UE Innovation

Apprentissage automatique pour la génération de mots de passe

Majed Haddad majed.haddad@univ-avignon.fr [1] Mohamed Morchid mohamed.morchid@univ-avignon.fr

Sommaire

- Objectif général
- Etapes du défi
- Modèles d’apprentissage des mots de passe : les réseaux de neurones récurrents
- Implementation
- Attendu et évaluation pour le défi

Objectif général

Proposer un système permettant de générer des mots de passe :

- couvrant un ensemble de mots de passe contenus dans une base de données test (diversité des résultats)

- appris dans un laps de temps le plus restreint (efficacité algorithmique)

- nécessitant le moins de données d’apprentissage

Étapes du défi

- Compréhension des modélisation neuronales de génération de séquences
- Choix de la stratégie d’apprentissage :
  - ✦ modèle (RNN, LSTM, etc.)
  - ✦ données (corpus, train/dev/val, etc.)
- Prise en main des outils de développement des algorithmes (PyTorch)
- Implémentation du systèmes
- Analyse des résultats

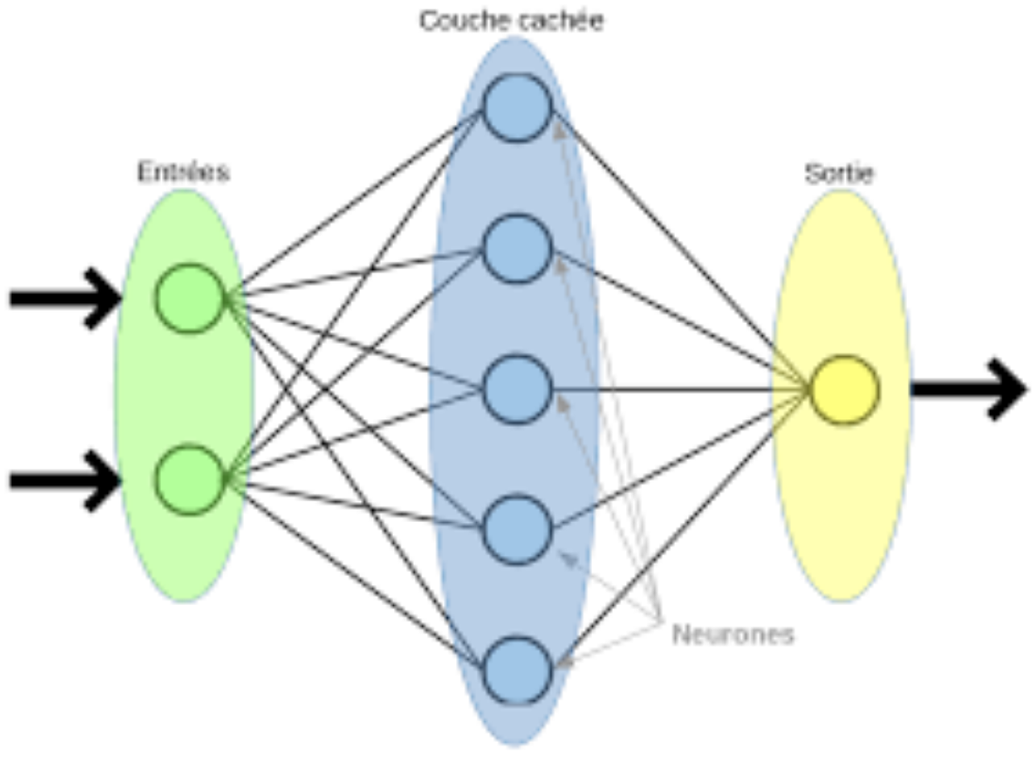
Réseaux de Neurones Artificiels

Phases d'apprentissage des Réseaux de Neurones Artificiels

A l'image de la biologie, le perceptron est un ensemble de neurones organisés en couche.

D'une couche à l'autre se propage le signal d'entrée jusqu'à la sortie à travers une ou plusieurs couches cachées, en activant ou non au fur et à mesure des neurones. (**forward**)

Le principe est de les renforcer ou les inhiber es liaisons entre les neurones en fonction de l'écart entre la sortie observée et la sortie attendue. (**backward**)



Réseaux de Neurones Artificiels

Le Perceptron Multicouche (MLP)

Modèle :

$$y_t = p(x_{t+1} \mid x_t, x_{t-1}, x_{t_2}, \ldots, x_1)$$

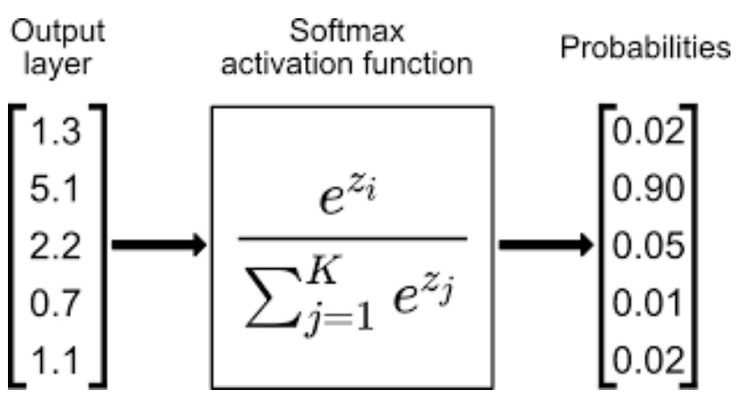
pour 1≤t<L

Minimiser la fonction objective cross-entropy :

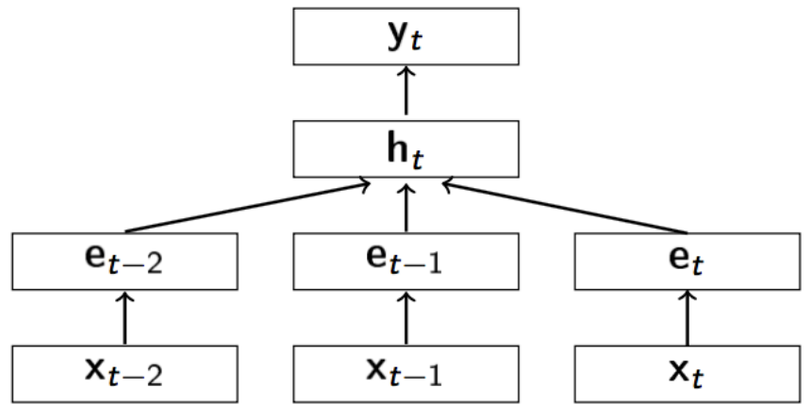
$$L = \sum_{t=1}^{L-1} H(y_t, x_{t+1}) = \sum_{t=1}^{V} \sum_{i=1}^{L-1} \log(y_{t,i}) x_{t+1,i}$$

σ (•) is a sigmoid-shaped function (e.g. logistic or tanh)

b• est un vecteur (biais), W • est une matrice de poids



Multi-Layer Perceptron (MLP)



$$\mathbf{y}_t = softmax\left(W^{yh}\mathbf{h}_t + \mathbf{b}^y\right)$$

$$\mathbf{h}_t = \sigma\left(W^{he}[\mathbf{e}_t; \mathbf{e}_{t-1}; \mathbf{e}_{t-2}] + \mathbf{b}^h\right)$$

$$\mathbf{e}_t = W^{ex}\mathbf{x}_t$$

Réseaux de Neurones Artificiels

Le Réseau de Neurones Récurrents (RNN)

Modèle :

$$y_t = p(x_{t+1} \mid x_t, x_{t-1}, x_{t_2}, \ldots, x_1)$$

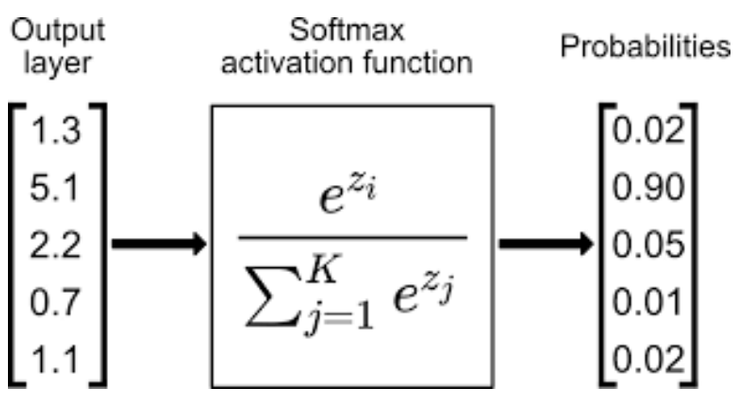
pour 1≤t<L

Minimiser la fonction objective cross-entropy :

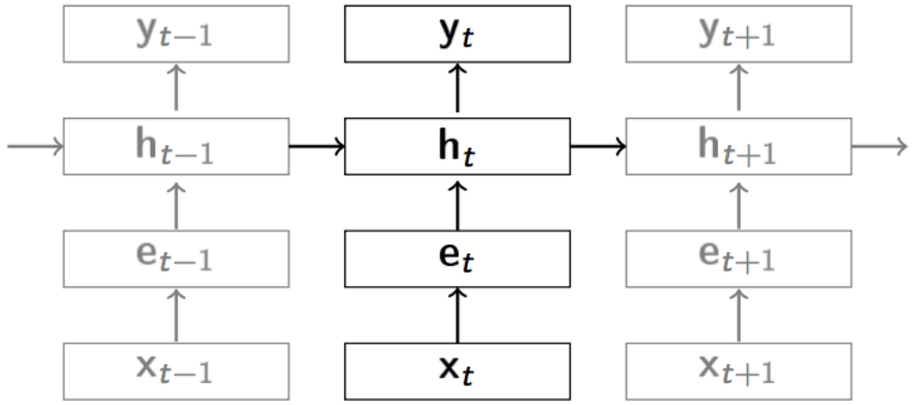
$$L = \sum_{t=1}^{L-1} H(y_t, x_{t+1}) = \sum_{t=1}^{V} \sum_{i=1}^{L-1} \log(y_{t,i}) x_{t+1,i}$$

σ (•) is a sigmoid-shaped function (e.g. logistic or tanh)

b• est un vecteur (biais), W • est une matrice de poids



Recurrent Neural Network (RNN)



$$\mathbf{y}_t = softmax\left(W^{yh}\mathbf{h}_t + \mathbf{b}^y\right)$$

$$\mathbf{h}_t = \sigma\left(W^{he}\mathbf{e}_t + W^{hh}\mathbf{h}_{t-1} + \mathbf{b}^h\right)$$

$$\mathbf{e}_t = W^{ex}\mathbf{x}_t$$

Réseaux de Neurones Artificiels

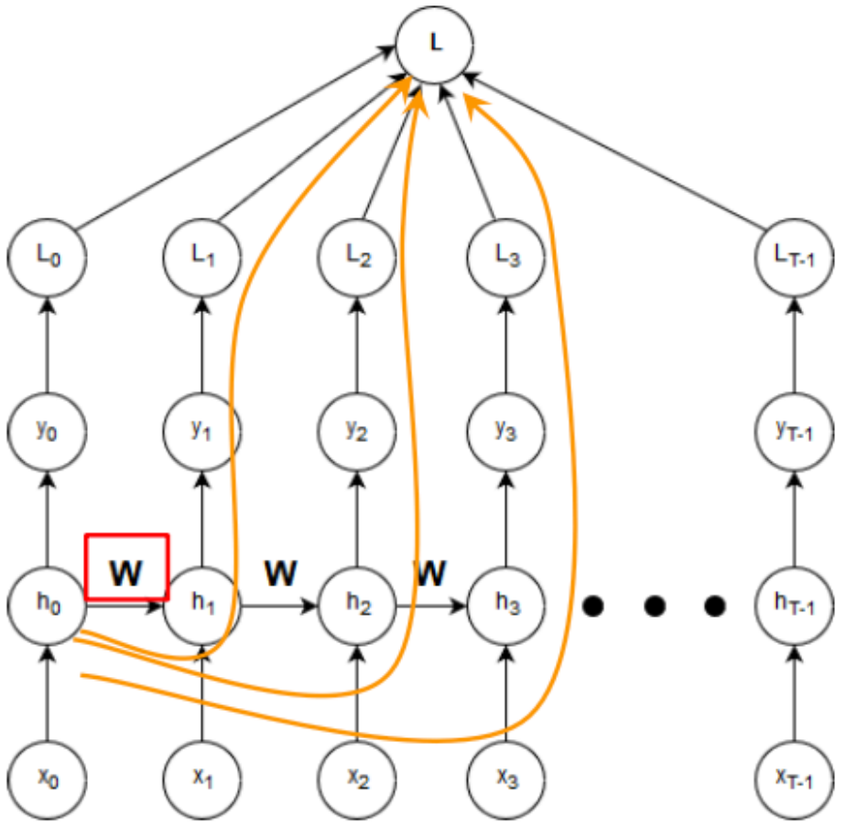
Rétropropagation du gradient (BPTT)

L'objectif est de mettre à jour la matrice de poids W

$$W \rightarrow W - \alpha \frac{\partial L}{\partial W}$$

Problème : W est partagée pour tous les éléments composant la séquence…

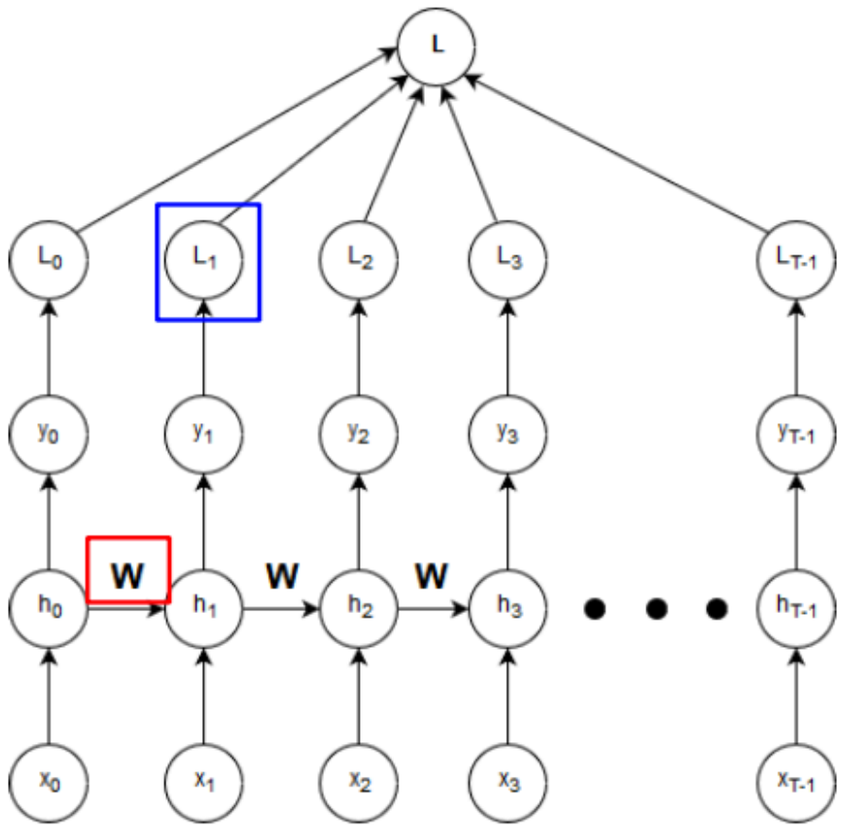
… Chaque chemin menant à L depuis W est une dépendance à considérer lors de la phase de rétropropagation du gradient de L



Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Combien de chemins sont possibles depuis W jusqu'à L en passant par L1 ?

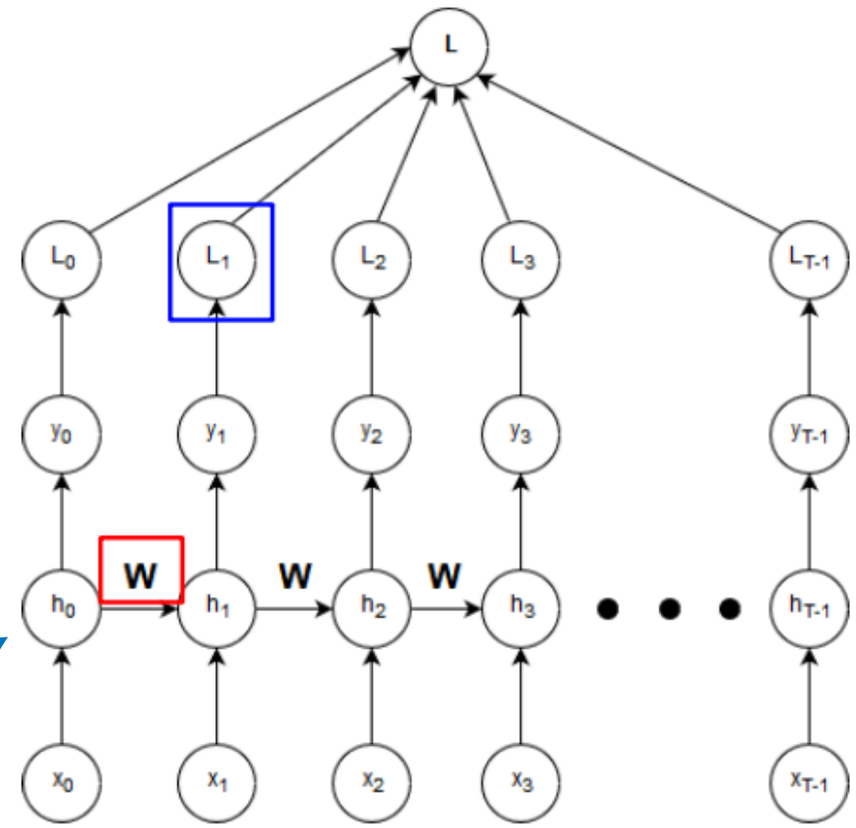


Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Combien de chemins sont possibles depuis W jusqu'à L en passant par L1 ?

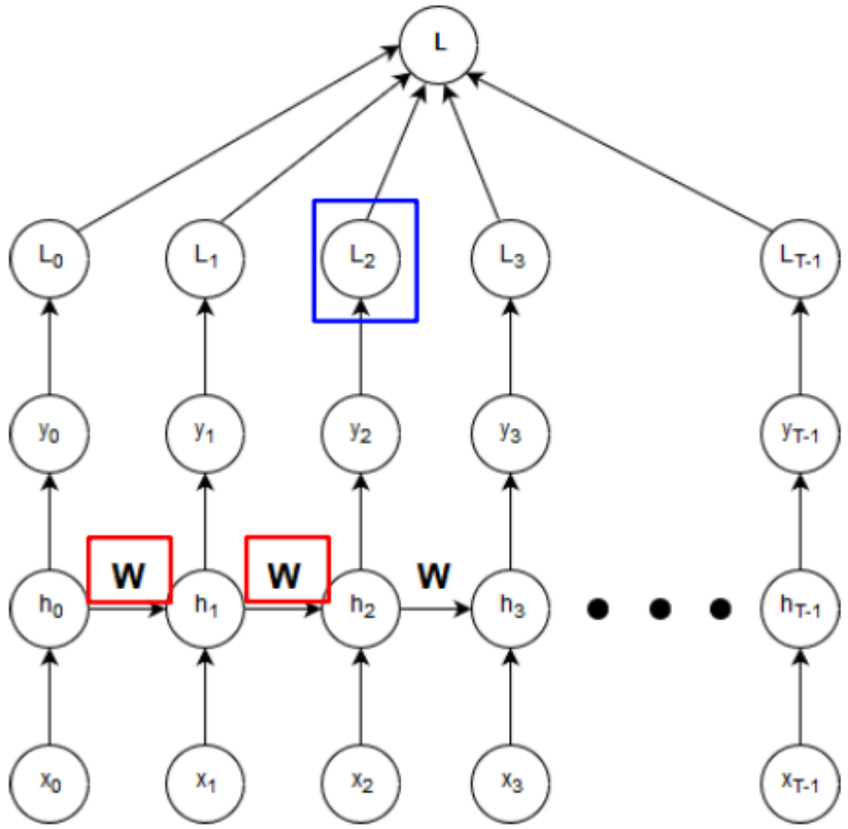
Uniquement 1 issu de h_0



Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

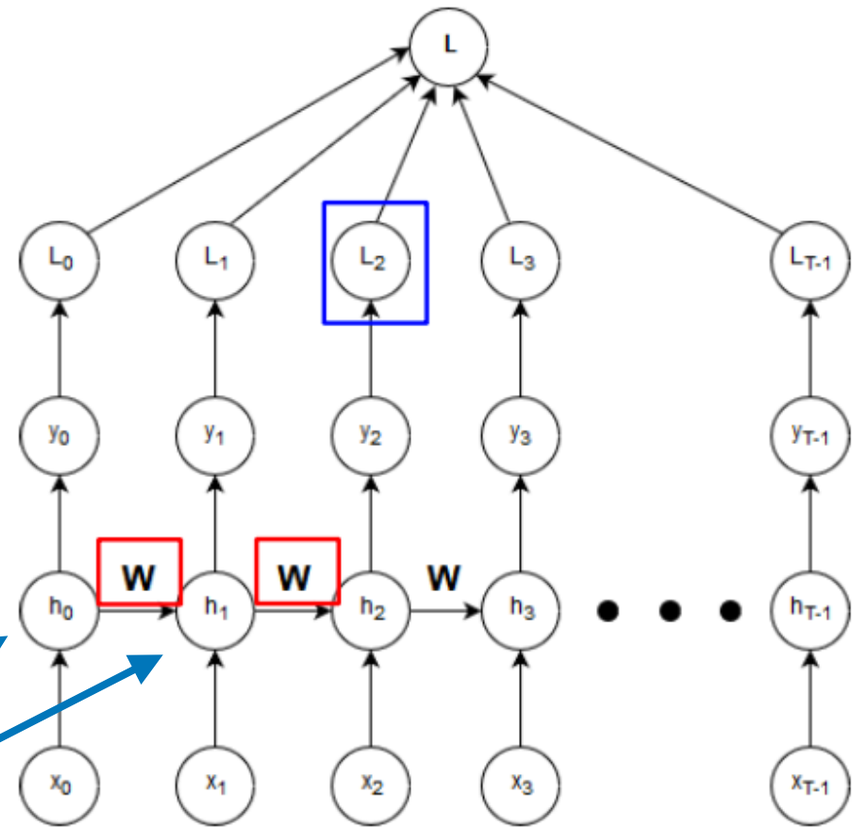
Combien de chemins sont possibles depuis W jusqu'à L en passant par L2 ?



Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Combien de chemins sont possibles depuis W jusqu'à L en passant par L2 ?

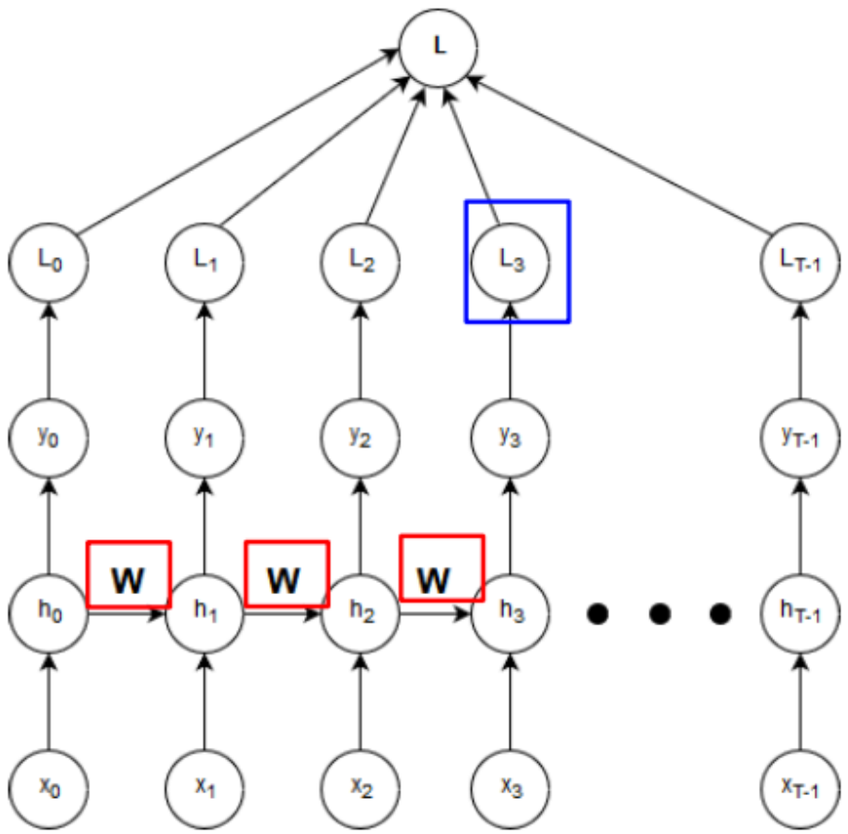


2 chemins sont possibles issu depuis h_0 **et** h_1

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Combien de chemins sont possibles depuis W jusqu'à L en passant par L3 ?



Réseaux de Neurones Artificiels

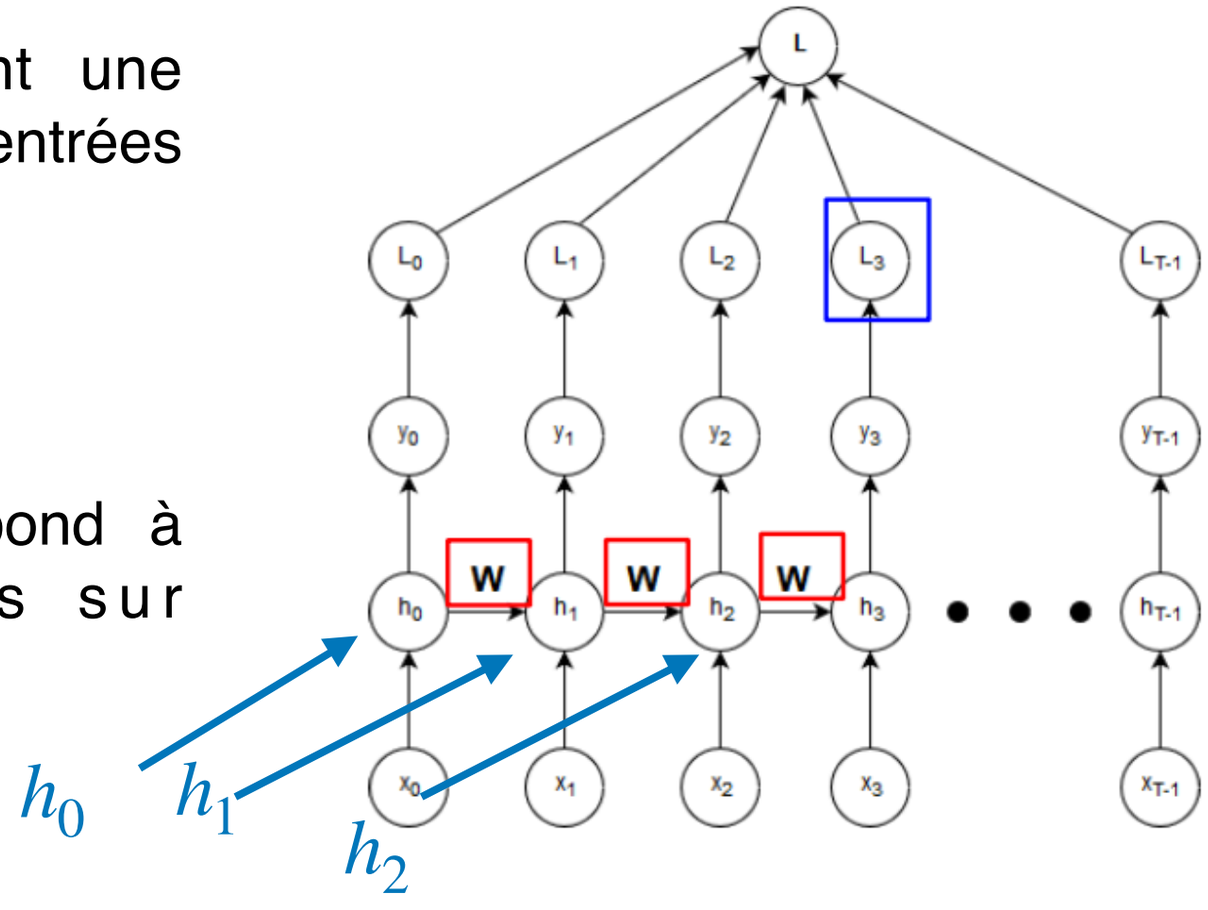
Rétropropagation du gradient (BPTT)

Dans ce dernier cas, il y a trois chemin possible.

Les chemins possibles forment une somme($\sum$)à travers toutes les entrées composant la séquence :

$$\frac{\partial L}{\partial W_h}$$

Le gradient de l'erreur correspond à deux sommes imbriquées sur l'ensembles des L_j et h_k .

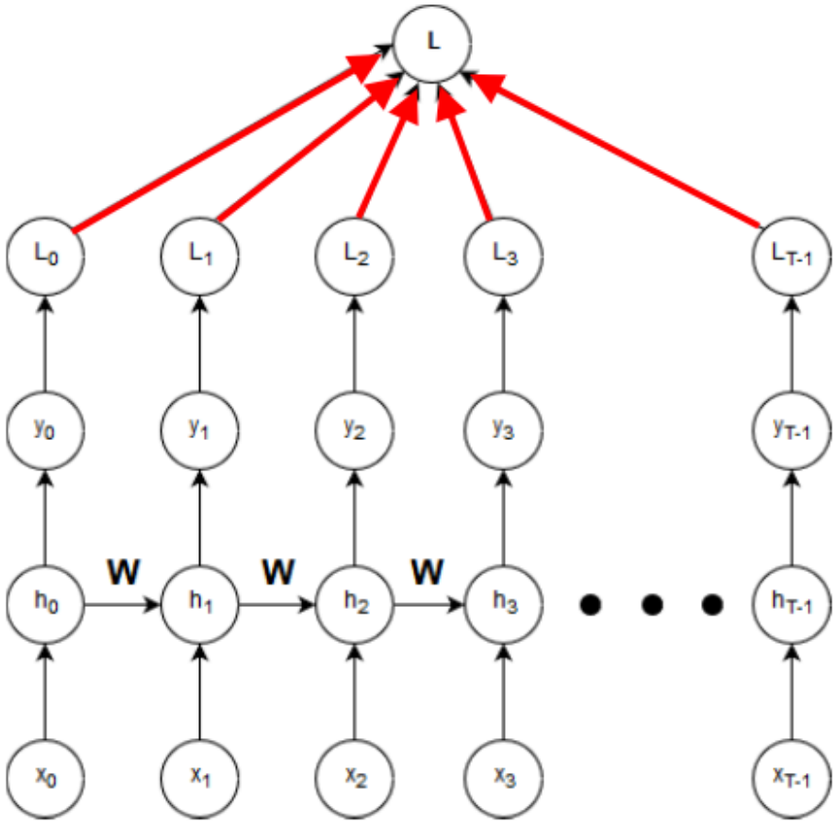


Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

La **première** somme concerne l'ensemble des L_j :

$$\frac{\partial L}{\partial W_h} = \sum_{j=0}^{T-1} \frac{\partial L_j}{\partial W_h}$$



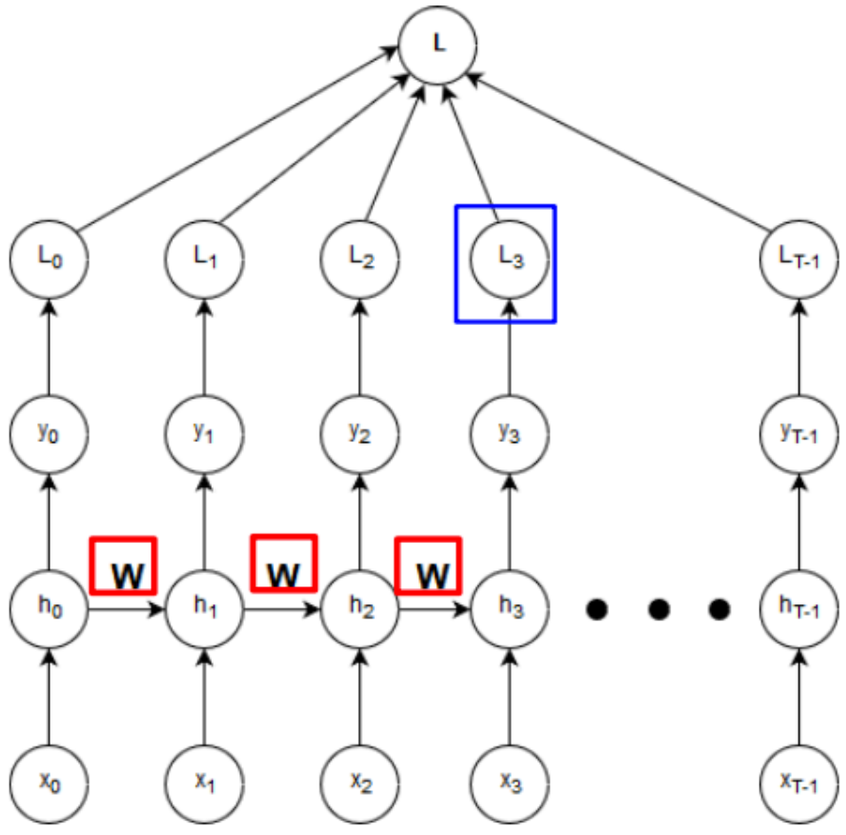
Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

La **seconde** traverse l'ensemble des états cachés précédents h_k.

Chaque L_j dépends des états précédent de la matrice W_h et des états cachés h_k :

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

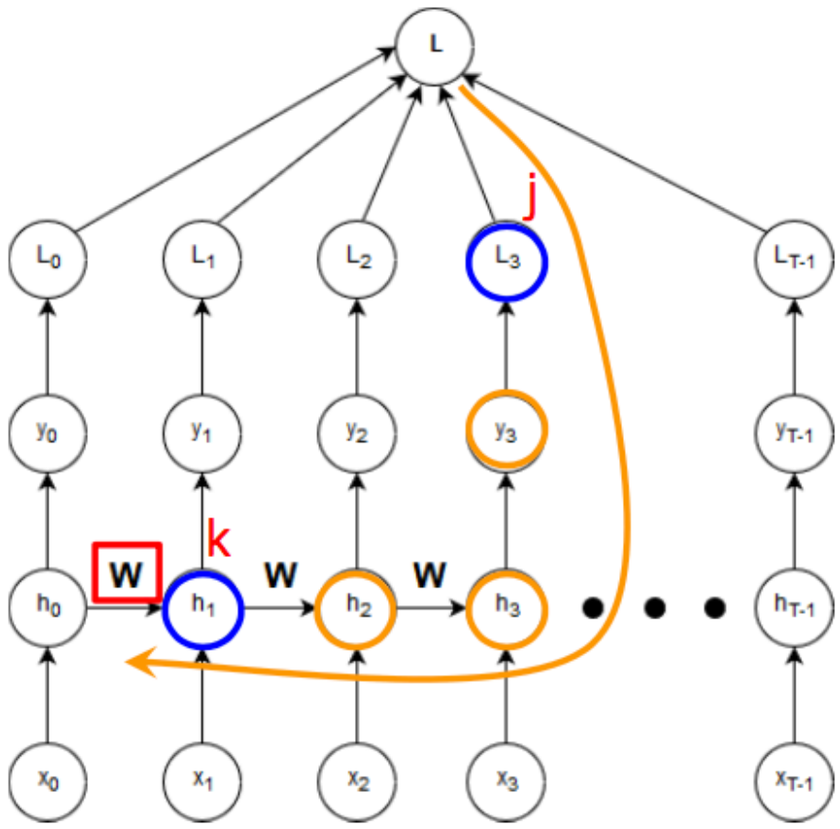


Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

Il n'y a pas d'**expression direct de L_j en fonction de h_k**.



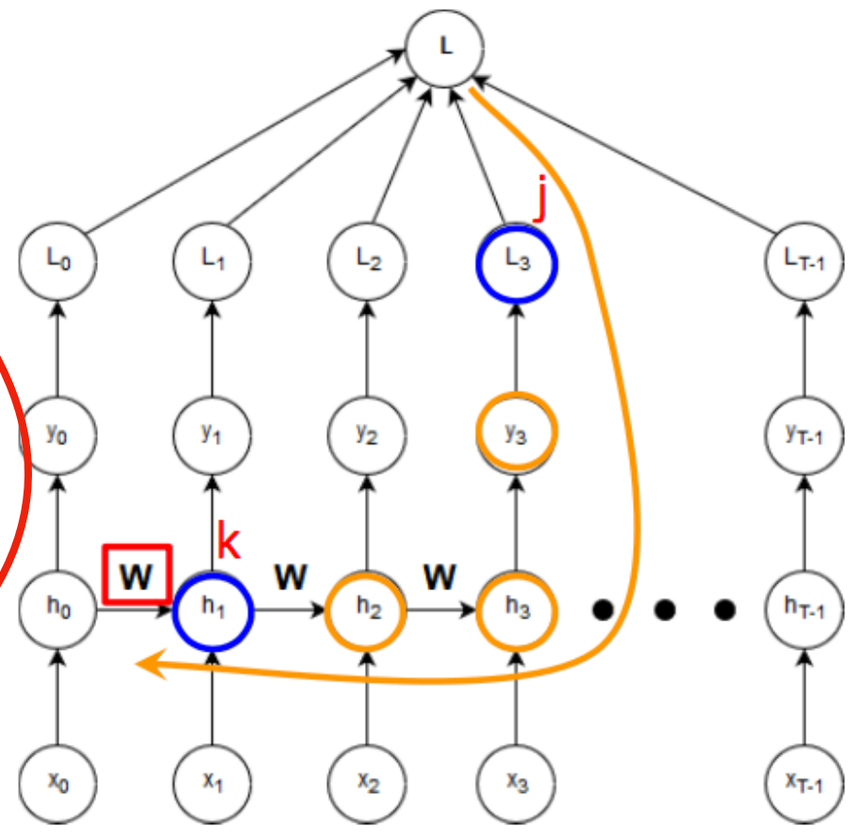
Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

Utilisation du théorème de dérivation des fonctions composées pour lier L_j à h_k au travers de tous les états cachés intermédiaires et de la sortie estimée y_j .

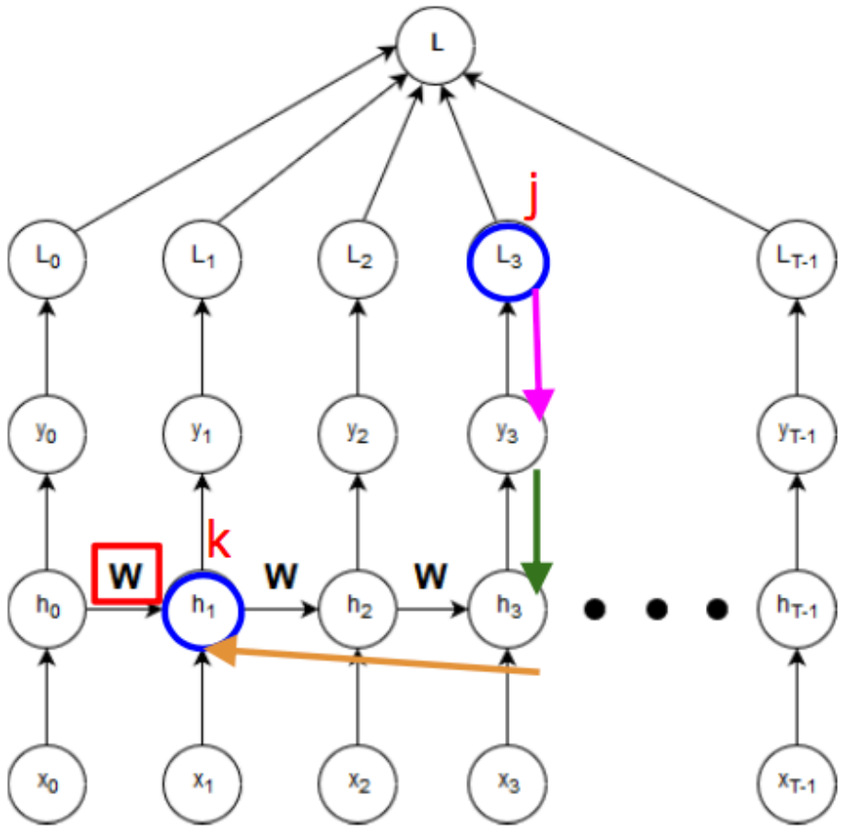
$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$



Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$



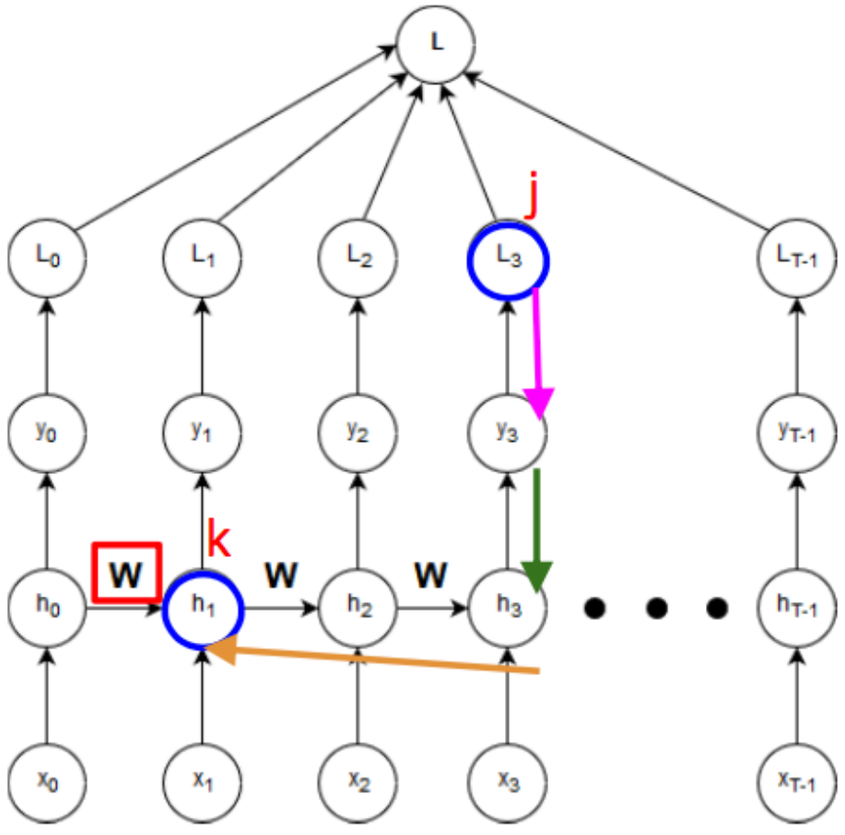
Utilisation du théorème de dérivation des fonctions composées pour lier L_j à h_k au travers de tous les états cachés intermédiaires et de la sortie estimée y_j .

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$



Cette partie est composée de dérivations basées sur des dépendances indirectes. En utilisant une dernière décomposition pour en extraire l'expression de :

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}}$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Ceci nous donne l'équation finale suivante pour la correction de la matrice de poids suivante :

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \left(\prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial h_k}{\partial W_h}$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Ceci nous donne l'équation finale suivante pour la correction de la matrice de poids suivante :

$$\frac{\partial L_j}{\partial W_h} = \sum_{k=1}^{j} \frac{\partial L_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \left(\prod_{m=k+1}^{j} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial h_k}{\partial W_h}$$

$$h_m = \alpha(W_x x_m + W_h h_{m-1})$$

On a donc :

$$\frac{\partial h_m}{\partial h_{m-1}} = W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression, c'est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

Matrice de poids

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T \, diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Dérivé de la fonction de transfert

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression, c'est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Multiplication de la matrice de poids qui entraîne ce phénomène

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression, c'est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Multiplication de la matrice de poids qui entraîne ce phénomène

Pourquoi ?

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression, c'est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Si la fonction α est la fonction identité et si W_h est une matrice diagonalisable, alors on peut décomposer W_h comme suit :

$$W_h = Q^{-1} A Q$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression, c'est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Si la fonction α est la fonction identité et si W_h est une matrice diagonalisable, alors on peut décomposer W_h comme suit :

$$W_h = Q^{-1} A Q$$

Matrice diagonale composée des valeurs propres de W_h

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

Le problème de cette expression est qu'elle a tendance à tendre soit vers 0 (vanishing) ou vers plus l'infini (exploding).

$$\frac{\partial h_j}{\partial h_k} = \prod_{m=k+1}^{j} W_h^T diag\left(\alpha'(W_x x_m + W_h h_{m-1})\right)$$

Si la fonction α est la fonction identité et si W_h est une matrice diagonalisable, alors on peut décomposer W_h comme suit :

$$W_h = Q^{-1} A Q$$

Matrice composée des vecteurs propres de W_h

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$W_h = Q^{-1} A Q$$

$$\left(W_h\right)^n = Q^{-1} \left(A\right)^n Q$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$W_h = Q^{-1} A Q$$

$$\left(W_h\right)^n = Q^{-1} \left(A\right)^n Q$$

Ce qui donne par exemple :

$$A = \begin{bmatrix} -0.71 & 0 & \\ 0 & b & 1.68 \end{bmatrix} \rightarrow A^{10} = \begin{bmatrix} 0.03 & 0 \\ 0 & 179.1 \end{bmatrix}$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$W_h = Q^{-1} A Q$$

$$\left(W_h\right)^n = Q^{-1} \left(A\right)^n Q$$

Ce qui donne par exemple :

Phénomène de disparition du gradient (vanishing)

$$A = \begin{bmatrix} -0.71 & 0 & \\ 0 & b & 1.68 \end{bmatrix} \rightarrow A^{10} = \begin{bmatrix} 0.03 & 0 \\ 0 & 179.1 \end{bmatrix}$$

Réseaux de Neurones Artificiels

Rétropropagation du gradient (BPTT)

$$W_h = Q^{-1} A Q$$

$$\left(W_h\right)^n = Q^{-1} \left(A\right)^n Q$$

Ce qui donne par exemple :

$$A = \begin{bmatrix} -0.71 & 0 & \\ 0 & b & 1.68 \end{bmatrix} \rightarrow A^{10} = \begin{bmatrix} 0.03 & 0 \\ 0 & \boxed{179.1} \end{bmatrix}$$

Phénomène d'explosion du gradient (exploding)

Réseaux de Neurones Artificiels

Solution au fortes variations du gradient

Différents solutions ont été apportées permettant de palier ce problème et de mieux traiter les dépendances longues :

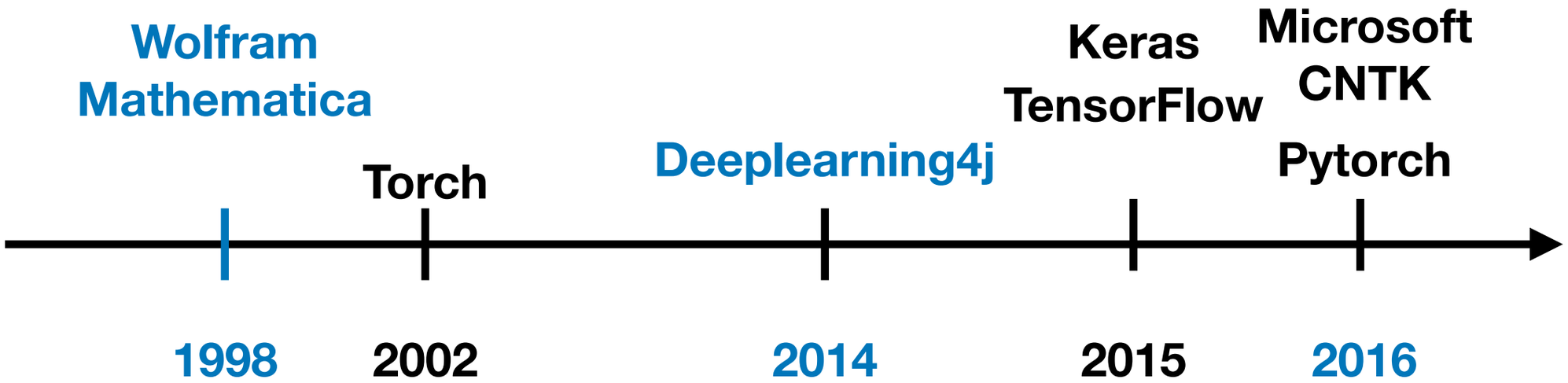
- RNN avec un Constant Error Carrousel (CEC)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- etc.

CEC : https://deepai.org/machine-learning-glossary-and-terms/constant%20error%20carousel
LSTM : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.4320&rep=rep1&type=pdf
GRU : https://arxiv.org/pdf/1412.3555.pdf?ref=hackernoon.com

Implementation

Historique des frameworks développés pour les NN

Différentes plateformes (GPU) de développement ont vu le jour :

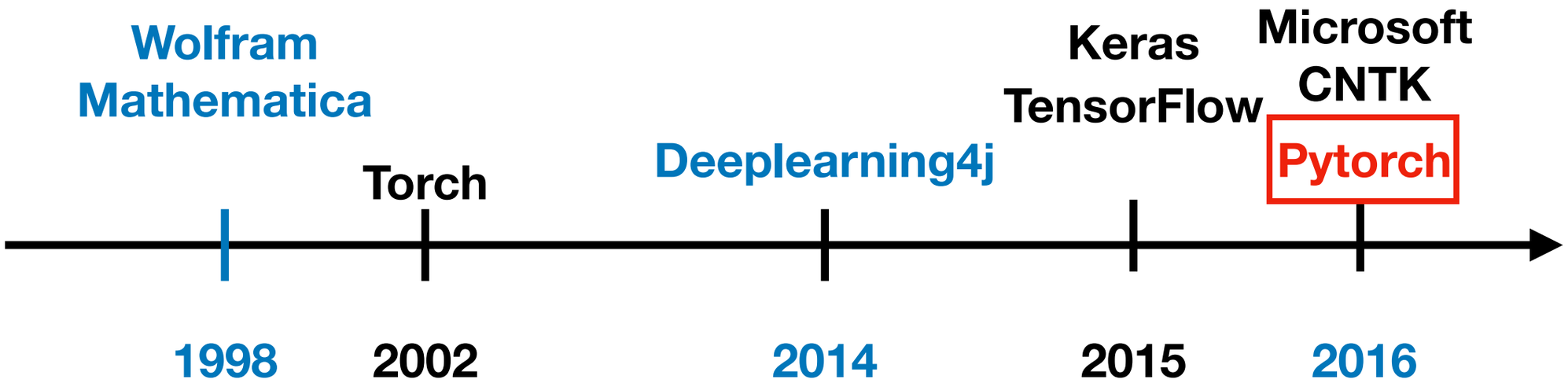


https://pytorch.org

Implementation

Historique des frameworks développés pour les NN

Différentes plateformes (GPU) de développement ont vu le jour :

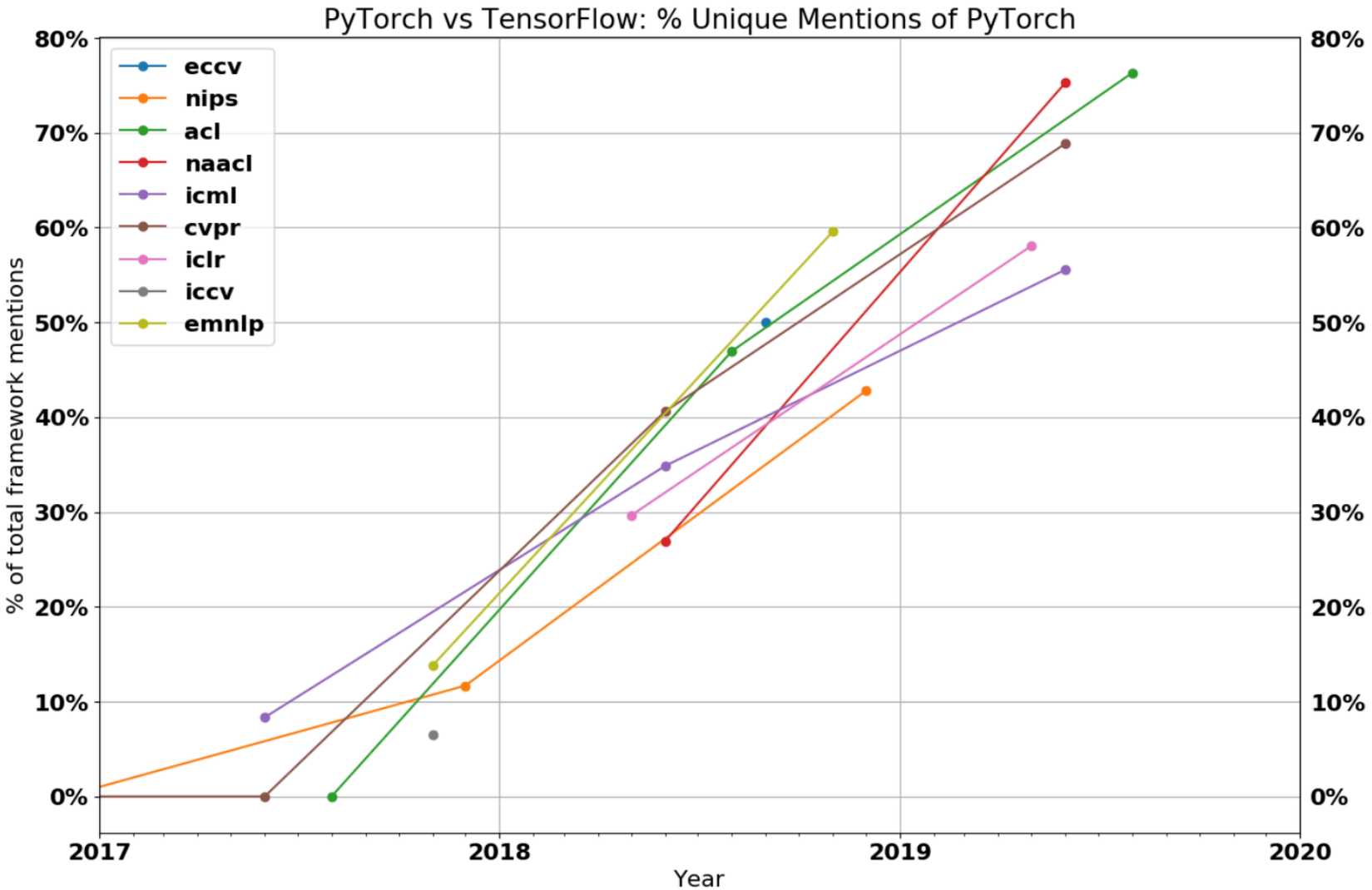


https://pytorch.org

Implementation

Pytorch *vs.* TensorFlow

Pourquoi Pytorch ?



https://pytorch.org

Implementation

Pytorch *vs.* TensorFlow

Pourquoi Pytorch ?

- **Pytorch s'intègre facilement** au reste de l'écosystème Python. Par exemple, vous pouvez simplement ajouter un point d'arrêt pdb n'importe où dans votre modèle PyTorch et cela fonctionnera. Dans TensorFlow, le débogage du modèle nécessite une session active et finit par être beaucoup plus délicat.

- **PyTorch est mieux conçu** et en partie parce que TensorFlow s'est handicapé en changeant souvent d'API (par exemple, «couches» -> «slim» -> «estimateurs» -> «tf.keras»).

- **PyTorch est** aussi **rapide** sinon plus rapide que TensorFlow.

https://pytorch.org

Implementation

Exemple de RNN avec Pytorch

Pourquoi Pytorch ?

```
class Model(nn.Module):
    def __init__(self, x_size, y_size, h_dim, n_layers):
        super(Model, self).__init__()
        self.hidden_dim = hidden_dim
        self.n_layers = n_layers

        self.rnn = nn.RNN(x_size, h_dim, n_layers)
        self.fc = nn.Linear(h_dim, y_size)

    def forward(self, x):
        batch_size = x.size(0)
        h = self.init_hidden(batch_size)

        y, h = self.rnn(x, h)
        y = out.contiguous().view(-1, self.hidden_dim)
        y = self.fc(y)

        return y, hidden
```

Attendu et évaluation

Attendus du travail pour ce défi : **Analyse des données**

Une étude des corpus :

- Choix du corpus d'apprentissage. Vous avez la possibilité de prendre d'autres corpus en plus de celui fournit (à justifier) mais avec les mêmes données de test.

- Définition de (ou des) stratégie(s) de gestion des données

- Analyses statistiques des données (variabilité intra-données avec une PCA par exemple, statistiques de cooccurrences *à minima*).

Attendu et évaluation

Attendus du travail pour ce défi : **Réflexions autour du modèle**

Compréhension des modèles existants :

- Analyse des modèles neuronaux existants pour le traitement et la génération de séquences

- Définition d’une stratégie d’apprentissage (DropOut, Normalisation, #layers, hyper-paramètres du modèle, etc.)

Attendu et évaluation

Attendus du travail pour ce défi : **Implémentation**

Implémentation à proprement dite du système :

- Prise en main de PyTorch (recherche au sein de la documentation, réalisation de petits exemples, etc)

- Implémentation du (ou des) système(s) de génération de mots de passe

Attendu et évaluation

Attendus du travail pour ce défi : **Implémentation**

Analyse des résultats :

- Evaluation du modèle à l'aide des données de validation (permettant de choisir la meilleure configuration des hyper-paramètres du modèle)

- Evaluation à l'aide des données de test communes à tous les étudiants en termes de temps de calcul est de taux de recouvrement (#mdp trouvés/#mdp contenus dans le test)

- Analyse des résultats et propositions d'améliorations

Des questions ?