Réseaux récurrents

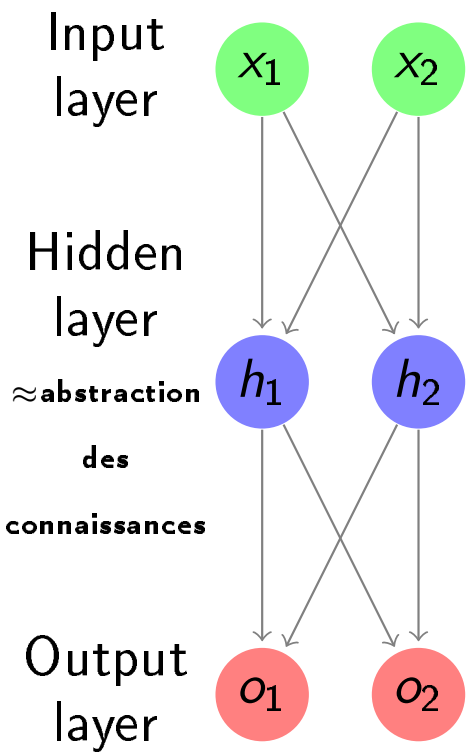
G. Durantin : gautier.durantin@e-i.com

ISAE-Supaero

Introduction

Activité introductive

Le problème de prédiction de la lettre suivante



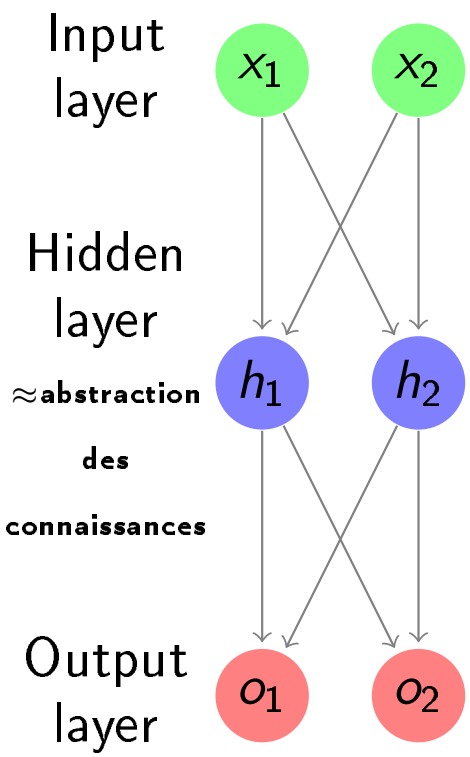
onceuponatimeaboyandagirl

- input : lettre de la séquence (encodée sur 5 bits=5 neurones)
- output : lettre suivante dans la séquence (encodée de la même manière)

Un réseau de neurones "classique" parvient-il à réaliser ce genre de tâche ?

Activité introductive

Le problème de prédiction de la lettre suivante



onceuponatimeaboyandagirl

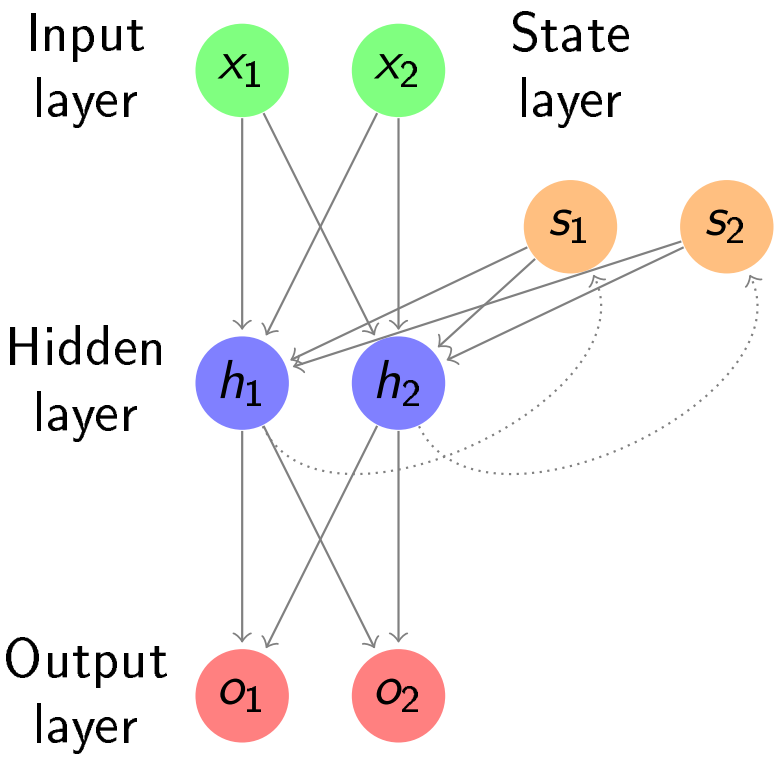
- input : lettre de la séquence (encodée sur 5 bits=5 neurones)
- output : lettre suivante dans la séquence (encodée de la même manière)

Le réseau de neurones présenté à gauche ne parvient pas à effectuer ce genre de tâche.

Même avec l'abstraction proposée par la *hidden layer*, la connaissance de la seule lettre en cours ne suffit pas pour prédire la suivante.

Activité introductive

Le problème de prédiction de la lettre suivante



[Elman, 1990] propose l'ajout d'un registre copiant à l'identique les activations de la *hidden layer* et servant d'entrée supplémentaire au système, pour conserver une mémoire de l'état du système. C'est la **State layer**[1]

[2] [Jordan, 1986] avait déjà proposé un raisonnement similaire, mais en utilisant un registre copiant la Output Layer. Elman constate toutefois que la copie d'instructions en mémoire sous forme abstraite -et donc via la Hidden Layer- semble mieux adaptée

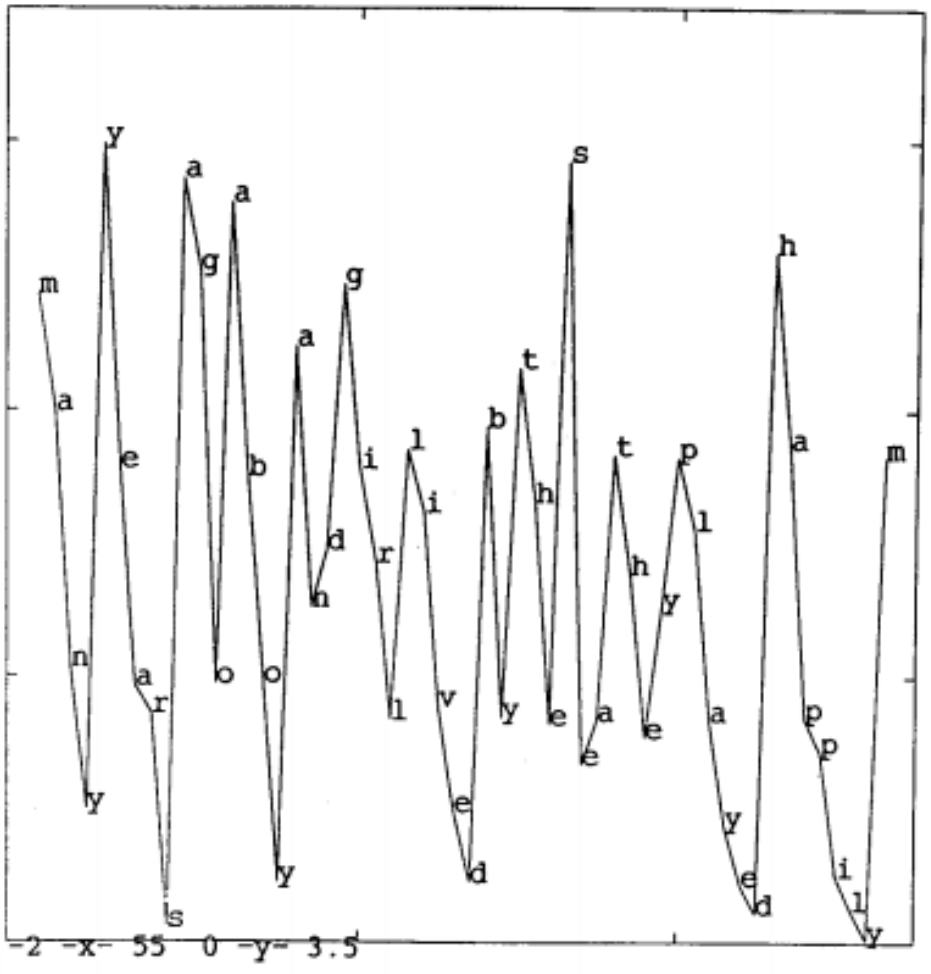


Figure 6. Graph of root mean squared error in letter-in-word precition task.

L'erreur de prédiction obtenue est forte en début de mot et diminue en fin.
Le réseau a appris à reconnaître les mots

Les séquences possédant une structure interne (même non linéaire, comme le langage) peuvent donc être apprises avec succès par des réseaux possédant une propriété de récurrence.

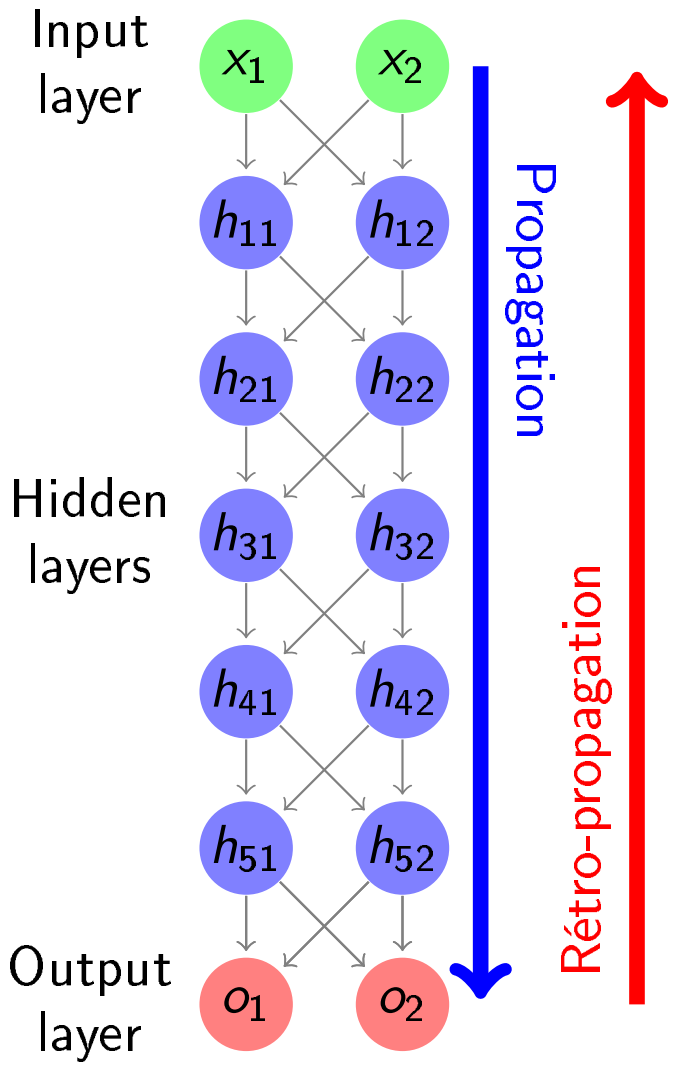
Réseaux récurrents

Activité principale de la séance

Activité principale de la séance : prédiction de séries temporelles (ici, sur un électroencéphalogramme)

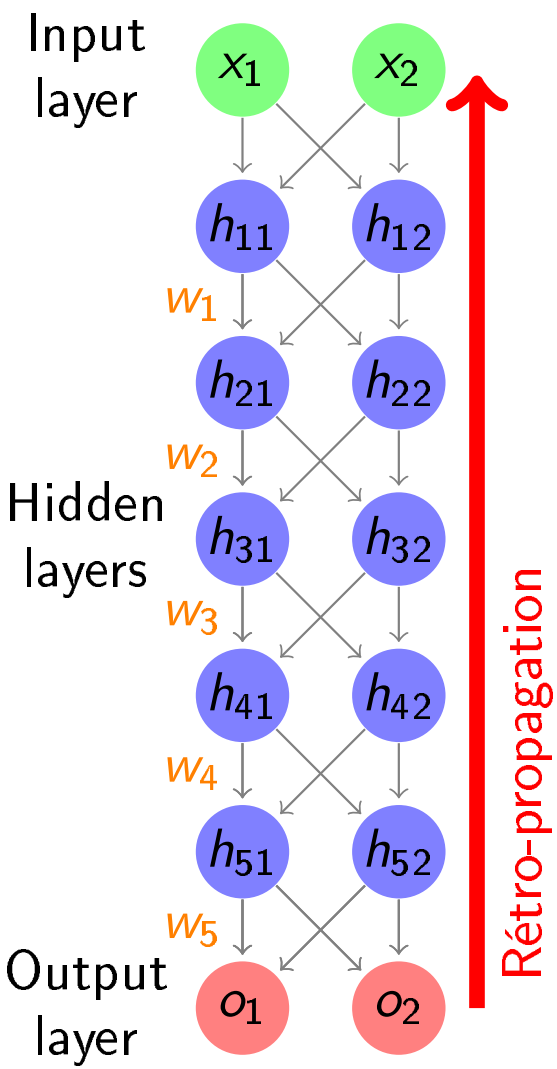
Rappel rétro-propagation du gradient

- En inférence, la propagation de l'input se fait au travers des différentes couches vers la couche de sortie.
- Pour ajuster les poids du réseau, la rétro-propagation du gradient de l'erreur de prédiction est faite depuis la couche de sortie vers l'input.



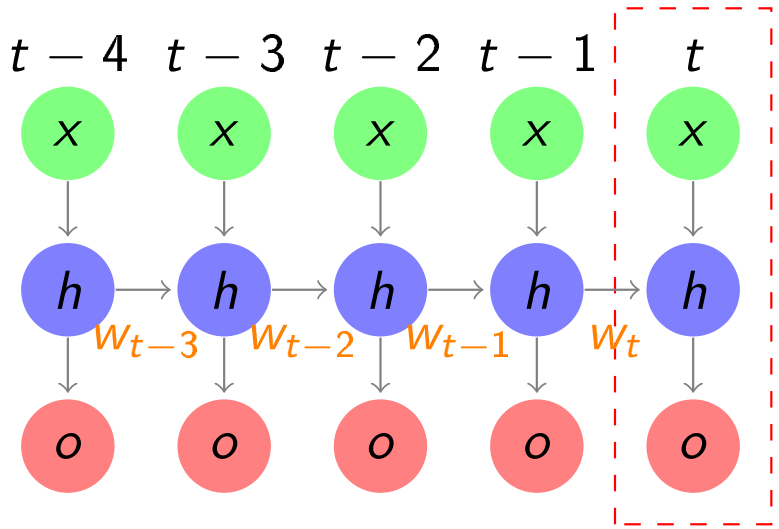
Rappel du problème de disparition/explosion du gradient

- Lors de la rétropropagation, le gradient de l'erreur est **propagé vers l'ensemble des neurones ayant contribué à produire cette erreur**, et subit une succession de produit matriciels.
- Dans l'exemple ci-contre, l'influence du gradient g sur la première couche cachée passe par la succession des w_j.
- Selon l'état des poids, ces multiplications successives peuvent conduire à une **disparition** (le plus souvent) ou à une **explosion** du gradient sur les couches éloignées de la sortie
- Ce problème affecte principalement les réseaux profonds[2]et limite l'apprentissage des couches proches de l'input.



[1]En pratique, plusieurs approches peuvent être déployées pour corriger ce problème : modification des fonctions d'activation, freezing ou clipping...

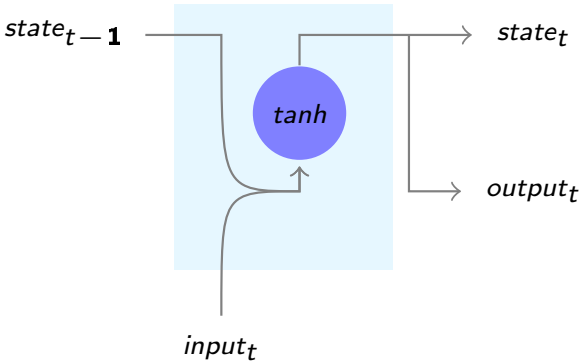
Rappel du problème de disparition/explosion du gradient



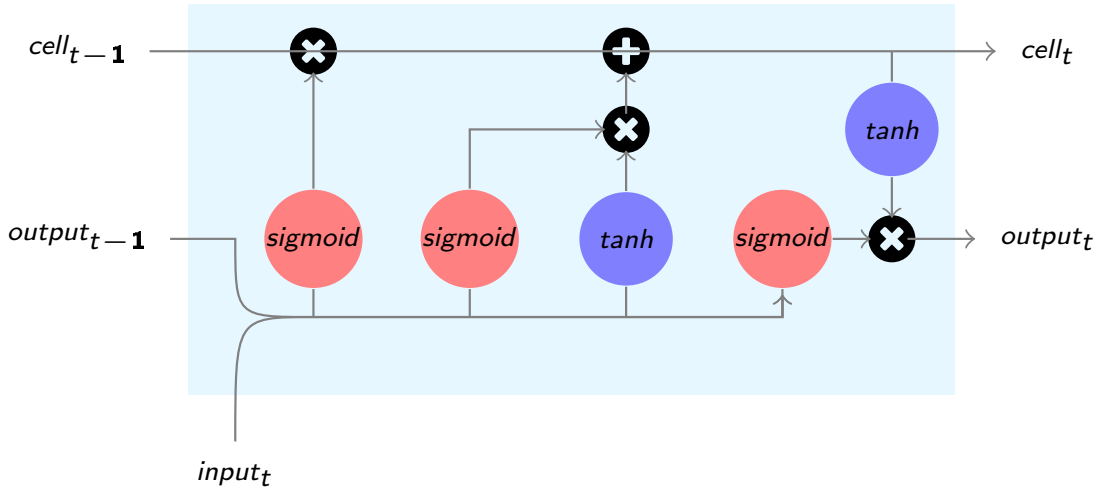
- Dans un réseau récurrent, même peu profond, le gradient de l'erreur est **rétropropagé vers l'ensemble des neurones ayant contribué à produire cette erreur, dans l'espace (couches) mais aussi dans le temps (couche récurrente)**
- A cause de l'influence des poids des couches récurrentes, le problème de disparition/explosion du gradient tend à se produire sur ces réseaux également, limitant ainsi leur apprentissage sur les événements éloignés dans le temps [Pascanu et al., 2012].

Des RNN aux LSTM (Long-Short Term Memory)

Architecture RNN

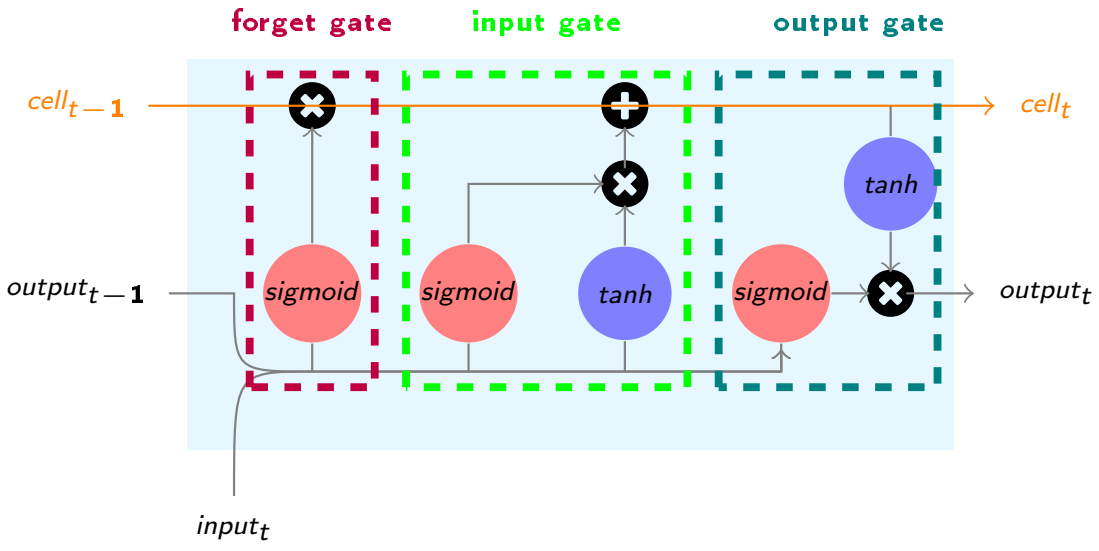


Architecture LSTM



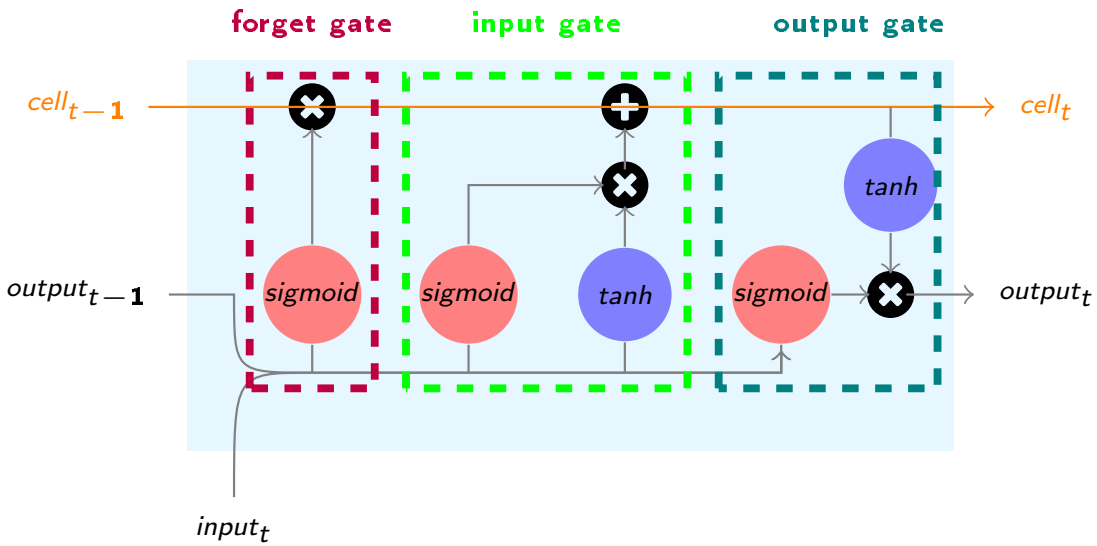
Pour corriger l'impossibilité des RNN d'apprendre sur le long terme à cause des problèmes de disparition du gradient, une architecture alternative de mémoire est proposée : la Long-Short Term Memory [Hochreiter and Schmidhuber, 1997]

LSTM



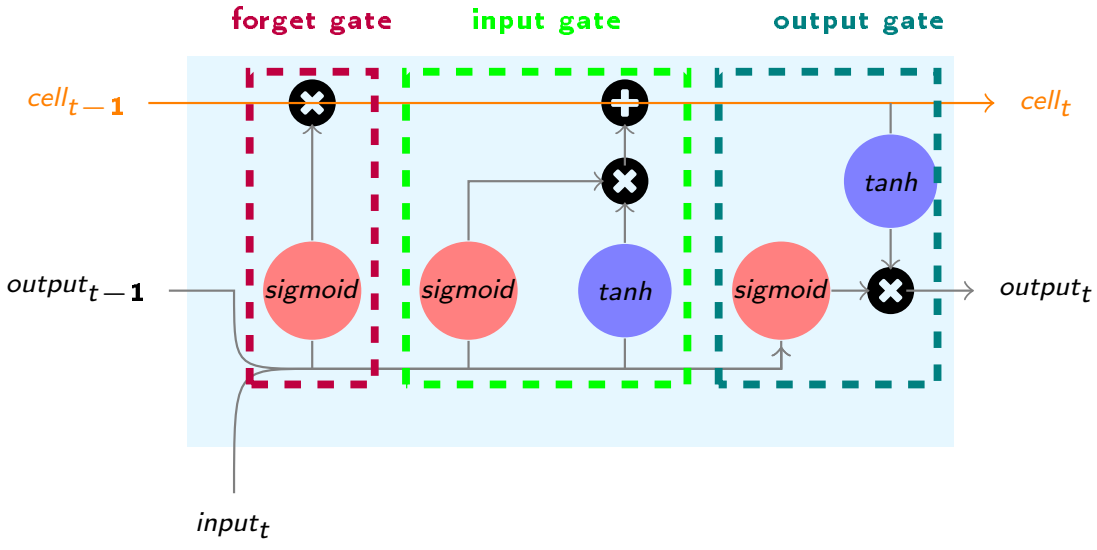
- En plus de l'état précédent de la sortie, le LSTM conserve une mémoire dans un vecteur **cell state**. Cette mémoire n'étant pas complètement remplacée à chaque itération, elle permet de conserver au besoin des informations sur le court ou long terme.

LSTM



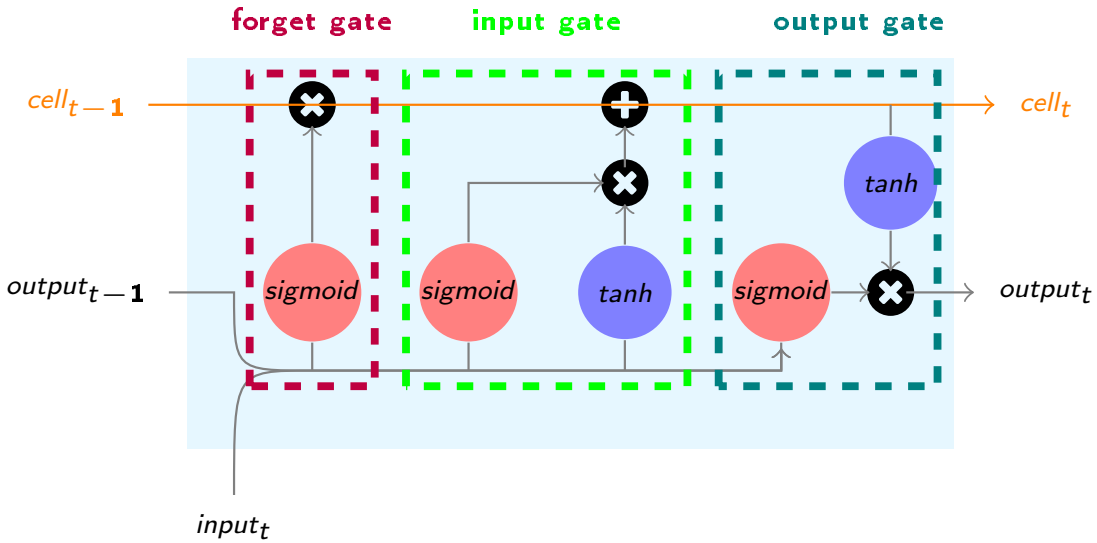
- Le LSTM utilise des portes (gates), basées sur l'utilisation de la fonction d'activation **sigmoïde**(à valeurs entre 0 et 1), comme mécanisme permettant d'atténuer/annuler certaines composantes.

LSTM



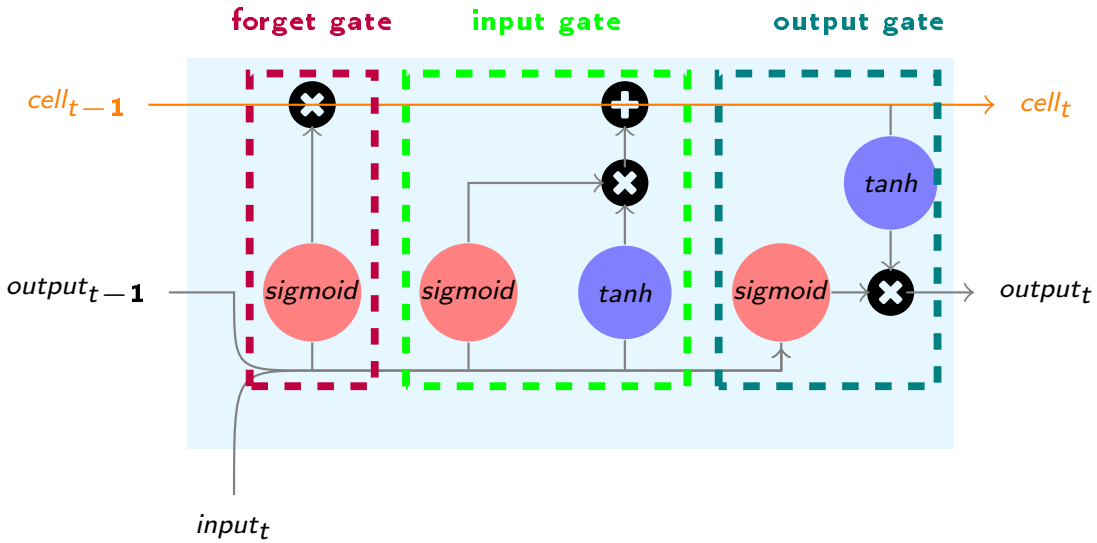
- La porte **forget** définit, à partir de l'entrée et de la sortie précédente, quelles composantes du Cell State peuvent être oubliées. Cela permet de n'utiliser qu'une partie du Cell State en fonction des itérations.

LSTM



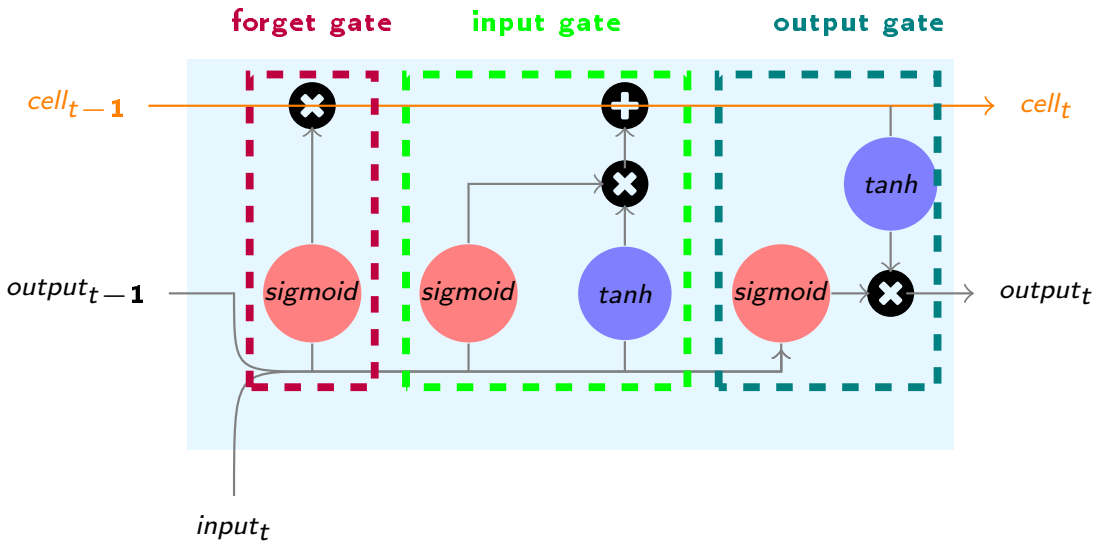
- La porte **input** met à jour certaines composantes du Cell State en fonction des entrées.

LSTM



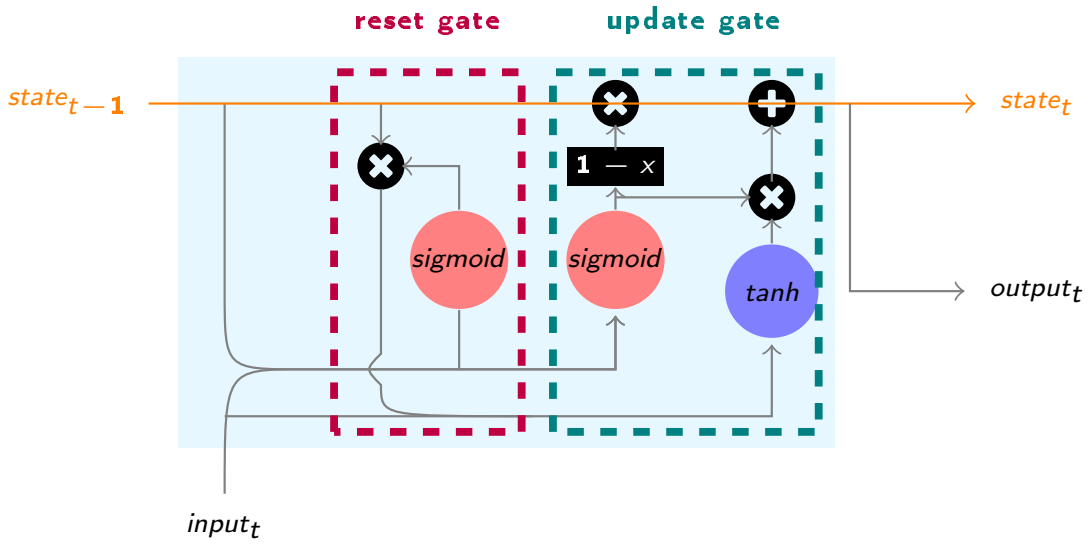
- La porte **output** permet de calculer la sortie du LSTM en utilisant certaines valeurs de la Cell State.

LSTM



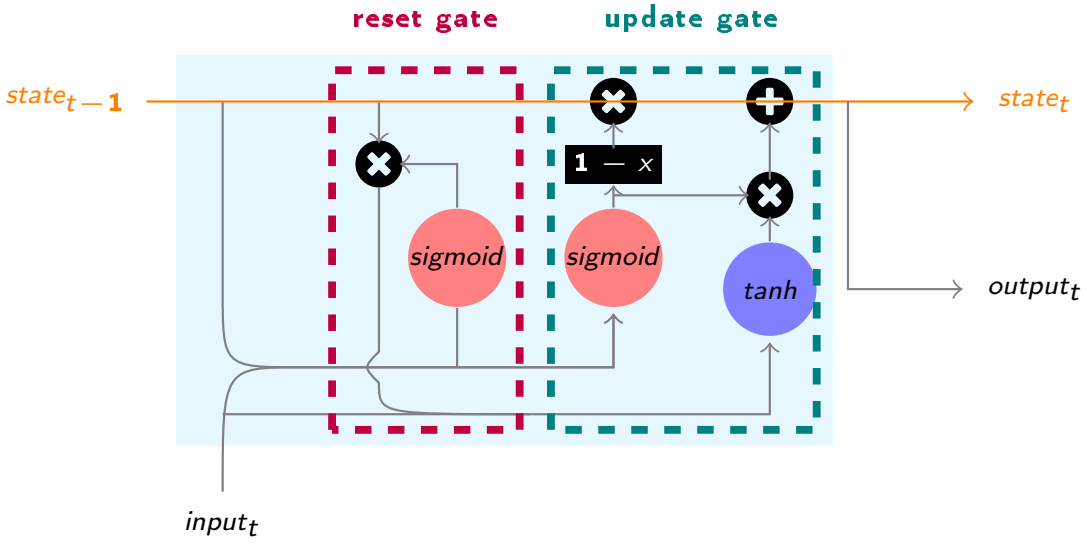
- Grâce aux **portes** permettant un mécanisme d'**oubli** et de **mise à jour** du **Cell State**, Le LSTM limite les risques de disparition du gradient à long terme. Ils sont ainsi de bons candidats pour le traitement du langage.

GRU (Gated Recurrent Units)



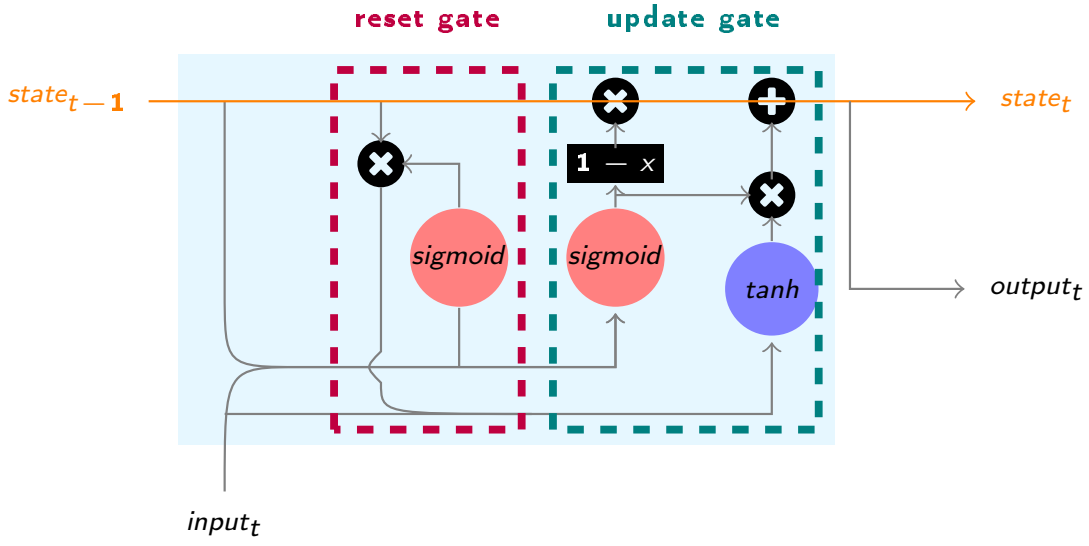
- Les GRU ont été introduits plus tardivement (2014), mais répondent au même problème que les LSTM. Toutefois, ils présentent une architecture simplifiée, **sans porte de sortie**. Ils reposent sur un **Hidden state** conservé d'itération en itération (à l'instar des RNN).

GRU (Gated Recurrent Units)



- A chaque itération, la porte **reset** définit quelles sont les composantes d'intérêt pour la prédiction de l'état suivant.

GRU (Gated Recurrent Units)



- La porte **update** réalise ensuite l'équivalent de ce que font les portes forget et input du LSTM : elle définit via une fonction sigmoïde les composantes du Hidden State à oublier. Elle met ensuite par opposition à jour les composantes du vecteur d'état en utilisant l'input. Comme pour le RNN, l'état caché mis à jour correspond à la sortie du GRU.

LSTM vs GRU

- Les GRU ont une architecture plus simple et sont par conséquent moins coûteux à mettre en place.
- Dans la majorité des cas d'usages NLP, leurs performances d'apprentissage sont équivalentes, et résolvent les problèmes de dépendance à longue distance.

  La jeune fille qui est venue me parler ce midi pendant ma pause déjeuner est française

- Le LSTM étant plus complexe, il peut en théorie apprendre plus de dépendances en présence d'une grande quantité de données.

Bibliographie et crédits image

Bibliographie


Elman, J. L. (1990).
Finding structure in time.
Cognitive science, 14(2):179–211.


Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural Computation, 9(8):1735–1780.


Jordan, M. (1986).
Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986.
Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science.


Pascanu, R., Mikolov, T., and Bengio, Y. (2012).
Understanding the exploding gradient problem.
CoRR, abs/1211.5063.