Deep Learning par la pratique

Jour 2, après midi : RNN

Recap

On a vu

- CNN
- Tensorboard
- Augmenter les données
- Batch size
- techniques de régularisation et Dropout

Plan

- RNN appliquées aux séries temporelles
- Cellules LSTM et GRU

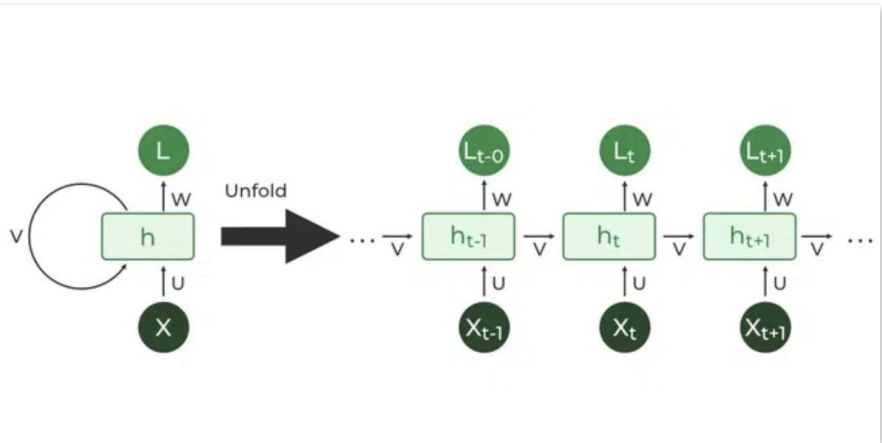
Ce qui nous mènera ensuite aux transformers

RNN

RNN

Recurrent Neural Networks : enchaîner les couches pour utiliser la sortie de la couche N-1 comme entrée de la couche N avec en plus un nouvel échantillon

- conçu pour traiter des données séquentielles
- le RNN a une mémoire et peut modéliser des dépendances temporelles
- on utilise les mêmes poids pour toutes les cellules
- à chaque cellule on ajoute l'échantillon suivant dans la séquence de données



RNN : Applications

Applications des Recurrent Neural Networks

1. Language Modelling and Generating Text
2. Speech Recognition
3. Machine Translation
4. Image Recognition, Face detection
5. Time series Forecasting

RNN : avantages et inconvenients

Avantages :

- Un RNN se souvient de chaque information au fil du temps. Il est utile pour la prédiction de séries temporelles uniquement grâce à sa capacité à se souvenir des entrées précédentes. C'est ce qu'on appelle la mémoire à long terme et à court terme (Long Short Term Memory).
- Les réseaux de neurones récurrents peuvent être utilisés avec des couches convolutives pour étendre le voisinage effectif des pixels.

Inconvénients :

- Problèmes de disparition et d'explosion du gradient.
- L'entraînement d'un RNN est une tâche difficile.
- pas efficace pour de très longues séquences

RNNs

Les RNN sont puissants pour capturer des relations séquentielles complexes dans les données, mais ils peuvent être difficiles à entraîner et souffrent de problèmes comme **l'explosion** ou la **disparition** du gradient.

Les variantes comme les **LSTM** et les **GRU** ont été développées pour atténuer ces problèmes et améliorer les performances sur des tâches séquentielles complexes.

En 2024, Pour tout ce qui est audio et NLP, les RNNs ont été remplacés par les Transformers

waaaay back in 2015!

Un article très populaire à l'époque sur l'efficacité des RNNs

The Unreasonable Effectiveness of Recurrent Neural Networks

https://karpathy.github.io/2015/05/21/rnn-effectiveness/

We can now afford to train a larger network, in this case lets try a 3-layer RNN with 512 hidden nodes on each layer. After we train the network for a few hours we obtain samples such as:

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.
```

Answering. In fact, I'd go as far as to say that

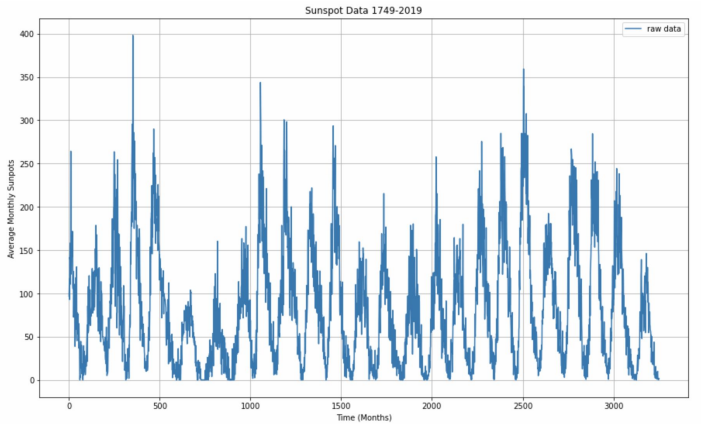
> *The concept of **attention** is the most interesting recent architectural innovation in neural networks.*

RNN simple

Serie temporelle

Variable(s) évoluant dans le temps : prix, quotation, temperature, humidité

- Air Passenger Dataset : https://www.kaggle.com/datasets/rakannimer/air-passengers
- Sunspots Dataset : https://www.kaggle.com/datasets/robervalt/sunspots
- Electricity Consumption Dataset https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption
- Stock Market Dataset https://www.kaggle.com/datasets/ehallmar/daily-historical-stock-prices-1970-2018
- Weather Data : https://www.kaggle.com/datasets/noaa/gsod
- Sales Data : https://www.kaggle.com/datasets/csafrit2/grocery-sales-data



https://www.christianhaller.me/blog/projectblog/2020-07-30-Sunspot-Activity-Time-Series/

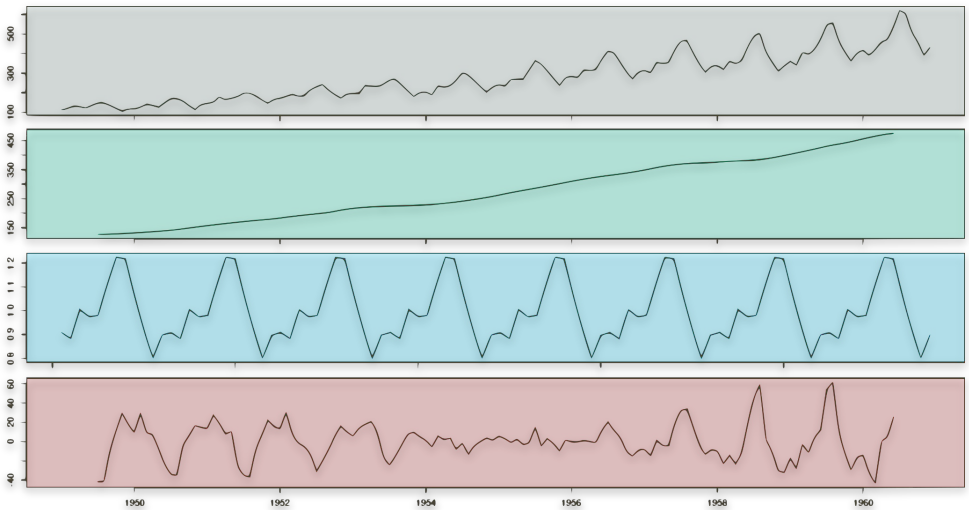
Spécificités de la prédiction des séries temporelles

- Train test split: la séparation des sets de train et test doit respecter la temporalité des données. pas de shuffling aléatoire.
- Evaluation de la performance : on utilise RMSE ou MAE, soit des métriques de régression
- rendre la série plus stationnaire en considérant son auto-différence d'écart 1 ou 2

Approches standard de prédiction des séries temporelles

- test statistiques de stationnarité
- modélisation ARIMA, AR, MA, ARMA, …
- Exponential Smoothing Models - Holt Winters
- Décomposition: tendance, saisonnalité et résidu
- Machine learning classique : XGBoost, …
- Facebook prophet
- Deep learning : RNN

Décomposition: tendance, saisonnalité et résidu



https://sthalles.github.io/a-visual-guide-to-time-series-decomposition/

RNN avec tensorflow - Keras

Les couches RNN de keras : https://keras.io/api/layers/recurrent_layers/

Recurrent layers

- LSTM layer
- LSTM cell layer
- GRU layer
- GRU Cell layer
- SimpleRNN layer
- TimeDistributed layer
- Bidirectional layer
- ConvLSTM1D layer
- ConvLSTM2D layer
- ConvLSTM3D layer
- Base RNN layer
- Simple RNN cell layer
- Stacked RNN cell layer

```
keras.layers.SimpleRNN(
    units,
    activation="tanh",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    unroll=False,
    seed=None,
    **kwargs)
```

Atelier - RNN - prédiction serie temporelle

Dans ce notebook nous allons construire des RNNs de prédiction de séries temporelles univariée.

Un premier dataset est le prix d'ouverture de cotation de l'action Google.

Nous implèmentons 3 RNNs : simple, LSTM et GRU

Votre mission :

- Modifier les architectures et parametres des modeles pour estimer les performances, rapidité d'exécution et complexité des modèles.
- N'hésitez pas à ajouter tensorboard pour analyser les modèles.
- Battre le score de **`mean_absolute_error`** de 1.78
- Dans un second temps, appliquez ces modèles sur un autre dataset comme celui des tâches solaires

LSTM et GRU

Long Short Term Memory

Long + Short ? contradiction dans les termes ?

Les LSTM sont un type spécial de RNN, capables de mémoriser des informations à long terme. Contrairement aux RNN classiques qui ont des difficultés à propager les informations pertinentes sur de longues séquences, les LSTM ont été conçus pour surmonter ce problème.

Un LSTM a une structure appelée "**cellule mémoire**" qui peut maintenir son état sur de longues périodes, permettant au réseau de se souvenir des informations pertinentes à long terme.

Un LSTM a aussi des "portes" (**gates**) qui régulent le flux d'informations entrant et sortant de la cellule mémoire. Ces portes permettent au réseau de décider quelles informations sont importantes à mémoriser, à oublier ou à transmettre à l'étape suivante.

Donc, "Long Term" fait référence à la capacité des LSTM à mémoriser et utiliser des informations pertinentes sur de longues séquences, tandis que "Short Term" fait référence à leur capacité à apprendre et à oublier des informations à court terme selon les besoins.

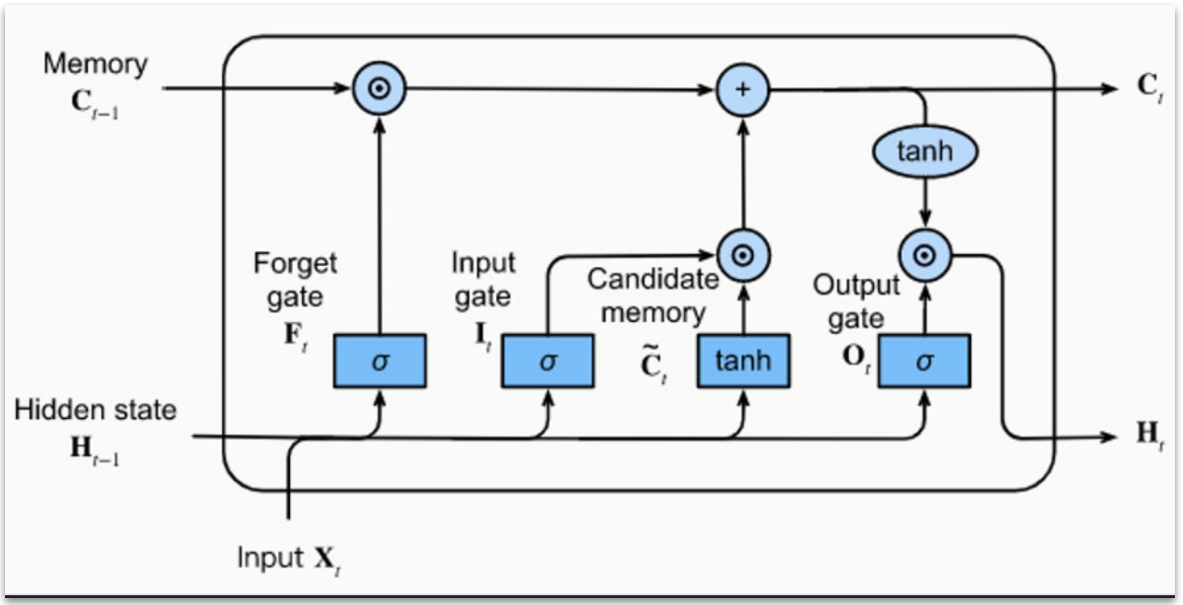
Malgré le nom apparemment contradictoire, les LSTM sont très efficaces pour traiter des séquences de données où les dépendances à long terme sont importantes, comme dans le traitement du langage naturel, la reconnaissance vocale et la prédiction de séries temporelles.

Variantes : LSTM

Long Short-Term Memory (LSTM) : variante des RNN conçus pour mieux gérer les dépendances à long terme grâce à des **cellules mémoire** et des mécanismes de **porte** pour contrôler le flux d'informations.

It introduces memory cells and gates to control the flow of information within the network. The key components of an LSTM cell are:

- **Cell state (Ct)**: It represents the memory of the LSTM unit and allows information to flow through the cell unchanged when necessary.
- **Input gate (i)**: Determines how much of the incoming information should be stored in the cell state.
- **Forget gate (f)**: Controls how much of the previous cell state should be forgotten or retained.
- **Output gate (o)**: Determines how much of the cell state should be exposed as the output of the LSTM unit.



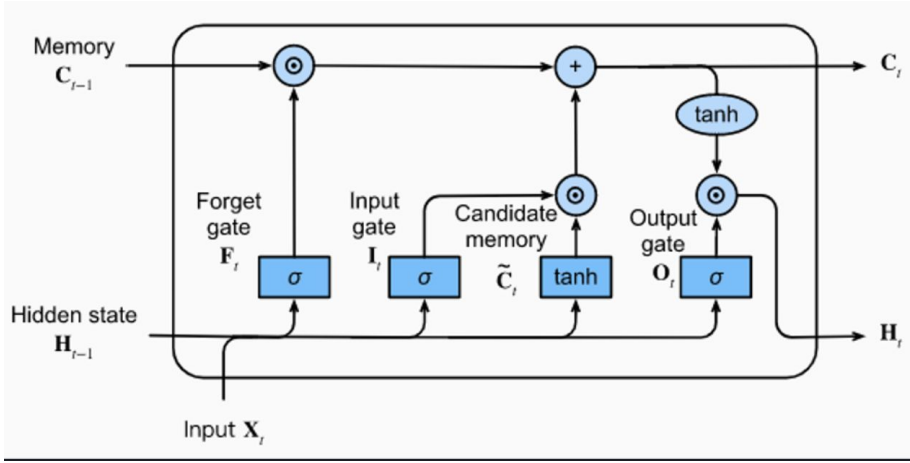
Ce schéma représente l'architecture interne d'une cellule LSTM (Long Short-Term Memory). Voici une explication détaillée des différents éléments :

1. Input Gate (Porte d'entrée) : Elle contrôle quelles nouvelles informations sont ajoutées à l'état de la cellule. Elle prend en entrée l'entrée actuelle (Xt) et l'état caché précédent (Ht-1), puis applique une fonction d'activation (généralement sigmoid) pour décider quelles valeurs mettre à jour.

2. Forget Gate (Porte d'oubli) : Elle décide quelles informations supprimer de l'état de la cellule. Elle prend également Xt et Ht-1 en entrée, puis applique une fonction sigmoid pour générer des valeurs entre 0 et 1. Les valeurs proches de 0 seront oubliées, tandis que celles proches de 1 seront conservées.

3. Candidate Memory (Mémoire candidate) : Elle crée un vecteur de nouvelles valeurs candidates qui pourraient être ajoutées à l'état de la cellule. Elle utilise une fonction d'activation (généralement tanh) pour transformer Xt et Ht-1.

4. Memory (Mémoire) : C'est l'état interne de la cellule LSTM. Il est mis à jour en multipliant l'ancien état par la sortie de la Forget Gate, puis en ajoutant la nouvelle information filtrée par l'Input Gate et la Candidate Memory.



5. Output Gate (Porte de sortie) : Elle contrôle quelles parties de l'état de la cellule sont transmises à la sortie. Elle applique une fonction sigmoid à Xt et Ht-1 pour décider quelles valeurs laisser passer, puis multiplie le résultat par l'état de la cellule transformé par tanh.

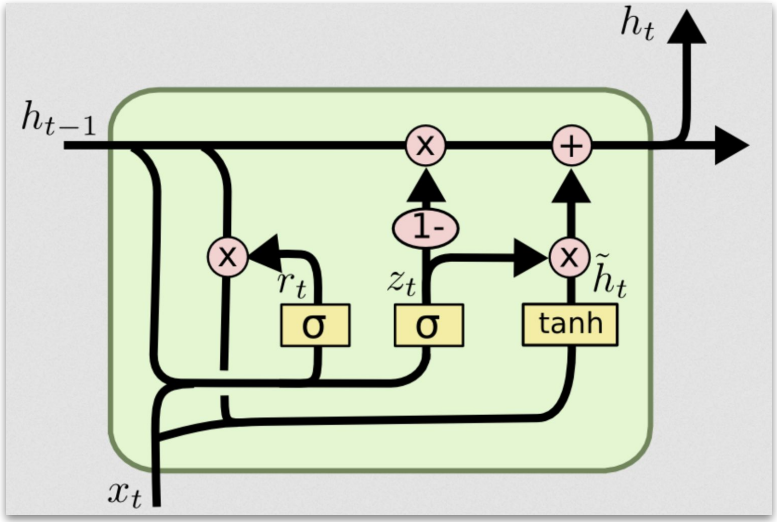
6. Hidden State (État caché) : C'est la sortie de la cellule LSTM à l'instant t. Il est calculé en multipliant la sortie de l'Output Gate par la valeur de l'état de la cellule transformé par tanh.

En résumé, les portes Input, Forget et Output permettent à la cellule LSTM de contrôler le flux d'informations, en décidant ce qu'il faut ajouter, supprimer ou transmettre à chaque étape. Cette architecture permet aux LSTM de capturer efficacement les dépendances à long terme dans les séquences de données.

Variantes : GRU

https://www.linkedin.com/pulse/day-01-basics-sequential-modelling-nlp-large-language-varshney/

Gated Recurrent Units (GRU) : Les GRU sont similaires aux LSTM mais avec une structure plus simple, utilisant moins de portes pour contrôler le flux d'informations.



la cellule GRU combine les portes d'oubli et d'entrée du LSTM en une seule porte de mise à jour. Les composants clés d'une cellule GRU sont :

- **Update Gate (z)** : Détermine la quantité de l'état caché précédent qui doit être transmise à l'étape de temps suivante.
- **Reset Gate (r)** : Contrôle la quantité de l'état caché précédent qui doit être oubliée.

LSTM ou GRU ?

	LSTM	**GRU**
Complexité du problème	Meilleur pour les séquences longues et les dépendances à long terme.	Plus efficace pour les problèmes plus simples et les dépendances plus courtes.
Ressources informatiques	Nécessite plus de ressources et de mémoire.	Moins de paramètres, plus efficace computationnellement.
Quantité de données d'entraînement	Performant avec de grandes quantités de données.	Bon résultat avec des ensembles de données plus petits.
Vitesse d'entraînement :	Entraînement plus long.	Plus rapide à entraîner.

Atelier RNN

- Sunspots https://www.christianhaller.me/blog/projectblog/2020-07-30-Sunspot-Activity-Time-Series/
- Weather data (beaucoup de manipulation des data)

https://colab.research.google.com/drive/1hkVoGLVuN0yC6hT6SI6mhk6miCIt7yIK#scrollTo=RBgR95ufJs35

- https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
- https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/

RNN pour le texte

https://www.geeksforgeeks.org/sentiment-analysis-with-an-recurrent-neural-networks-rnn/?ref=lbp