Projet ODATA - Recherche de données climatiques par similarité

Gilles Vanwormhoudt

November 3, 2023

1 Introduction

Dans ce projet, nous nous intéressons à la recherche de données climatiques par similarité en utilisant le langage Python. Avec ce type de recherche, un utilisateur sélectionne un exemple de données climatiques comme requête et le processus de recherche trouve dans une base la ou les données climatiques qui sont les plus similaires par rapport à des caractéristiques prédéfinies. Ce type de recherche peut servir par exemple à déterminer si des conditions climatiques ont déjà été rencontrées par le passé pour éventuellement prévoir les conditions climatiques à venir ou par faire des analyses variées sur le climat (par regroupement par exemple).

Dans le cas de ce projet, vous utiliserez un jeu de données climatiques appelés $JENA$. Ce jeu de données a été enregistré par une station météo à l'institut Max Planck pour la biogéochimie en allemagne durant une période situé entre le 1er janvier 2009 et le 31 décembre 2016 (https://www.bgc-jena.mpg.de/wetter/). Il comprend des relevés constitués de 14 quantités (tel que la température de l'air, la pression atmosphérique, l'humidité, la direction du vent, ...). Le tableau 3 ci-dessous présente en détail ces quantités. Ces relevés ont été effectués toutes les 10 minutes au cours de la période indiquée précédemment. Il s'agit donc d'une série temporelle de données multivariées.

Les objectifs principaux du projet sont d'une part de mettre en oeuvre les différentes étapes nécessaires à la réalisation de la recherche par similarité pour ce jeu de données et d'autre part de faire les expérimentations pour obtenir de bonnes performances à la manière de ce que nous avons accompli durant les séances de travaux pratiques. La conception d'une interface pour permettre à un utilisateur d'efffectuer des recherches ne fait pas partie des objectifs du projet.

Les étapes à réaliser sont décrites dans les prochaines sections. Ces étapes sont les suivantes:

1. Récupération et analyse du jeu de données
2. Préparation et transformation du jeu de données
3. Recherche par similarité: méthode exacte et méthode approximative
4. Exploitation pratique des résultats

2 Récupération et analyse du jeu de données

Le jeu de données est disponible dans la page de l'UV ODATA au sein de la section 'Recherche par similarité'. Il se présente sous la forme d'un fichier nommé 'jena_climate_2009_2016.csv' au format CSV. Ce fichier est donc facilement manipulable avec la bibliothèque Pandas.

Après téléchargement du fichier correspondant au jeu de donnée, vous pouvez analyser son organisation grâce aux opérations classiques fournies par Pandas (nombre et type de colonnes, nombres de lignes, statistiques, est-ce que toutes les valeurs sont quantitatives, etc). Il est également important de regarder si le jeu contient des valeurs manquantes ou aberrantes.

Index	Features	Format	Description
1	Date Time	01.01.2009 00:10:00	Date-time reference
2	p (mbar)	996.52	The pascal SI derived unit of pressure used to quantify internal pressure. Meteorological reports typically state atmospheric pressure in millibars.
3	T (degC)	-8.02	Temperature in Celsius
4	Tpot (K)	265.4	Temperature in Kelvin
5	Tdew (degC)	-8.9	Temperature in Celsius relative to humidity. Dew Point is a measure of the absolute amount of water in the air, the DP is the temperature at which the air cannot hold all the moisture in it and water condenses.
6	rh (%)	93.3	Relative Humidity is a measure of how saturated the air is with water vapor, the %RH determines the amount of water contained within collection objects.
7	VPmax (mbar)	3.33	Saturation vapor pressure
8	VPact (mbar)	3.11	Vapor pressure
9	VPdef (mbar)	0.22	Vapor pressure deficit
10	sh (g/kg)	1.94	Specific humidity
11	H2OC (mmol/mol)	3.12	Water vapor concentration
12	rho (g/m ** 3)	1307.75	Airtight
13	wv (m/s)	1.03	Wind speed
14	max. wv (m/s)	1.75	Maximum wind speed
15	wd (deg)	152.3	Wind direction in degrees

Figure 1 – Types de quantité recueillis dans le jeu de données

3 Préparation et transformation du jeu de données

Après avoir analysé la donnée, vous avez pu observer que les lignes sont indicées par des nombres entiers. En général, quand on a affaire à des données de type 'série temporelle', il est préférable d'indicer les lignes par la date de collecte de la donnée. Cela facilite le calcul de statistiques sur une période donnée ou de statistiques selon un certain fréquence temporel (semaine, mois, ...). Pour permettre un tel indiçage, il est nécessaire que chaque donnée possède une date unique dans le jeu ce qui est le cas ici et plus généralement dans les séries temporelles.

La première opération est donc d'effectuer ce ré-indiçage en utilisant Pandas. Pour cela, il est nécessaire de convertir au préalable la colonne contenant la date sous forme de chaîne en une date sous forme d'objets. Cette conversion peut être réalisée à l'aide de la fonction `to_datetime` avec le format '

Le ré-indiçage étant effectuée, il s'agit à présent de normaliser les valeurs des différentes colonnes. Cette opération a pour but de mettre sur une même échelle toutes les mesures. Non seulement cela va égaliser le poids de chaque mesure et standardiser leur moyenne et écart-type mais cela va également faciliter les comparaisons pour la similarité (i.e le calcul des distances) et lui donner plus de sens. Il existe plusieurs façon de normaliser. Vous pouvez appliquer la normalisation Min-Max ou la normalisation Standard (voir les formules dans Wikipedia).

Comme indiqué dans l'introduction, les données initiales correspondent à des relevés effectués toutes les 10 minutes. Cela représente un nombre important de données pour jour. L'étape suivante consiste à aggréger les données journalières. Pour cela, Pandas dispose d'une fonction 'resample'. Cette fonction permet d'aggréger des données temporelles sur une période spécifique (par exemple, aggréger des données par heure, par jour, par semaine, ...) et effectuer des opérations statistiques sur ces groupes. Dans le cas présent, il est demandé d'aggréger les données de chaque jour sur des périodes de 2h en se basant sur la moyenne (mean). Il est à noter que la fonction 'resample' de Pandas est capable d'appliquer l'aggregation sur toutes les quantités en même temps.

En sortie, cette fonction produit un nouveau DataFrame (il est important de vérifier la diminution du nombre de lignes dans le tableau).

La figure suivante montre sur un graphe le résultat recherché pour un jour particulier et la quantité correspondant à la température de l'air. Au préalable, une journée correspond à 144 valeurs (6*24h) de température. Après transformation, une journée correspond à 12 valeurs (24h/2h).

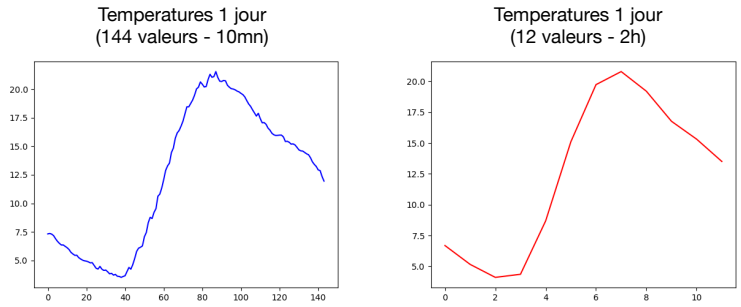

Figure 2 – Aggrégation des températures sur une journée (en utilisant des périodes de 2h)

4 Extraction des caractéristiques

A présent, nous allons nous pencher sur la construction des vecteurs qui vont être utilisés pour la recherche par similarité. Le principe est de construire un vecteur pour chaque jour inclus dans le jeu de données. La méthode de construction consiste à concatener les séries temporelles journalière correspondant à chaque quantité. La figure suivante résume le principe pour trois quantités (température, pression et humidité).

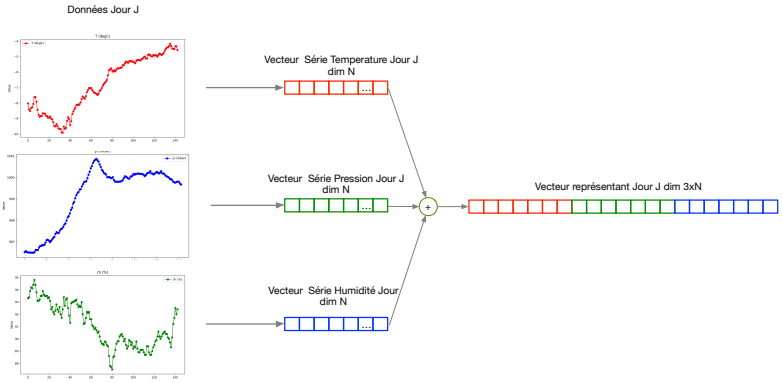

Figure 3 – Construction du vecteur pour 1 journée par concaténation des données

Maintenant que vous savez comment construire le vecteur caractéristique d'un journée, il s'agit d'appliquer cette opération sur l'ensemble des journées. Les quantités à retenir pour construire ces vecteurs sont:

— p (mbar)
— T (degC)
— rh ()
— VPact (mbar)
— H2OC (mmol/mol)
— wv (m/s)

Il est également important de stocker ces vecteurs pour éviter de les recalculer à chaque lancement du programme. Pour faciliter ce stockage, vous pouvez construire une matrice Numpy (NB Jours x (12 X NB Quantités)) à partir de tous les vecteurs obtenus et utiliser la fonction save de Numpy pour enregistrer la matrice dans un fichier. Cela facilitera le chargement de tous les descripteurs en une opération (fonction Numpy load).

En parallèle au stockage des vecteurs correspondant aux quantités des jours, il est important de construire un tableau Numpy correspondant avec le dates puis de stocker ce tableau. Cela permettra d'indiquer le jour concerné lors de l'affichage des résultats de recherche (cf l'utilisation des vecteurs de mots dans les TP où ce principe a été appliqué : une matrice numpy contenant les vecteurs des mots et un tableau numpy des mots).

5 Recherche par similarité

5.1 Méthode exacte

Nous pouvons à présent mettre en oeuvre une première recherche par similarité en utilisant un algorithme de force brute vue en cours et en TP. Pour rappel, une recherche par similarité s'effectue en calculant une distance ou un mesure de similarité entre un objet requête et les objets formant la base. La recherche retourne les objets avec le meilleur score par rapport à l'objet requête (c'est à dire les plus proches ou les plus similaires selon la fonction choisie).

Dans notre cas, les objets formant la base et une requête auront chacun la forme d'un vecteur correspondant à une journée. D'un point de vue pratique et dans le but de faciliter l'analyse des résultats, votre programme de recherche doit retourner les dates correspondant aux plus proches voisins ainsi que le score obtenu. Cela facilitera l'analyse des résultats.

Pour l'algorithme de force brute, vous pouvez utiliser la version réalisée lors du premier TP ou celui offert par une bibliothèque comme Scikit (vu également en TP). Il faut également construire un jeu de requêtes pour tester les recherches. Une solution est de décomposer le tableau de départ en deux sous-ensembles, un gros par la base et un petit pour les requêtes. On peut réserver par exemple les 100 dernières lignes du tableau comme ensemble de requêtes.

Dans un premier temps, effectuez quelques tests de recherche en utilisant la distance euclidienne comme distance entre les vecteurs correspondants aux données. Cette distance produit-elle de bons résultats ? Puis expérimentez d'autres distances parmi celles vues en TP.

Pour les tests, vous utiliserez la recherche des plus proches voisins avec plusieurs valeurs de k (1,3, 5, 10). Une vérification peut être faite en traçant des graphes de comparaison entre la requête et les plus proches voisins pour différentes quantités.

5.2 Méthode approximative

On indexe maintenant les données climatiques à l'aide d'une méthode LSH. Etant donné que nous n'avons pas de connaissances préétabli sur l'application de LSH pour ce type de donnée, il faut procéder à des évaluations de performances en faisant varier les différents paramètres offerts par cette méthode: le *nombre de projections*, le *nombre de table*, la *valeur de W*. Pour cette partie, vous pouvez vous réutiliser l'algorithme LSH fourni en TP.

Commencez par mettre en oeuvre la méthode LSH avec une valeur spécifique pour les paramètres de façon à vérifier le bon fonctionnement de cette méthode avec le jeu de donnée et le jeu de requêtes fournis.

Pour déterminer les valeurs de paramètres de LSH donnant de bonnes performances, le protocol expérimental pratiqué en TP peut-être repris. Pour rappel et selon ce protocol, on commence par calculer une vérité terrain sur l'ensemble des requêtes avec l'approche force-brute puis on évalue ensuite l'impact des différents paramètres sur la qualité des résultats ainsi que sur la vitesse d'exécution. Dans la mise en oeuvre de ce protocole, il est permis de réutiliser tout ou partie du code ayant été mis en oeuvre durant les TPs.

A l'issue de vos expérimentations, il est demandé de préparer une synthèse écrite que vous incluerez dans le rapport. Dans cette synthèse, vous donnerez votre analyse et interprétation des résultats. Vous indiquerez également de façon argumentée la sélection des paramètres que vous conseillez pour appliquer LSH au jeu de données en question. Le choix des mesures d'évaluation est libre. L'usage de graphiques ou de tableaux est recommandé pour illuster et expliquer vos résultats.

6 Exploitation pratique des résultats de recherche

Dans cette partie, nous nous intéressons à une exploitation pratique des résultats de recherche.

Une exploitation intéressante des résultats est de prévoir la météo qu'il fera à court terme (par exemple le lendemain) en exploitant la similarité des quantités obtenues pour une journée (requête) avec celles obtenus dans le passé. Cette méthode est connu sous le nom "Regression avec les plus proches voisins".

Dans le cas présent, on vous demande de mettre en oeuvre cette idée de façon élémentaire en utilisant les 2 plus proches voisins. Pour cela, vous sélectionnerez une journée parmi les objets requêtes construites précédemment puis vous soumettrez cette requête à une recherche par similarité. Une fois les 2 plus proches voisins obtenus, il est demandé de construire un graphe par quantité retenu dans lequel vous afficherez les valeurs temporelles pour la requête et les voisins. Cela permettra de comparer visuellement si les courbes sont proches ou pas. En analysant les graphes obtenus que pouvez-vous dire ? Est-ce que la distance utilisée se révèle satisfaisante ?

Pour terminer, vous produirez des graphes de comparaisons similaires en utilisant le jour suivant de chaque objet (c'est-à-dire celui de l'objet requête et ceux de ses 2 plus proches voisins). L'idée est d'examiner si cette approche peut servir afin de prédire les conditions météorologiques du lendemain.

Peut-on dire que cette méthode est efficace pour ce type de prédiction ? Justifiez votre réponse.

Pensez-vous que l'aggrégation des données des voisins serait utile pour des prédictions plus fiables ? Justifiez votre réponse.

7 Livrables

Les livrables à remettre à l'issue du projet sont:

1. Un rapport au format PDF contenant:
   - un rappel succint des objectifs du projet
   - une description succinte du système mis en oeuvre
   - une description du protocole expérimental et une description des résultats obtenus pour la méthode LSH
   - une description des résultats obtenus dans l'application pratique pour obtenir des prédictions
   - un bilan sur cette étude (ce que vous avez appris et apprécié, les difficultés rencontrées, ce que vous auriez voulu explorer d'avantage, ...)
   - une mise en perspective de votre travail
2. Le code permettant de reproduire les expérimentations, suffisamment documenté. Cela peut-être un ensemble de fichiers Python ou un notebook Jupyter selon vos préférences.