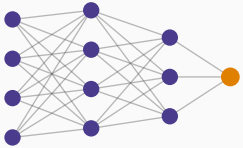
Apprentissage profond

Chapitre 9 : Les réseaux récurrents : RNN, LSTM, GRU



Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Traiter une séquence

Le problème de la mémoire longue

Le LSTM

En pratique sur des séries financières

Ce qu'il faut retenir

Traiter une séquence

Ce qui manque aux architectures précédentes

Deux limites

Un perceptron exige une entrée de **taille fixe**. Une convolution ne voit qu'une **fenêtre locale**.

Or une série financière peut exiger de se souvenir d'un choc survenu cent séances plus tôt, et les séquences n'ont pas toutes la même longueur.

Le réseau récurrent

Un **réseau récurrent** maintient un **état caché** h_t, mis à jour à chaque pas de temps :

$$h_t = g(W_h\, h_{t-1} + W_x\, x_t + b)$$

h_t résume tout ce qui a été vu jusqu'à t. Les **mêmes** poids servent à chaque pas.

La parenté avec les processus autorégressifs

Un AR(1) du chapitre 5 de *Modélisation ARMA* s'écrit $y_t = \phi y_{t-1} + \varepsilon_t$.

Un réseau récurrent obéit à la même structure : l'état présent dépend de l'état précédent. Deux différences seulement — l'état est un **vecteur** et non un scalaire, et la mise à jour est **non linéaire**.

Le problème de la mémoire longue

Pourquoi le RNN simple échoue

La rétropropagation dans le temps

Pour entraîner, on **déplie** le réseau sur toute la longueur de la séquence, puis on rétropropage.

Une séquence de cent pas devient un réseau de **cent couches**.

Le chapitre 5, appliqué au temps

Remonter cent pas multiplie cent facteurs. Le gradient **s'évanouit** ou **explose**, comme dans un réseau profond ordinaire.

Conséquence pratique : un RNN simple retient efficacement une dizaine de pas, guère plus. **Il oublie exactement ce qu'on voulait qu'il retienne.**

Un remède partiel

L'**écrêtage du gradient** traite l'explosion, en plafonnant la norme avant la mise à jour.

Il ne traite **pas** l'évanouissement, qui exige un changement d'architecture.

Le LSTM

Une mémoire qu'on peut ne pas toucher

L'idée centrale

Le **LSTM** ajoute un **état de cellule** C_t, qui traverse le temps en subissant surtout des opérations **additives**.

Trois **portes** contrôlent ce qui s'y passe.

Les trois portes

Porte	Décide
Oubli	quelle part de la mémoire effacer
Entrée	quelle information nouvelle écrire
Sortie	quelle part de la mémoire exposer

Chacune est un neurone à sortie sigmoïde : une valeur entre 0 — tout fermer — et 1 — tout laisser passer.

Pourquoi cela résout l'évanouissement

Intuition

Dans un RNN simple, l'information est **multipliée** par des poids à chaque pas : elle se dégrade géométriquement.

Dans un LSTM, si la porte d'oubli reste ouverte, l'état de cellule est simplement **transporté** — le gradient circule sans facteur multiplicatif.

C'est la même idée que les connexions résiduelles du chapitre 8 : **ménager au gradient un chemin sans multiplication**.

Définition

Le **GRU** simplifie : deux portes au lieu de trois, pas d'état de cellule séparé.

Il compte moins de paramètres, s'entraîne plus vite, et obtient des performances généralement comparables. C'est un bon point de départ sur des séries de taille modeste.

En pratique sur des séries financières

Ce qu'il faut faire, et ne pas faire

La préparation des données

1. Travailler sur les **rendements**, non sur les prix — la stationnarité du chapitre 2 de *Modélisation ARMA*.
2. Découper en **fenêtres glissantes** : les T dernières valeurs prédisent la suivante.
3. Normaliser avec les statistiques de la période d'**apprentissage seulement**.
4. Découper **chronologiquement**, jamais au hasard.

Le piège le plus coûteux

Normaliser sur toute la série avant de découper introduit dans la période d'apprentissage une information sur la volatilité **future**.

Le modèle paraît excellent en validation et échoue en production. C'est la fuite de données du chapitre 1 d'*Apprentissage statistique*, dans sa forme la plus difficile à repérer — parce que rien dans le code ne semble anormal.

Une attente à calibrer

Intuition

Sur les rendements quotidiens d'un indice liquide, aucun modèle — récurrent ou non — ne prévoit le **signe** de façon exploitable après coûts de transaction.

Ce qui se prévoit bien, en revanche, c'est la **volatilité**. Un LSTM peut y rivaliser avec un GARCH, sans le battre nettement, et à un coût de calcul très supérieur.

Le chapitre 11 chiffrera cette comparaison. Il vaut mieux aborder ces architectures sans en attendre ce qu'elles ne donneront pas.

Ce qu’il faut retenir

Le résumé du chapitre

Cinq points

1. Un **RNN** maintient un état caché h_t ; c'est un **AR non linéaire** à état vectoriel.
2. La rétropropagation dans le temps transforme une séquence de T pas en un réseau de T couches.
3. Le gradient s'évanouit : un RNN simple ne retient qu'une **dizaine** de pas.
4. Le **LSTM** transporte un état de cellule par des opérations additives, contrôlé par trois **portes**.
5. Sur séries financières : rendements, fenêtres glissantes, normalisation sur l'apprentissage **seulement**.

La suite

Le récurrent traite la séquence **pas à pas**, ce qui interdit la parallélisation et limite encore la portée mémoire.

Le chapitre 10 présente l'architecture qui a résolu les deux problèmes d'un coup.