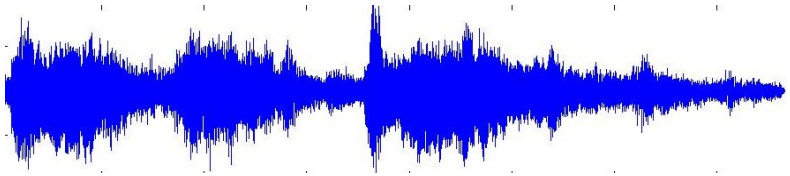
Réseaux de neurones récurrents

A. Carlier

2025

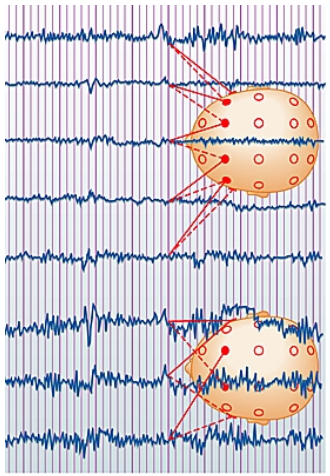
Plan du cours

Données séquentielles

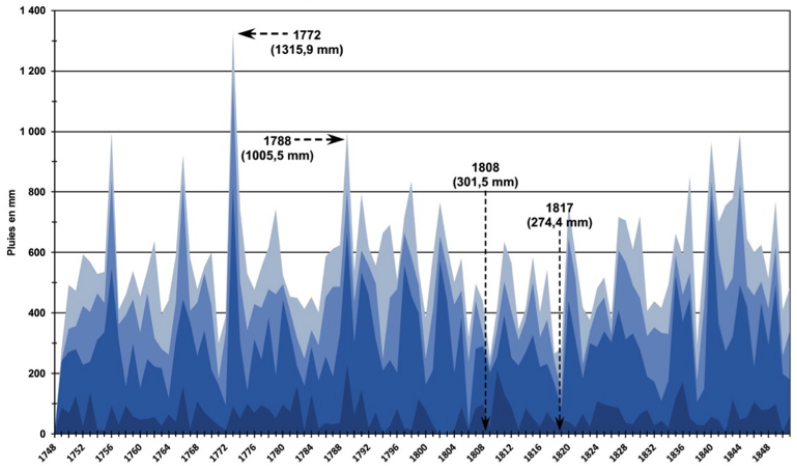


Audio

Vidéo



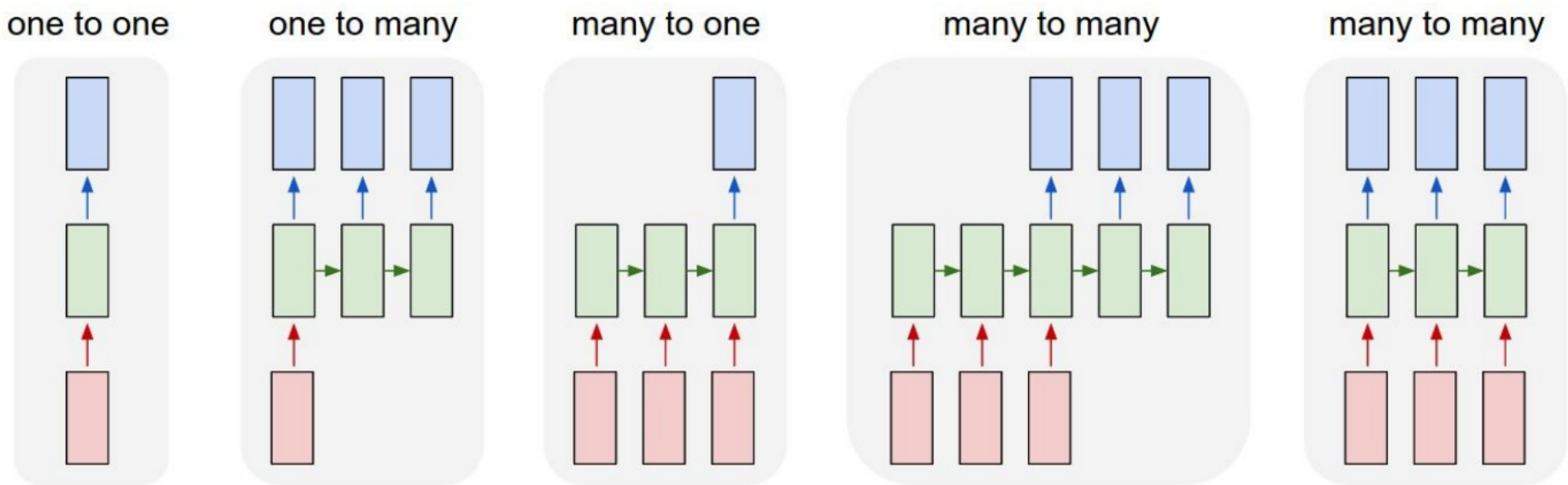
Données médicales



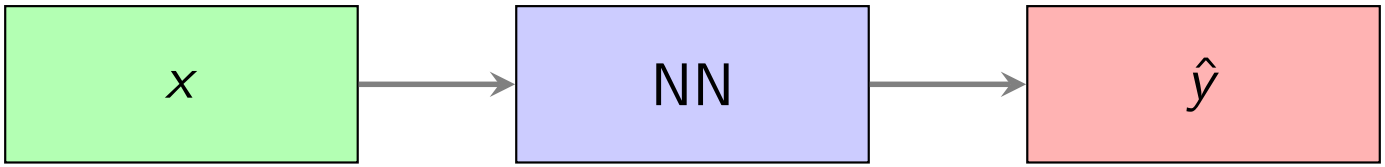
Données physiques

Mais aussi... le texte !

Problèmes séquentiels

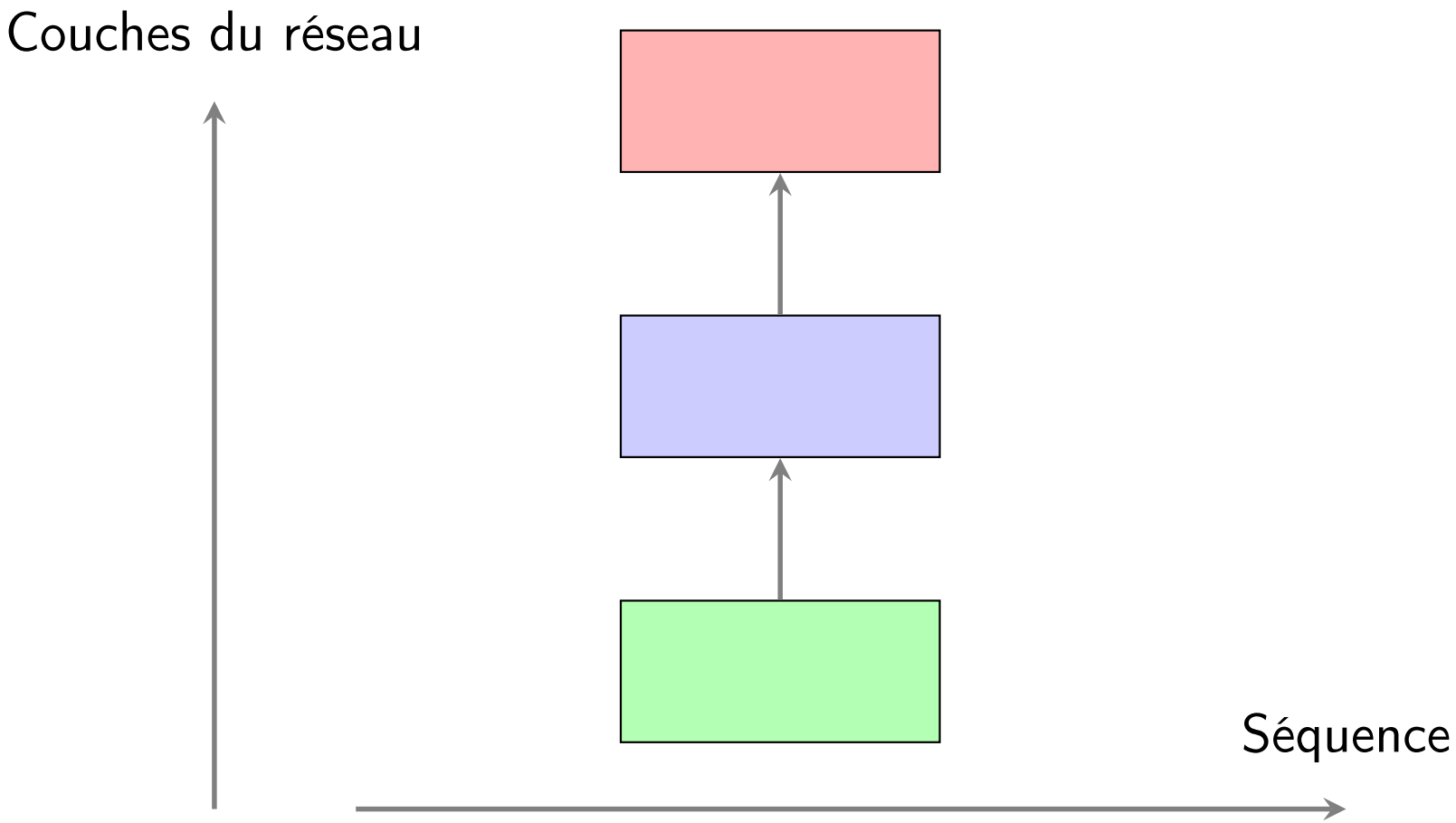


One to one



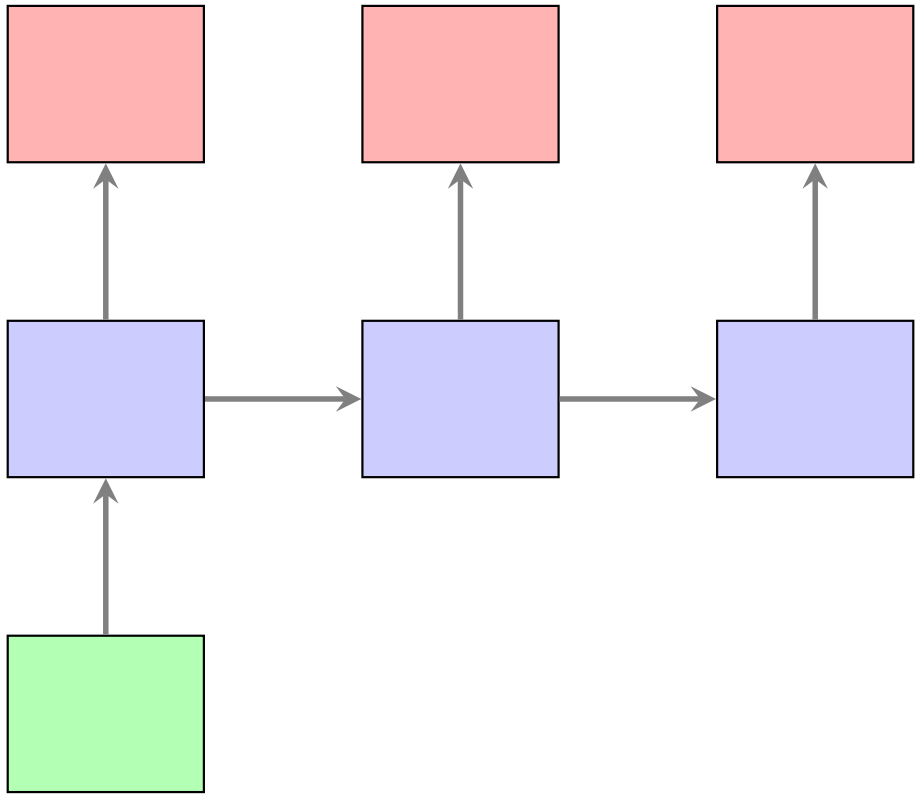
Exemples : Perceptrons multi-couches, Réseaux de neurones convolutifs, etc.

One to one

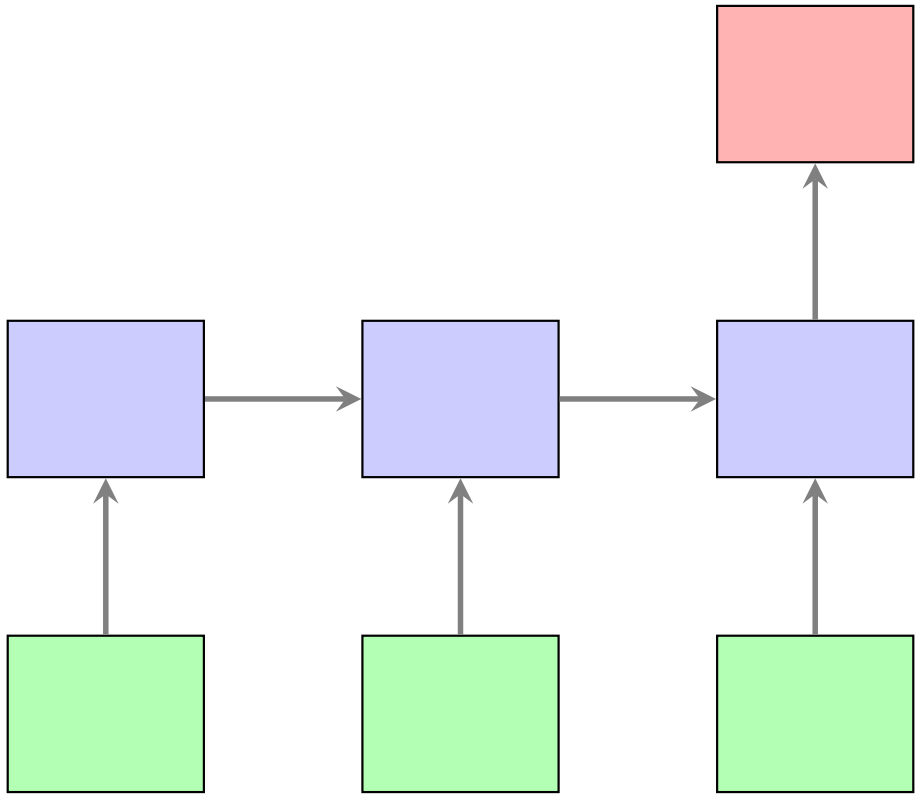


Une autre représentation : entrée en bas et sortie en haut.

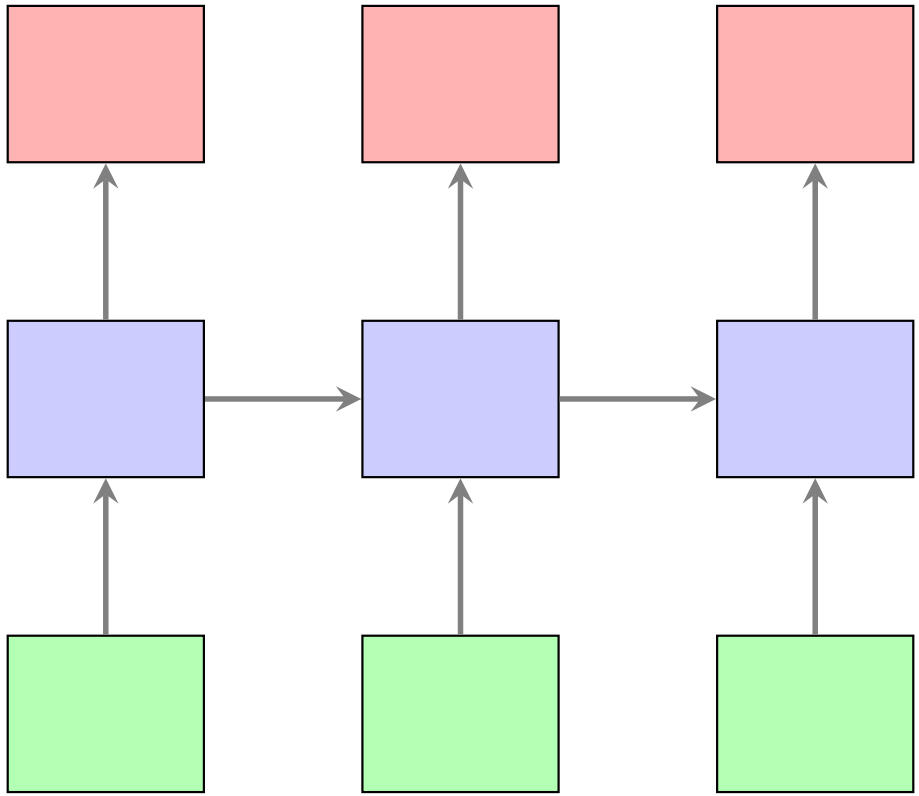
One to many



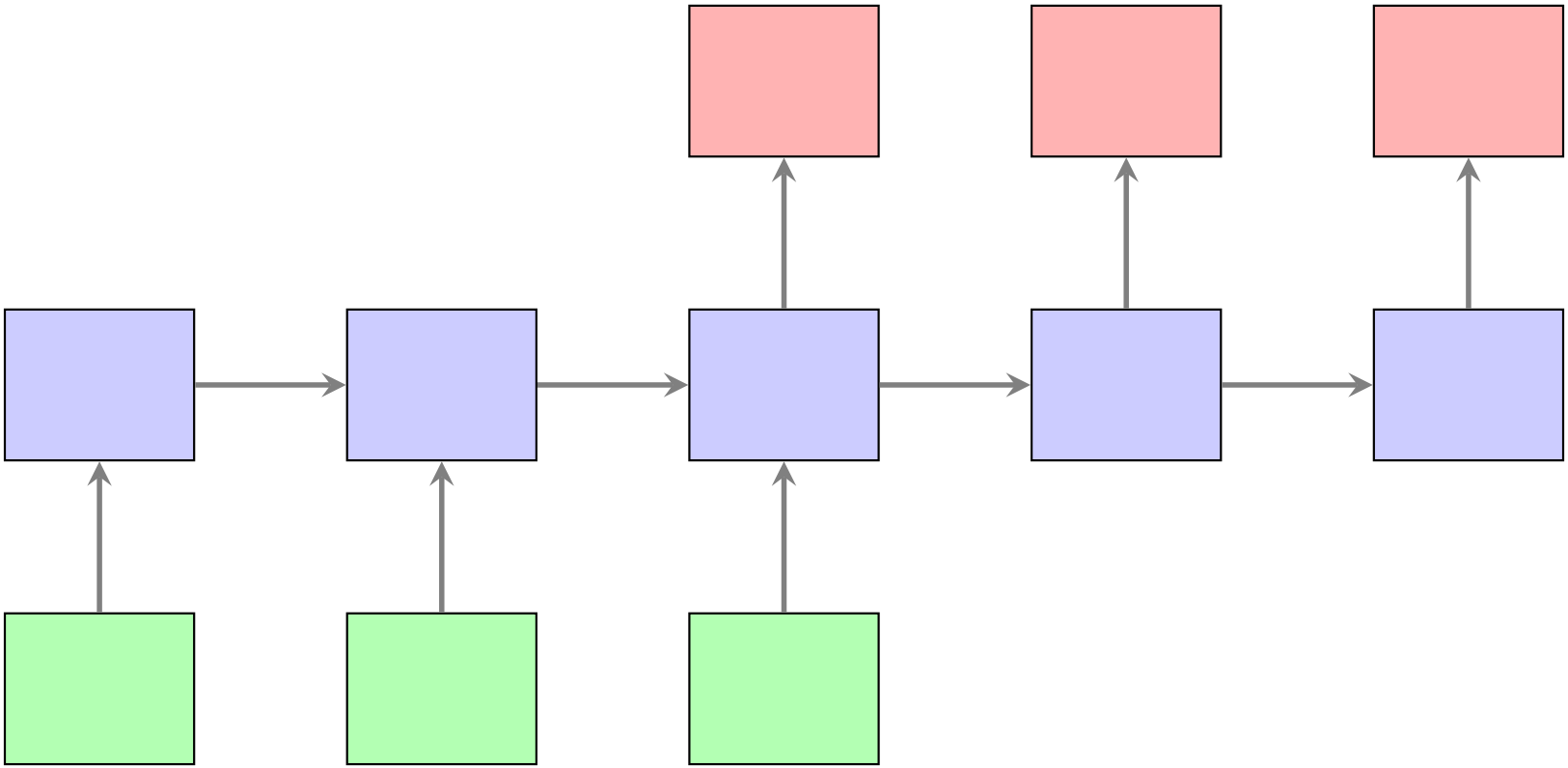
Many to one



Many to many



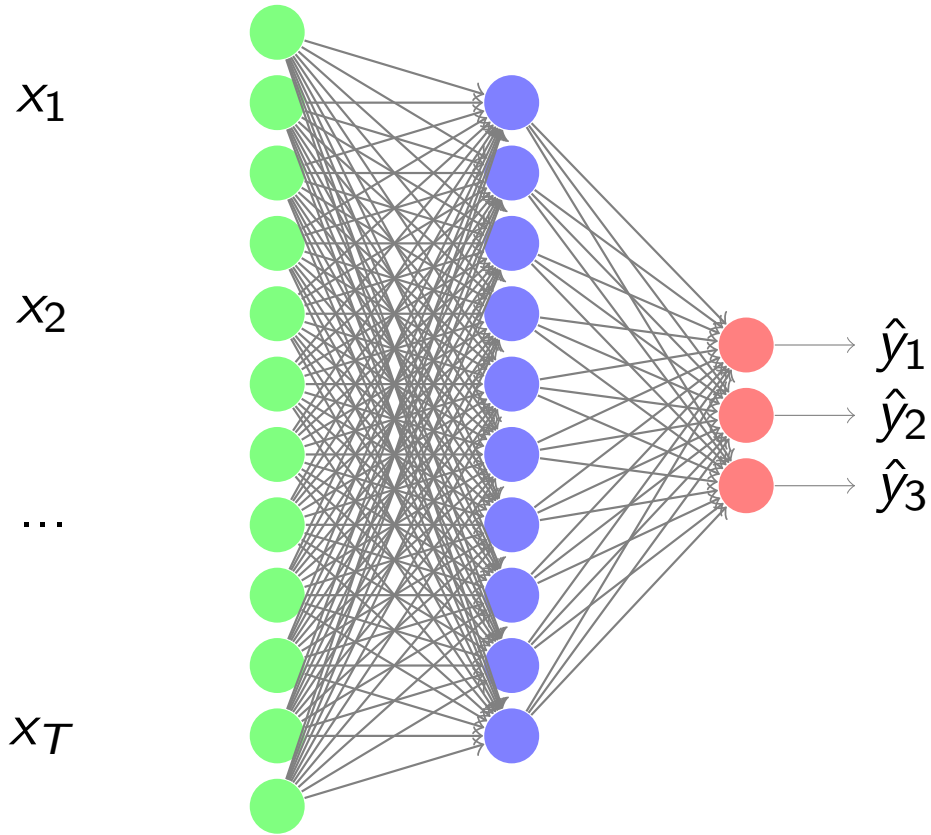
Many to many



Plan du cours

Données séquentielles

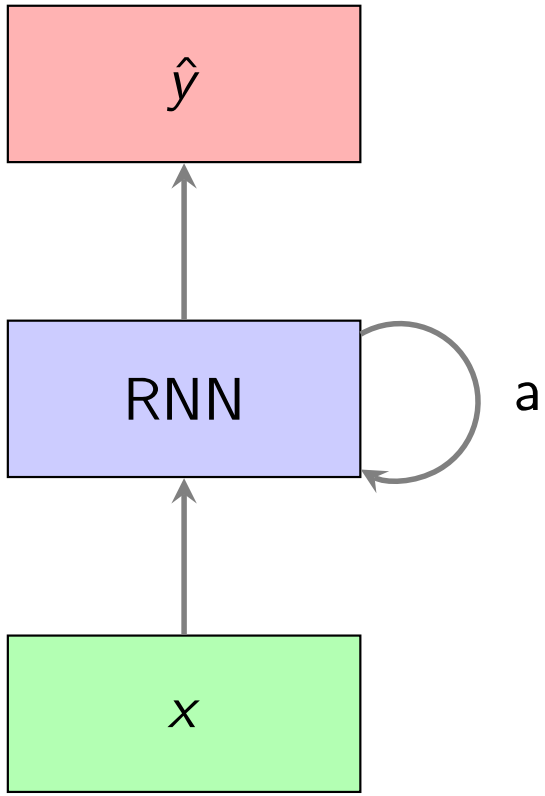
Un perceptron multicouches classique n'est pas adapté au traitement des données séquentielles :



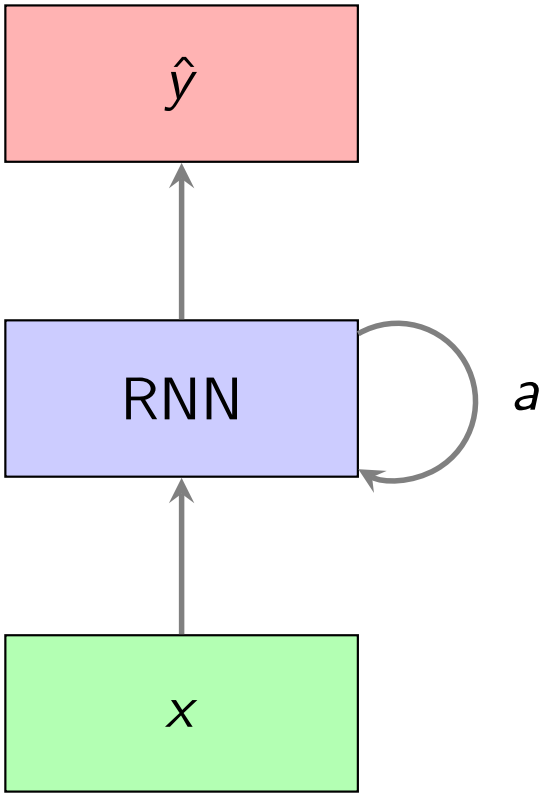
Les séquences sont de longueur variable et on doit apprendre un traitement différent pour chaque élément de la séquence !

Neurone récurrent

On introduit le concept du neurone récurrent :



Neurone récurrent

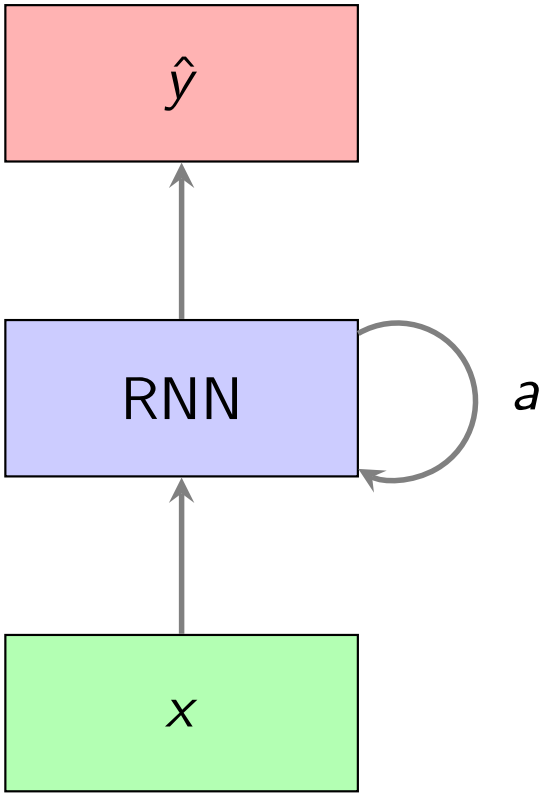


Soit une séquence d'entrées $(x^{<i>})_{i=1..T_x}$

$$a^{<t>} = f_W(a^{<t-1>}, x^{<t>})$$

La même fonction f et les mêmes paramètres W sont utilisés à chaque pas de temps.

Neurone récurrent classique

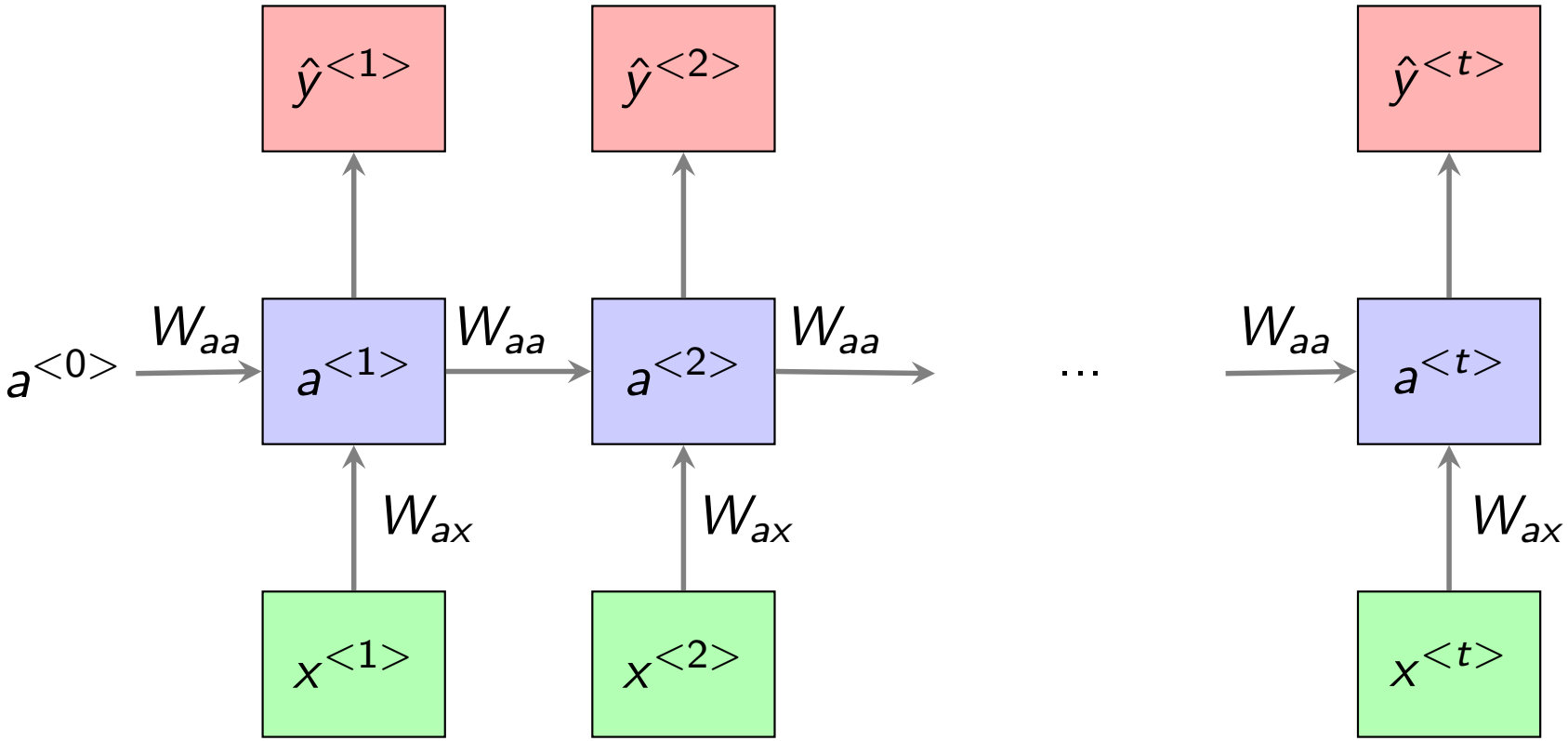


Soit une séquence d'entrées $(x^{<i>})_{i=1..T_x}$

$$a^{<0>} = \vec{0}$$
$$a^{<t>} = \tanh(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g(W_{ya}h^{<t>} + b_y)$$

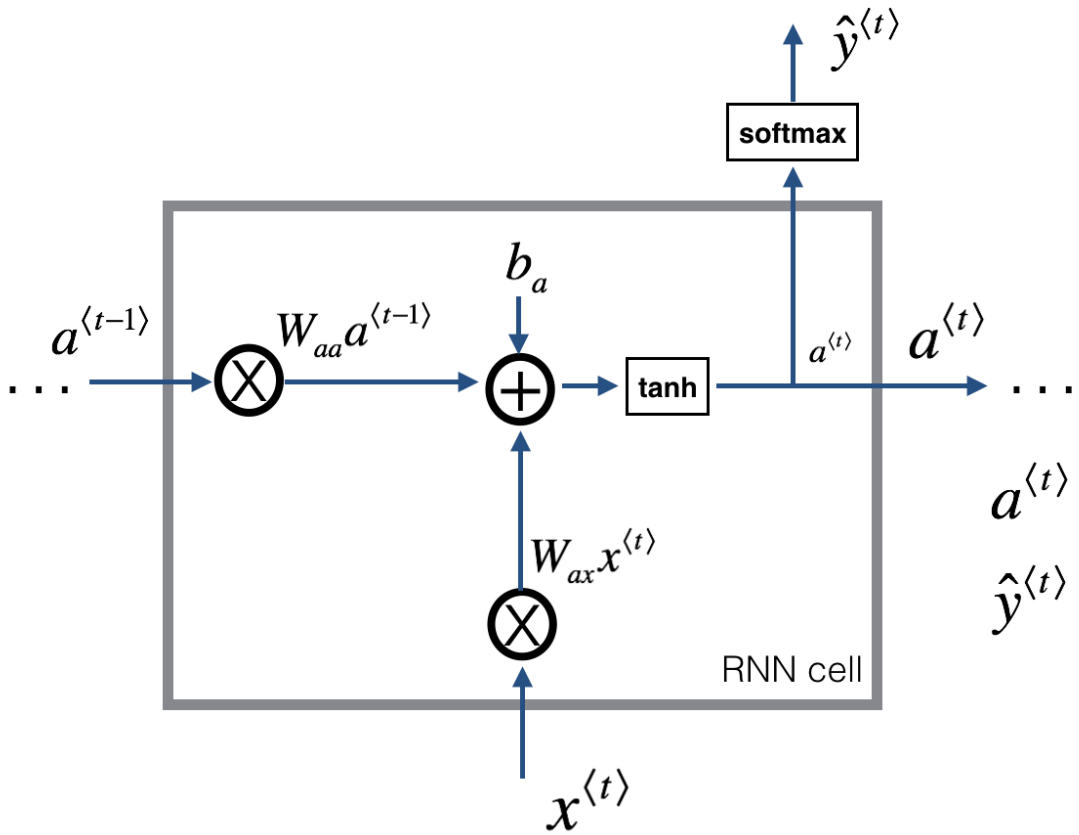
La fonction g représente la fonction d'activation de la couche de sortie, qui dépend du problème considéré (typiquement `sigmoid`, `softmax`, ou `linear`).

Neurone récurrent - représentation développée



Les mêmes paramètres sont réutilisés pour tous les éléments de la séquence !

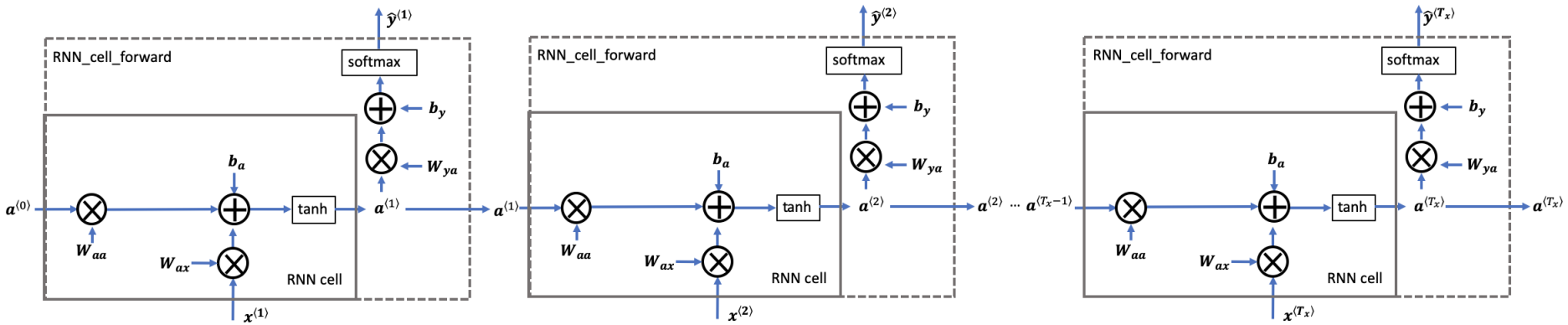
Neurone récurrent : passe *forward*



$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

Neurone récurrent : passe *forward*

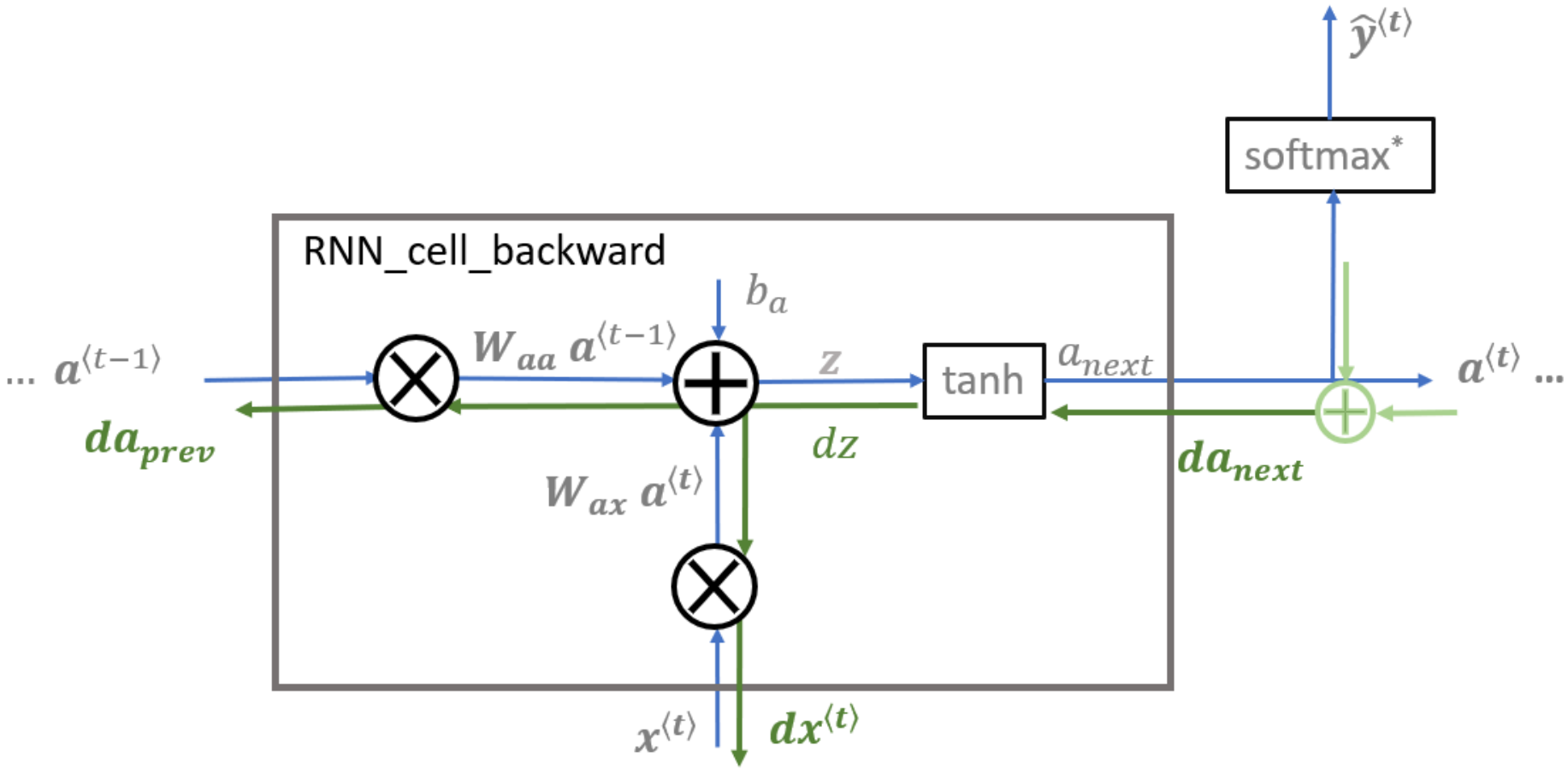


Les prédictions sont réalisées séquentiellement. On ne peut pas paralléliser simplement les calculs dans un réseau récurrent, ce qui les rend très peu efficaces (en temps).

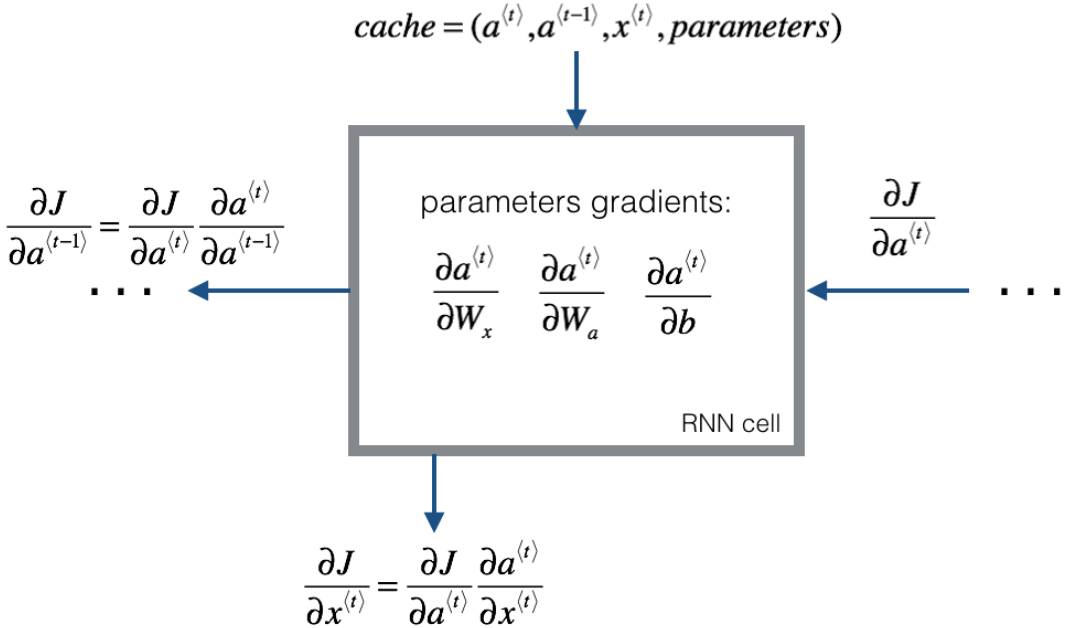
En revanche, puisque l'on réutilise les mêmes paramètres pour chaque pas de temps, ils sont très économes en paramètres et donc moins sujets au sur-apprentissage.

Neurone récurrent : passe *backward*

Rétropropagation temporelle (*backpropagation through time*)



Neurone récurrent : passe *backward*



$$a^{\langle t \rangle} = \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{ax}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) x^{\langle t \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{aa}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) a^{\langle t-1 \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial b} = \sum_{batch} (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial x^{\langle t \rangle}} = W_{ax}{}^T . (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial a^{\langle t-1 \rangle}} = W_{aa}{}^T . (1 - \tanh(W_{ax} x^{\langle t-1 \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

Neurone récurrent : passe *backward*

Le gradient de la perte par rapport aux paramètres inclut le terme suivant :

$$\prod_{i=1}^{T-1} \frac{\partial a^{<i+1>}}{\partial a^{<i>}}$$

Ce terme peut causer des problèmes d'évanescence ou d'explosion des gradients... temporelles !

Gradient clipping

Pour prévenir l'explosion du gradient, on peut utiliser la technique du *gradient clipping* :

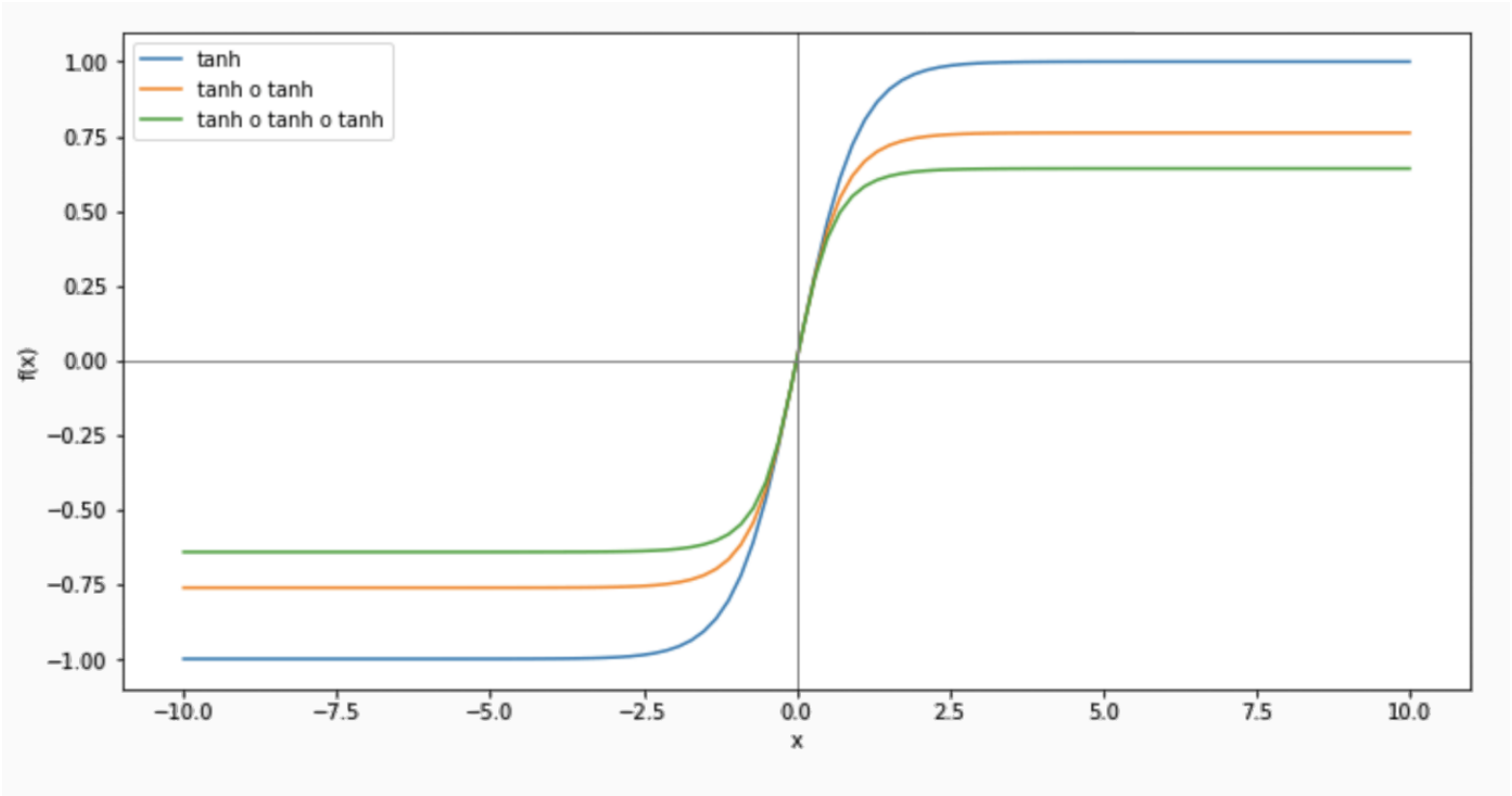
Si $||g|| > c$, alors

$$g \leftarrow c\frac{g}{||g||}$$

Par exemple, si l'on veut instancier un optimiseur en `Keras` qui utilise cette technique, on peut utiliser l'argument *clipnorm* ainsi :

```
opt = SGD(lr=0.01, momentum=0.9, clipnorm=1.0)
```

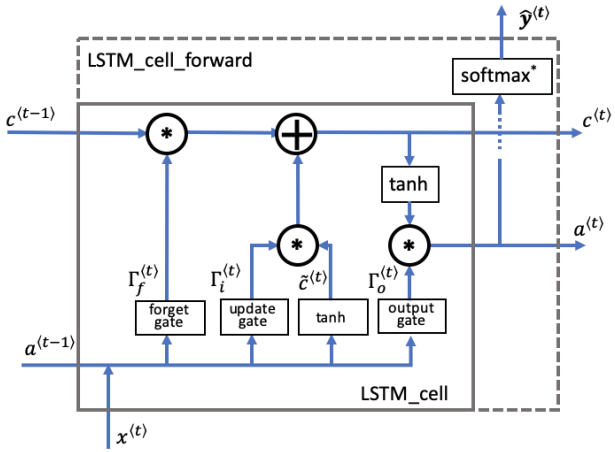
Dépendances à long terme



L'usage de la fonction `tanh` comme fonction d'activation pose des problèmes pour les longues séquences :
tanh(tanh(... x)...) tend vers 0 !

Evanescence du gradient

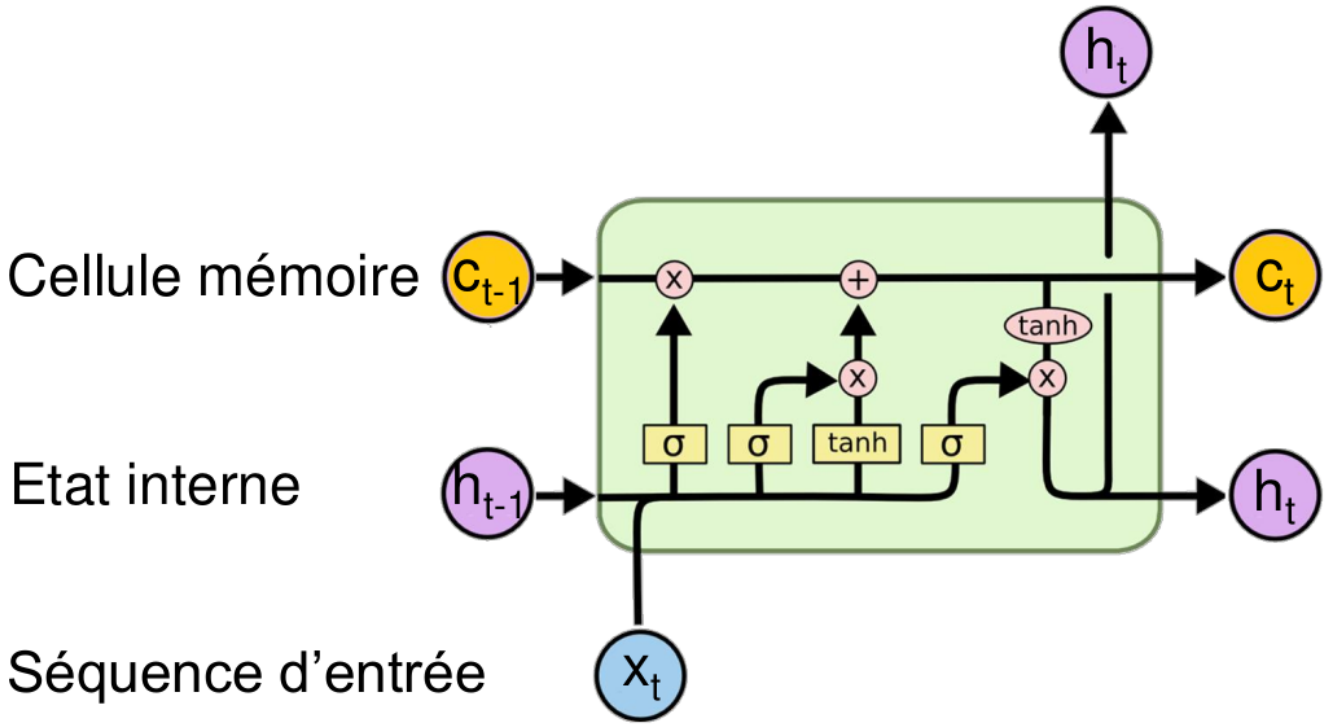
En 1997, Hochreiter et Schmidhuber proposent une nouvelle cellule récurrente qui rend possible l'apprentissage de dépendance à long terme, et limite les problèmes d'evanescence du gradient : les `LSTM`.



Plan du cours

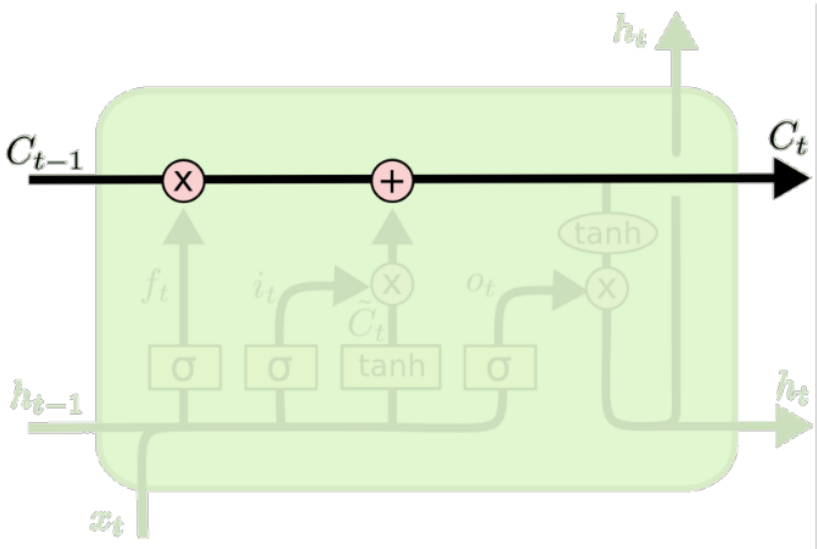
Long Short-Term Memory

Réduction du problème de dissipation avec **un mécanisme de gates** et une **cellule mémoire**.



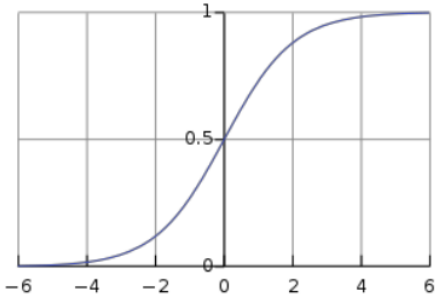
Long Short-Term Memory

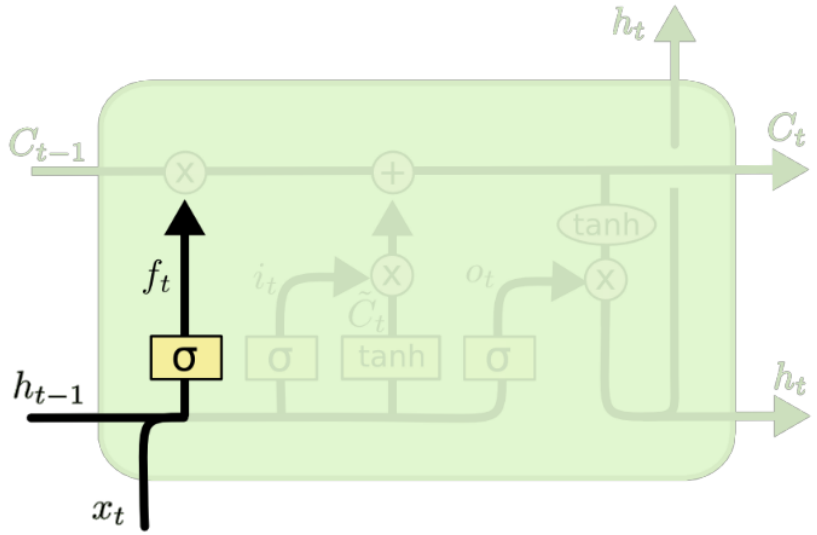
- Le point clé de la LSTM est sa cellule mémoire.
  - Très peu d'opérations dessus.
  - L'information peut passer très facilement.



Long Short-Term Memory

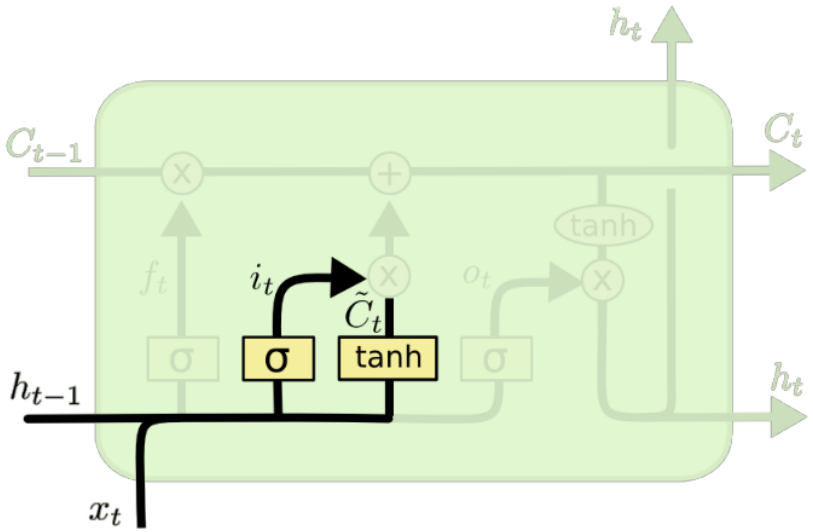
- Calcul de la forget gate à partir de x_t et h_{t-1}.
  - $f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$
  - σ est la fonction sigmoïde (bornée entre 0 et 1).





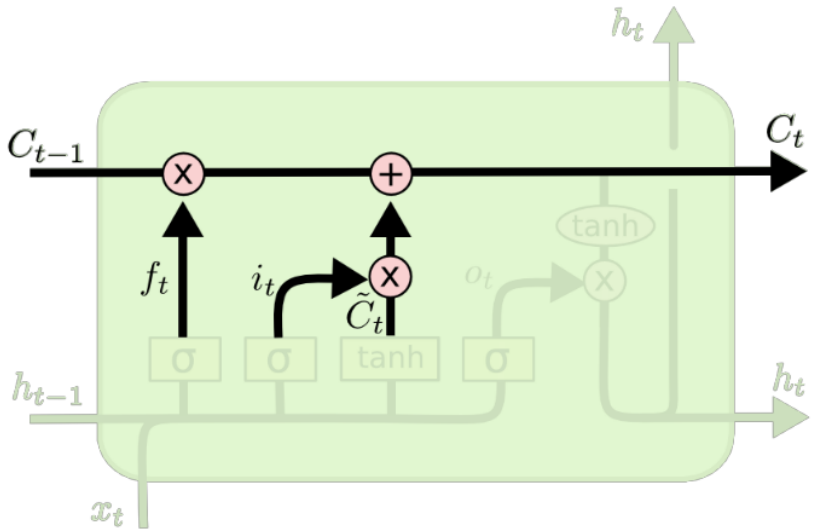
Long Short-Term Memory

- Calcul de l'input gate à partir de x_t et h_{t-1}.
  - $i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$
  - L'input gate contrôle ce qui entre dans la cellule mémoire.
- Calcul de ce qui va être ajouté à la cellule mémoire.
  - $g_t = \tanh(U_g x_t + W_g h_{t-1} + b_g)$



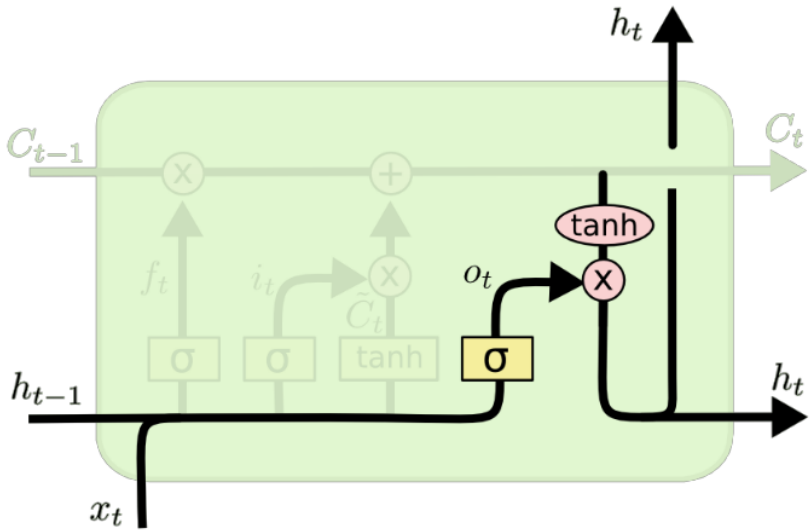
Long Short-Term Memory

- Mise à jour de la cellule mémoire à l'aide de l'input et de la forget gate.
  - $c_t = i_t \odot g_t + f_t \odot c_{t-1}$
  - $\odot$ = multiplication terme à terme.
  - L'input gate permet d'ajouter de l'information dans la cellule, et la forget gate permet d'oublier l'information déjà présente dans la cellule.



Long Short-Term Memory

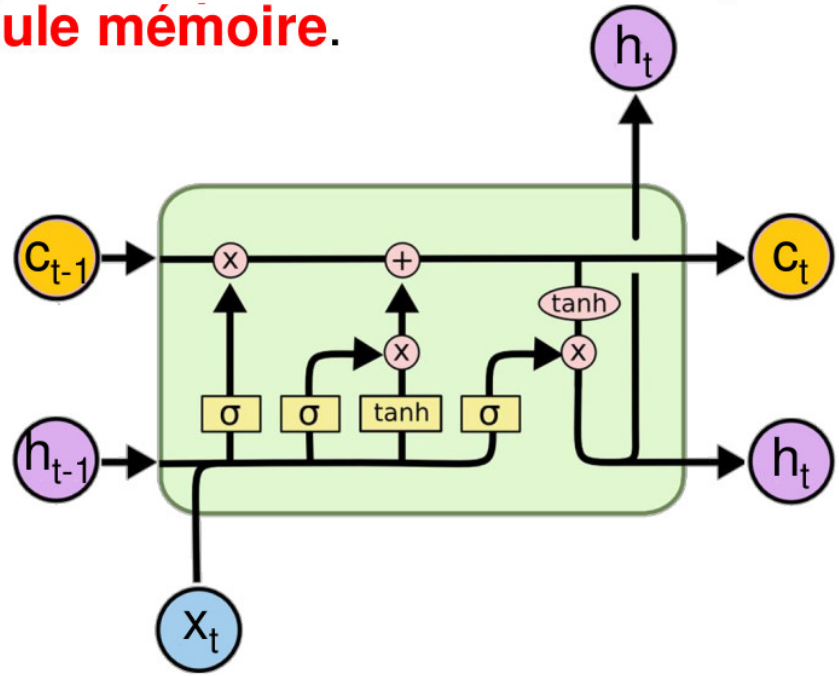
- Calcul de l'output gate à partir de x_t et h_{t-1}.
  - $o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$
  - L'output gate contrôle ce qui sort de la cellule mémoire.
- Calcul de l'état interne.
  - $h_t = o_t \odot \tanh(c_t)$



Long Short-Term Memory

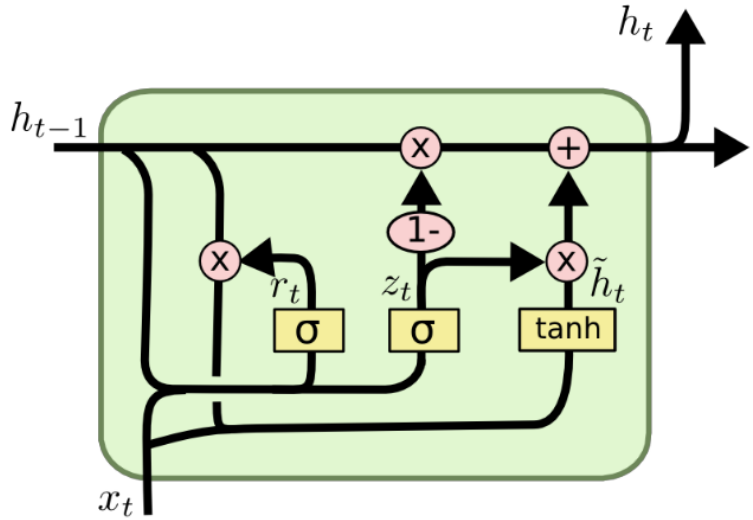
Réduction du problème de dissipation avec **un mécanisme de gates** et une **cellule mémoire**.

$$
\begin{aligned}
i_t &= \sigma(U_i x_t + W_i h_{t-1} + b_i) \\
f_t &= \sigma(U_f x_t + W_f h_{t-1} + b_f) \\
o_t &= \sigma(U_o x_t + W_o h_{t-1} + b_o) \\
g_t &= \tanh(U_g x_t + W_g h_{t-1} + b_g) \\
c_t &= i_t \odot g_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$



Gated Recurrent Units

- Une variante populaire de la LSTM.
  - Pas de cellule mémoire explicite.
  - Input et forget gates combinées.
- En pratique, performances égales à la LSTM.
  - Plus rapide à calculer.

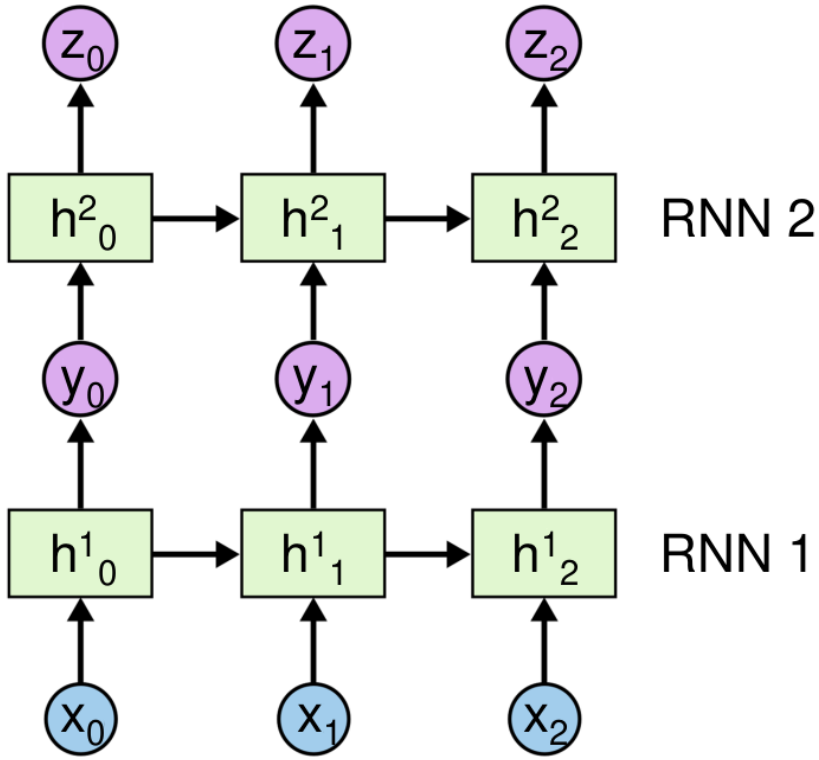


$$z_t = \sigma(U_z x_t + W_z h_{t-1} + b_z)$$
$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$
$$g_t = \tanh(U_g x_t + W_g(r_t \odot h_{t-1}) + b_g)$$
$$h_t = z_t \odot g_t + (1 - z_t) \odot h_{t-1}$$

Plan du cours

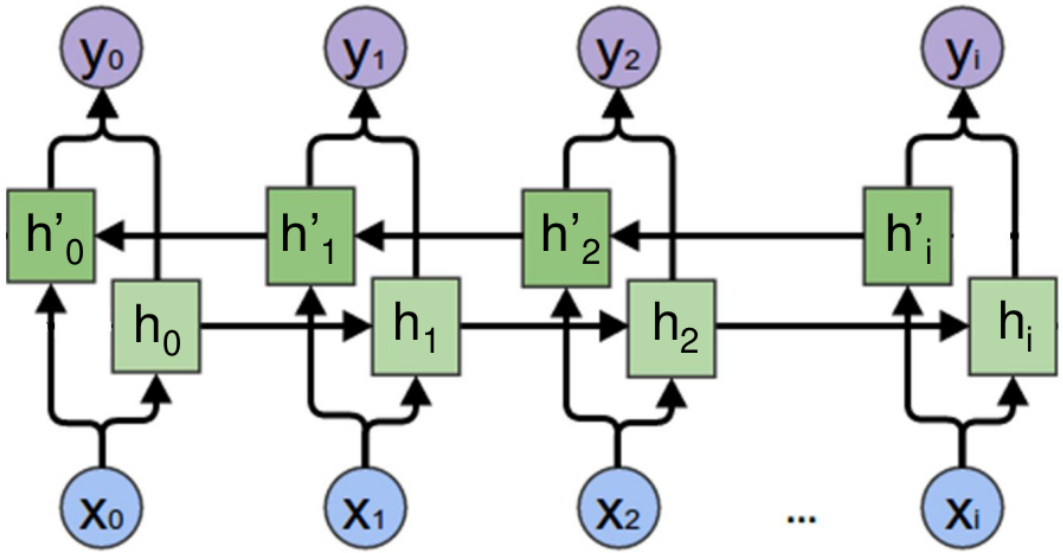
Réseaux récurrents

- Pour créer des RNNs profonds, on peut empiler des couches
  - Chaque RNN peut être un RNN simple, une LSTM, GRU,…
- La séquence de sortie de la première couche est la séquence d'entrée de la seconde couche, etc.



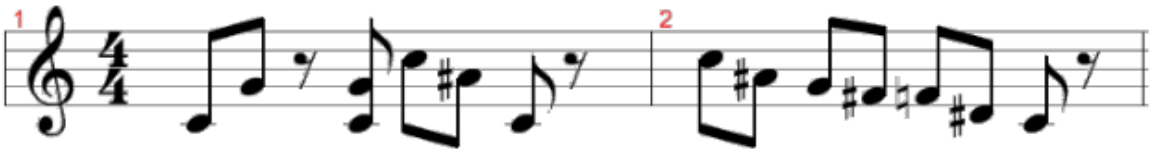
Réseaux bidirectionnels

- Ajout d'un second RNN qui lit la séquence à l'envers.
- Permet d'avoir de l'information sur ce qui se passe avant **et** après.
- Les 2 RNNs sont différents (paramètres différents)!

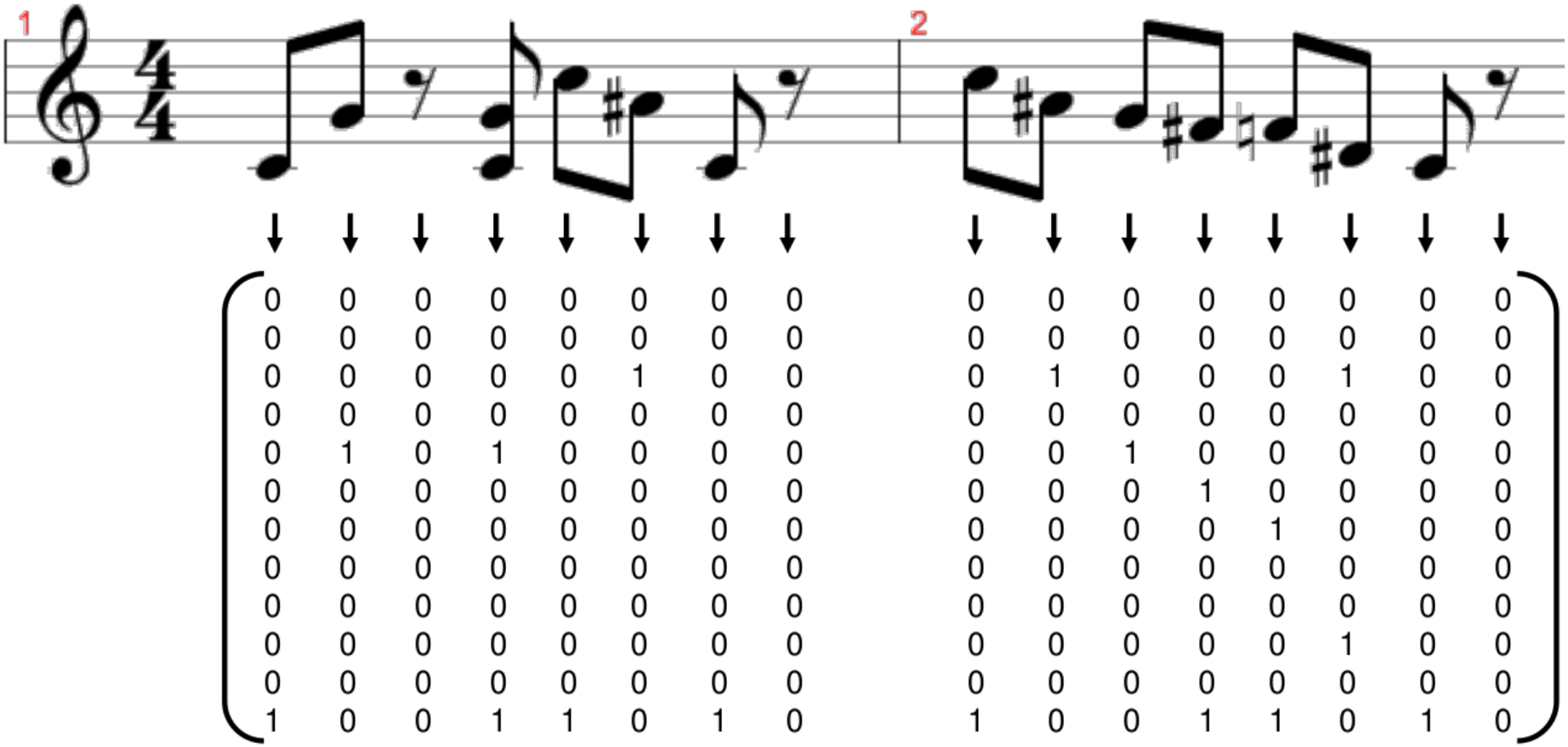


Exemple : classification de séquence

But: Reconnaitre le genre musical à partir de la partition

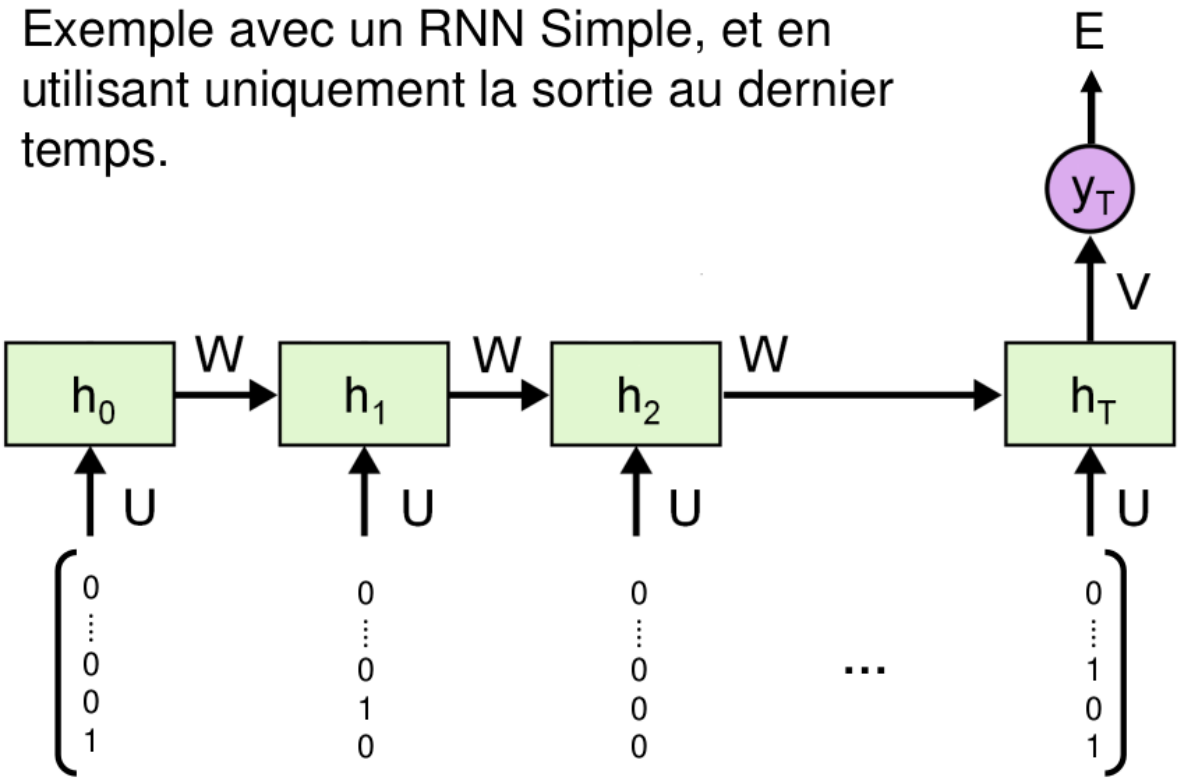
- Données: 500 partitions de musique (durée variable):
  - x: 
- 3 Classes: Blues, Rock, Classique
  - c: (1, 0, 0) ou (0, 1, 0) ou (0, 0, 1)

Exemple : classification de séquence



Exemple : classification de séquence

Exemple avec un RNN Simple, et en utilisant uniquement la sortie au dernier temps.



$$E = \text{cross\_entropy}(y_T, c)$$

$$y_T = \text{Softmax}(Vh_T + d)$$

$$h_t = \tanh(Ux_t + Wh_{t-1} + b)$$

$$x$$

Exemple : classification d'événements sonores

- But: Détecter et classifier des événements sonores dans des enregistrements.
  - Début et fin.
  - Type d'événement (voiture qui passe, oiseau qui chante,…).
  - Polyphonie (plusieurs événements peuvent avoir lieu en même temps).

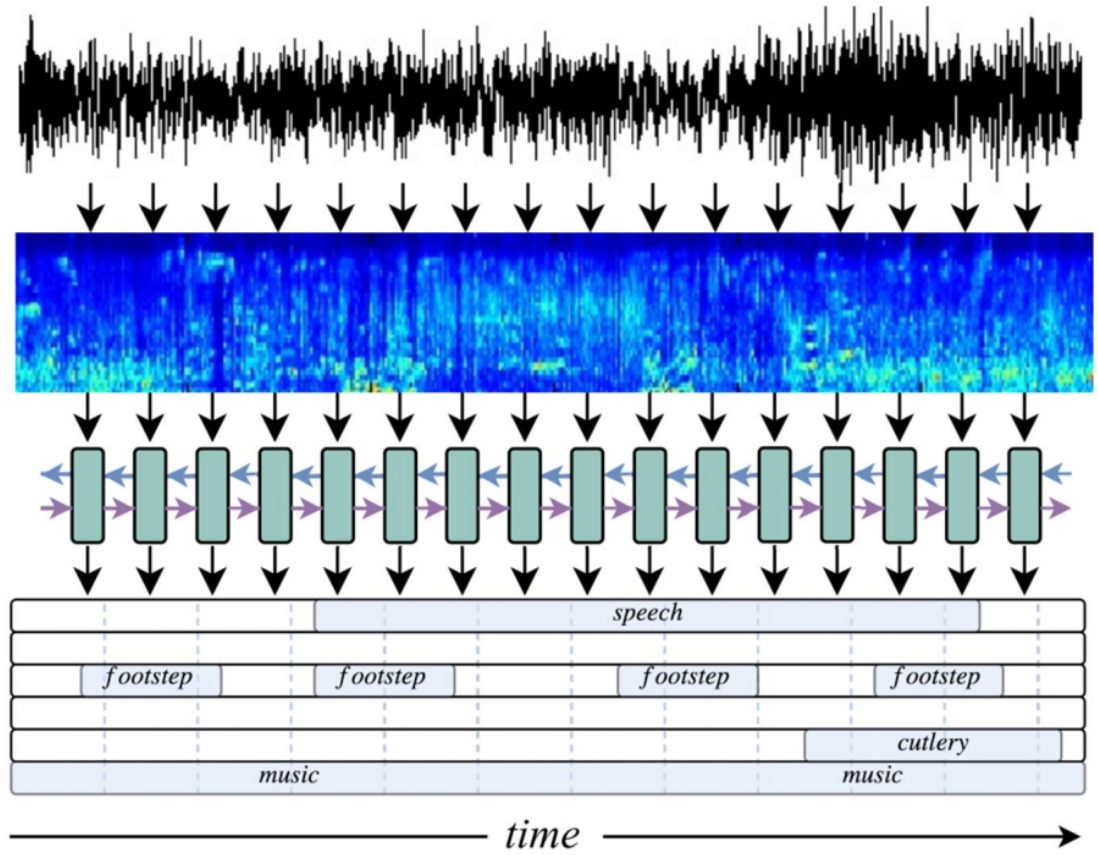
Exemple : classification d'événements sonores

Enregistrement brut

Spectrogramme

LSTMs bidirectionnels (4)

Cibles pour chaque classes



Plan du cours

Données séquentielles

Une phrase est une séquence :

- de mots :

J ’ apprécie les fruits au sirop .

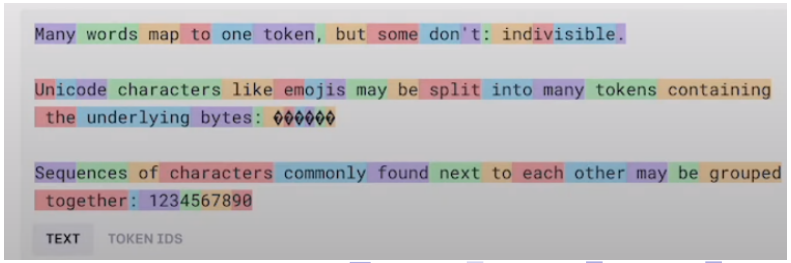
- de syllabes :

J ’ ap pré cie les fruits au si rop

- ou de caractères :

J ’ a p p r é c i e l ...

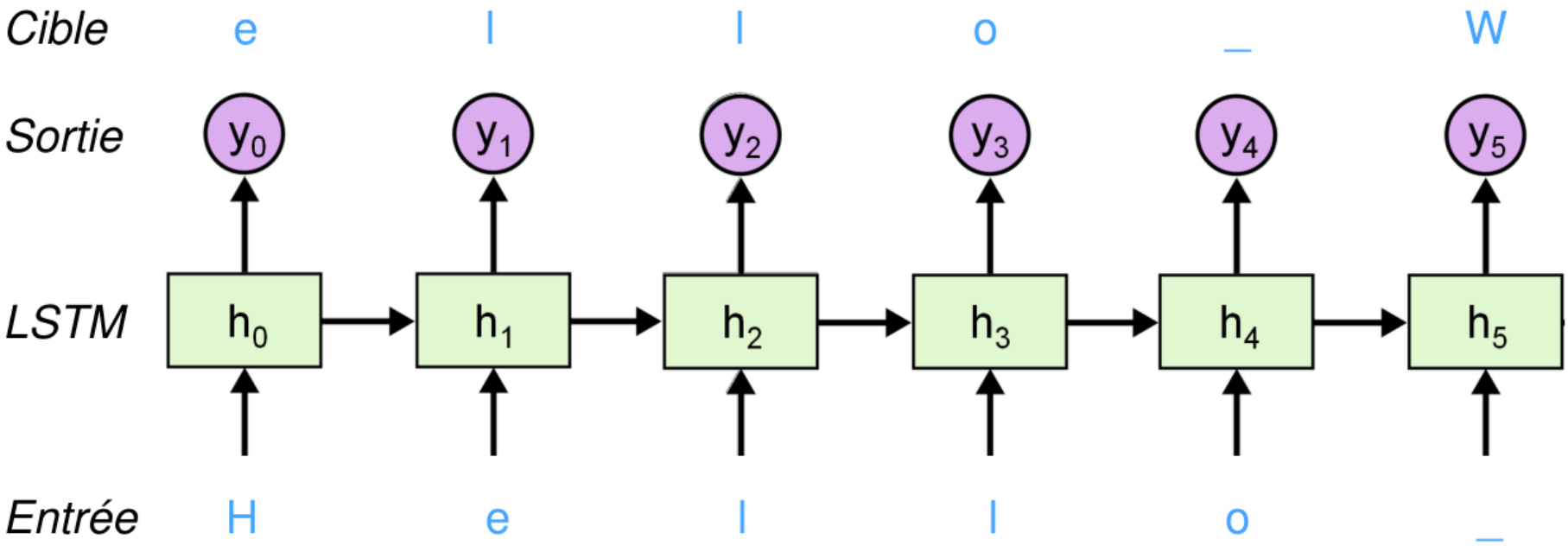
On parle de *tokens*. (En réalité, une *tokenization* optimale peut être apprise du corpus).



Modèle de Langage

Objection : prédiction du prochain *token* d'une séquence.

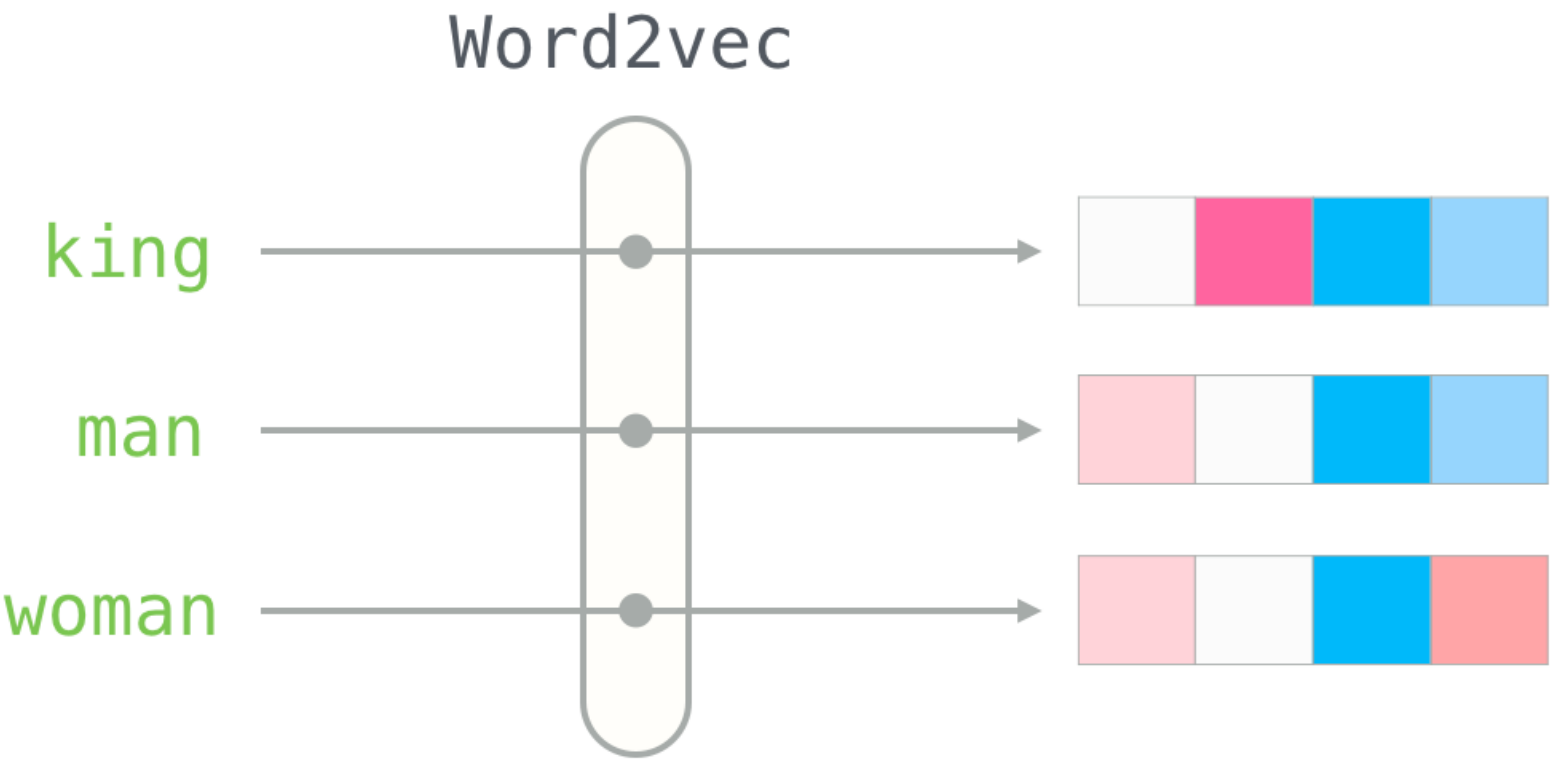
- Ensemble d'apprentissage : texte "tokenisé"
- Entrée : séquence de *tokens* $x^{<1>}, ..., x^{<t>}$
- Cible (label) : $x^{<2>}, ..., x^{<t+1>}$
- Fonction de coût : entropie croisée moyennée sur la séquence.



Erreur: Entropie croisée à chaque temps.

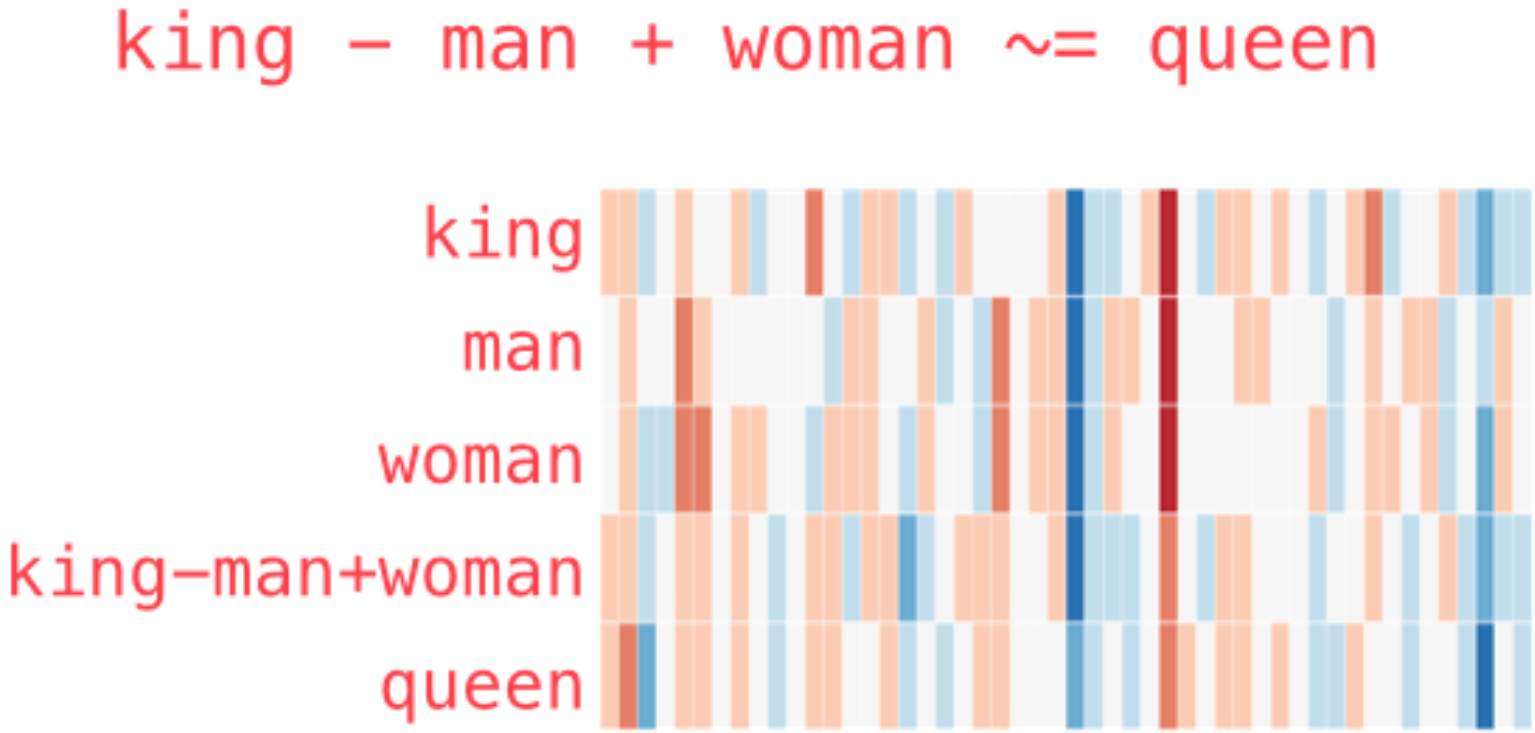
Word (token) embedding

Comment représenter numériquement les *tokens* (des chaînes de caractère) ?



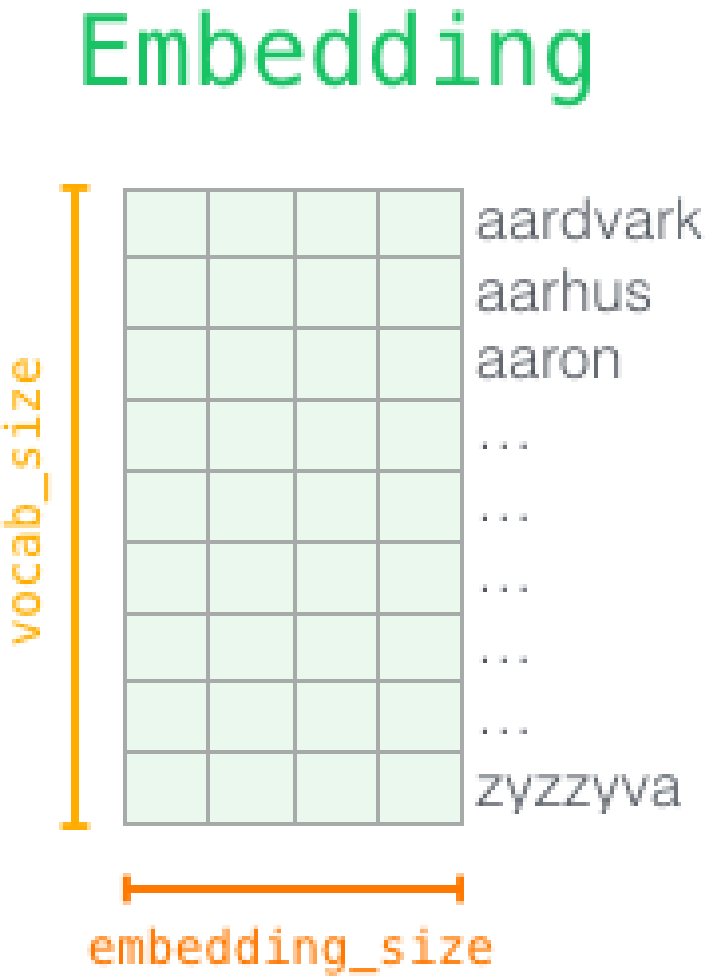
Word (token) embedding

Un *Embedding* ("plongement") est une représentation numérique d'un *token*. Ces représentations portent une signification sémantique :



Couche d'Embedding

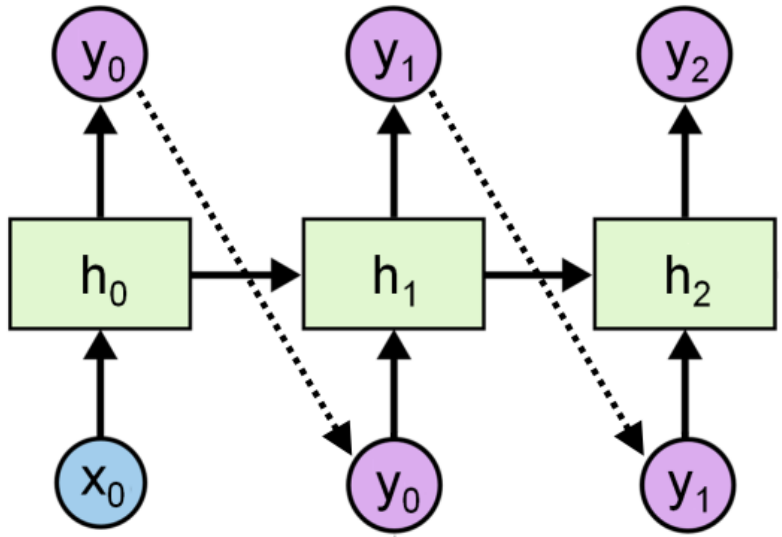
Une couche d'Embedding est une table de correspondance (*Look-up table*) dont les coefficients sont apprenables sur un ensemble d'apprentissage.



Génération de texte

On peut utiliser un RNN pour générer des séquences:

- On donne la sortie au temps t comme entrée au temps t+1
- Le modèle génère une séquence lui-même!



Génération de texte à partir d'un RNN

At initialization :
"usb9xkrd9ruaiasdsaqj'4lmjwyd61se.lcn6jey0pbco40ab'65<8um324 nqdhm<ufwty*/w5bt'nm.zq«2rqm-a2'2mstu315wtNwdqNafqh"

After one epoch :
"to will an apple for a N shares of the practeded to working rudle and a dow listed that scill extressed holding a"

After 70 epochs :
"president economic spokesman executive for securities was support to put used the sharelike the acquired who "