Application du deep learning à la traduction automatisée

soutenance de stage

Alex AUVOLAT
Montreal Institute for Learning Algorithms
sous l'encadrement de Yoshua Bengio et Pascal Vincent

Avril - Août 2014

Plan

Section 1

Généralités sur le deep learning

Introduction

- Deep learning : réseaux de neurones, sous un nouveau nom
- Une méthode de base : la backpropagation (rétropropagation du gradient)
- Variations sur les architectures de réseaux utilisés : profondeur, largeur, récurrent, convolutionnel
- Capacité d'entraîner des réseaux très larges

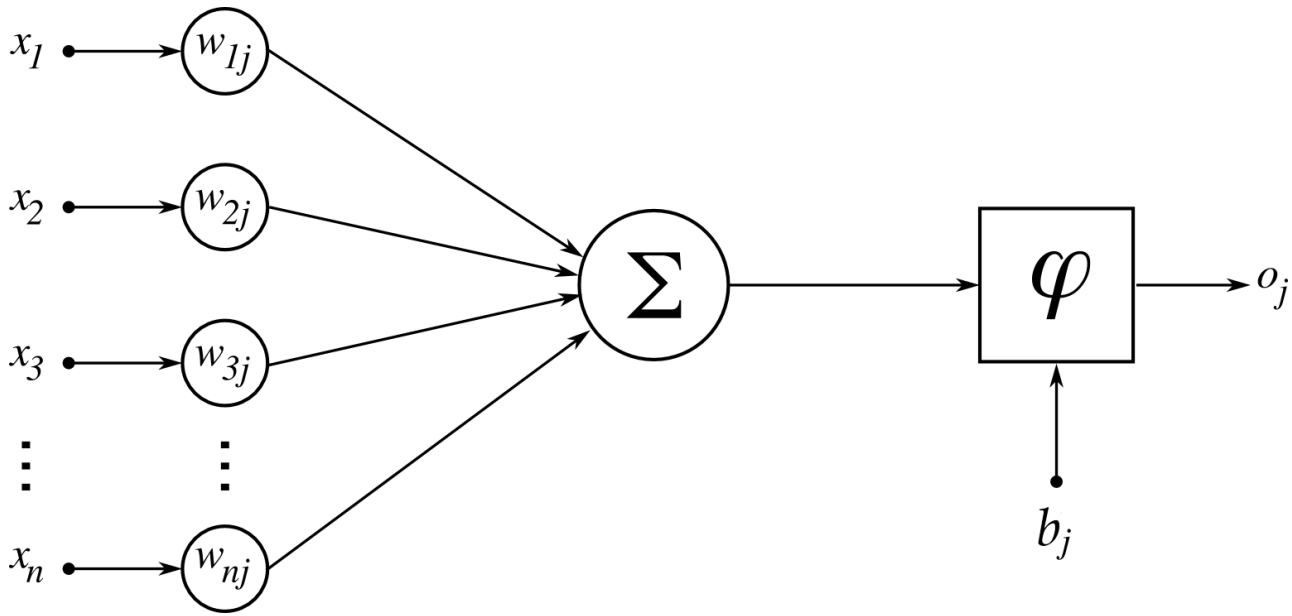
Fonctions paramétriques

Les réseaux de neurones sont une classe de fonctions paramétriques :

$$f_\theta : X \to Y$$

θ : vecteur de tous les paramètres de la fonction
f_θ peut être pensée comme un graphe de calcul (Theano), c'est-à-dire comme une composition de fonctions élémentaires

Neurones artificiels



Neurones artificiels

$$o_j = \varphi\left(x_1 w_{1j} + x_2 w_{2j} + \cdots + x_n w_{nj} + b_j\right)$$

Paramètres :

- w_{ij} : poids
- b_j : biais

Hyperparamètre :

- φ : fonction d'activation, généralement non-linéaire

Couches de neurones

Tous les neurones d'une couche ont les mêmes entrées, donc le calcul est essentiellement une multiplication matricielle :

$$o = \varphi(Wx + b)$$

- W : matrice de poids
- b : vecteur de biais
- φ : fonction d'activation point par point

Intérêt : une multiplication matricielle peut être parallélisée très rapidement sur un GPU.

Modèles profonds

- Un modèle avec une couche cachée peut approximer n'importe quelle fonction
- Dans certains cas, rajouter des couches permet de diminuer exponentiellement le nombres de neurones requis pour calculer une certaine fonction
- Problèmes de gradients qui explosent ou qui disparaissent, partiellement résolu avec la fonction d'activation $\varphi(x) = max(0, x)$

Mesure d'erreur

Dans le cas de l'apprentissage supervisé, on dispose d'un ensemble $\mathcal{T}$ de couples (x, y) qui servent à entraîner le modèle.
On définit une mesure d'erreur, ou fonction de coût :

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{T}} l(f_\theta(x), y)$$

- Classification : $f_\theta(x)$ est une distribution de probabilité sur les classes,
$$l(f_\theta(x), y) = -log(f_\theta(x)_y)$$
- Régression :
$$l(f_\theta(x), y) = ||f_\theta(x) - y||_2^2$$

Descente de gradient

On adapte itérativement les paramètres θ en suivant le gradient descendant du coût :

$$\theta \leftarrow \theta + \lambda \nabla_\theta \mathcal{L}$$

λ : taux d'apprentissage, à choisir avec précision.
Descente de gradient *stochastique* (SGD) : on utilise un seul exemple pour calculer le gradient :

$$\theta \leftarrow \theta + \lambda \nabla_\theta l(f_\theta(x), y)$$

Généralement on utilise une SGD avec un *minibatch* d'environ 100 exemples.
Alternatives : momentum, AdaDelta, Adam, ...

Avec Theano, la backprop c'est magique

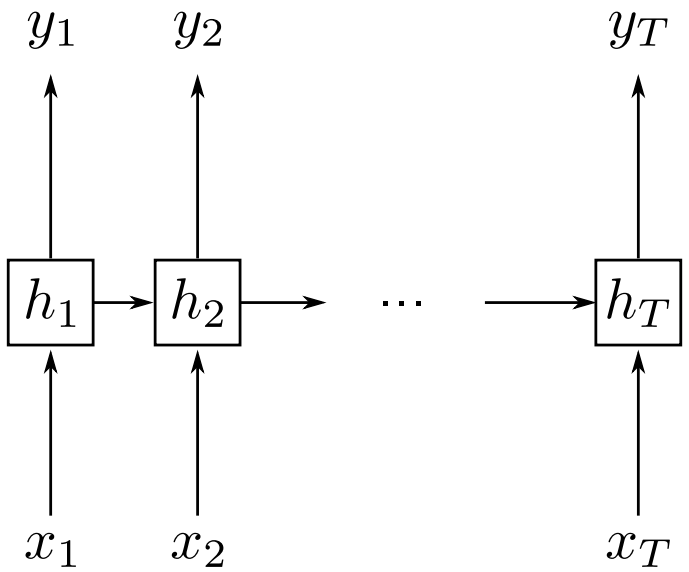
Avec Theano, pas besoin de calculer de gradients manuellement.

Étant donné le graphe de calcul de la fonction originale, Theano calcule automatiquement l'expression symbolique pour les gradients.

Cela permet de se concentrer sur l'architecture du modèle en faisant complètement abstraction de la backpropagation.

Réseaux récurrents

Un réseau récurrent permet de traiter des entrées/sorties de longueur arbitraire :

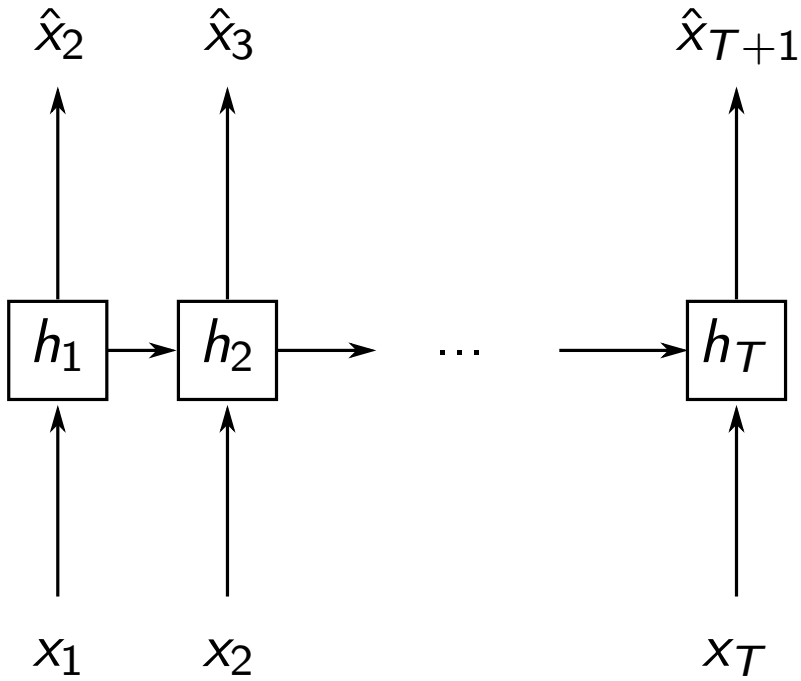


Correspond à la primitive *scan* dans Theano.

Section 2

Modèles de traduction

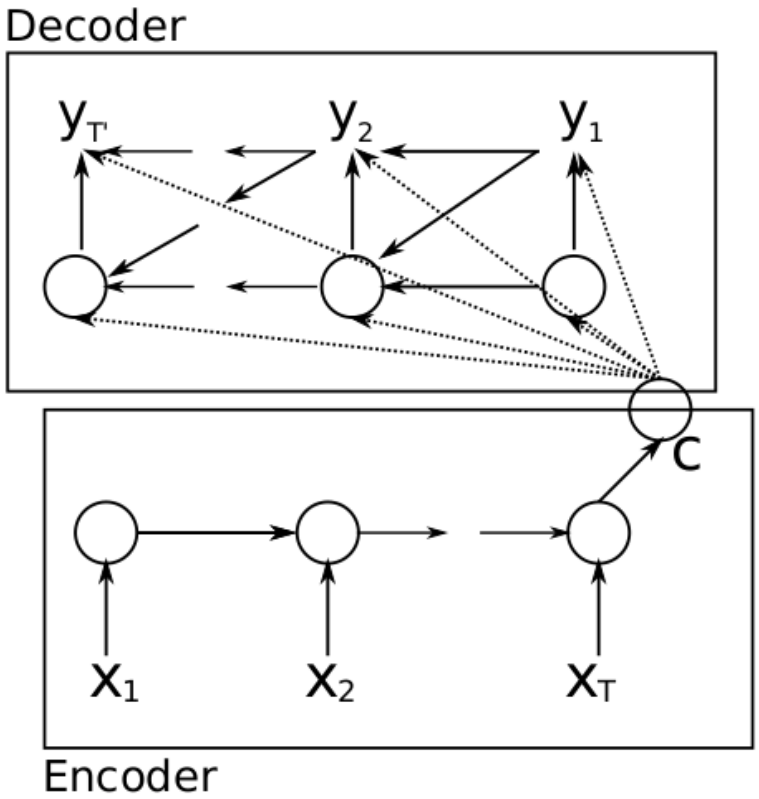
Génération séquentielle avec un réseau récurrent



Entraînement : les x_i sont une phrase qui existe, issue du dataset.
Exploitation : on choisit un mot à chaque étape selon la distribution $\hat{x}_i$, qui devient l'entrée suivante x_i.

Modèle encodeur-décodeur simple

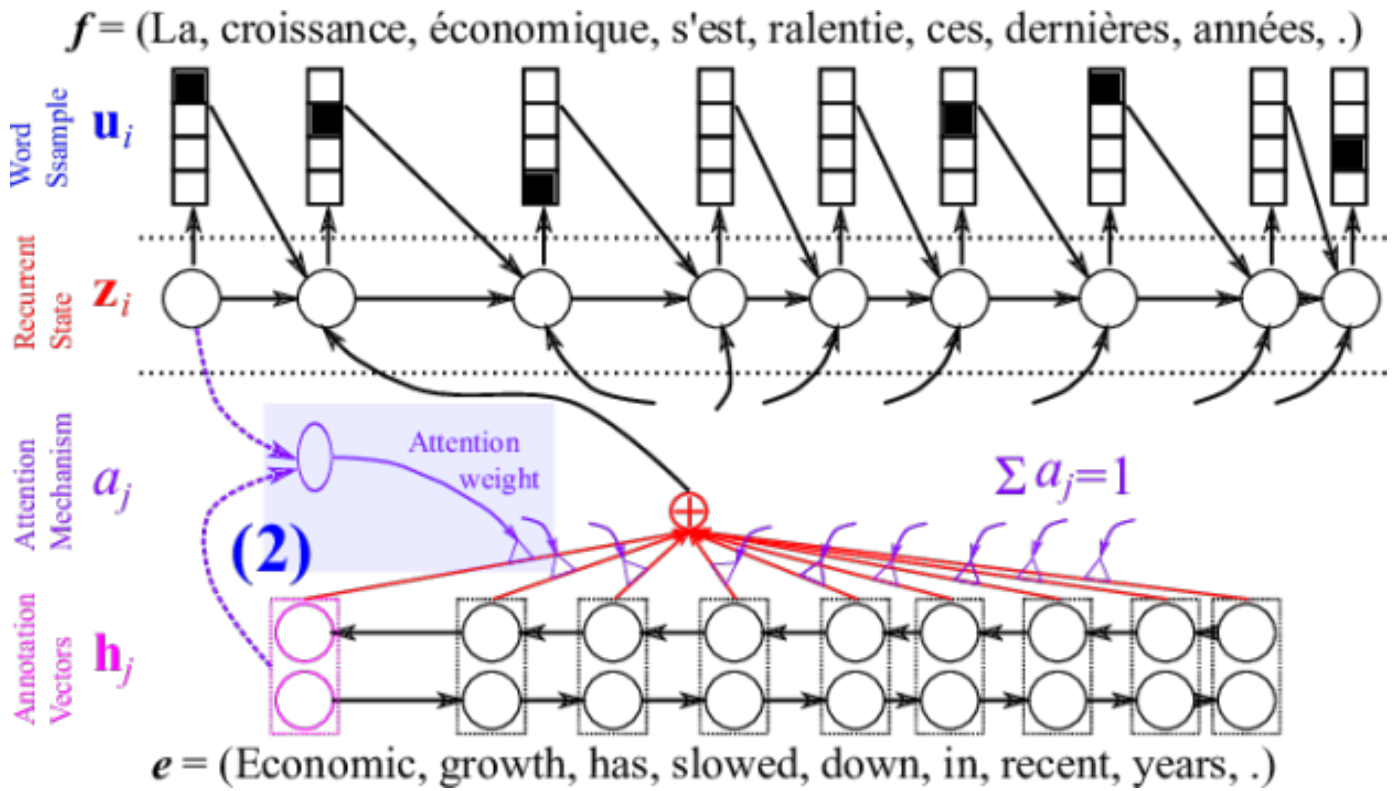
On utilise un réseau récurrent pour encoder la phrase dans une représentation de taille fixe, puis on utilise un autre réseau récurrent pour générer la traduction :



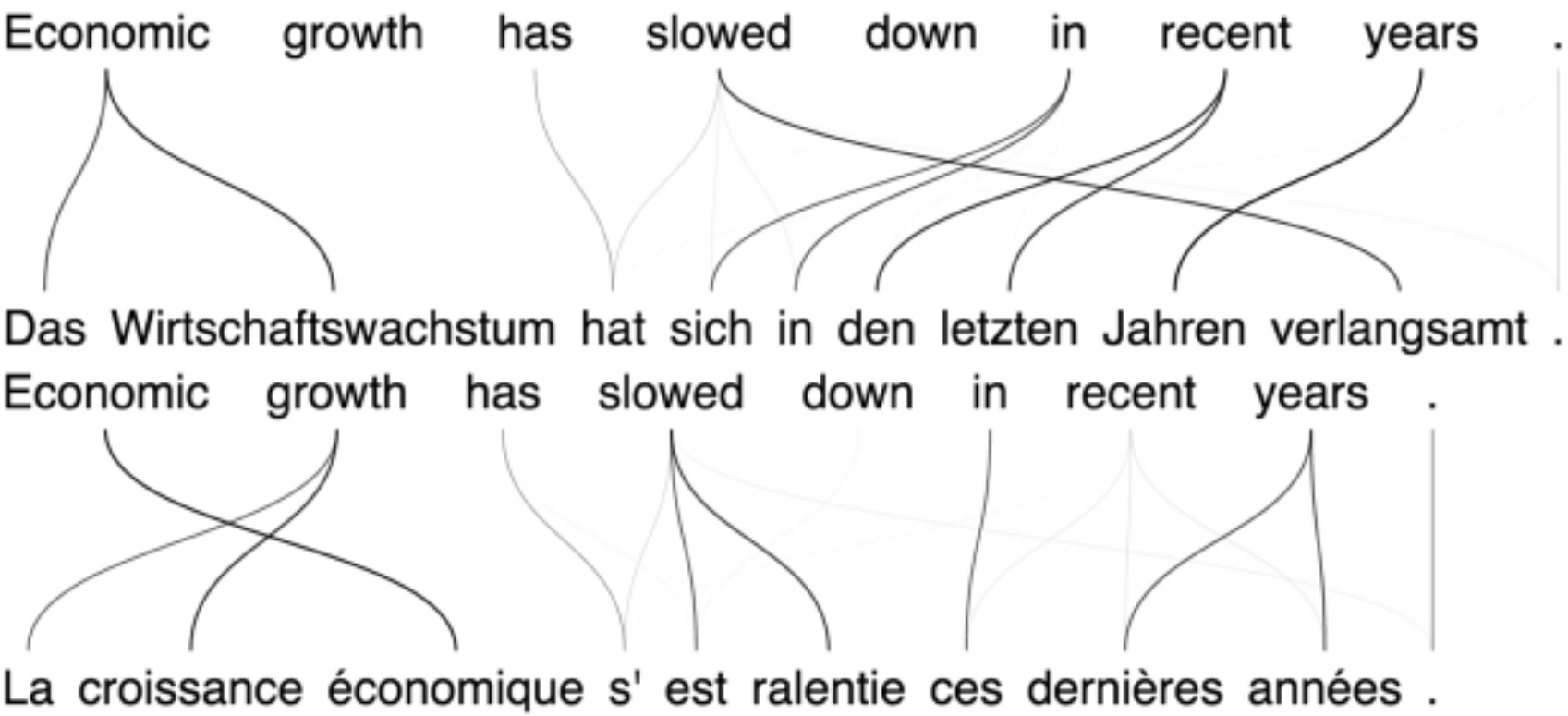
Modèle encodeur-décodeur simple

Limitation : la phrase entière ne peut en général pas être représentée dans un vecteur C de taille fixe.

Modèle avec mécanisme d'attention



Modèle avec mécanisme d'attention



Section 3

Optimisation du softmax

Couche de sortie : opération softmax

La couche de sortie doit donner un vecteur de probabilité sur l'ensemble des mots. On utilise un softmax pour normaliser :

$$o = \mathrm{softmax}(Wx + b)$$

$$\mathrm{softmax}(a)_i = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}}$$

$$o_j = \frac{e^{W_j x + b_j}}{\sum_{k=1}^{N} e^{W_k x + b_k}}$$

Optimisation

- En général la dimension de sortie est très grande (taille du vocabulaire, plusieurs centaines de milliers de mots)
- Même pour calculer une seule valeur, on doit calculer le produit matriciel en entier
- Approximation : dans le dénominateur, un grand nombre de termes sera négligeables devant les quelques termes dominants

Maximum inner product search

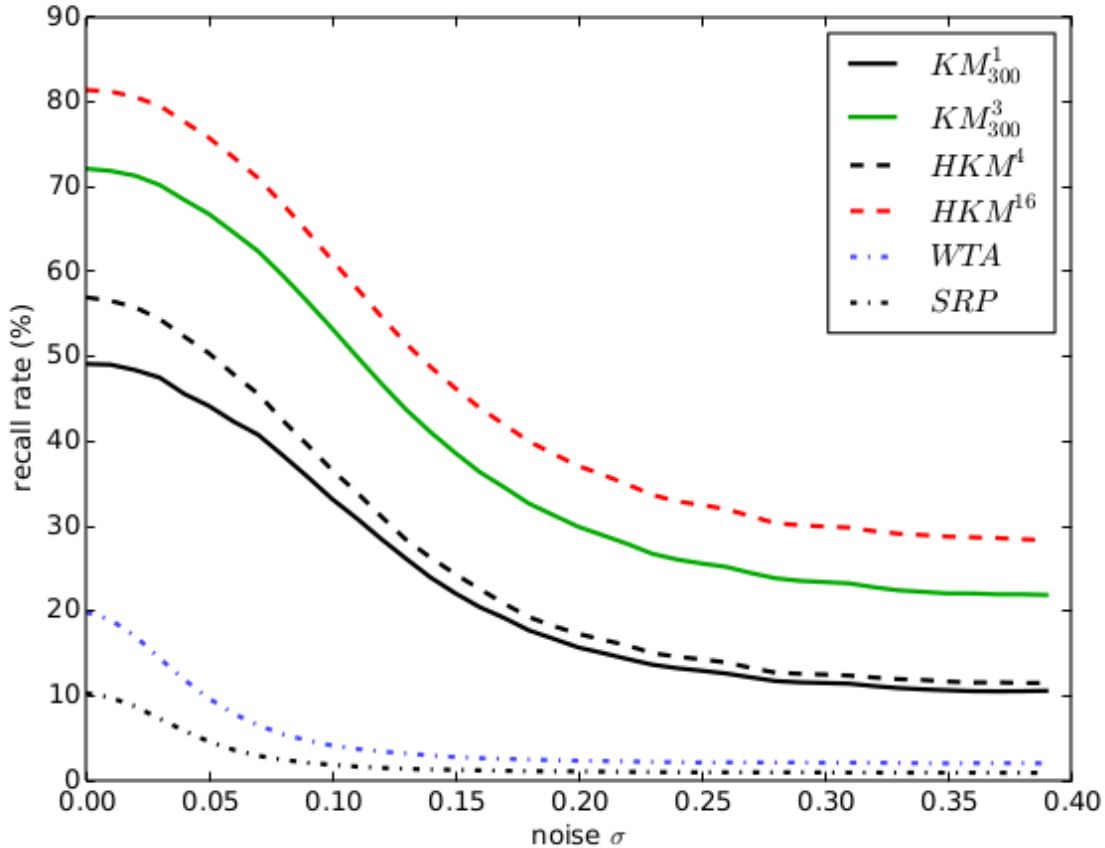
- Il faut trouver quels termes seront dominants dans le dénominateur
- Cela correspond à trouver les termes qui maximisent $W_k x + b_k$
- C'est un problème de *maximum inner product search*

Solution

- On peut approximer MCSS (maximum cosine similarity search) avec un clustering k-means sphérique
- On peut transformer MIPS en MCSS en rajoutant des composantes
- On peut donc trouver approximativement les termes dominants avec une méthode de clustering

Résultats

L'approche clustering est plus précise que d'autres basées sur des hash LSH (locality-sensitive hashing)



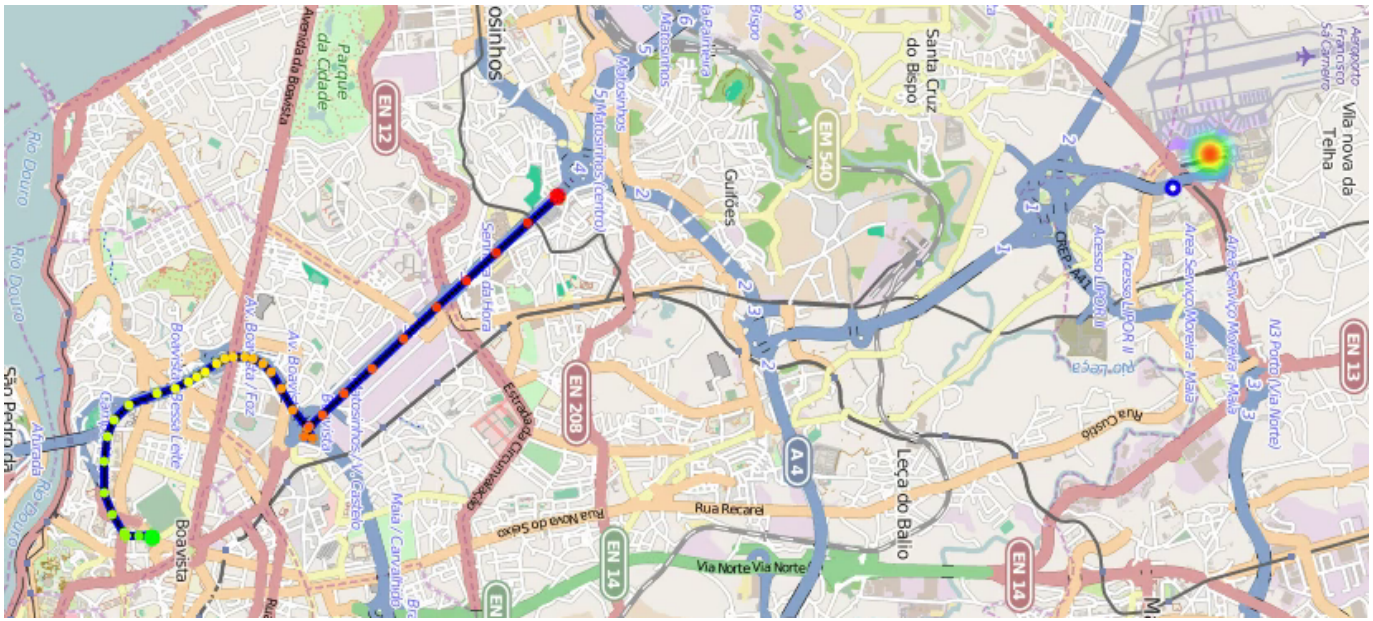
Utilisation

- Exploitation du modèle : cela permet de trouver rapidement les p meilleurs candidats
- Entraînement du modèle : en approximant le dénominateur du softmax, l'entraînement peut être accéléré
- Mise en pratique : ça ne fonctionne pas encore

Section 4

Compétition Kaggle

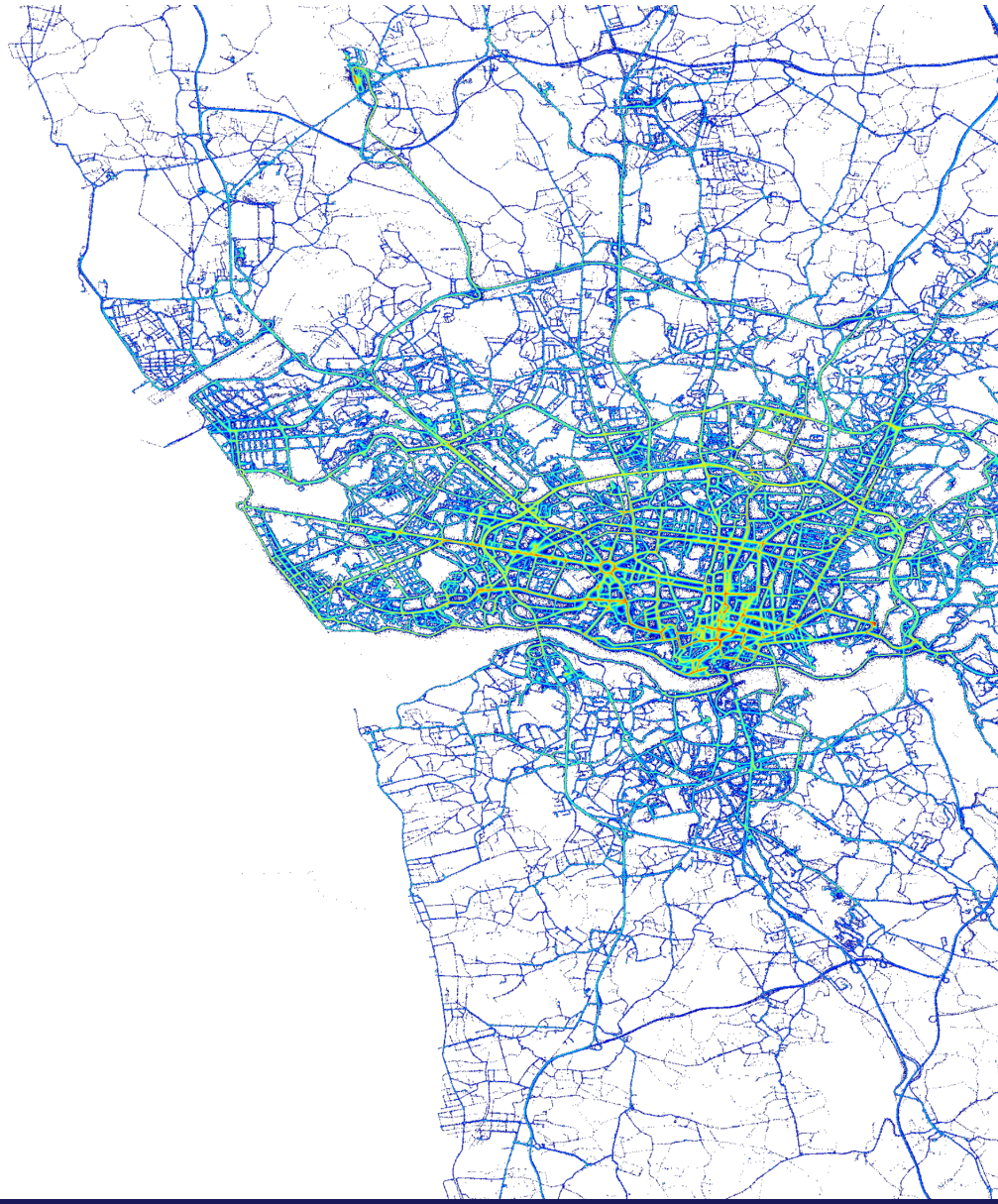
Description du problème



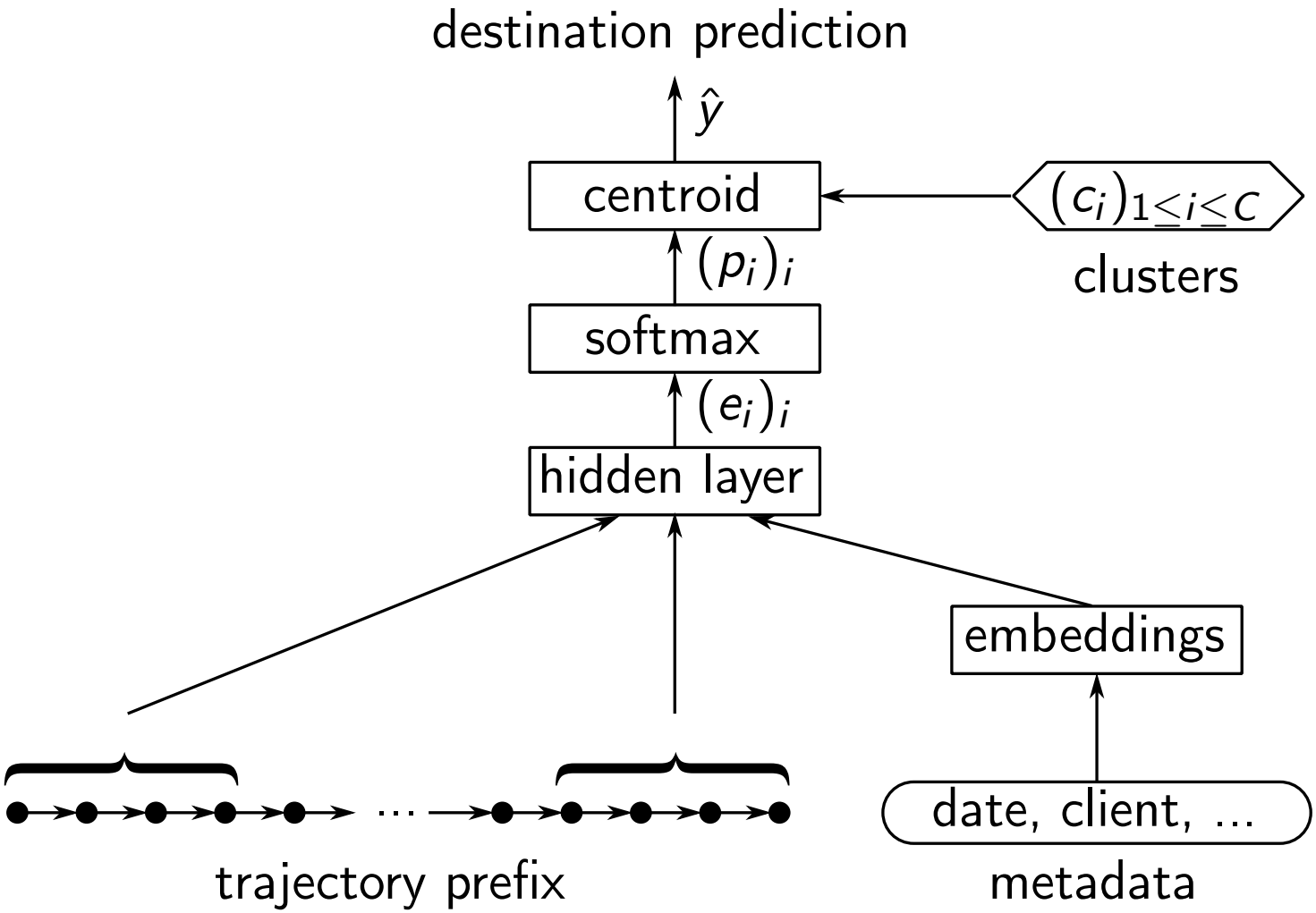
Problème : prédire la destination d'un taxi à partir du début de son trajet

Entrées

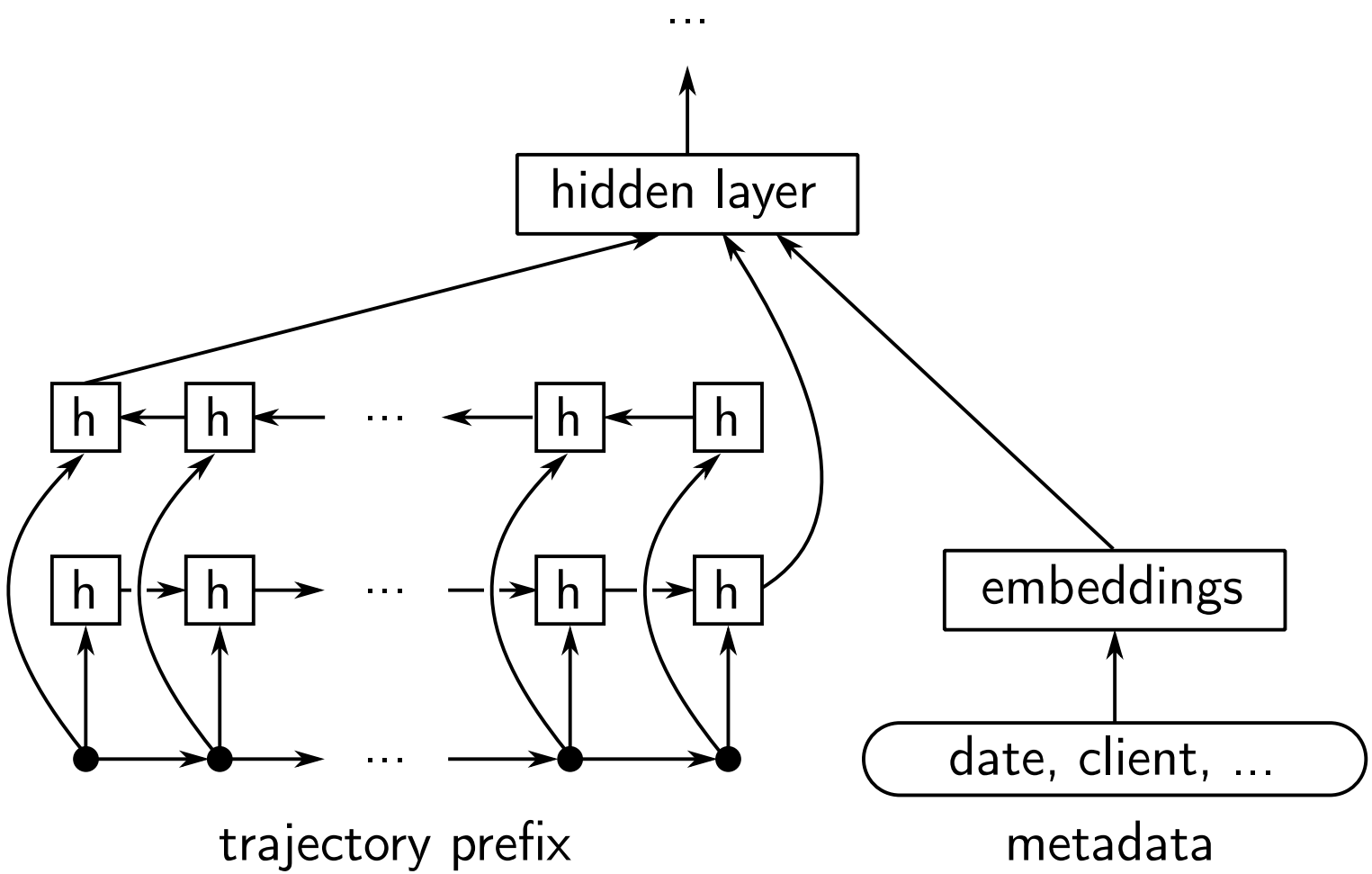
- trajectoire : un point GPS toutes les 15 secondes
- métadonnées du trajet :
  - ID client
  - ID stand taxi
  - ID taxi
  - heure de début du trajet



Solution gagnante



Amélioration



Conclusion

- On peut adapter les réseaux de neurones à n'importe quel problème
- Beaucoup moins de travail manuel (feature engineering, choix de modèle) qu'avec du machine learning classique
- Et des résultats meilleurs !

Papiers

Alex Auvolat and Pascal Vincent.
Clustering is efficient for approximate maximum inner product search.
arXiv preprint arXiv :1507.05910, 2015.

Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio.
Artificial neural networks applied to taxi destination prediction.
arXiv preprint arXiv :1508.00021, 2015.