

Apprentissage automatique 1 (EMINF2C1)

Cours 3 : Réseaux de neurones récurrents

Contributeur :

Stergos Afantenos

Philippe Muller

Contact : prenom.nom@irit.fr

Introduction

- Lors du cours précédent nous avons vu comment nous pouvons traiter les données de taille fixe, et le cas particulier des images à l'aide des Réseaux Convolutifs (Convolutional Neural Networks, CNNs).
- La question se pose naturellement : comment pouvons nous traiter les *données séquentielles* qui peuvent ne pas avoir une taille fixe, comme par exemple les signaux sonores, le texte, etc ?

Motivation

Analyse de sentiments

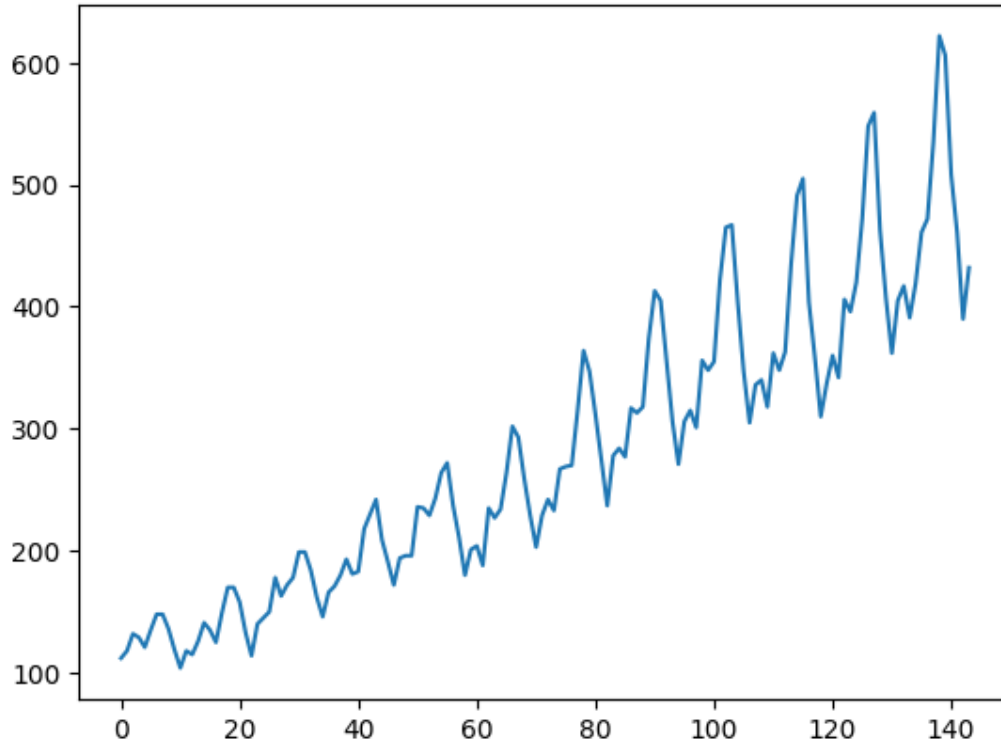
Loves the German bakeries in Sydney. Together with my imported honey it feels like home	Positive
@VivaLaLauren Mine is broken too! I miss my sidekick	Negative
Finished fixing my twitter...I had to unfollow and follow everyone again	Negative
@DinahLady I too, liked the movie! I want to buy the DVD when it comes out	Positive
@frugaldougal So sad to hear about @OscarTheCat	Negative
@Mofette brilliant! May the fourth be with you #starwarsday #starwars	Positive
Good morning thespians a bright and sunny day in UK, Spring at last	Positive
@DowneyisDOWNEY Me neither! My laptop's new, has dvd burning/ripping software but I just can't copy the files somehow!	Negative

Image credit

Motivation

Séries temporelles

Exemple : données du trafic aérien international



Motivation

Reconnaissance de la parole

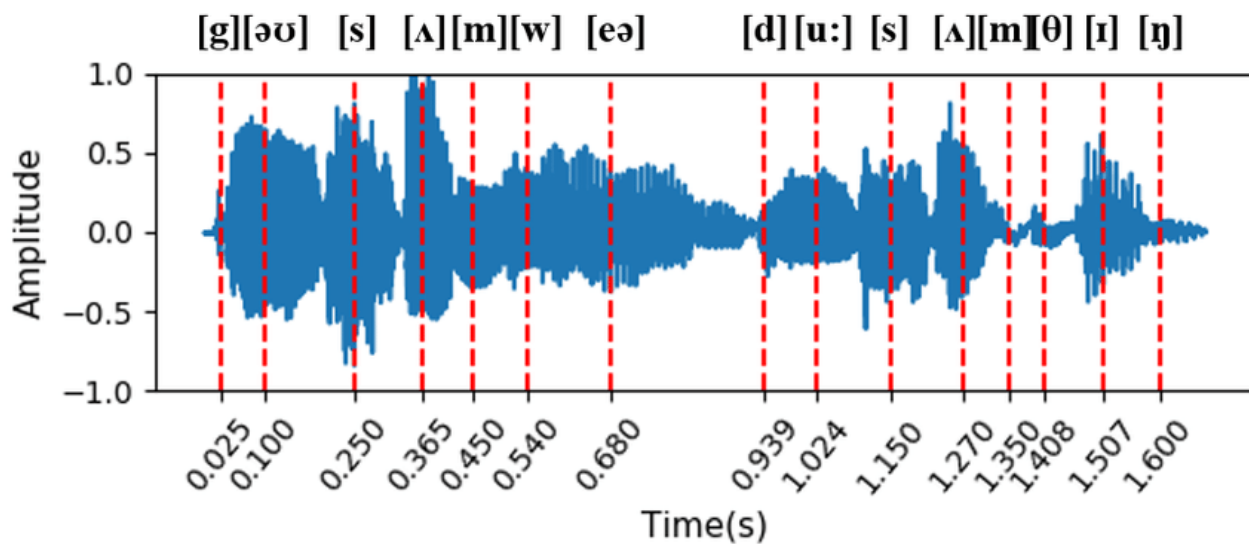


Image credit

Motivation



Traduction automatique

TextDocuments

DETECT LANGUAGEENGLISHGREEKFRENCH





↔FRENCHENGLISHSPANISH

In terms of energy efficiency, on bikes humans have even surpassed natural evolution: to move 1kg of body mass 1km, a cyclist on a normal bike uses only 0.136 calories while a seagull uses 1.433 calories to travel the same distance.




232 / 5,000


En termes d'efficacité énergétique, à vélo, les humains ont même dépassé l'évolution naturelle : pour déplacer 1 kg de masse corporelle sur 1 km, un cycliste sur un vélo normal utilise seulement 0,136 calories tandis qu'une mouette utilise 1,433 calories pour parcourir la même distance.



Send feedback

Image credit





6/54

Motivation

Génération de légendes à partir d'une image



a little girl sitting on a bench holding an umbrella.



a herd of sheep grazing on a lush green hillside.



a close up of a fire hydrant on a sidewalk.



a yellow plate topped with meat and broccoli.



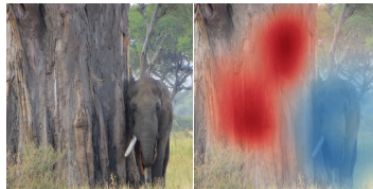
a zebra standing next to a zebra in a dirt field.



a stainless steel oven in a kitchen with wood cabinets.



two birds sitting on top of a tree branch.



an elephant standing next to rock wall.



a man riding a bike down a road next to a body of water.

Image credit

Motivation

Création de résumés

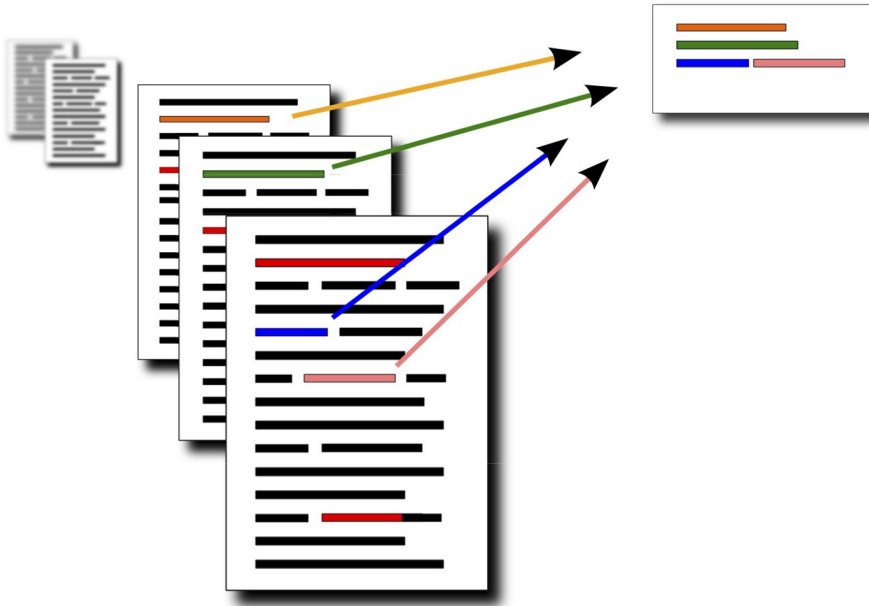


Image credit

Motivation

Entrée structurée

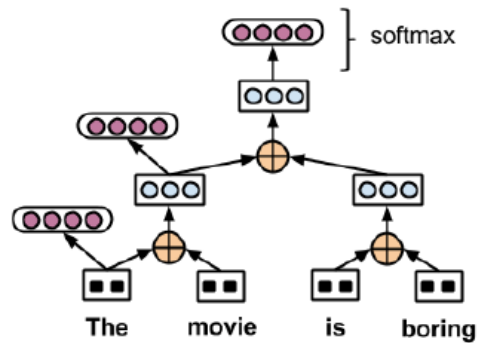
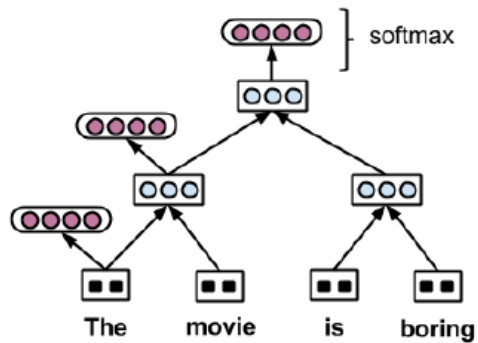


Image credit

Motivation

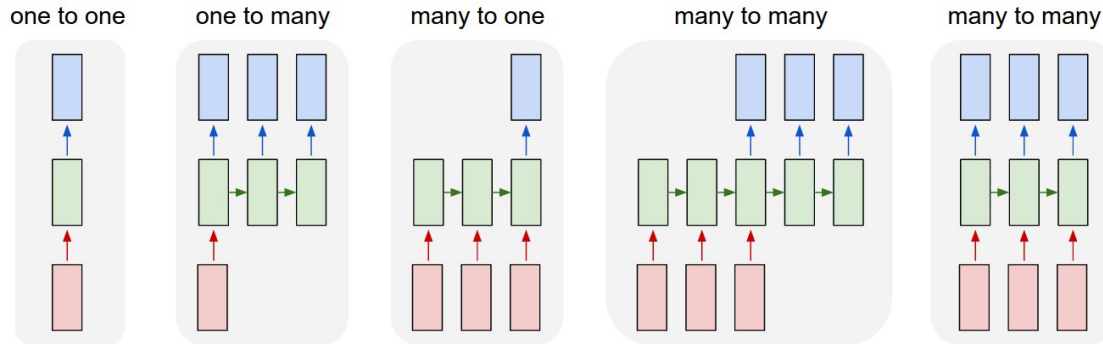
Applications pour lesquelles nous avons des données de taille variable :

- Reconnaissance de la parole
- Traitement de données textuelles :
 - Traduction automatique
 - Création de résumés
 - Analyse de sentiments
 - ...
- Génération de légendes à partir d'une image
- Vidéos
- Séries financières
- Données biologiques
- Signaux médicaux
- ...

Modélisation de séquences

- Dans un problème “classique” nous avons un “objet” et nous voulons le classier dans une classe, parmi un ensemble fini de classes. C’est un problème **one-to-one**
- Cette situation n’est pas toujours convenable, et d’autres configurations peuvent exister.

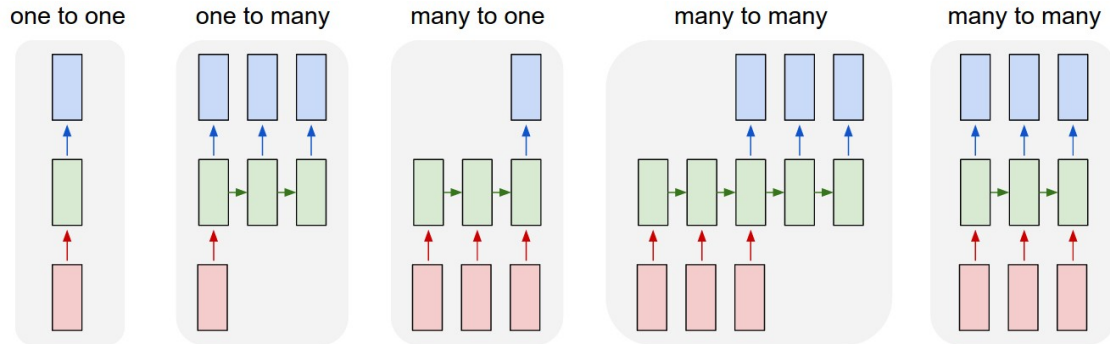
Modélisation de séquences



one-to-many : Prendre en entrée un objet (une image, par exemple) et produire une séquence (une légende, par exemple)

Image credit

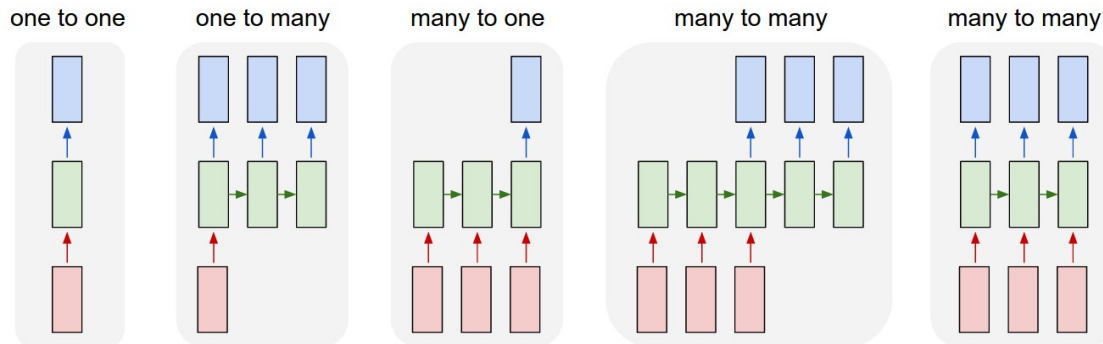
Modélisation de séquences



many-to-one : Prendre en entrée une séquence (un texte, par exemple) et le classifier (faire une analyse de sentiments, par exemple).

Image credit

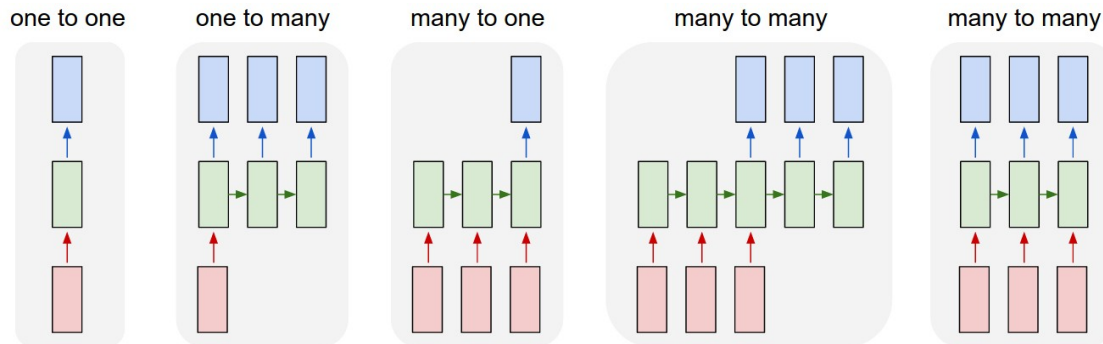
Modélisation de séquences



many-to-many : Prendre en entrée une séquence (un texte en français, par exemple) et produire une autre séquence (sa traduction en anglais, par exemple)

Image credit

Modélisation de séquences



many-to-many (bis) : Prendre en entrée une séquence (un signal sonore discrétisé) et produire une autre séquence pour chaque élément de l'entrée (une transcription)

Image credit

Réseaux de Neurones Récurrents : RNN

- Pour beaucoup de problèmes nous avons une suite de données

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(\tau)}$$

et nous souhaitons produire une séquence de sortie

$$\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(\tau)}$$

- L'idée au cœur des RNN : nous pouvons avoir un modèle séparé pour chaque $\mathbf{x}^{(i)}$ mais ensuite **partager** les paramètres entre les différents modèles via des états internes

$$\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \dots, \mathbf{h}^{(\tau)}$$

Réseaux de Neurones Récurrents : RNN

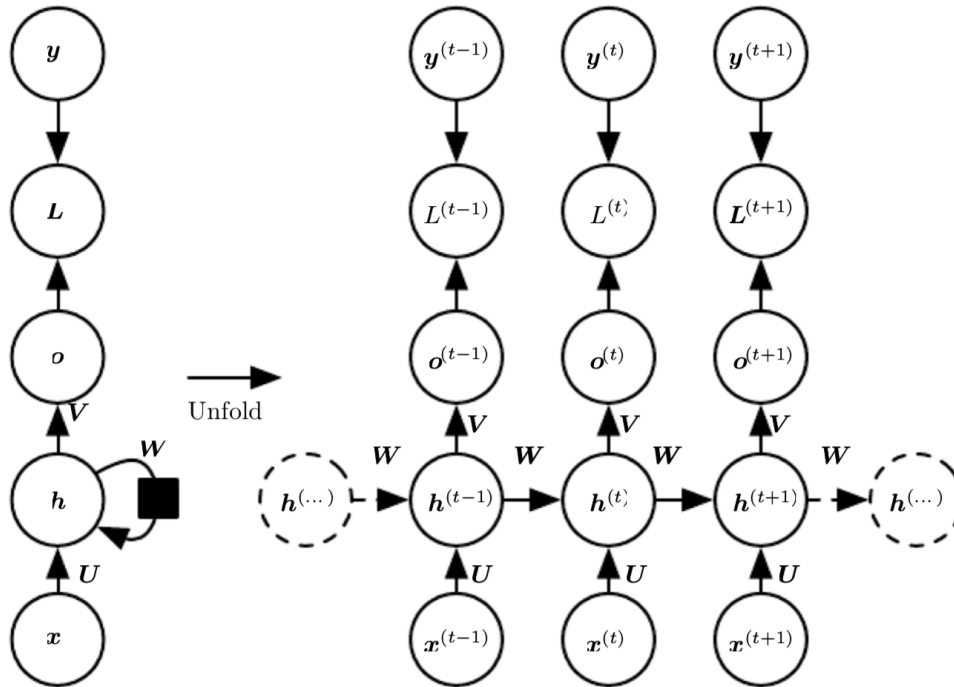


Image taken from <https://github.com/janishar/mit-deep-learning-book-pdf>

Réseaux de Neurones Récurrents : RNN

- Nous pouvons décrire le RNN précédent avec les équations suivantes :

$$\mathbf{h}^{(t)} = \tanh(\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{\mathbf{y}}^{(t)} = f(\mathbf{o}^{(t)})$$

où \mathbf{b} , \mathbf{W} , \mathbf{U} ; \mathbf{c} , \mathbf{V} sont des paramètres à apprendre.

- Le RNN le plus simple est le suivant :

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

$$\hat{\mathbf{y}}^{(t)} = f(\mathbf{V}\mathbf{h}^{(t)})$$

avec \mathbf{W} , \mathbf{U} , \mathbf{V} des paramètres à apprendre.

Réseaux de Neurones Récurrents : RNN

- Ce qui est important est que les paramètres sont **partagés** à travers le temps.
- $\mathbf{h}^{(t-1)}$ peut également être un paramètre.

Une autre architecture encore :

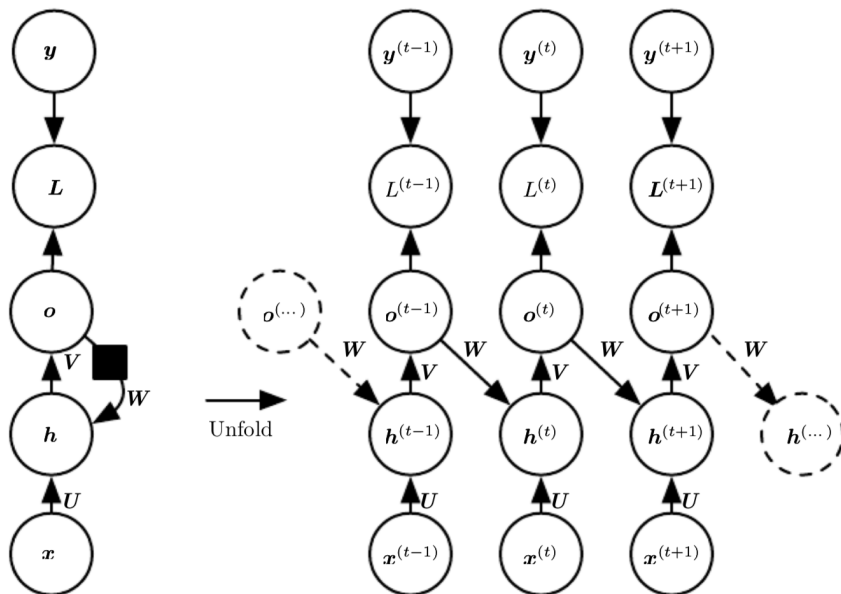


Image taken from <https://github.com/janishar/mit-deep-learning-book>

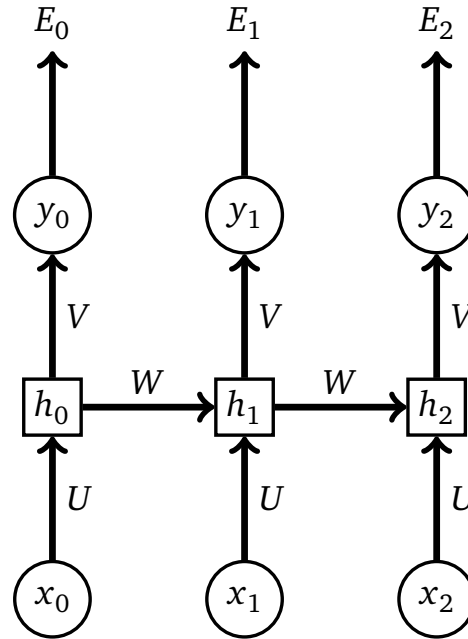
Rétropropagation à travers le temps

- Erreur global :

$$E = \sum_{t=0}^T E_t$$

- Rétropropagation pour adapter les paramètres

$$\frac{\partial E}{\partial U} = \sum_{t=0}^T \frac{\partial E_t}{\partial U}$$

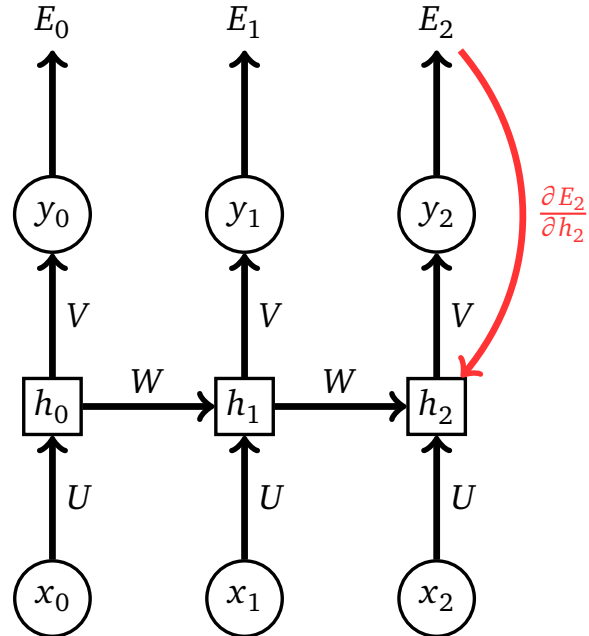


Inspiré par César Laurent, Réseaux Récurrents, École d'été IVADO, 2017

Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- et nous propageons l'erreur jusqu'à h_2

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \dots$$

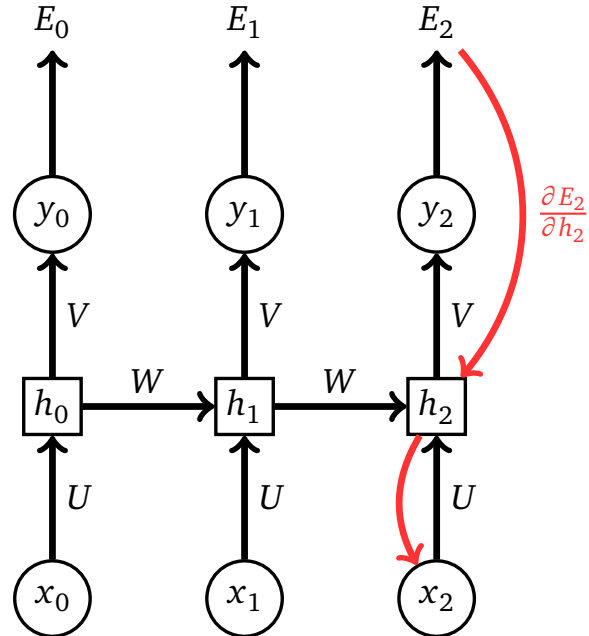


Inspiré par César Laurent, Réseaux Récurrents, École d'été IVADO, 2017

Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- l'erreur est ensuite propagée à U

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} (x_2^T \dots$$

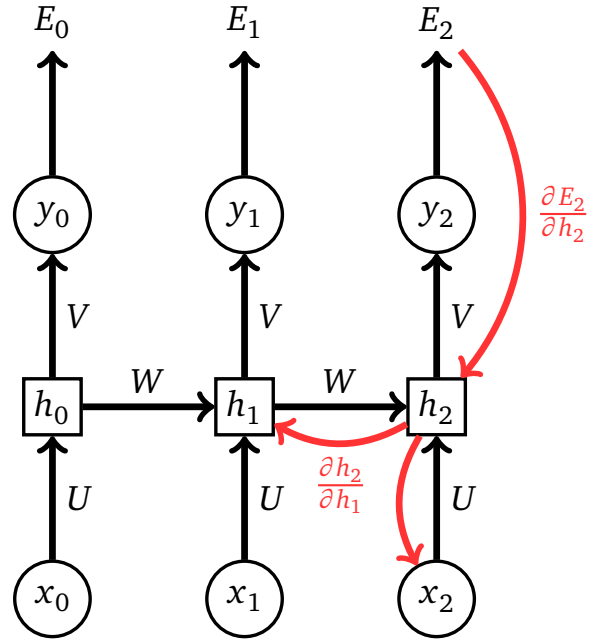


Inspiré par César Laurent, Réseaux Récurrents, École d'été IVADO, 2017

Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- Nous devons continuer à rétropropager vers h_0 et h_1

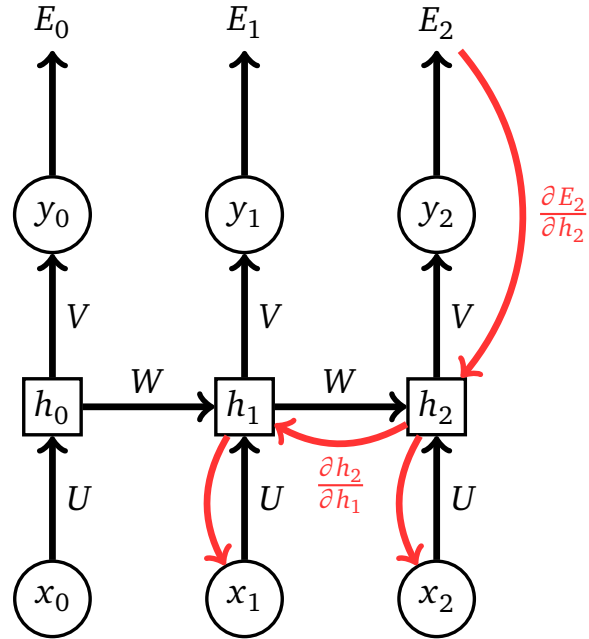
$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} (x_2^T + \frac{\partial h_2}{\partial h_1} (\dots$$



Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- L'erreur est ensuite propagée à U au premier pas

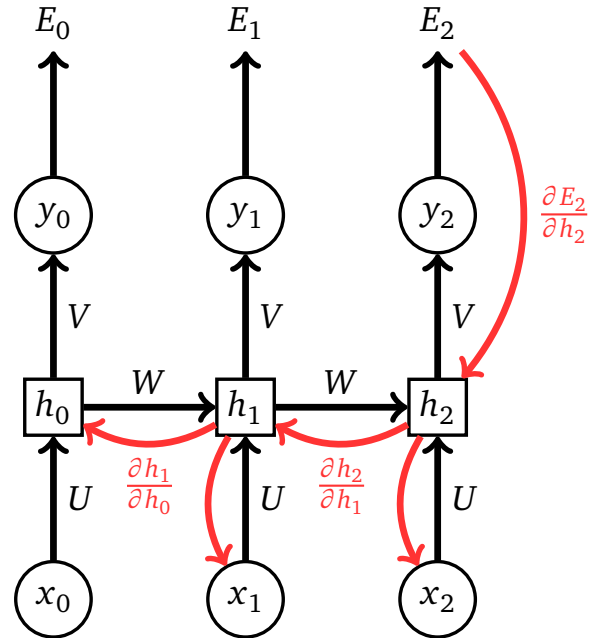
$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2}(x_2^T + \frac{\partial h_2}{\partial h_1}(x_1^T \dots$$



Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- et nous continuons avec h_0

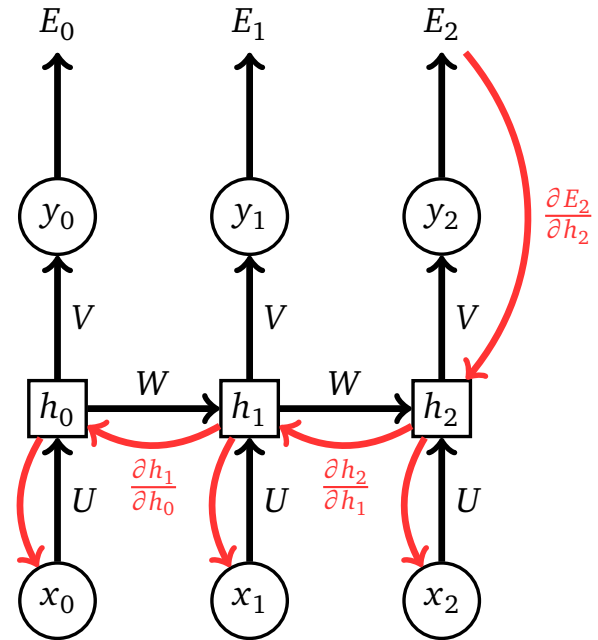
$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \left(x_2^T + \frac{\partial h_2}{\partial h_1} \left(x_1^T + \frac{\partial h_1}{\partial h_0} \dots \right) \right)$$



Inspiré par César Laurent, Réseaux Récurrents, École d'été IVADO, 2017

Rétropropagation à travers le temps

- Pour calculer $\frac{\partial E}{\partial U}$ nous calculons d'abord $\frac{\partial E_2}{\partial U}$
- et nous finissons avec la rétropropagation sur U au premier pas



$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \left(x_2^T + \frac{\partial h_2}{\partial h_1} \left(x_1^T + \frac{\partial h_1}{\partial h_0} x_0^T \right) \right)$$

Rétropropagation à travers le temps

- Les mêmes opérations sont effectuées afin de calculer $\frac{\partial E_1}{\partial U}$ et $\frac{\partial E_0}{\partial U}$ pour obtenir à la fin :

$$\frac{\partial E}{\partial U} = \sum_{t=0}^T \frac{\partial E_t}{\partial U}$$

- Nous devons également calculer les gradients sur les autres paramètres :

$$\frac{\partial E}{\partial V} = \sum_{t=0}^T \frac{\partial E_t}{\partial V} \qquad \frac{\partial E}{\partial W} = \sum_{t=0}^T \frac{\partial E_t}{\partial W}$$

Comme pour les autres modèles déjà vus, les librairies de Deep Learning (torch, tensorflow) font ces opérations automatiquement.

Limites des RNN

Les problèmes auxquels les RNNs font face sont les suivants :

- Dépendances à long distance
- Explosion du gradient
- Dissipation du gradient

Dépendances à longue distance

- Question : Quelle ville est la capitale de France ?

Texte pris de wikipedia et légèrement modifié.

Dépendances à longue distance

- Question : Quelle ville est la capitale de France ?
- Paris ... est la capitale de France.

Texte pris de wikipedia et légèrement modifié.

Dépendances à longue distance

- Question : Quelle ville est la capitale de France ?
- Paris ... est la capitale de France.
- Paris, qui se situe au cœur d'un vaste bassin sédimentaire aux sols fertiles et au climat tempéré, le bassin parisien, sur une boucle de la Seine, entre les confluents de celle-ci avec la Marne et l'Oise et qui est également le chef-lieu de la région Île-de-France et le centre de la métropole du Grand Paris, créée en 2016, est la capitale de France.

Texte pris de wikipedia et légèrement modifié.

Dépendances à longue distance

- Si on veut apprendre des dépendances à longue distance il faut être capable de propager le gradient à travers le réseau.
- Si notre séquence est longue, cette propagation peut être rapidement instable.
- Deux problèmes surviennent : explosion du gradient ou bien dissipation du gradient.

Explosion de gradient

- Si à chaque étape le gradient est amplifié, nous aurons une explosion du gradient à la fin.
- Une solution, mais qui n'est pas forcément optimale, est le **gradient clipping** (Pascanu et al. 2013)

$$g = \frac{\partial E}{\partial W}$$

if $\|g\| \geq \text{seuil}$ **then**

$$g \leftarrow \frac{\text{seuil}}{\|g\|} g$$

end if

Dissipation du gradient

- Si l'inverse est vraie, c'est à dire qu'à chaque étape le gradient s'atténue, nous aurons une dissipation du gradient.
- L'apprentissage de dépendances à longue distance n'est pas possible.
- Une solution simple, comme avant, n'est pas possible non plus.
- Il faudra utiliser des architectures particulières pour pouvoir améliorer la performance.

Dissipation/Explosion du gradient

- Est-ce un problème spécifique aux RNNs ?

Dissipation/Explosion du gradient

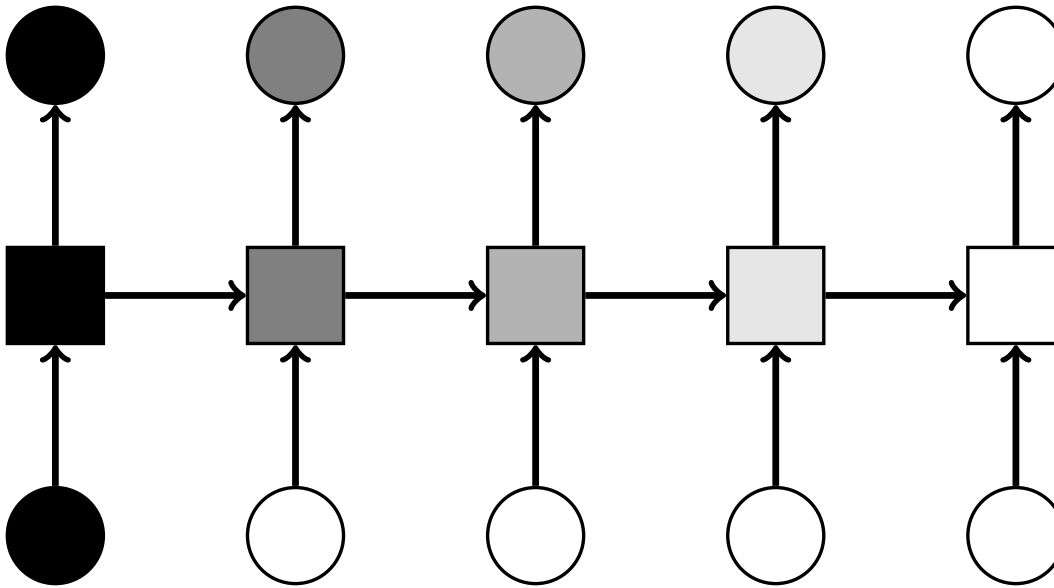
- Est-ce un problème spécifique aux RNNs ?
- Non ! Même les réseaux de convolution et les MLP peuvent souffrir de ce problème, surtout quand on a des couches très profondes.

Dissipation/Explosion du gradient

- Est-ce un problème spécifique aux RNNs ?
- Non ! Même les réseaux de convolution et les MLP peuvent souffrir de ce problème, surtout quand on a des couches très profondes.
- Une solution que beaucoup d'architectures adaptent est l'ajout de connections directes entre les couches distantes (cf connexions résiduelles cours précédent).

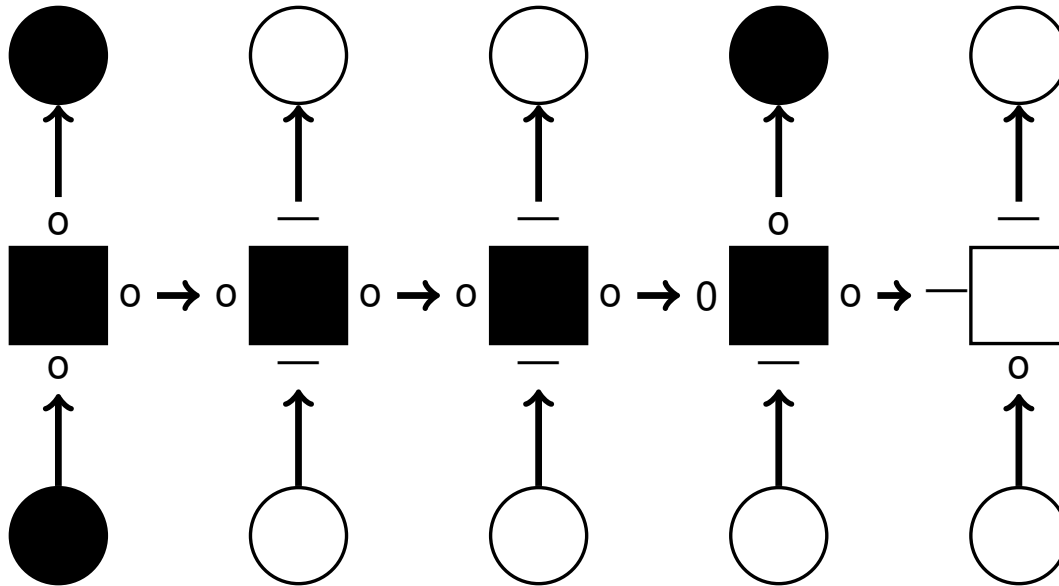
Mémoire dans le temps

Un RNN classique n'est pas capable de retenir de l'information pendant longtemps



Mémoire dans le temps

Solution : portes



LSTMs

Long Short Term Memory networks

Les LSTMs sont une forme de RNN qui sont capables d'apprendre des dépendances à longue distance, quelque chose que les RNNs classiques n'arrivent pas à faire très bien.

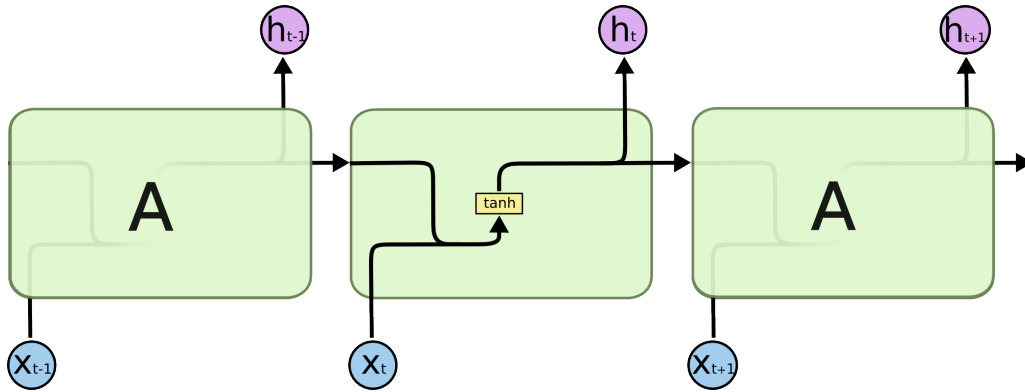


Figure 1 – Un RNN classique avec une couche

Images for the following are taken from

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

LSTMs

Long Short Term Memory networks

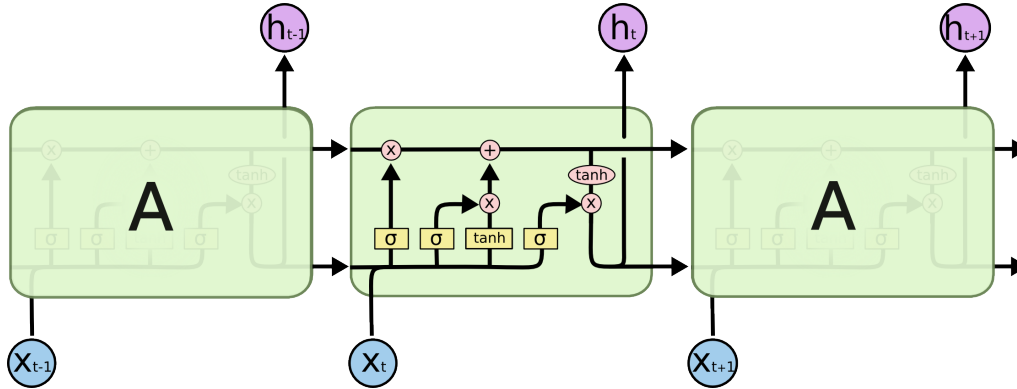
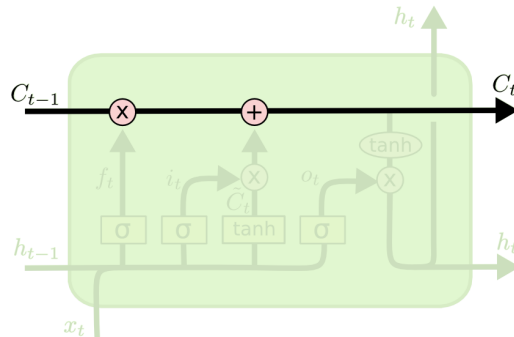


Figure 2 – Un LSTM contient 4 couches cachées qui interagissent

LSTMs

Long Short Term Memory networks

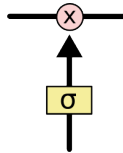
- L'idée principale derrière les LSTMs est le fait que nous pouvons avoir une sorte de mémoire (cell state) qui est partagée entre les différents étapes
- Cette mémoire est modifiée par quelques interactions linéaires à chaque étape via des « portes » (gates).
- Nous avons très peu d'opérations sur la cellule mémoire et l'information peut passer très facilement.



LSTMs

Long Short Term Memory networks

Les portes sont un moyen de permettre aux informations d'être sélectivement transmises. Elles sont composées par une couche de neurones utilisant la fonction sigmoïde et une opération de Hadamard.



La couche sigmoïde génère des valeurs entre 0 et 1. 0 signifie que rien ne passe et 1 signifie que tout passe.

Un LSTM contient 3 portes de contrôle.

Produit Hadamard

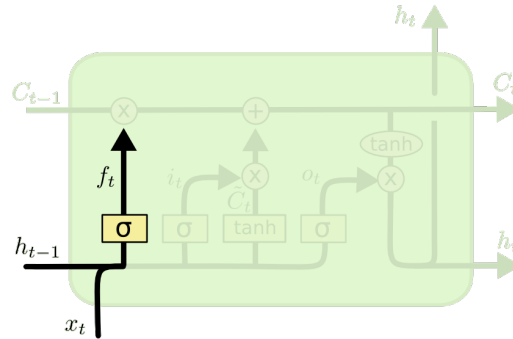
Petit rappel

Si nous avons deux matrices $A = [a_{ij}]_{n \times m}$ et $B = [b_{ij}]_{n \times m}$ le produit Hadamard est une troisième matrice $C = [c_{ij}]_{n \times m}$ tel que

$$c_{ij} = a_{ij} \cdot b_{ij} \quad \forall i \in [1 \dots n], j \in [1 \dots m]$$

LSTMs

Le réseau doit d'abord déterminer ce qu'il va «oublier» (forget gate) en prenant en compte le \mathbf{h}_{t-1} et le \mathbf{x}_t



$$f_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

- σ est la fonction sigmoïde bornée entre 0 et 1.
- Ce qu'on va « oublier » concerne la cellule mémoire.

LSTMs

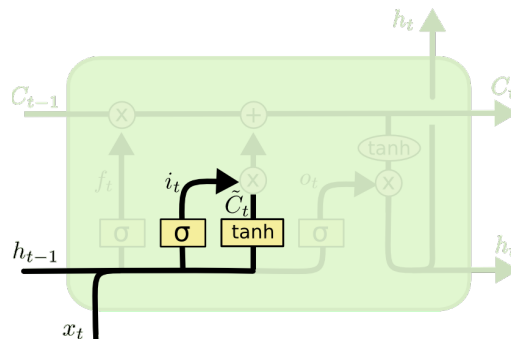
Ensuite le réseau doit déterminer les informations à retenir dans la mémoire (cell)

- D'abord une couche sigmoïde (input gate layer) décide des valeurs qui vont être misent à jour.

$$i_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

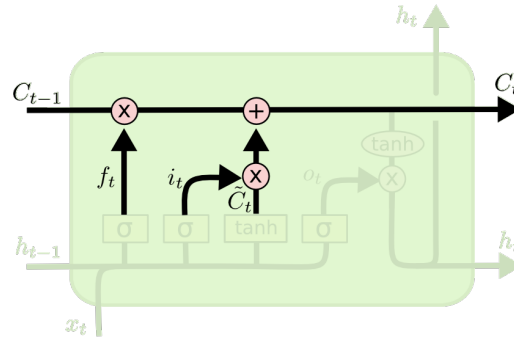
- Ensuite une couche \tanh créer un vecteur de valeurs candidats $\tilde{\mathbf{C}}_t$ qui peuvent être ajoutés à la mémoire.

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)$$



LSTMs

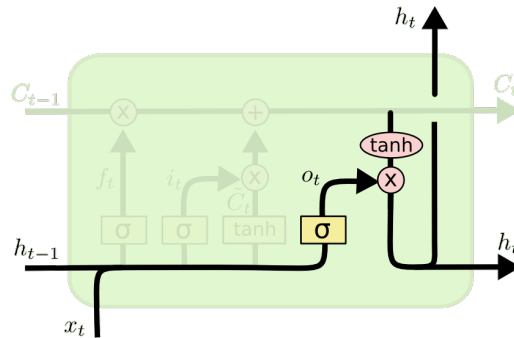
Nous pouvons donc maintenant mettre à jour la mémoire C_{t-1} pour obtenir C_t en combinant i_t et \tilde{C}_t .



$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t$$

LSTMs

- Finalement le réseau doit produire la sortie, selon C_t
- D'abord nous nous appliquons une sigmoïde pour déterminer quel morceaux de la mémoire nous allons sortir
- Ensuite nous passons C_t par une \tanh (pour obtenir des valeurs entre -1 et 1) et nous multiplions par la sortie du sigmoïde



$$o_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$h_t = o_t \otimes \tanh(C_t)$$

LSTMs bidirectionnnels

- Les LSTMs sont capables de représenter le contexte d'une entrée (un mot par exemple)

LSTMs bidirectionnnels

- Les LSTMs sont capables de représenter le contexte d'une entrée (un mot par exemple)
- Ce contexte est limité à gauche seulement, par contre !

LSTMs bidirectionnnels

- Les LSTMs sont capables de représenter le contexte d'une entrée (un mot par exemple)
- Ce contexte est limité à gauche seulement, par contre !
- Quid du contexte à droite ?

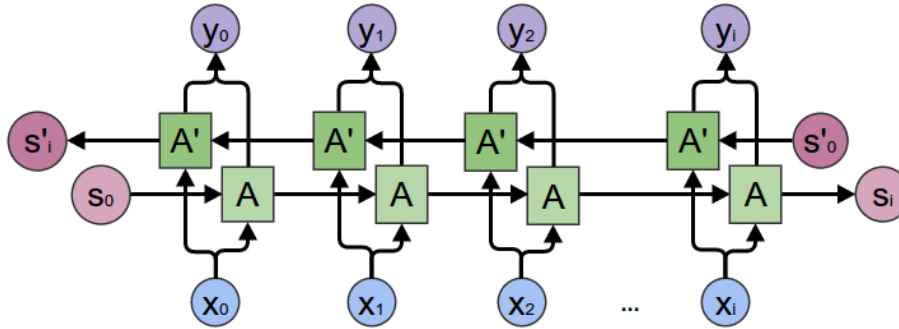
LSTMs bidirectionnnels

- Les LSTMs sont capables de représenter le contexte d'une entrée (un mot par exemple)
- Ce contexte est limité à gauche seulement, par contre !
- Quid du contexte à droite ?
- Ce n'est pas toujours possible d'avoir un contexte à droite (par exemple dans un scénario génératif), mais dans beaucoup de cas on peut y avoir accès

LSTMs bidirectionnnels

- Les LSTMs sont capables de représenter le contexte d'une entrée (un mot par exemple)
- Ce contexte est limité à gauche seulement, par contre !
- Quid du contexte à droite ?
- Ce n'est pas toujours possible d'avoir un contexte à droite (par exemple dans un scénario génératif), mais dans beaucoup de cas on peut y avoir accès
- Solution : LSTMs Bidirectionnnels

LSTMs bidirectionnnels



$$\vec{\mathbf{h}}_t = LSTM_{FW}(\vec{\mathbf{h}}_{t-1}, \mathbf{x}_t)$$

$$\overleftarrow{\mathbf{h}}_t = LSTM_{BW}(\overleftarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t)$$

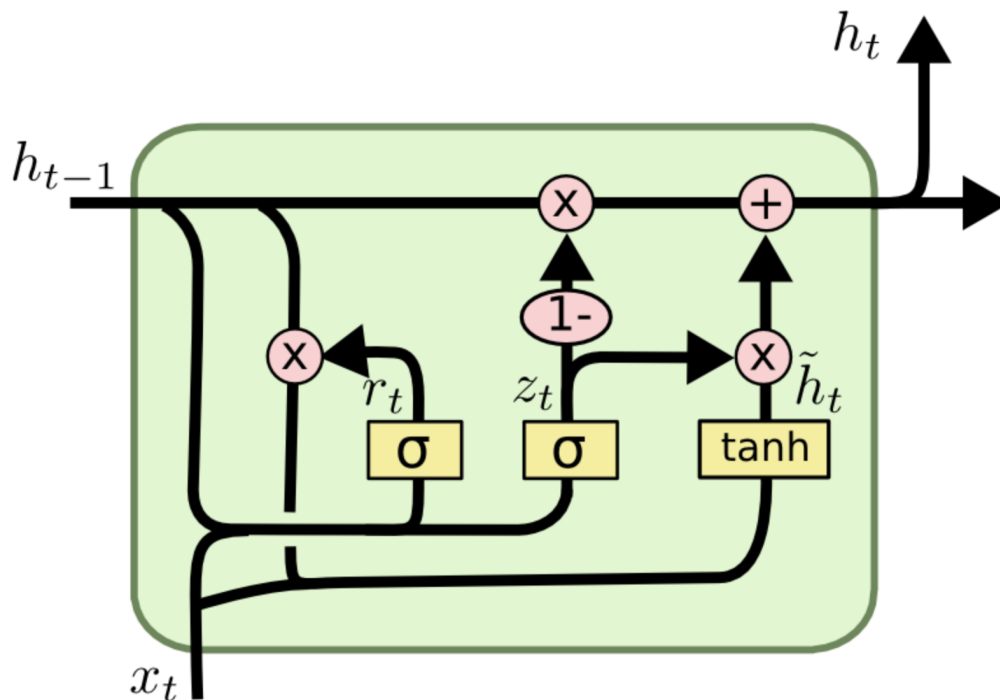
$$\mathbf{h}_t = [\vec{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$$

Bien évidemment, un RNN quelconque peut être bidirectionnel

Gated Recurrent Units (GRUs)

- Les GRUs est une autre architecture qui contient une mémoire permettant de capturer les dépendances à long distance
- La décision concernant les informations que le réseaux va « retenir » et les informations qu'il va « oublier » se fait via deux unités avec gates :
 1. une unité « update »
 2. une unité « reset »

Gated Recurrent Units (GRUs)



Gated Recurrent Units (GRUs)

Les équations pour le GRU sont les suivantes, u représente la gate pour le « update » et r pour « reset » :

$$u_t = \sigma(\mathbf{b}_u + \mathbf{U}_u \mathbf{x}_t + \mathbf{W}_u h_{t-1})$$

$$r_t = \sigma(\mathbf{b}_r + \mathbf{U}_r \mathbf{x}_t + \mathbf{W}_r h_{t-1})$$

$$h_t = u_t \otimes \tanh(\mathbf{U}_g \mathbf{x}_t + \mathbf{W}_g (r_t \otimes h_{t-1}) + \mathbf{b}_g) + (1 - u_t) \otimes h_{t-1}$$

Gated Recurrent Units (GRUs)

- Les GRUs sont souvent utilisés au lieu des LSTMs
- Ils n'ont pas une mémoire explicite
- Souvent on obtient des résultats similaires
- Ils sont plus rapide à calculer
- Nous pouvons avoir des GRU bidirectionnels

RNNs génératifs

Question : peut-on utiliser les réseaux récurrents afin de apprendre un modèle et ensuite l'utiliser pour produire des séquences ?

RNNs génératifs

Question : peut-on utiliser les réseaux récurrents afin de apprendre un modèle et ensuite l'utiliser pour produire des séquences ?

- Entrée : un texte (séquences de caractères UTF, par exemple, 1000 caractères)
- Sortie : Prédire le caractère suivant
- C'est une forme de modèle de langage
- Pour calculer l'erreur nous pouvons utiliser l'entropie croisée

RNNs génératifs

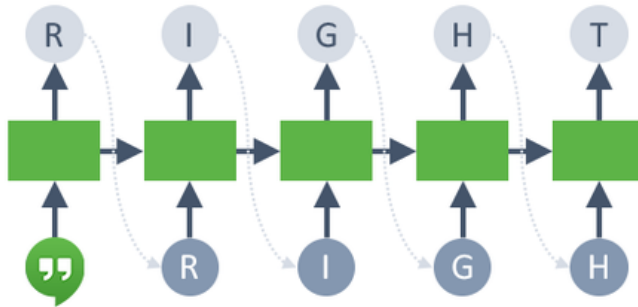


Image credit

RNNs génératifs

Exemple de génération entraîné sur des textes journalistiques

- Après initialisation :
usb9xkrd9ruaias\$dsajj'4lmjwyd61/se.lcn6jey0pbco40ab'65<8um324nqdhm<ufwt#y/w5bt'nm.zq«2rqm-a2'2mst#u315w&tNwdqNafqh*
- Après une époque d'entraînement :
to will an apple for a N shares of the practeded to working rudle and a dow listed that scill extressed holding a
- Après 76 époques d'entraînement :
president economic spokesman executive for securities was support to put used the sharelike the acquired who pla

c'est plus ou moins la stratégie de base des modèles de génération de texte actuels

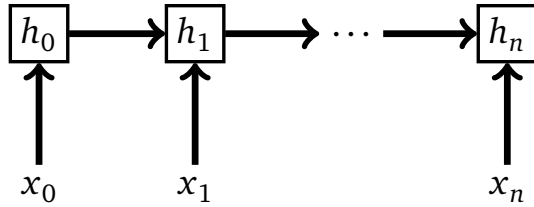
<http://deeplearningathome.com/2016/10/>

Text-generation-using-deep-recurrent-neural-networks.html

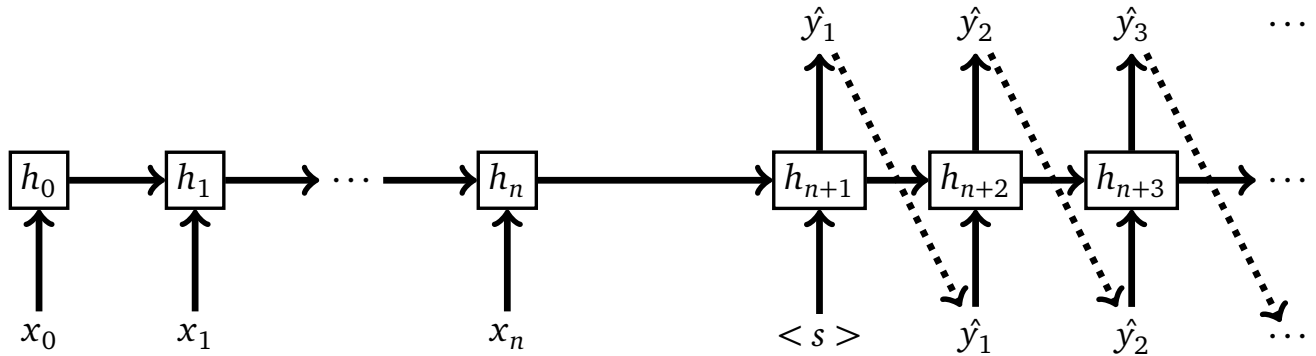
Exemple d'utilisation des RNNs génératifs

- Génération de musique
- Traduction automatique
- Production des résumés
- Génération des dialogues
- Chatbot
- Analyse syntaxique
- Génération du code (python, C, etc) à partir du langage naturel
- Génération d'écriture manuscrite
- Génération de parole

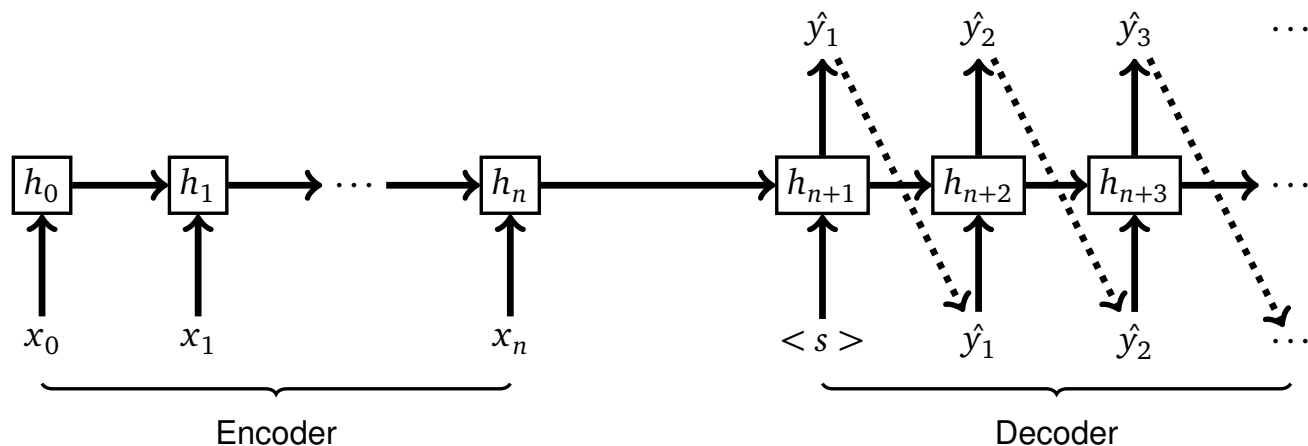
RNN génératif et le modèle Sequence to Sequence (seq2seq)



RNN génératif et le modèle Sequence to Sequence (seq2seq)



RNN génératif et le modèle Sequence to Sequence (seq2seq)



Entraînement de modèles RNNs génératifs

- **Entraînement** : étant donné $(x, y_{1:T})$ le modèle est entraîné en prenant toute la séquence d'un coup.
- Nous utilisons le Negative Log Likelihood (vraisemblance)

$$\mathcal{L}_{NLL}(\theta) = - \sum_t \log p(w_t = y_t | y_{1:t-1}, x; \theta)$$

- **Test** : nous avons un scénario de prédiction structuré où la structure est une séquence

$$\hat{y}_{1:T} = \operatorname{argmax}_{w_{1:T}} \sum_t \log p(w_t | w_{1:t-1}, x; \theta)$$

« Décodage »

- À chaque étape nous avons une distribution de probabilité sur tous les symboles (mots)
- Le problème est que nous souhaitons « décoder » ces probabilités afin de trouver la séquence la plus « intéressante » (score optimal) : mais problème combinatoire, recherche exhaustive trop coûteuse.
- → Stratégies de décodage approximatives
 - Approche « glouton » (greedy search)
 - Beam search
 - Top-K sampling

Recherche exhaustive

- Nous cherchons essentiellement de maximiser (pour un taille T) :

$$\begin{aligned} P(y, x) &= P(y_1|x)P(y_2|y_1, x) \cdots P(y_T|y_1, \cdots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \cdots, y_{t-1}, x) \end{aligned}$$

Recherche exhaustive

- Nous cherchons essentiellement de maximiser (pour un taille T) :

$$\begin{aligned} P(y, x) &= P(y_1|x)P(y_2|y_1, x) \cdots P(y_T|y_1, \cdots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \cdots, y_{t-1}, x) \end{aligned}$$

- Nous pouvons chercher exhaustivement toutes les séquences et choisir celle qui maximise le score globale

Recherche exhaustive

- Nous cherchons essentiellement de maximiser (pour un taille T) :

$$\begin{aligned} P(y, x) &= P(y_1|x)P(y_2|y_1, x) \cdots P(y_T|y_1, \cdots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \cdots, y_{t-1}, x) \end{aligned}$$

- Nous pouvons chercher exhaustivement toutes les séquences et choisir celle qui maximise le score globale
- **Problème** : Le coût est trop élevé : $O(V^T)$, V étant le vocabulaire et T la longueur de la séquence produite, c'est prohibitive.

Greedy Decoding

- Très simplement, à chaque étape nous prenons le symbol (mot) le plus probable.

$$\hat{y}_t = \operatorname{argmax}_w P(w|\hat{y}_1, \dots, \hat{y}_{t-1}, x)$$

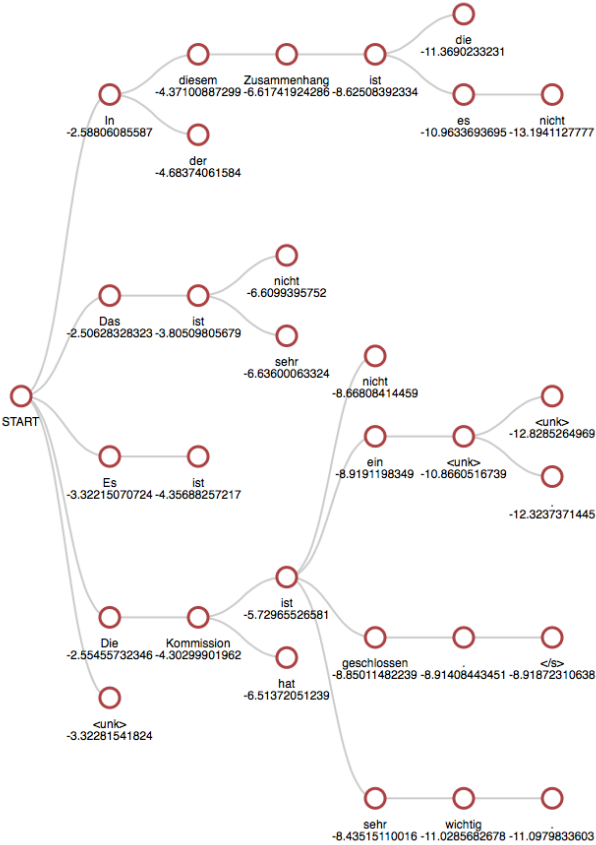
- **Problème** : aucun moyen de revenir en arrière ; les séquences ne font pas toujours sens.

Beam Search

L'idée de base :

- À chaque étape nous choisissons les k prédictions plus probables.
- Une arborescence commence à se développer
- À chaque étape nous n'explorons que k branches de l'arborescence
- Quand le symbole de fin arrive, nous choisissons la séquence qui maximise le score.

Beam Search



src OpenNMT

Top-k sampling

- on ne génère qu'une série, un élément à la fois
- au lieu de conserver k branches, on tire au hasard parmi les k étapes les plus probables
- donne plus de variété, ce qui est parfois souhaitable pour de la génération (ex texte)
- variante : tirer au hasard parmi les tokens de probabilité totale $> p$ fixé ("top-p sampling")

Quiz

Lequel de ces problèmes est un problème de séquence ou qui peut être ramené à une séquence ? (sans être nécessairement la seule méthode)

- saisie intuitive (par exemple complétion de caractères, prochain mot, T9)
- reconnaissance de sons
- nettoyage d'image (inpainting)
- génération de musique