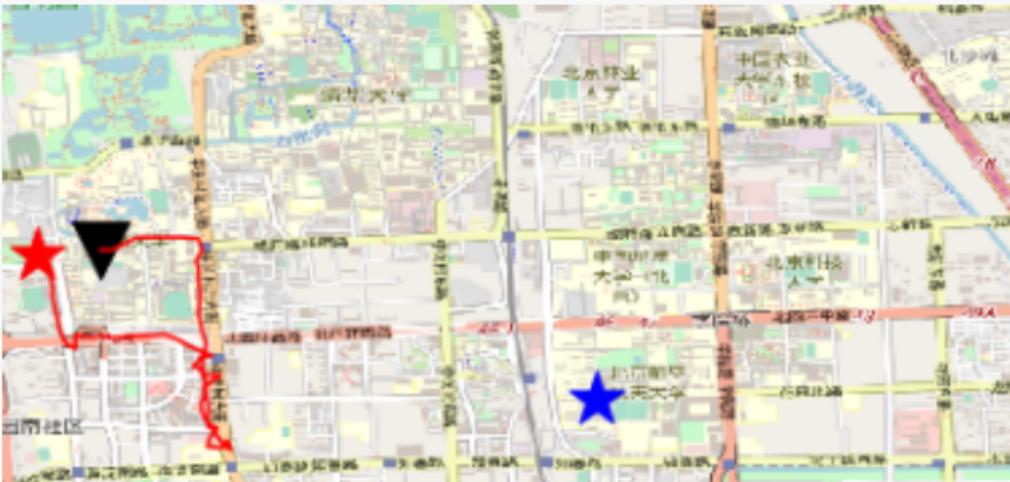
Predicting a driver's destination



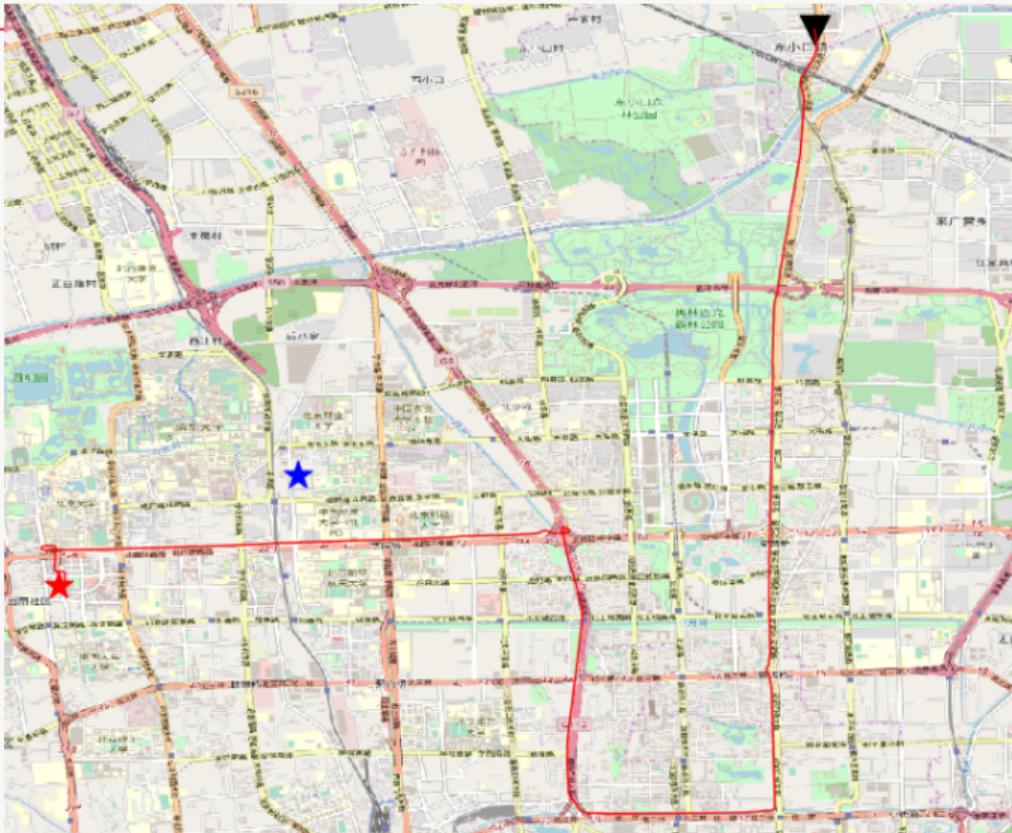
Prédire la destination d'un conducteur



Prédire la destination d'un conducteur



Prédire la destination d'un conducteur



Prédire la destination d'un conducteur

Problèmes

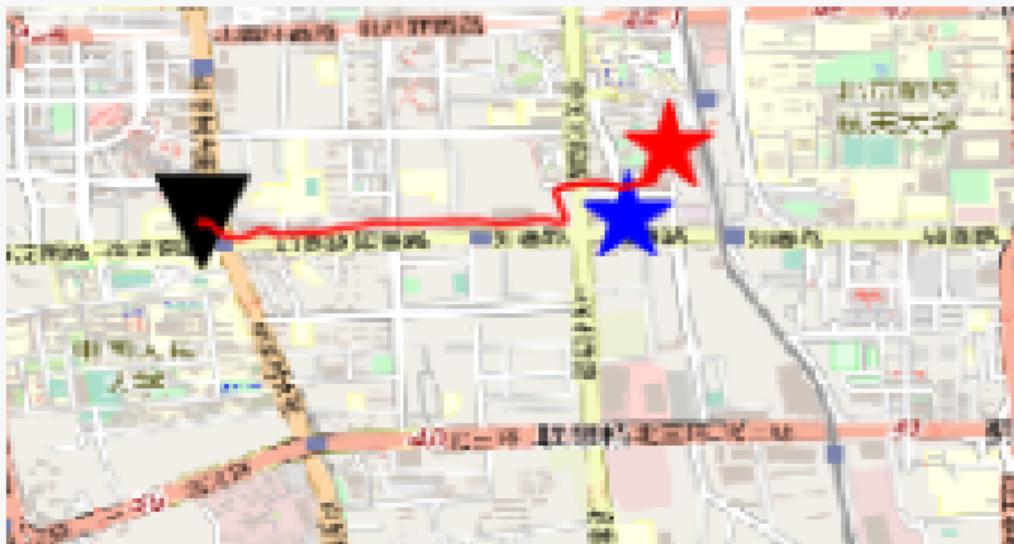
- ▶ Le modèle prédit toujours des points dans le plus grand *cluster* des destinations vu précédemment.
- ▶ C'est le "Safe bet" pour réduire la distance entre la destination prédite et la vraie destination.
- ▶ Le modèle n'a aucune information à propos du réseau routier à toutes les trajets se produisent.

Prédire la destination d'un conducteur

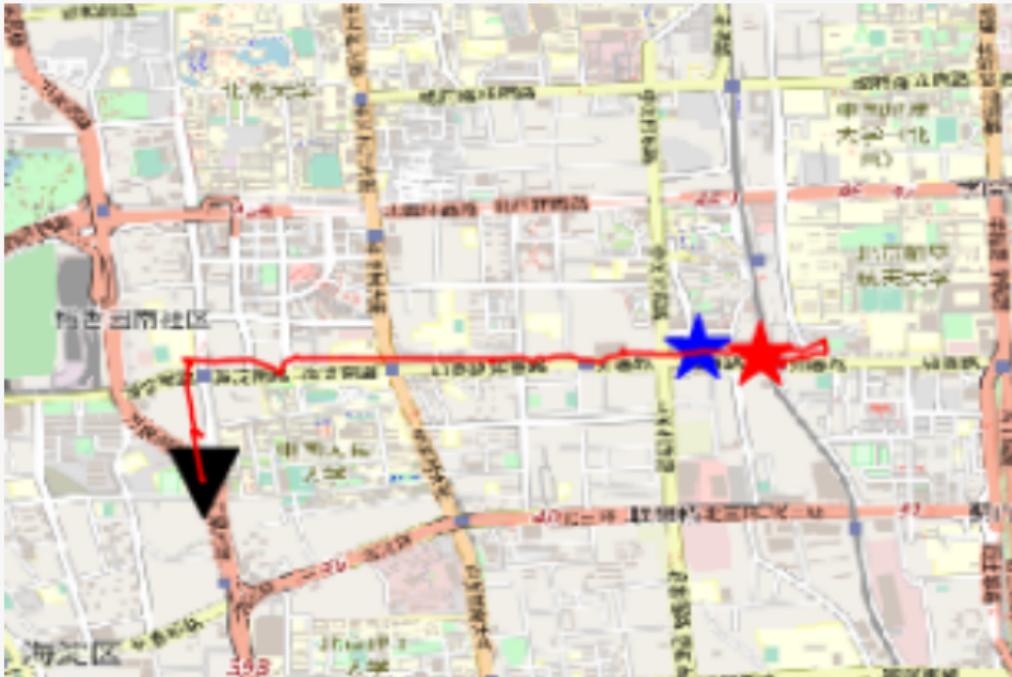
Notre deuxième essai

- ▶ On fournit en entrée une séquence de longueur de variable de tuples (lat, long) au modèle.
- ▶ On fournit la moitié du trajet pour faire la prédiction.
- ▶ On essaie de prédire le noeud du réseau routier le plus rapproché de la destination.
- ▶ C'est maintenant un problème de classification.

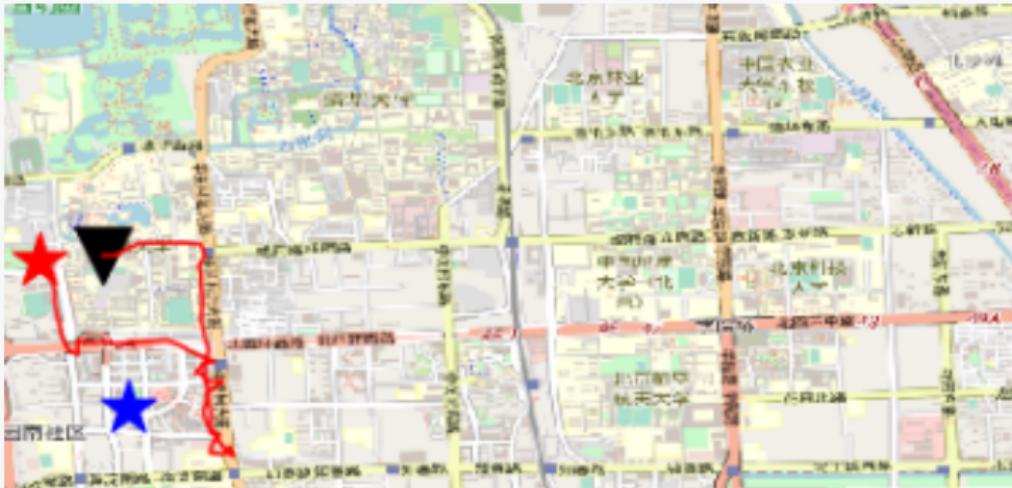
Prédire la destination d'un conducteur



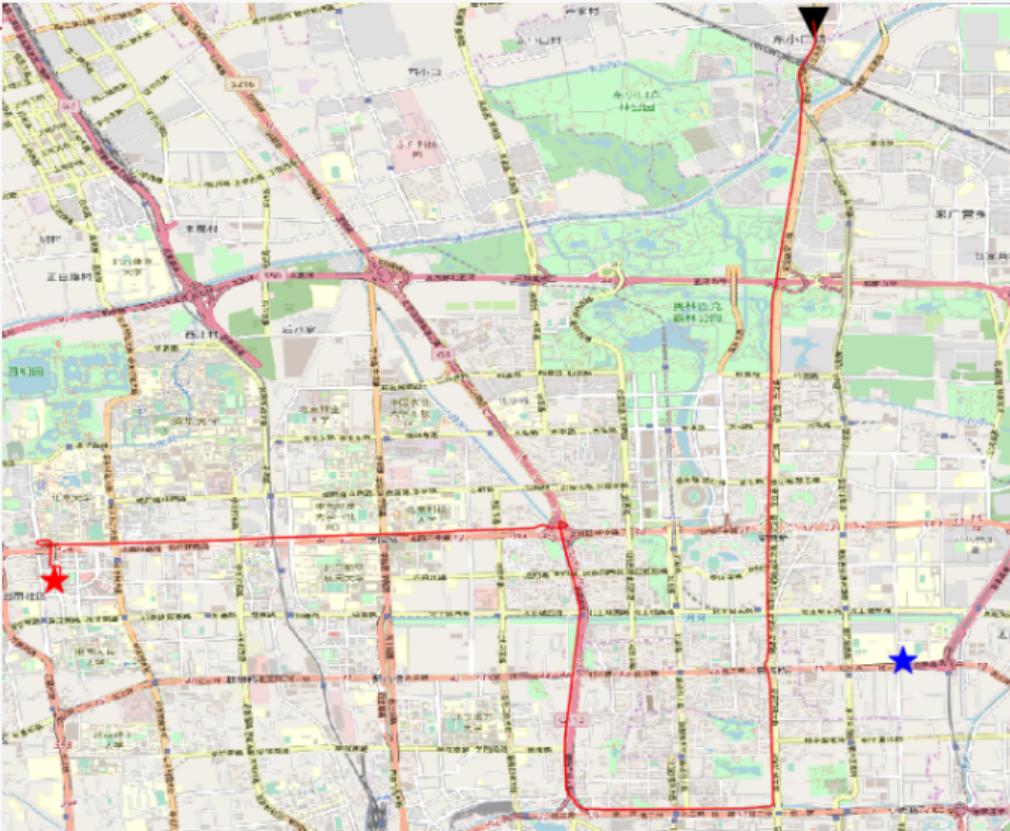
Prédire la destination d'un conducteur



Prédire la destination d'un conducteur



Prédire la destination d'un conducteur



Prédire la destination d'un conducteur

Améliorations possibles

- ▶ Ajouter le temps de la journée ou le jour de la semaine comme *feature* pour chaque point.
- ▶ Utiliser les noeuds du réseau routier au lieu des données GPS brutes comme entrée ("node embeddings")
- ▶ Caractériser comment l'erreur évolue en fonction de la fraction du trajet qu'on donne en entrée au modèle.

Science des données géospatiales chez Intact

- ▶ L'équipe UBI (*Usage Based Insurance*) chez Intact essaie de mieux comprendre le comportement des usagers de la route en fonction de leurs habitudes de conduite.
- ▶ On utilise de grandes quantités de données géospatiales pour :
 - ▶ Déetecter des comportements dangereux ou hors du commun sur la route
 - ▶ Déetecter des rues ou intersections dangereuses où les accidents sont plus fréquents
 - ▶ Aider les usagers à améliorer leur conduite via une application mobile
- ▶ Nous sommes toujours à la recherche de stagiaires !



Références 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.

Références 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.

Références 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.