EARIA 2016

RÉSEAUX DE NEURONES ET REPRÉSENTATION EN RECHERCHE D'INFORMATION

Benjamin Piwowarski

PLAN

1. Introduction
2. Du perceptron aux réseaux de neurones
3. Optimisation: Sorties et coûts, apprentissage
4. APIs pour l'apprentissage
5. Les architectures
6. Applications en RI

REPRÉSENTATION

Logique

"One-hot", "sac de mots", etc.

- Index facile
- Pas de notion de proximité
- précision 1, rappel 0

Distribuée ou continue

Chaque entité est plongée dans $\mathbb{R}^n$

- Index : difficile
- Notion de proximité
- rappel 1, précision 0

👉 Selon l'application, on peut utiliser l'un ou l'autre – ou les deux !

L'APPRENTISSAGE DE REPRÉSENTATION

De nombreux modèles

- Latent Semantic Indexing/Analysis (LSI ou LSA)
- Latent Dirichlet Allocation (LDA)
- Word2vec
- Glove

Utilisation en RI

- Clustering & Classification
- Recherche documentaire
- Question-réponse

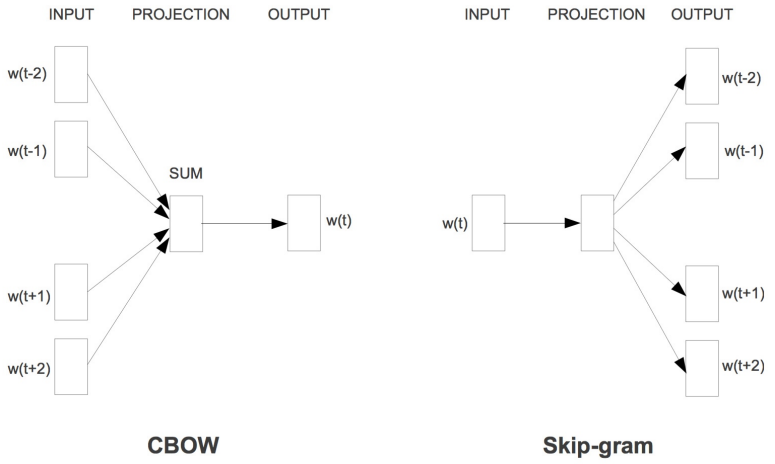
LES REPRÉSENTATIONS CONTINUES

Beaucoup d'applications

- Images
- Son
- n-grammes
- Mots
- Documents
- ...

MOTS

Word2Vec: D-BOW et SKIP



Modèle (skip-gram): pour un contexte c autour du mot a

$$p(a|c) \propto \exp(x_a \cdot c)$$

Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient Estimation of Word Representations in Vector Space. 2013.

[analyse] Levy, O., & Goldberg, Y. Neural Word Embedding as Implicit Matrix Factorization. NIPS 2014

MOTS: GLOVE

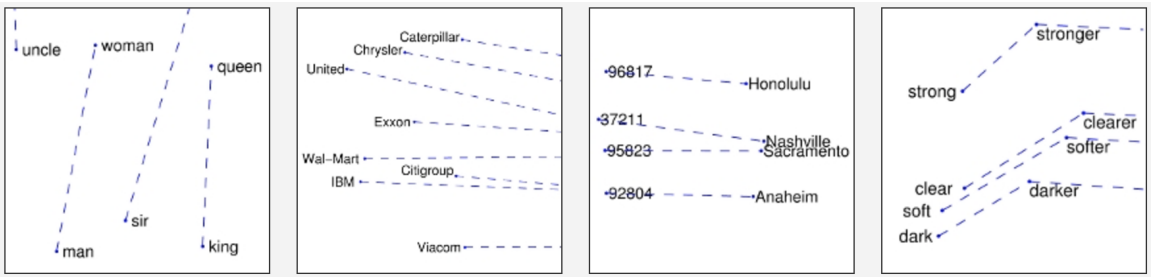
On estime le logarithme de la fréquence de co-occurrence

$$\boldsymbol{x}_i \cdot \boldsymbol{x}_j + \boldsymbol{b}_i + \boldsymbol{b}_j \approx \log \boldsymbol{X}_{ij}$$

PROPRIÉTÉS

Pour Word2Vec & Glove, on observe des relations de proximité

... et des relations algébriques



man - woman | company - ceo | city - zip code | comparative - superlative

REPRÉSENTATION DES RELATIONS

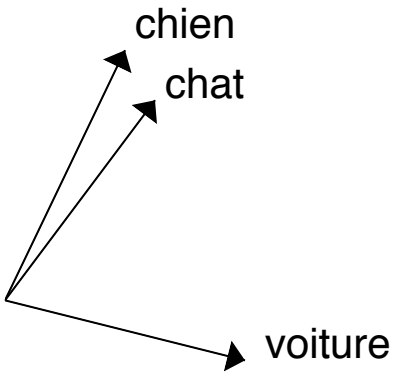
👉 Cette idée (relation algébrique) se retrouve dans d'autres apprentissages de représentation

- Connaissances sous forme de triplets (e_1, r, e_2)
- Les entités dans $\mathbb{R}^n$
- Score $s(e_1, r, e_2)$ appris par un modèle

> 👁 **Modèle de translation**
>
> $$s = \|(x_{e_1} - x_{e_2}) - x_r\|^2$$

RÉSEAUX DE NEURONES ET REPRÉSENTATION

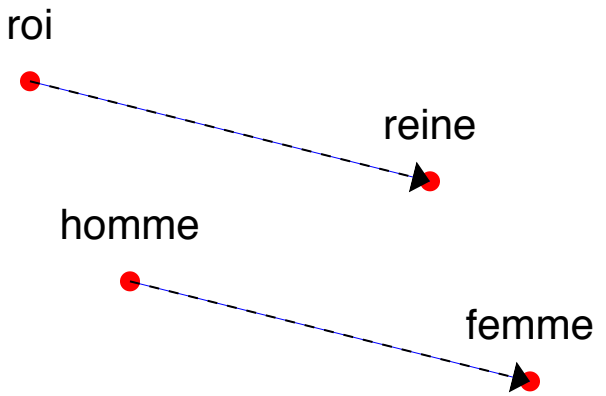
Proximité = propriété partagées



Mots proches ~ sémantique proche

RÉSEAUX DE NEURONES ET REPRÉSENTATION

Relation algébrique = propriété (Word2Vec, Mikolov)



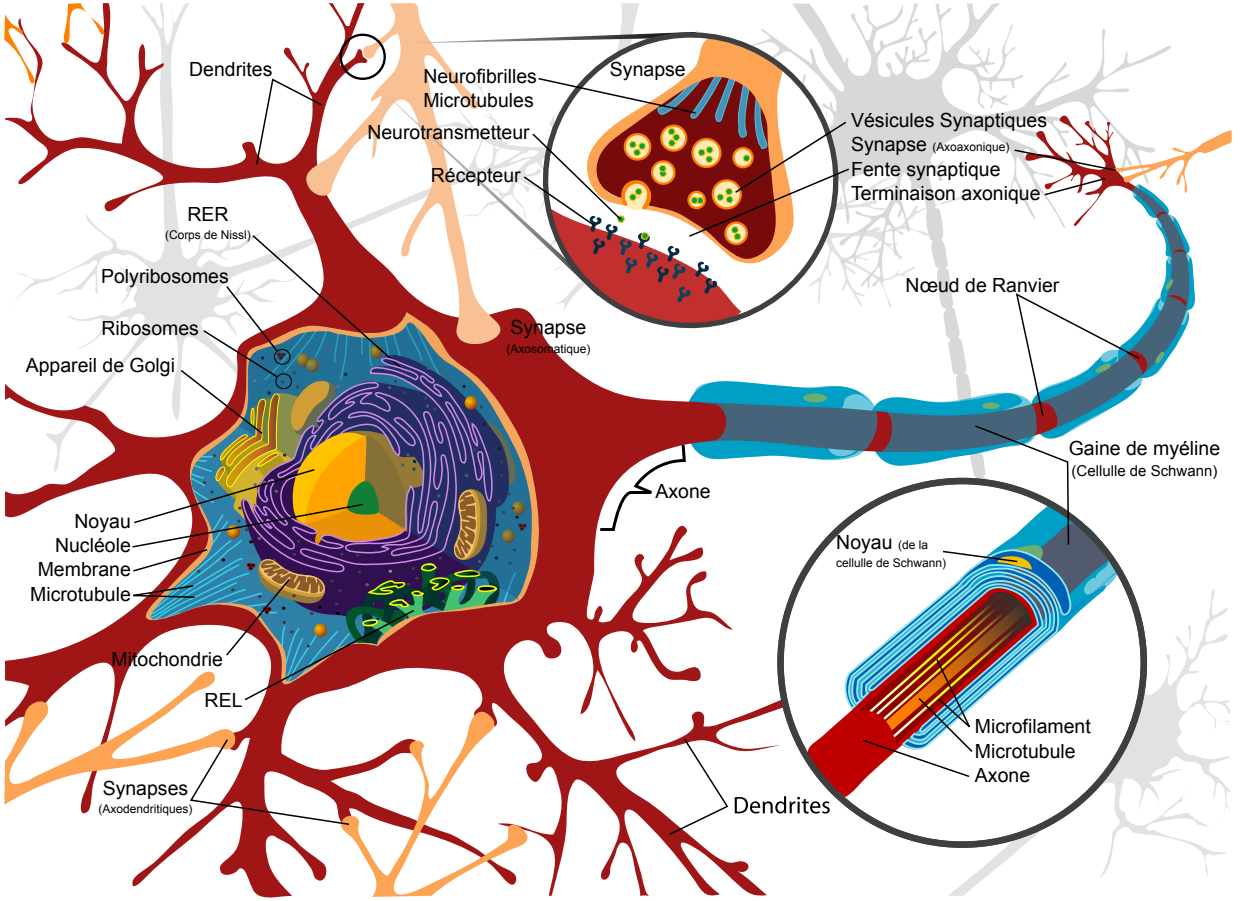
Il existe un sous-espace vectoriel qui correspond à des relations

POURQUOI LES RÉSEAUX DE NEURONES ?

- Des succès récents (~ 4 ans) dans le traitement du signal (image, son, etc.)
- 👉 Adaptés au traitement des représentations distribuées

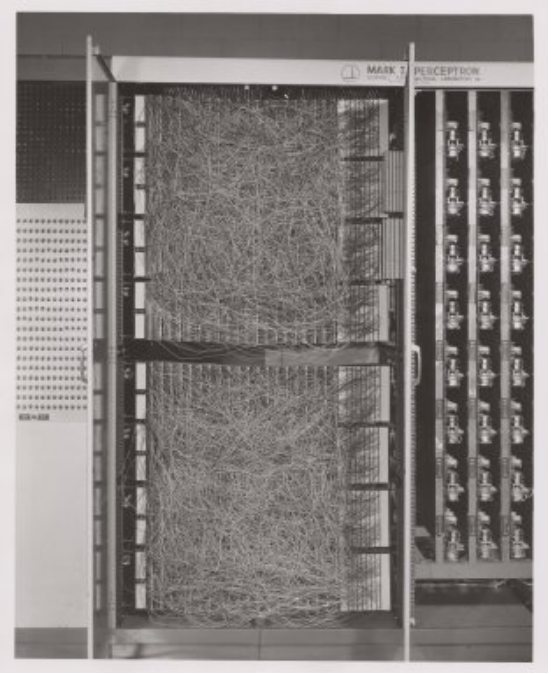
⚠ Attention:
il n'y a pas de miracle - le théorème du "No Free Lunch" frappe encore !

INSPIRATION BIOLOGIQUE



Source Wikipedia

LE PERCEPTRON



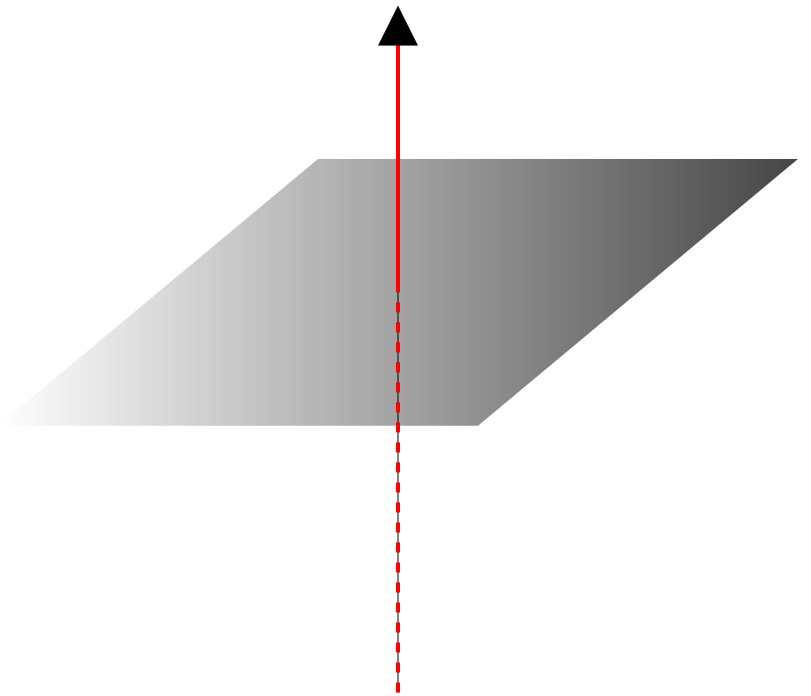
- Un neurone y est connecté à un ensemble de neurones afférents $x_1, \ldots, x_n$
- Son activation est determinée par ses entrées (et un terme de biais)

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f(\mathbf{w} \cdot \mathbf{x} + b)$$

- $\mathbf{w} \cdot \mathbf{x} + b = 0$ ➔ hyperplan dans $\mathbb{R}^n$

PERCEPTRON: FRONTIÈRE DE DÉCISION

Le perceptron est une machine de **classification linéaire**. Il modélise un hyperplan dans un espace vectoriel



SÉPARATION LINÉAIRE

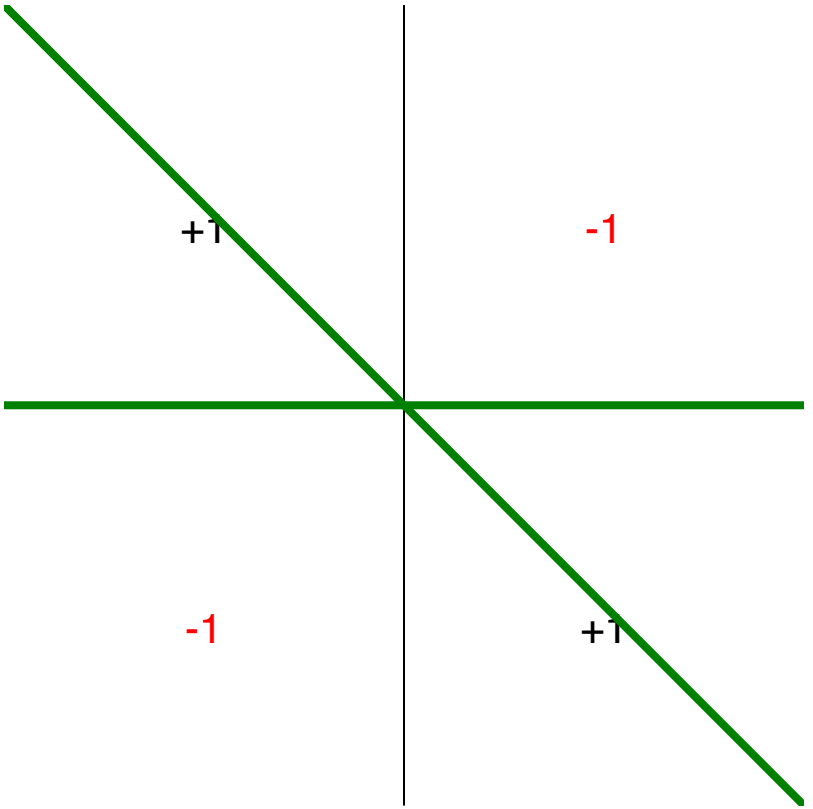
En pratique

- On prend en compte une "direction" (espace de dimension 1)
- Un hyperplan en dimension n sépare parfaitement $n + 1$ points distincts

En RI

- Espace des termes
- Espace des n-grammes
- Espace des documents
- ...

PERCEPTRON: CRITIQUE



LA SOLUTION :
LE PERCEPTRON MULTI-COUCHE

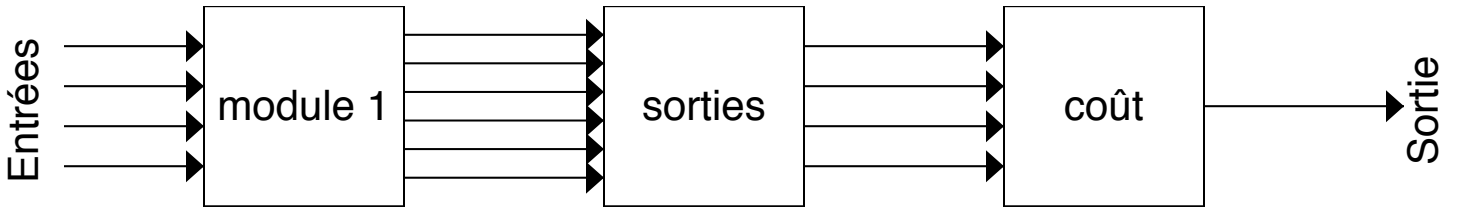
- Plusieurs couches
- Non linéarités qui transforment les sorties
  - sigmoide
    $$\sigma(x) = 1/\left(1 + e^{-x}\right)$$
  - tangente hyperbolique
    $$\tanh(x) = \left(1 - e^{2x}\right) / \left(1 + e^{-2x}\right)$$
  - ReLU (rectified linear unit)
    $$relu(x) = \max(0, x)$$
  - ...

⚠ Point important: saturation

Demonstration avec le XOR

LA COMPOSITION DE FONCTIONS

- Avant: **transformation linéaire** suivie d'une **activation** non linéaire
- Aujourd'hui : vision modulaire



Les modules

- Module linéaire (ou convolution) = **détection de motifs**

$$f_\theta : \mathbb{R}^n \longrightarrow \mathbb{R}^p$$

- Module d'activation = **saturation**

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$$

- Module de coût = **objectif**

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

SORTIES

La dernière couche correspond à la sortie:

- Régression:

  👁 score d'un document pour une question

- Classification:

  👁 Est-ce une bonne réponse ? Catégorie du document ?

👉 Pour interpréter de façon *probabiliste* les sorties, on utilise

- Cas binaire, une sigmoïde

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

- Cas binaire, le softmax

$$softmax(x) = \frac{\exp x_i}{\sum \exp(x_j)}$$

COÛTS

👉 But: mesurer la différence entre l'objectif et le comportement

On calcule sur l'ensemble de la base d'apprentissage $\mathcal{D}$

$$\sum_{d \in \mathcal{D}} \Delta(d, f_\theta(d))$$

où Δ mesure l'erreur par rapport à la sortie désirée en fonction de la tâche.

COÛTS : RÉGRESSION

L'ensemble d'apprentissage = paires entrée/sortie désirée

$$\mathcal{D} = \{(x_i, \hat{y}_i)\}$$

👉 Coût quadratique

$$\Delta(f_\theta(d), \hat{y}) = \|y - \hat{y}\|^2$$

Applications

- Estimer des poids

COÛTS : ORDONNANCEMENT

L'ensemble d'apprentissage = paires d'objet (ex. document) avec $x_+ \succ x_-$

$$\mathcal{D} = \left\{(x_i^+, x_i^-)\right\}_i$$

La fonction f donne le score d'un objet

👉 max-margin loss

$$\Delta(f_\theta(x_+), f_\theta(x_-)) = \max\{0, 1 - (f_\theta(x_+) - f_\theta(x_-))\}$$

Le minimum (0) est atteint si

$$f_\theta(x_+) > f_\theta(x_-) + 1$$

Application

- Apprendre à ordonner des documents, des relations, etc.

COÛTS : CLASSIFICATION

L'ensemble d'apprentissage = paires entrée / classe $c \in \{1, \dots, k\}$

$$\mathcal{D} = \{(x_i, \hat{c}_i)\}_i$$

La fonction f_θ renvoie un vecteur $\mathbb{R}^k$,

$$f_\theta(x) = (p(c = 1|x; \theta), \dots, p(c = k|x; \theta))$$

👉 Coût cross-entropique

$$\Delta(f_\theta(x), \hat{c}) = \log p(c = \hat{c}|x; \theta) = \log \left(f_\theta(x)\right)_{\hat{c}}$$

COMMENT APPRENDRE ?

- On a un ensemble de paramètres $\theta \in \mathbb{R}^p$
- On veut essayer de minimiser le coût (ou risque R)

$$\mathbb{E}(\Delta(\theta))$$

- Il n'y a pas de solution analytique...

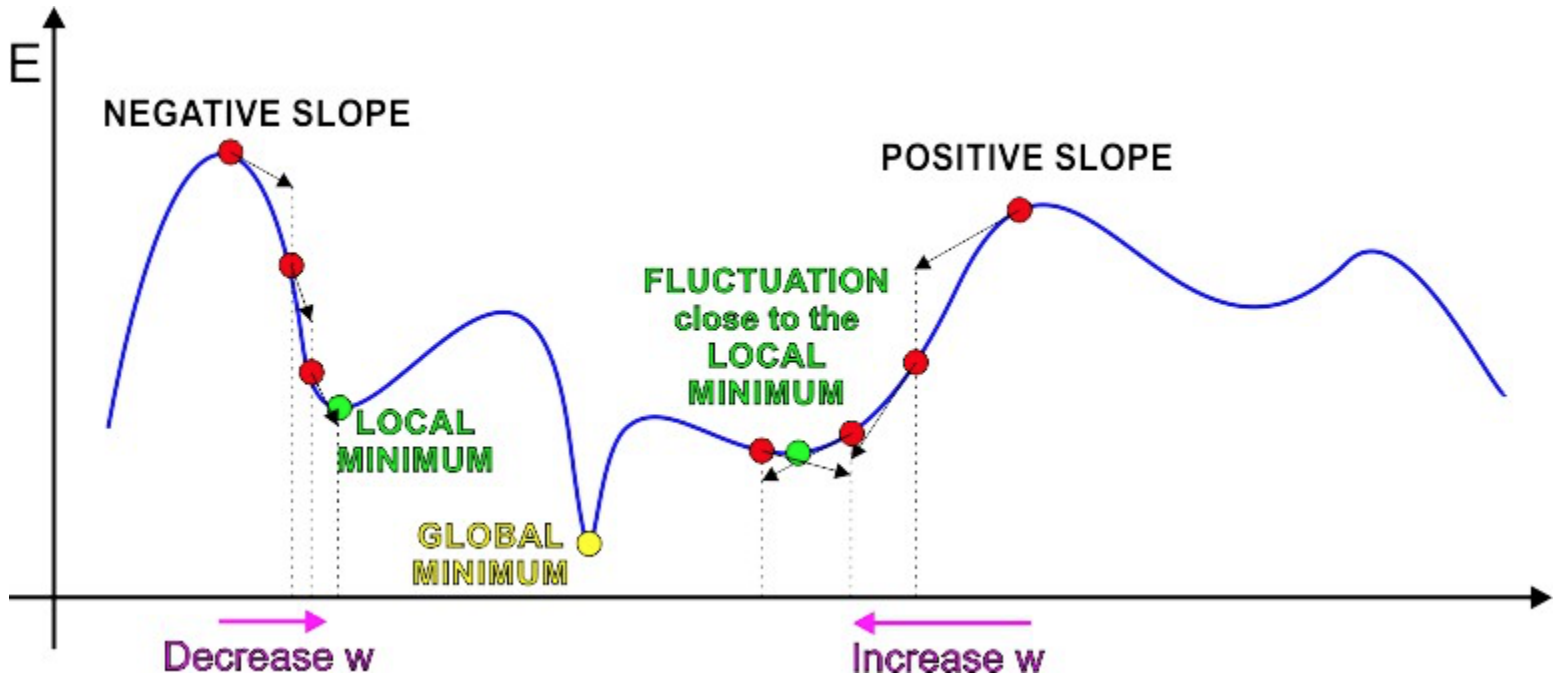
👉 On utilise la descente de gradient

$$\theta_{t+1} = \theta_t - \epsilon \nabla_\theta \mathbb{E}(\Delta)$$

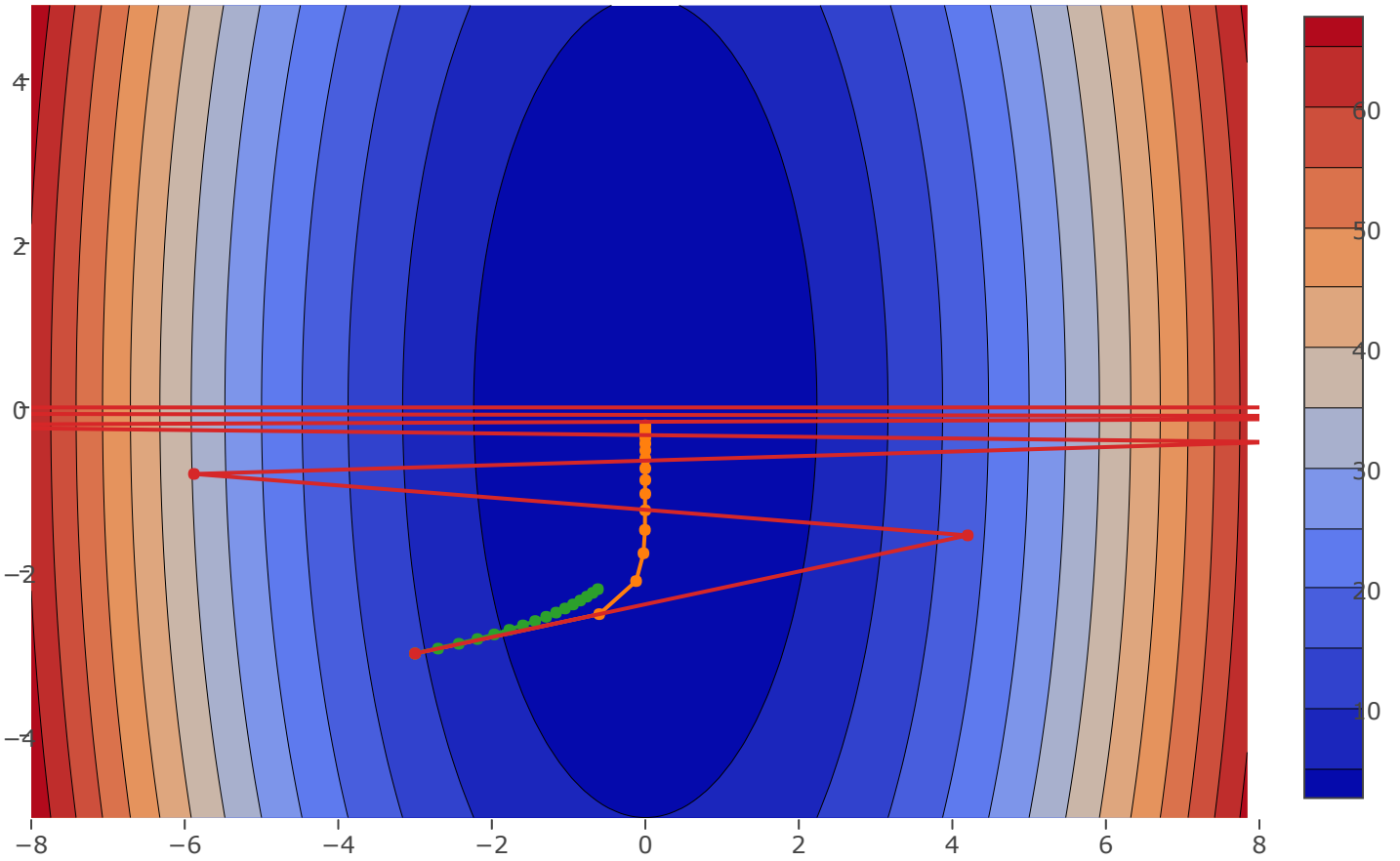
avec

$$\nabla_\theta \mathbb{E}(\Delta) = \begin{pmatrix} \frac{\partial \mathbb{E}(\Delta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \mathbb{E}(\Delta)}{\partial \theta_N} \end{pmatrix}$$

LA DESCENTE DE GRADIENT EN IMAGE



LES TECHNIQUES DU SECOND ORDRE - POURQUOI ?



EN PRATIQUE

Apprentissage stochastique ou par lots (batch)

- Moins de mémoire utilisée
- Marche mieux

Utilisation de méthodes du second degré approchées :

- L-BFGS (jeu complet)
- ADAGRAD (stochastique)
- ADAM (stochastique)
- ...

Kingma D., Lei Ba J. Adam: A Method for Stochastic Optimization. 2014.

LA RÉTRO-PROPAGATION DE GRADIENT

Problématique

Comment mettre à jour les paramètres du modèle ? i.e. comment calculer

$$\nabla_\theta \mathbb{E}(\Delta(\theta)) \approx \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta R(d_i; \theta)$$

Solution

Utilisation de la dérivation de fonction composées.

Si $h = g \circ f$ avec

$$f : x \in \mathbb{R}^n \to y \in \mathbb{R}^p$$

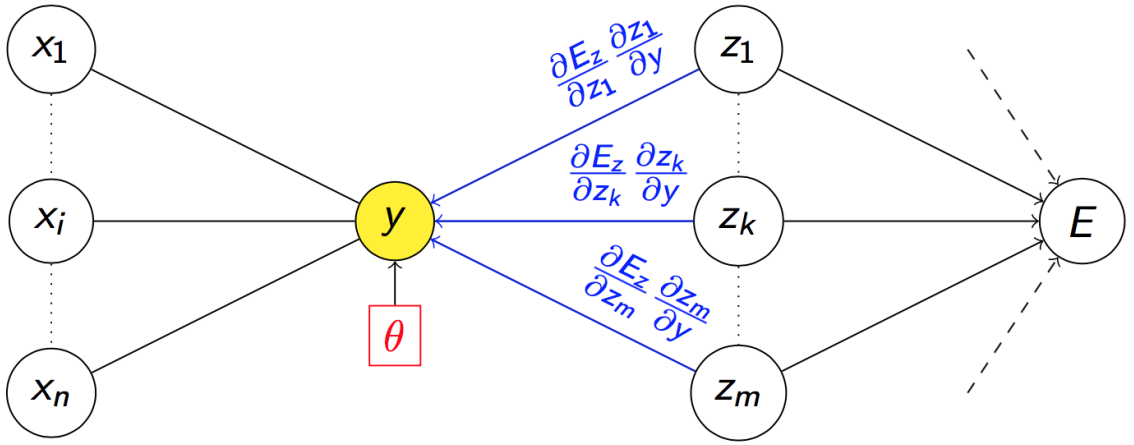
et

$$g : y \in \mathbb{R}^p \to z \in \mathbb{R}$$

alors

$$\frac{\partial h}{\partial \theta} = \sum_{i=1}^{p} \frac{\partial g}{\partial y_i}(f_\theta(x)) \frac{\partial f_i}{\partial \theta}(x)$$

RÉTRO-PROPAGATION: RÉSUMÉ



$$\Delta E \text{ en fonction du paramètre } = \frac{\partial E_x}{\partial \theta} = \frac{\partial E_y}{\partial y}\frac{\partial y}{\partial \theta}$$

$$\Delta E \text{ en fonction d'un neurone } = \frac{\partial E_z}{\partial y} = \sum_{k=1}^{m} \frac{\partial E_z}{\partial z_k}\frac{\partial z_k}{\partial y}$$

INITIALISATION

⚠ L'objectif est de donner un départ le moins pourri possible : on évite de saturer trop $\frac{\partial E}{\partial \theta} \approx 0$: évolution très lente des poids

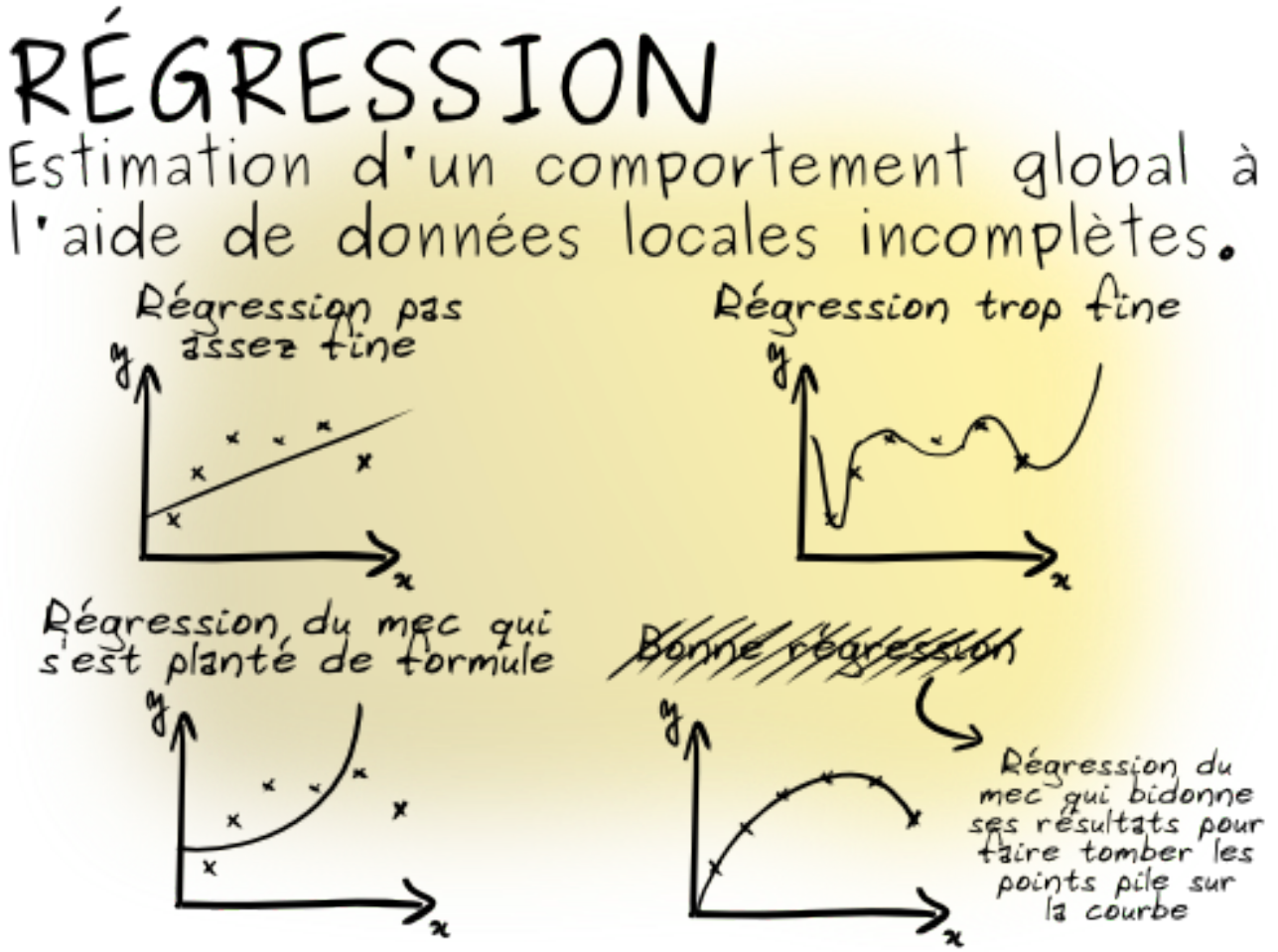
Idée : Rester dans une zone linéaire

- Centrer / réduire les données pour chaque variable d'entrée
- Initialiser les poids aléatoirement dans $[-1/\sqrt{n}, 1/\sqrt{n}]$ avec n le nombre d'entrées du neurone
- Utiliser $\tanh$ pour les couche cachées

Cela vaut aussi pour les entrées !

- Entrées centrées
- Variance réduite
- Peu de corrélation si possible

LE SUR ET SOUS-APPRENTISSAGE



RÉGULARISATION: A PRIORI SUR LES PARAMÈTRES

Pour éviter le sur-apprentissage, un a priori sur les paramètres

- Correspond à un *a priori* Gaussien

$$L_2(\theta) = \|\theta\|_2 = \sqrt{\sum_j \theta_j^2}$$

- Correspond à un *a priori* Laplacien (plus parcimonieux)

$$L_1(\theta) = \|\theta\|_1 = \sum_j |\theta_j|$$

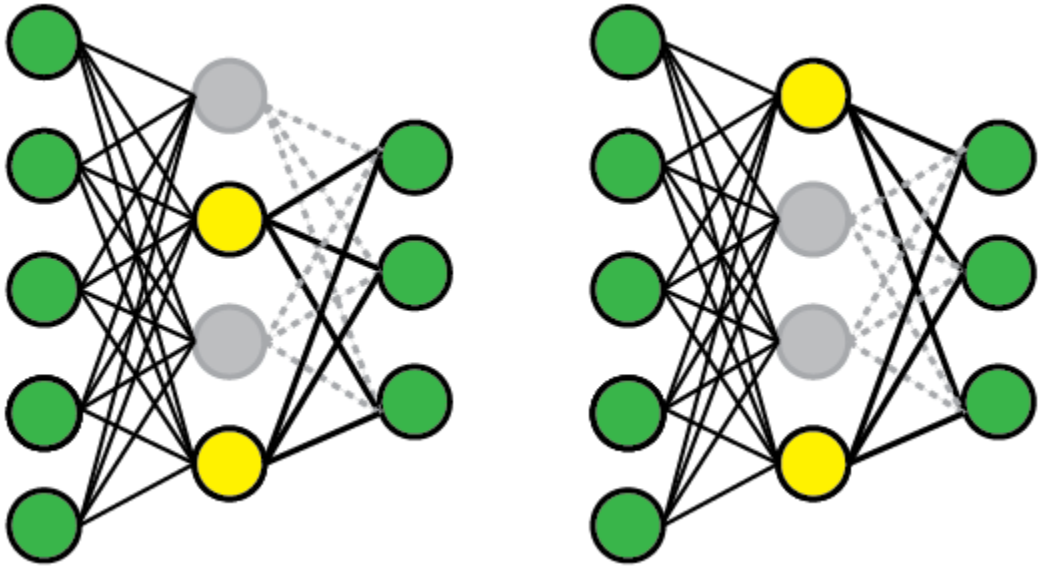
RÉGULARISATION: DROPOUT

Pour éviter le sur-apprentissage

- on peut aussi utiliser un moyennage de modèles

$$p(y|x) = \int p(y|x,\theta)p(\theta|D)$$

- Le dropout en est une forme approchée mais plus pratique



QUAND S'ARRÊTER ?

- Théorie : Fonction convexe => $\|\nabla\theta\| < \tau$
- En pratique (heuristiques) :
  - Utilisation d'un lot de validation (contrôle de l'erreur de généralisation)
  - Nombre d'itérations
  - Les paramètres varient peu

LES GRANDES LIBRAIRIES

Bas niveau

- Theano (Python)
- Tensorflow (Python / C++) développé par google

Haut niveau

- Torch7 (soutenu par facebook)
- Keras (base: theano ou tensorflow)

THEANO OU TENSORFLOW

On définit

- Des variables (paramètres)
- Des inconnues (les entrées et sorties)
- Des opérations (multiplication, convolution, addition, etc.)

Une fois le **graphe de calcul** défini, on peut l'utiliser

```
import tensorflow as tf
# Définit une variable
w = tf.Variable(tf.truncated_normal((5,5)), name="w")
# Définit une inconnue
x = tf.placeholder(tf.float32, (5), name="x")
# Définit une multiplication
wx = tf.matmul(w, tf.reshape(x, [-1, 1]))

# Initialisation et calcul
init_op = tf.initialize_all_variables()
with tf.Session() as session:
    session.run(init_op)
    print(session.run(wx, {x: [1.2, -2.2, 1.3, 0.4, 0.5]}))
```

KERAS

Plus simple: utilisation de modules (torch7 et Keras)

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))
# Définition de la fonction de coût
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

# Optimisation
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)
```

LES ARCHITECTURES DE RÉSEAU

Problème

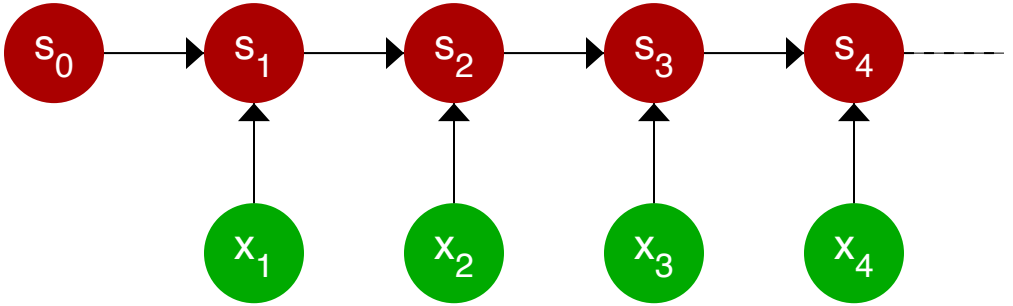
- Le texte est une séquence (de mots, de caractères)
- Les réseaux de neurones traitent des entrées de taille fixe
- Comment faire ???

LES ARCHITECTURES DE RÉSEAU

Deux façons

- On exploite la séquentialité = récurrence
- On exploite l'invariance temporelle = convolution

RÉSEAUX RÉCURRENTS



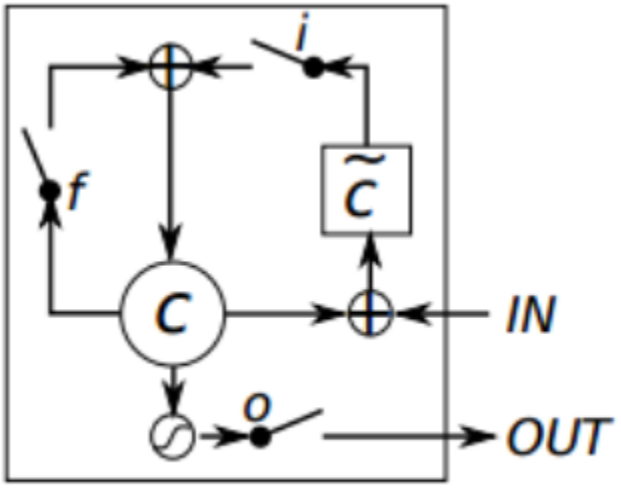
- À chaque pas de temps, on met à jour $\boldsymbol{s_t}$
- L'état final représente la séquence

⚠ Attention

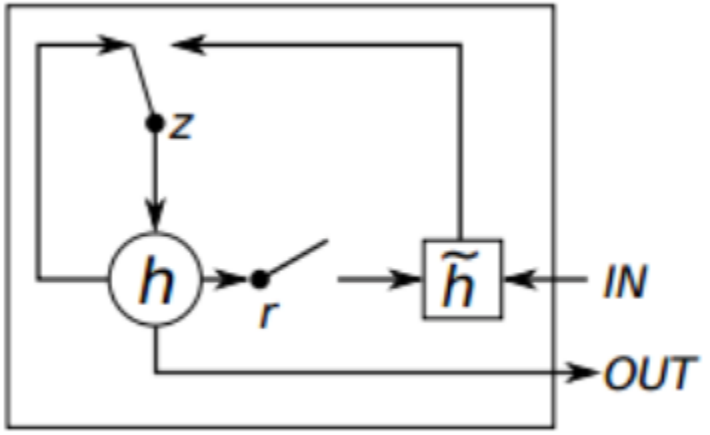
- Perte d'information sur des séquences longues (> phrase)
- Problème d'apprentissage (vanishing/exploding gradient)

RÉSEAUX RÉCURRENTS: GATED UNITS

Utilisation d'unités avec des "portes"



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

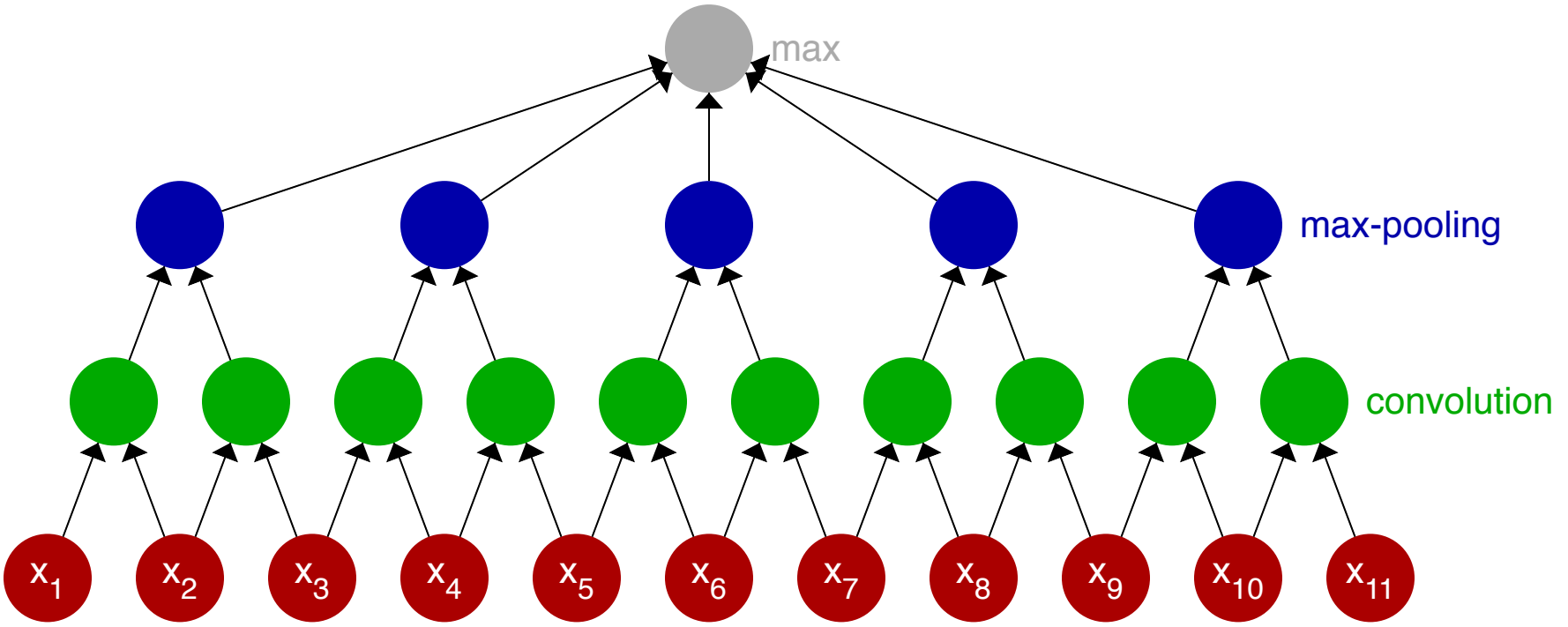
MASQUES : CONVOLUTION ET POOLING

Idée : Composition de représentations de plus en plus abstraites

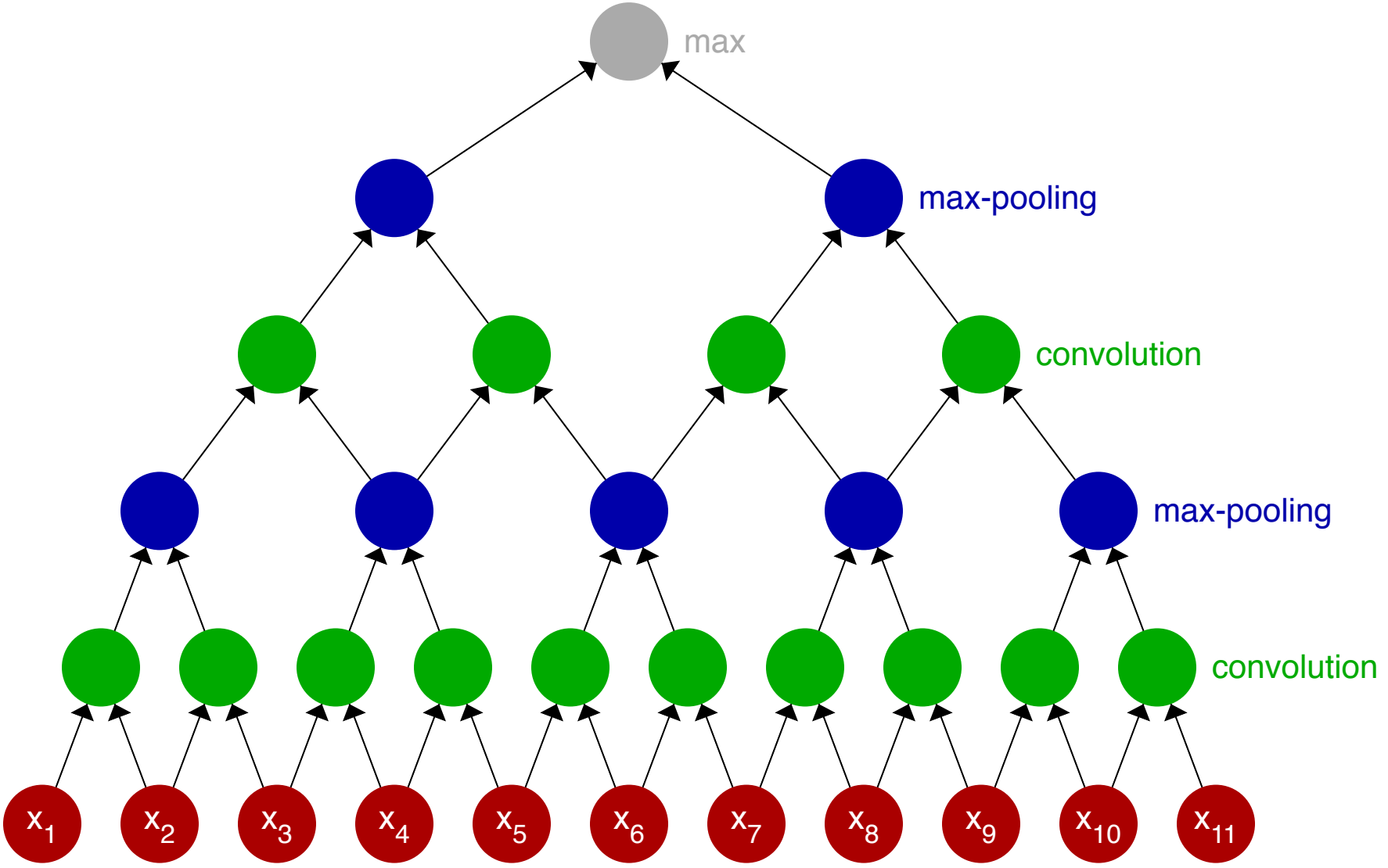
- Texte : caractère → mot → clause → phrase → histoire
- Image : pixel → bords → motif → partie → objet

⚠ À chaque fois : invariance (1D, 2D) :
Opération répétée sur une sous partie de l'entrée

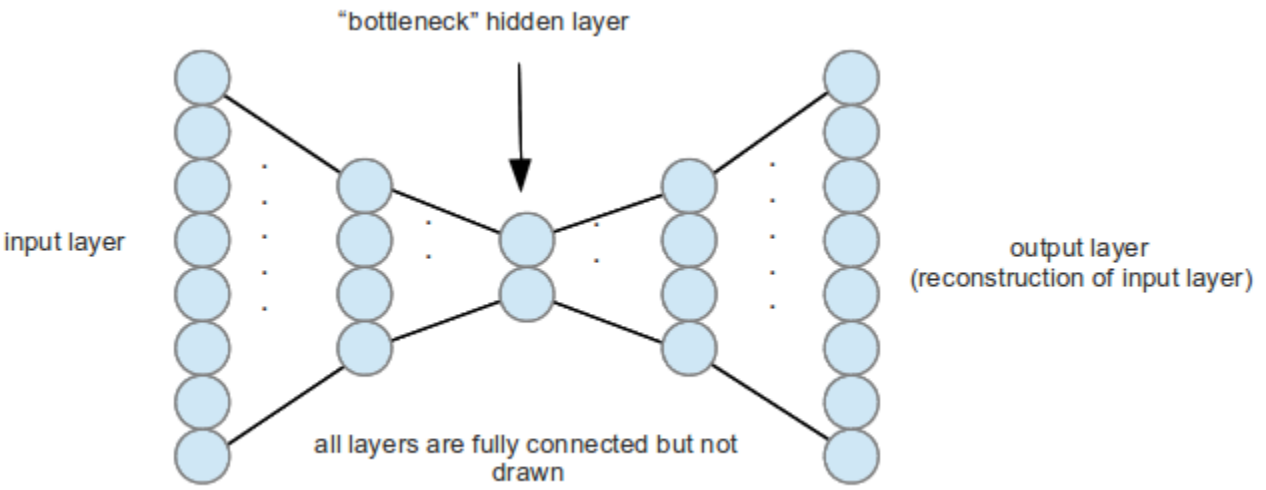
CONVOLUTION



CONVOLUTION



AUTRE: AUTO-ENCODEURS



http://nghiaho.com/

Tout un bestiaire...

- RBM et DBM
- Auto-Encoder
- Denoising Auto-Encoder
- Bi-Generative Adversarial Nets
- ...

À utiliser...

- Pour obtenir une représentation
- Beaucoup de données non supervisées

MÉCANISME D'ATTENTION ET DE MÉMOIRE

👉 **But :** améliorer la capacité des réseaux de neurones à traiter des tâches complexes. Appliqué dans les tâches de question-réponse (entre autres)

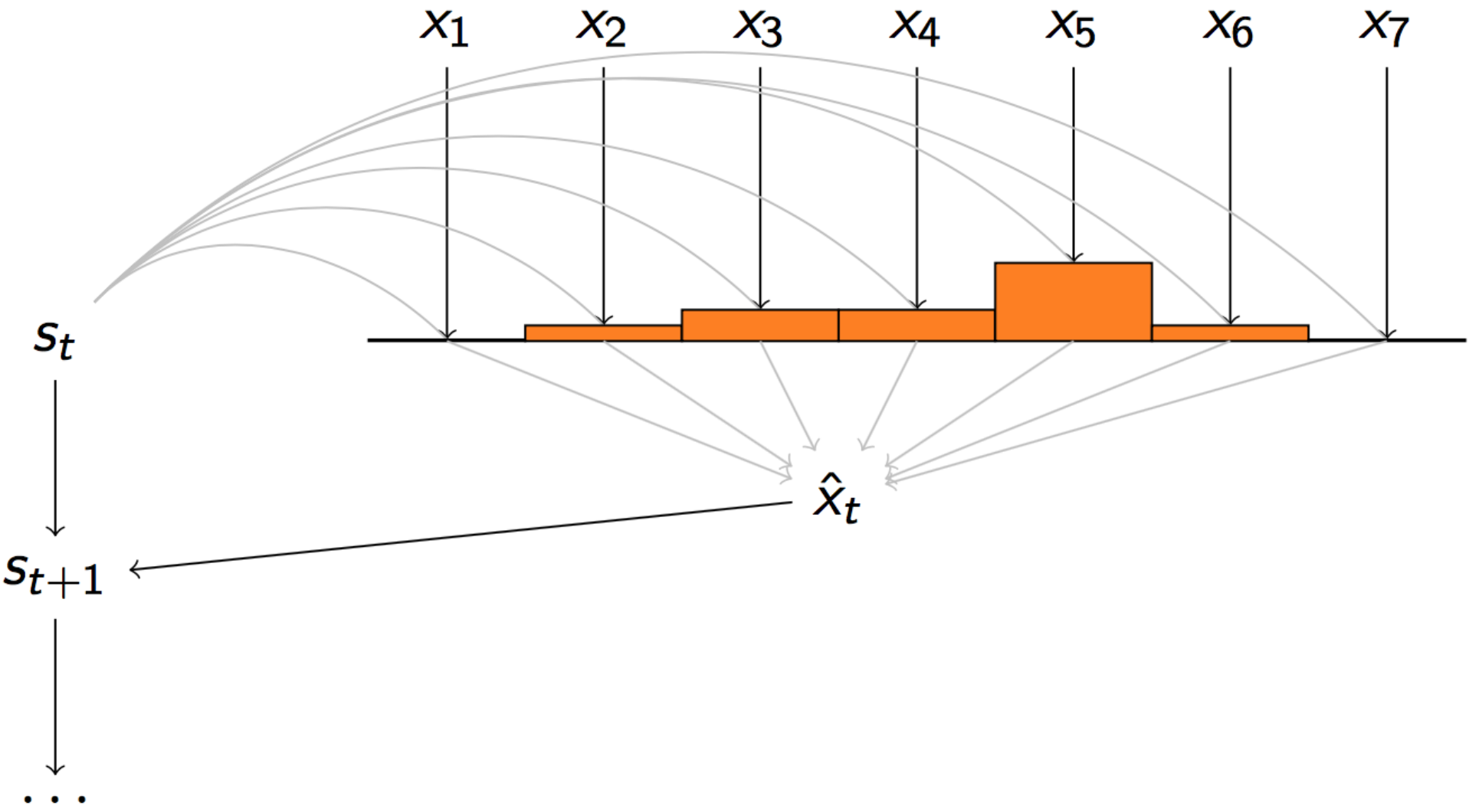
Attention

Permet de se focaliser l'attention sur certaines parties des entrées

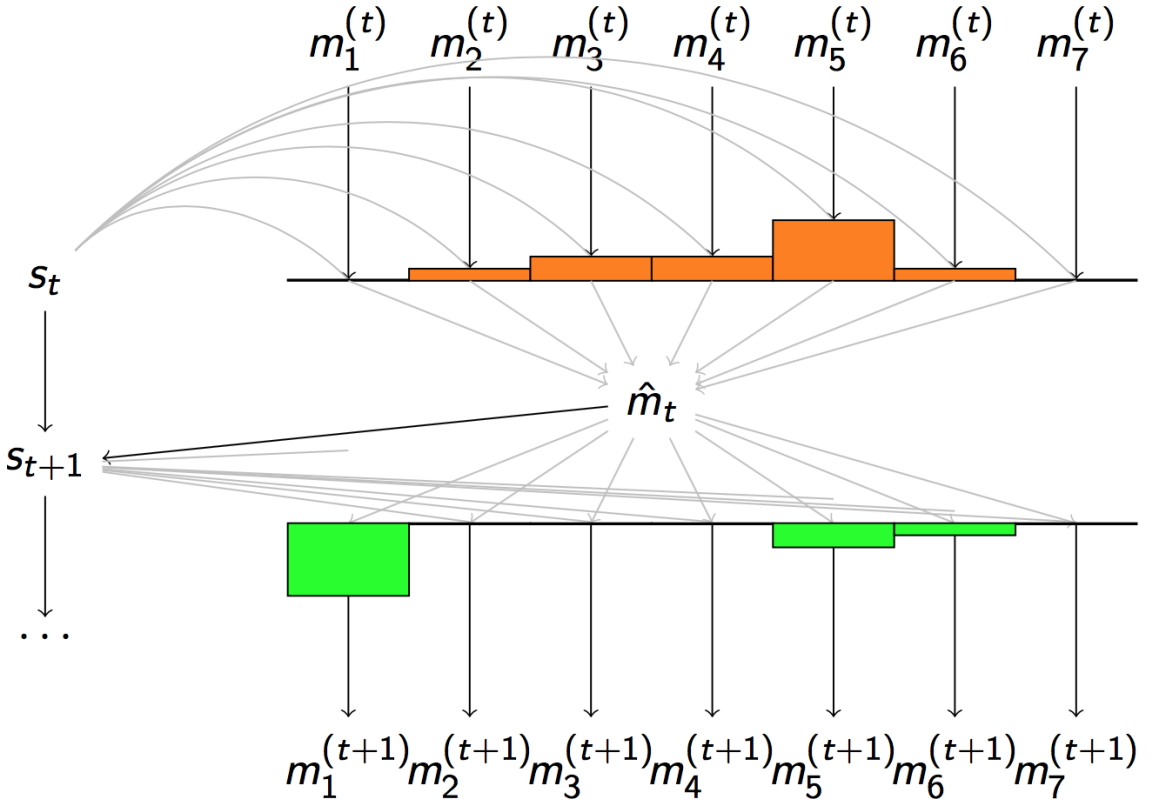
Mémoire

Permet de lire et d'écrire dans une mémoire

ATTENTION



MÉMOIRE


$m_1^{(t)}$ $m_2^{(t)}$ $m_3^{(t)}$ $m_4^{(t)}$ $m_5^{(t)}$ $m_6^{(t)}$ $m_7^{(t)}$
s_t
$\hat{m}_t$
s_{t+1}
...
$m_1^{(t+1)}$ $m_2^{(t+1)}$ $m_3^{(t+1)}$ $m_4^{(t+1)}$ $m_5^{(t+1)}$ $m_6^{(t+1)}$ $m_7^{(t+1)}$

LES APPLICATIONS

Différents cas d'usage en fonction des données disponible et de la granularité du traitement

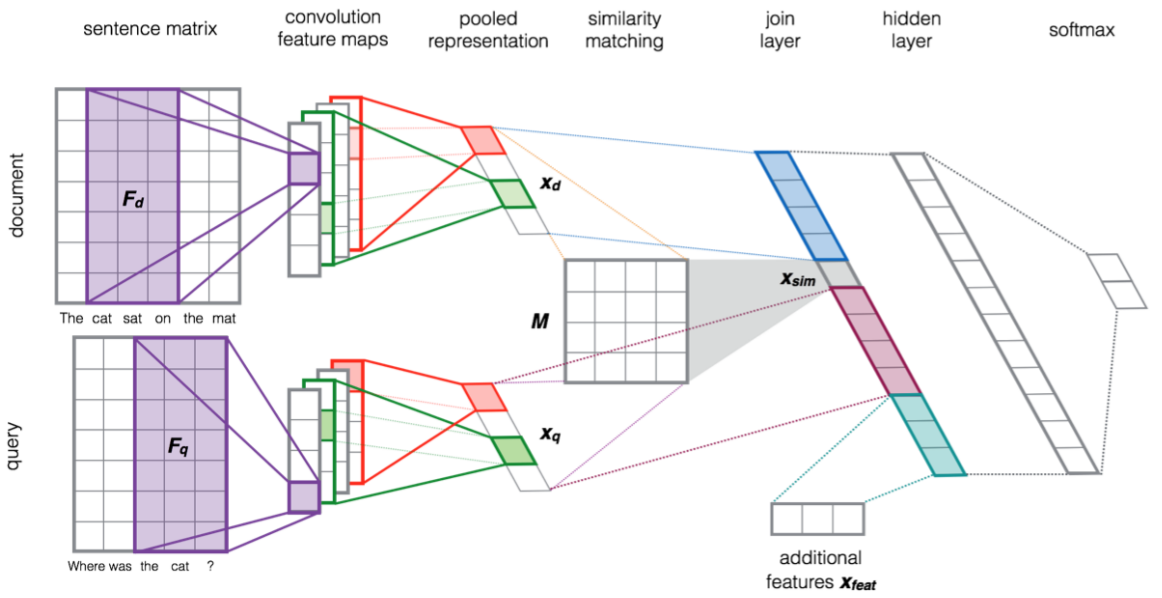
👉 Données

- On a plein de données: utilisation directe
- On a peu de données : utilisation indirecte

👉 Granularité

- Fine (phrase)
- Moyenne (passage)
- Grande (document)

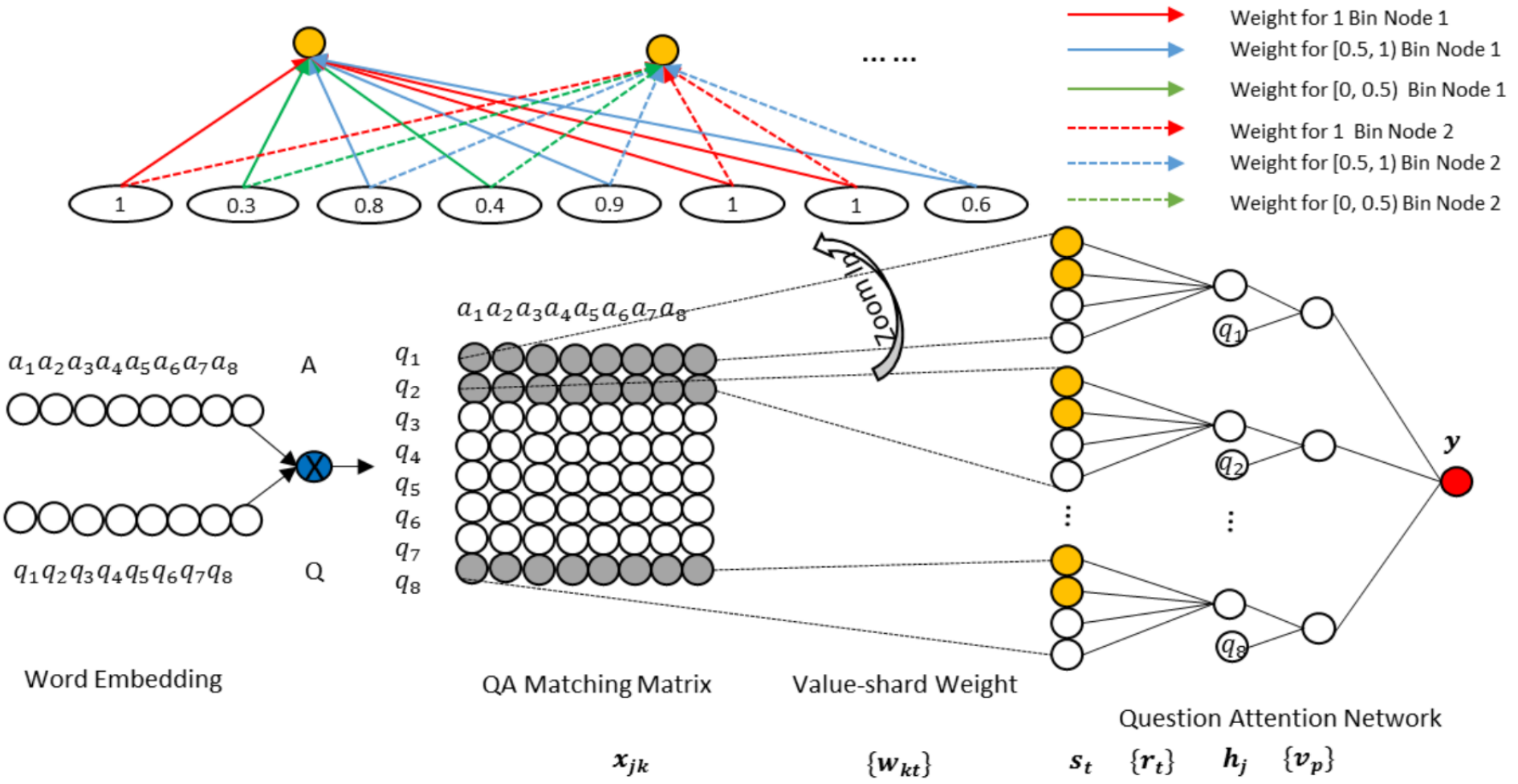
TEXTES COURTS: RE-RANKING DE RÉPONSES



Experiments: QA Trec Dataset

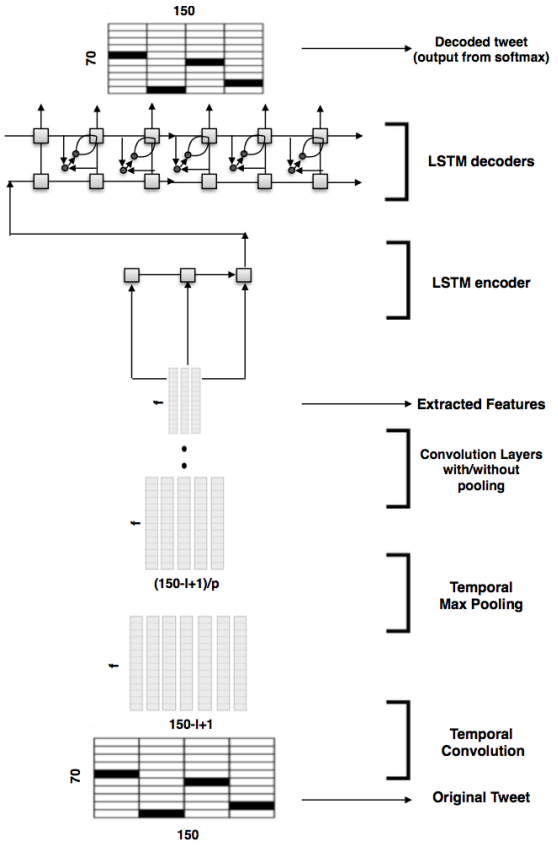
A. Severyn and A. Moschitti, “Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks,” SIGIR 2015.

MODÈLE D'ATTENTION (ANMM)



L. Yang, Q. Ai, J. Guo, and W. B. Croft, aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. CIKM 2016.

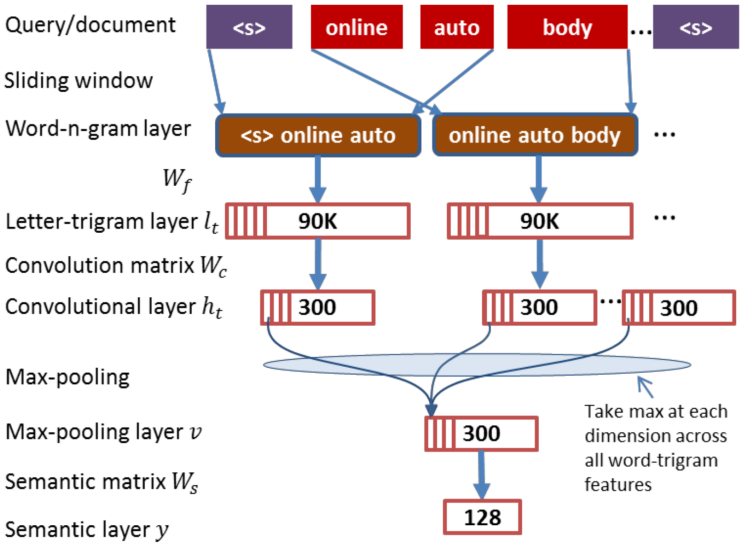
TEXTES COURTS: TWEET2VEC



- CNN (caractères) + LSTM
- Entrainement: auto-encodage sur 3 million de tweets (SemEval 2015-Task 1)
- Analyse de sentiments et similarité sémantique

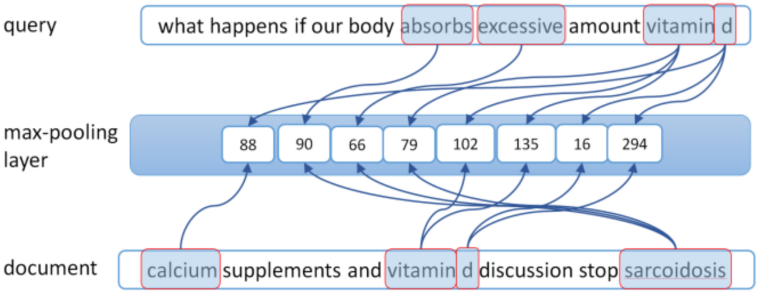
Vosoughi, S., Vijayaraghavan, P., & Roy, D. Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder. SIGIR 2016

TEXTES COURTS : RI ADHOC



(SUITE)

- Entrainé sur 12.071 questions (non public)
- Document = titre
- Visualisation :



Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. CIKM 2014.

LEARNING TO REWEIGHT QUERY TERMS

Dans le BIM (Binary Independance Model), on a

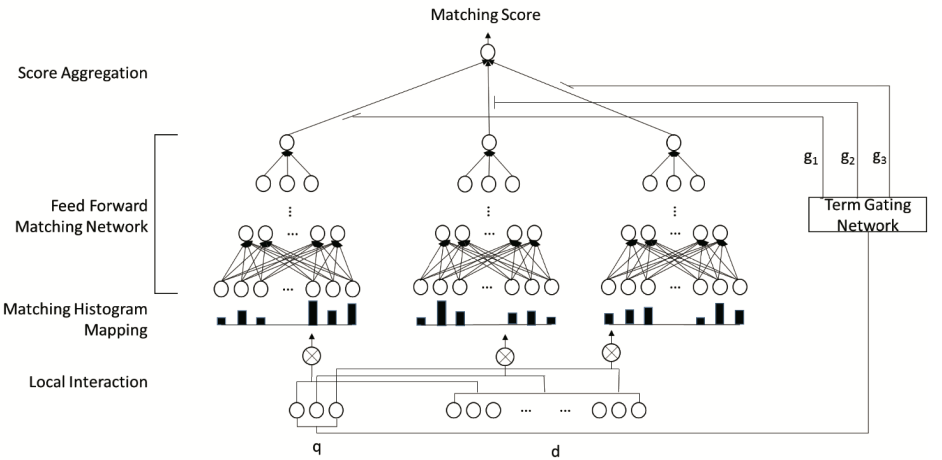
$$Score(q,d) = \sum_{t \in q \cap d} \log\left(\frac{p(t|R,q)}{1-p(t|R,q)} \times \frac{p(t|\neg R,q)}{1-p(t|\neg R,q)} \right) \times w(t,d)$$

👉 But: estimer $p(t|R,q)$

- Utilise la représentation $\boldsymbol{x}_t$ de chaque terme
- Moyenne $\bar{\boldsymbol{x}} = \frac{1}{|q|} \sum_{t \in q} \boldsymbol{x}_t$
- Entrée = déviation par rapport à la moyenne = $\boldsymbol{x}_t - \bar{\boldsymbol{x}}$
- Sortie désirée : $p(t|R,q)$
- Coût quadratique

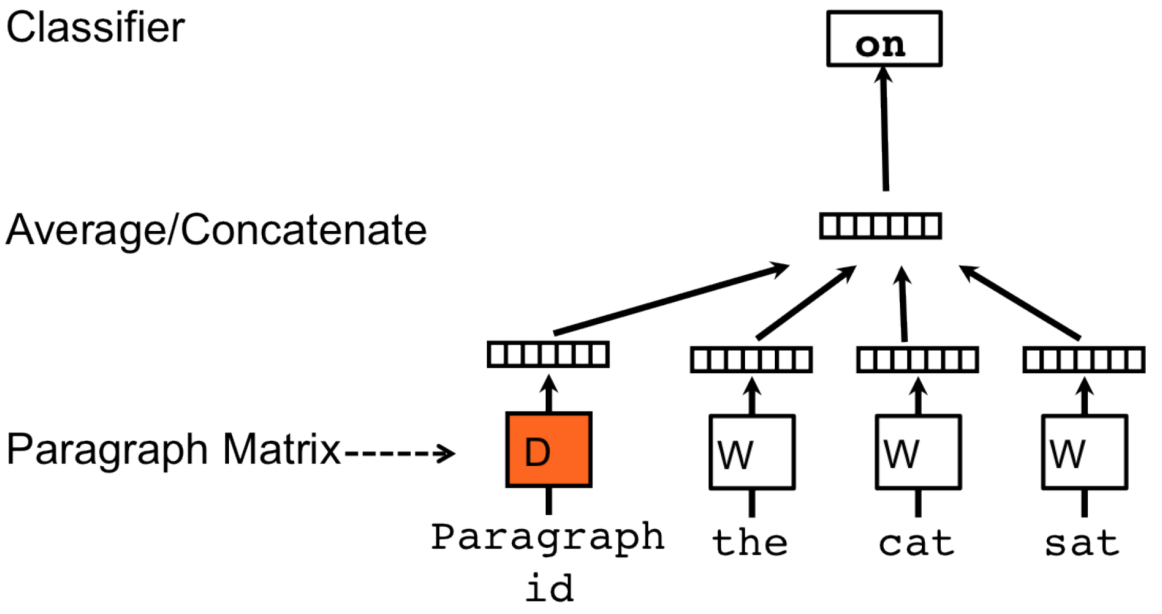
📖 Zheng, G., & Callan, J.. Learning to Reweight Terms with Distributed Representations. SIGIR 2015

A DEEP RELEVANCE MATCHING MODEL FOR AD-HOC RETRIEVAL



J. Guo, Y. Fan, Q. Ai, and W. B. Croft, “A Deep Relevance Matching Model for Ad-hoc Retrieval,” CIKM 2016.

PV-DBOW: INTRODUCTION



Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents (Vol. cs.CL). Presented at the International Conference on Machine Learning.

PV-DBOW

Point de départ: modèle de langue

$$p(w|d) = (1-\lambda)p_{QL}(w|d) + \lambda p_{PV}(w|d)$$

Une adaptation du modèle C-BOW (Word2Vec)

- Échantillonage spécifique
- Prise en compte de la longueur
- Prise en compte du document

Q. Ai, L. Yang, J. Guo, and W. B. Croft, “Improving Language Estimation with the Paragraph Vector Model for Ad-hoc Retrieval,”, SIGIR 2016

CONCLUSION

Les réseaux de neurones en RI

- CNN satisfaisants pour des textes courts
- LSTM pour des passages
- Au niveau des documents, ça ne marche pas... 👉 La représentation distribuée sert de manière détournée

📖 [discussion] Cohen, D., Ai, Q., & Croft, W. B. (2016). Adaptability of Neural Networks on Varying Granularity IR Tasks.

PRÉSENTATIONS GÉNÉRALES

Colah's blog (réseaux de neurones, convolution, LSTM)

OPTIMISATION

Y. LeCun, L. Bottou, G. B. Orr, and K.R. Müller, “Efficient BackProp” (2012)
R. Fletcher, Practical Methods of Optimization, 2000.