Multivariate Time Series Forecasting of Weather using PyTorch

Aurélien Blanco

01 Janvier 2026

Abstract

Nous nous intéressons ici à la prévision de séries temporelles multivariées appliquées aux données météorologiques. L'objectif est de prédire la température future (T) pour près de 10 000 localisations géographiques distinctes, en se basant sur un historique de variables atmosphériques (précipitations, humidité, pression, etc.). Pour ce faire, nous avons implémenté un réseau de neurones récurrents de type LSTM (*Long Short-Term Memory*) avec le framework PyTorch. Une attention particulière a été portée au pré-traitement des données, notamment la normalisation et l'encodage cyclique des dates, afin d'éviter les problèmes de convergence. Le modèle final est capable de générer des prédictions auto-régressives sur quelques jours, démontrant sa capacité à capturer les dynamiques saisonnières et locales.

Contents

1 Introduction

La prévision météorologique est un défi classique de la science des données, traditionnellement résolu par des modèles physiques complexes de dynamique des fluides. Cependant, avec l'avènement du Deep Learning, les approches basées sur les données historiques offrent une alternative puissante.

Dans ce projet, nous traitons un problème de régression sur séries temporelles multivariées. Notre jeu de données contient des relevés quotidiens pour plusieurs milliers de stations. Ce projet est complexe par la densité d'informations présentes dans le dataset, nous avons pour chaque jour, 10 000 lignes correspondants à des mesures prises dans différentes villes. Le dataset étant riche, nous allons essayer de tirer profit de toutes ces informations pour prédire 10 000 températures à chaque prédiction.

1.1 Related work

Les réseaux de neurones récurrents (RNN) sont naturellement adaptés aux données séquentielles. Cependant, les RNN classiques souffrent du problème de disparition du gradient (*vanishing gradient*) sur de longues séquences. L'architecture LSTM (*Long Short-Term Memory*), introduite par Hochreiter et Schmidhuber, résout ce problème grâce à un système de portes (*gates*) permettant de conserver l'information sur de plus longues périodes. C'est l'architecture que nous avons choisie pour ce projet, car la température d'un jour dépend fortement des conditions des jours précédents (inertie thermique).

2 Method

2.1 Data Pre-processing

Le succès d'un modèle de Deep Learning dépend fortement de la qualité des données en entrée. Nous avons appliqué les transformations suivantes :

- **Normalisation :** Les variables ont des échelles très différentes (ex: la pression atmosphérique ≈ 1000, les précipitations ≈ 0, la température ≈ 15). Sans normalisation, le modèle peine à converger et tend à prédire une valeur moyenne constante. Nous avons utilisé un `StandardScaler` (moyenne 0, écart-type 1) sur toutes les features ainsi que sur la cible.
- **Encodage cyclique des dates :** La date brute (ex: 20260101) est une valeur croissante qui ne capture pas la notion de saisonnalité. Nous avons transformé la date en deux composantes sinusoïdales pour que le modèle comprenne que le 31 décembre est proche du 1er janvier :

$$x_{sin} = \sin(\frac{2\pi \times \text{day}}{366}), \quad x_{cos} = \cos(\frac{2\pi \times \text{day}}{366})$$

- **Séquençage :** Les données tabulaires ont été transformées en fenêtres glissantes (*Sliding Windows*). Pour chaque localisation, nous utilisons les 7 jours précédents pour prédire le jour suivant.

2.2 Architecture

Nous avons conçu une architecture compacte mais efficace pour gérer la dimensionnalité du problème :

- **Entrée :** Tenseur de dimension `(Batch_Size, Sequence_Length=7, Input_Size=29)`. Les 29 entrées correspondent aux variables météorologiques + les dates encodées.
- **Couche LSTM :** Une couche unique avec une taille cachée (*Hidden Size*) de 64 neurones.
- **Couche Linéaire (Fully Connected) :** Une couche de sortie connectée au dernier pas de temps du LSTM, projetant la dimension 64 vers 1 seule valeur (la Température).

2.3 Parameters & Hyperparameters

Les hyperparamètres retenus pour l'entraînement sont les suivants :

- **Fonction de perte (Loss) :** MSE (*Mean Squared Error*), adaptée à la régression.
- **Optimiseur :** Adam, avec un taux d'apprentissage (*Learning Rate*) de 0.01.
- **Batch Size :** 1024, pour accélérer le calcul sur GPU.
- **Split Train/Test :** 90% pour l'entraînement, 10% pour le test.
- **Epochs :** 10 (suffisant après normalisation pour observer une convergence).

La stratégie de prédiction future (Forecasting) est **récursive** : pour prédire à $J+1$, le modèle utilise les données réelles jusqu'à J. Pour prédire à $J+2$, il utilise sa propre prédiction de $J+1$ comme entrée, et ainsi de suite sur 30 jours.

3 Experiments

3.1 Weather dataset

Le jeu de données provient de fichiers CSV concaténés, représentant des relevés météorologiques pour 9892 localisations distinctes identifiées par leurs coordonnées `LAMBX` et `LAMBY`.

Les caractéristiques (*features*) incluent : `PRENEI` (neige), `PRELIQ` (pluie), `FF` (vent), `Q` (humidité spec.), `DLI`, `SSI`, `HU` (humidité), ainsi que les dates transformées. La variable cible est `T` (Température).

Le volume total des données dépasse 14 millions de lignes, pour 29 attributs (ou features).

Le dataset provient du gouvernement : https://meteo.data.gouv.fr/datasets/6569b27598256cc583c917a7 . Et seules les données datant d'après 2022 ont été utilisé.

3.2 Results

Avant la normalisation des données, nous avons observé un phénomène de "collapse to the mean", où le modèle prédisait une valeur constante d'environ 12°C. Après l'application du `StandardScaler`, la *Loss* a chuté drastiquement (de ≈ 88.0 à ≈ 0.02 sur les données scalées).

La fonction de test retourne une erreur moyenne faible, et les graphiques montrent que le modèle parvient à suivre les tendances de température sur l'ensemble de test.

3.3 Analysis

L'analyse des prédictions sur 30 jours futurs montre que le modèle a du mal a prédire au long terme.

Un point critique a été l'implémentation de la fonction `predict_future_all_locs`. Plutôt que de prédire location par location (ce qui serait trop lent), nous passons un tenseur de dimension `[9892, 7, 29]` au modèle. Le modèle prédit alors simultanément la température du jour suivant pour toutes les villes. Cette approche vectorisée utilise pleinement la puissance du GPU.

4 Conclusion

Ce projet a permis de mettre en œuvre un pipeline complet de prévision météorologique avec PyTorch. L'architecture LSTM s'est révélée efficace pour modéliser la température, à condition de rigoureusement normaliser les données en entrée.

Nous avons réussi à générer des prévisions très approximative sur 30 jours pour près de 10 000 points géographiques. Des améliorations pourraient être apportées, notamment en utilisant une couche d'*Embedding* pour que le modèle apprenne les spécificités géographiques de chaque `LOCATION`, ou en remplaçant le LSTM par un Transformer pour mieux capturer les dépendances à long terme.

Nous tenons à préciser que le modèle fournit d'excellent résultat à cours terme, cela est du que pour les prédictions futures, une récursion est effectué sur les nouvelles données prédites, or juste la température est prédite est non pas les 29 attributs, de ce fait les prédictions futures sont de plus en plus mauvaises au fil des jours.

Note technique : Enfin, tout ceci a été réalisé sur un ordinateur équipé de 32Gb de RAM ainsi que d'une carte graphique Nvidia P100 16Gb, ce qui a nécessité l'utilisation de `DataLoader` optimisés (`pin_memory=True`) pour gérer les transferts de données volumineux.