Modèle de Markov à état caché : Prédiction de la consommation française d'électricité

WANNENMACHER LUCAS

28 septembre 2020

Table des matières

Chapitre 1

Etude bibliographique : Présentation du modèle de Markov caché

Introduction au modèle

Le modèle de Markov caché est un modèle graphique, qui est généralement utilisé pour prédire des états (cachés) en utilisant des données séquentielles comme le temps, le texte, la parole, etc.

Prenons un exemple : un casino malhonnête utilise deux dés , l'un d'eux est juste, l'autre est pippé. Pippé signifie que l'un des dés n'a pas les probabilités définies comme (1/6, 1/6, 1/6, 1/6, 1/6, 1/6,/ 1/6). Le casino lance au hasard l'un des dés à un moment donné. Supposons maintenant que nous ne savons pas quel dé a été utilisé à quel moment (l'état est caché). Cependant, nous connaissons le résultat des dés (1 à 6), c'est-à-dire la séquence des lancers (observations). Le modèle de Markov caché peut utiliser ces observations et prédire quand le dé dé déloyal a été utilisé (état caché).

Dans le modèle de Markov, le futur état ne dépend que de l'état/événement actuel et non d'autres états plus anciens.

Dans l'exemple précédent, la probabilité de l'état au temps t ne dépendra que de l'état au temps t-1 : en d'autres termes, la probabilité de s(t) sachant s(t-1), soit P(s(t)|s(t-1)) . C'est ce qu'on appelle le **modèle de Markov du premier ordre**.

Si la probabilité de l'état s au temps t dépend des états en t-1 et t-2, on parle de **modèle de Markov de second ordre** . Le modèle de Markov du 2e ordre peut s'écrire : P(s(t)|s(t-1),s(t-2)).

Exemple introductif

- Voici un petit exemple qui a pour but d'introduire quelques notions utiles.

Contexte Imaginons deux urnes, nommées urne 1 et urne 2, dans lesquelles il y a à la fois des boules blanches et noires.

Dans chaque urne nous plaçons un certain nombre de boules blanches et un certain nombre de boules noires. Par exemple, mettons dans l'urne 1 9 boules blanches et 1 boule noire. Dans l'urne 2, on met 5 boules noire et une boule blanche.

L'exemple fonctionne de la manière suivante :

— Si, dans l'urne 1, nous tirons une boule noire, alors nous restons sur l'urne 1. Sinon, nous allons à l'urne 2 ;

— De la même manière, si, dans l'urne 2, on tire une boule noire, on change de sac, et si l'on tire une boule blanche, on reste à l'urne 2 ;

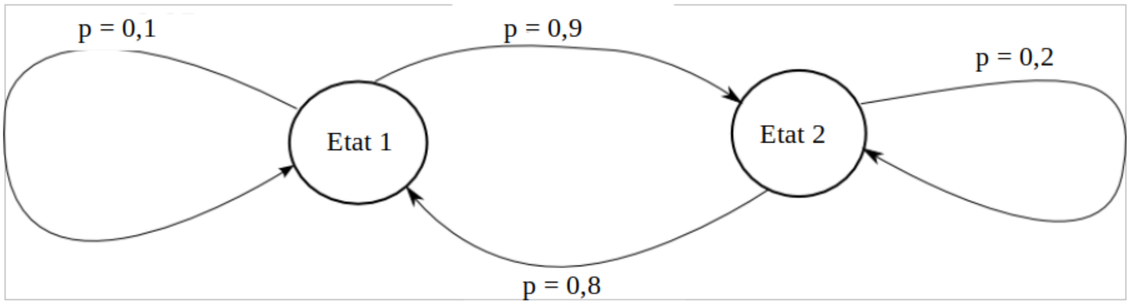
— Les tirages s'effectuent avec remise.

On peut recencer dans un tableau les probabilités de changer ou non d'urne au tirage suivant, suivant l'urne sur laquelle on se trouve :

/	Prochain tirage : Urne 1	Prochain tirage : Urne 2
Tirage : Urne 1	0.1	0.9
Tirage : Urne 2	5/6	1/6

Le fait que l'on se trouve sur l'urne 1 ou l'urne 2 avant un tirage représente un état (notons les respectivement Etat 1 et Etat 2)

On peut modéliser cette expérience par une **Chaine de Markov** que l'on peut représenter avec un **automate de Markov** :



Introduction d'états cachés Reprenons l'exemple précédent en introduisants deux nouvelles urnes : l'urne 1bis et l'urne 2bis telles que :

— L'urne 1bis contient 5 boules bleue et 1 boule rouge ;
— L'urne 2bis contient 1 boule rouge et 5 boules bleues ;

On introduit ces urnes de la manière suivante :

— On commence sur le groupe d'urnes 1/1bis, on tire une boule dans l'urne 1 bis, on note sa couleur et on la remets. ;
— On tire une boule dans l'urne 1 pour savoir dans quel groupe d'urne sera le prochain tirage ;
— On recommence autant que l'on veut ;

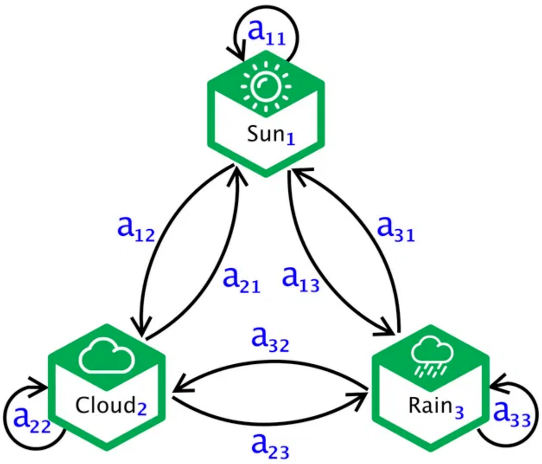
Deux séquences sont générées :

—
— La séquence dite de sortie : celle des boules rouges ou bleues. Celle-ci est **connue**.
— La séquence dite de transition : celle des boules blanches ou noires. Celle-ci est **inconnue**.

On a donc ici des **états cachés**.

Probabilité de transition

La probabilité que l'on passe d'un état à un autre est la probabilité de transition. Imaginons qu'on fait de la météo et qu'il y ait 3 états : Sun, Cloud et Rain. Voici un diagramme résumant les transitions qu'il y a entre ces états :



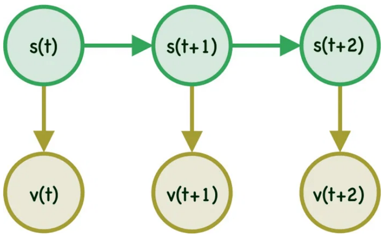
Ici, les termes a_{ij} désignent les **probabilités de transition** : c'est, au moment t+1, la probabilité du système de passer de l'état i à l'état j.

Les probabilités de transition sont définies à l'aide d'une **matrice de transition**, souvent notée A.

Voici la matrice de Transition correspondant au diagramme ci-dessus : $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$

Du modèle de Markov au modèle de Markov caché

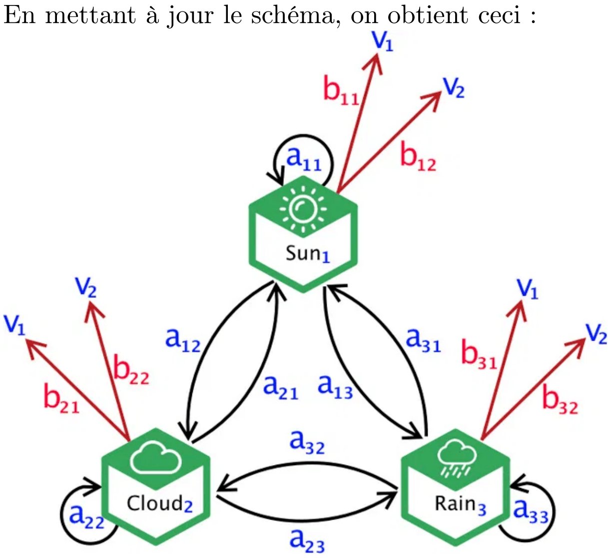
Dans le modèle de Markov à caché, l'état du système est caché. Le système en l'état s(t) émettra un symbole observable v(t). On peut représenter cela avec le schéma suivant :



On appelle cela une **chaine de Markov cachée**

Pobabilités d'émission et matrice d'émission

Reprenons l'exemple de la météo. Supposons que, en fonction du temps qu'il fait, l'humeur d'une personne passe de joyeuse à triste (notons v_1 et v_2). Supposons aussi que nous ne savons aps le temps qu'il fait. Dans ce cas, la météo est l'état caché dans le modèle et l'humeur (heureuse ou triste) est le symbole visible/observable. En mettant à jour le schéma, on obtient ceci :



Les **probabilités d'emission** sont notées b_{ij}.

On peut les recenser dans une **matrice d'emission**, souvent notée B, qui, dans ce cas, est la suivante : B $= \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$

Chapitre 2

Etude bibliographique sur l'application

Le modèle de Markov caché a donc les paramètres suivants :
— un ensemble d'états cachés généralement noté S ;
— Une matrice de probabilité de transition noté A ;
— Une séquence d'observations noté V_T ;
— Une matrice de probabilité d'émission notée B ;
— Une distribution initiale de probabilités, généralement notée π ;

Un premier défi est d'estimer $P(V_T|\theta)$, θ étant le modèle connu constitué des éléments listés ci-dessus. Une solution est la suivante :
— Nous devons d'abord trouver toutes les séquences possibles de l'état S_M où M est le nombre d'états cachés. ;
— Ensuite, à partir de toutes ces séquences de S_M, trouver la probabilité de quelle séquence a généré la séquence visible de symboles V_T via une formule des probabilités totale ;

Afin de calculer la probabilité du modèle généré par la séquence particulière de T symboles visibles V_T, nous devons prendre chaque séquence concevable d'état caché, calculer la probabilité qu'ils aient produit V_T et ensuite additionner ces probabilités.

Phase d'apprentissage

Lors de l'application du modèle de Markov caché, le nombre d'états cachés n'est pas en général connus. On peut d'essayer différentes options, mais cela peut entraîner une augmentation du temps de calcul. C'est pourquoi souvent, un nombre spécifique d'états caché est utilisé pour entrainer le modèle.

Il existe des algorithmes pour effectuer l'apprentissage du modèle de Markov caché, dont l'algorithme de Baum-Welch . C'est celui-ci que j'ai choisi d'implémenter. C'est un algorithme itératif qui vise à préciser les valeurs des matrices d'émission et de transition.

Chapitre 3

Présentation du jeu de données choisi : La consommation électrique de la France de 2012 à 2020

Le jeu de données choisi recense, de 2012 à 2020, la consommation électrique de la France par tranche de 30 minutes. Celui-ci est disponible sur le site data.gouv.fr. Les données de 2020 sont encore parcellaires, c'est pourquoi elles n'entreront pas dans cette étude.

Près de 157000 valeurs de consommation semi-horaire d'électricité sont disponibles dans ce jeu de données. J'ai jugé ce jeu de données suffisamment grand pour que l'on puisse utiliser une partie pour l'apprentissage, puis le reste pour la prédiction. En effet, l'idée est d'utiliser les données de 2012 à par exemple 2018 pour la phase d'apprentissage, puis de conserver les données de 2019 pour tirer ds conclusions sur la précision des prédictions.

Concernant la courbe de consommation, on peut relever plusieurs motifs réccurents. Tout d'abord, un motif d'une période d'environs 24h qui correspond au cycle jour/nuit. Ensuite, en prenant une échelle de temps plus large, il y a plus légèrement une variation sur l'année. Enfin, il y a des variations réccurentes à l'échelle de la semaine.

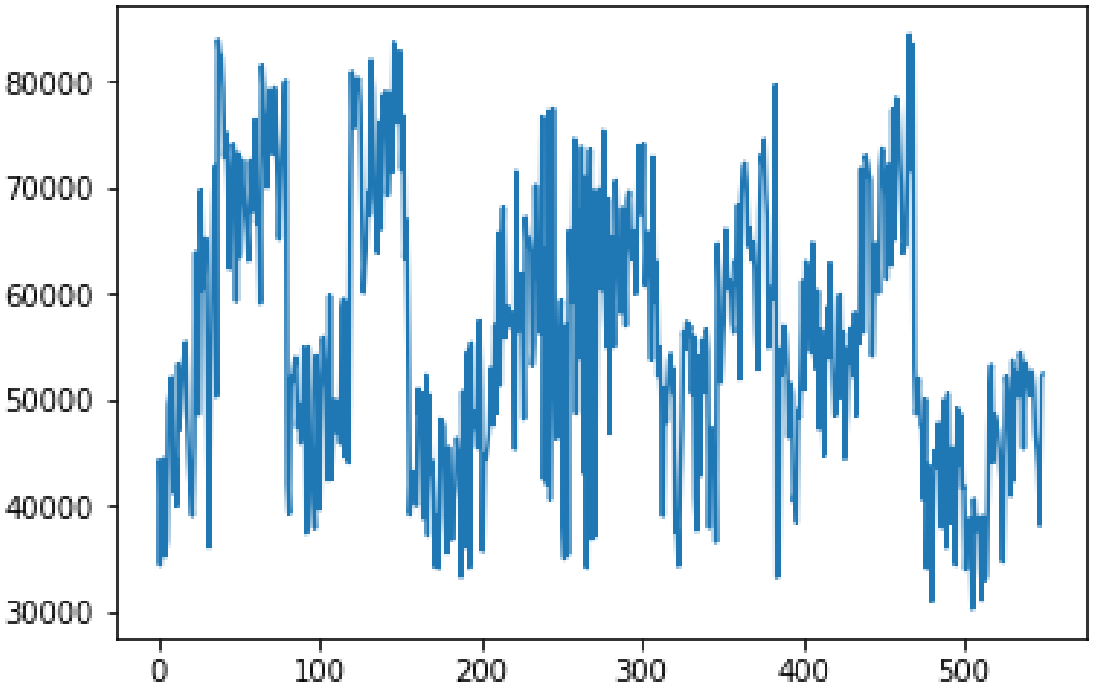


FIGURE 3.1 – Extrait du jeu de données : motifs d'une période de 1 jour

Chapitre 4

Choix des hyper-paramètres et implémentation

Dans un premier temps, il a fallu définir ce qu'était les états dits visibles. D'abord, on s'intéresse simplement aux valeur brutes, en les discrétisant pour avoir un nombre fini d'états possibles. Aussi, on pouvais s'intéresser à la dérivée, et ainsi ajouter des états relatifs aux faits que la tendance soit, par exemple, plutôt à la hausse ou à la baisse.

Si cette méthode était suivi, le dilemme était d'ajuster la précision avec laquelle on discrétisait les valeurs. Par exemple, si l'on regroupais les valeurs de consommation dans m classes et les valeurs de la dérivée dans n classes, il y aurait donc n*m états visibles possibles, ce qui peut poser des problèmes de complexité.

Un premier choix a été fait de faire 40 états visibles différents :

— L'axe des ordonnées a d'abord été découpé en 10 : on a découpé les valeur entre 0 et 100000 en 10 classes (on a donc un pas de 10000) ;

— la fonction dérivée est divisée en 4 classes selon la tendance : très baissière, légèrement baissière, légèrement haussière, très haussière. Si la dérivée est négative, la tendance est baissière, et inversement. La valeur de la dérivée qui sépare légèrement (haussière ou baissière) et très (haussière ou baissière sera déterminée plus tard. ;

J'ai fait le choix de calculer la dérivée sur les 5 dernières valeurs. En effet, j'ai fait une **régression linéaire** sur ces valeurs, car la dérivée est très sensible si on la calcule juste sur les 2 dernières valeurs. Par ailleurs, cela ne pose pas de problème de complexité, étant donné que le nombre de régression linéaires effectuées est en o(n), n étant la taille de l'échantillon (environs 150000).

On aurait également pu s'intéresser à la dérivée seconde, c'est-à-dire au pendant de "l'accélération". Cela aurait cependant encore ajouté des états supplémentaires.

Remarque : La dérivée étant calculée sur les 5 derniers états, on ne peut pas rigoureusement parler de Modèle de Markov d'ordre 1, car des états antérieurs sont nécéssaires. Cependant, 5 valeurs étant négligeable devant la taille de l'échantillon, j'ai jugé que faire cette approximation était acceptable.

Concernant le nombre d'états cachés, je n'ai pas trouvé de réelle méthode pour déterminer ce nombre. Etant donné qu'on avait approximativement 2 états pour les jours/nuits, 2 pour les périodes de la semaines, et 2 par an (été/hiver), j'ai approximé la multiplication de cela à 10.

La matrice de Transition était donc de dimension 10*10. Concernant les valeur initials, j'ai simplement choisi de les équirépartir.

Voici la matrice de transition initiale (avant entrainement) : $$\begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

Il y a 40 états visibles et 10 états cachés, la matrice d'émission sera donc de dimension 40*10. Concernant les probabilités dans la matrice d'émission , j'ai aussi choisi de les équirépartir.

Voici donc la matrice d'émission initiale (avant entrainement) :

$$\begin{pmatrix} 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & ... & 1/40 & 1/40 & 1/40 & 1/40 \end{pmatrix}$$

De la même manière, il a fallu définir un vecteur initial de distribution de probabilité, noté π. J'ai aussi fait le choix simple d'y répartir les probabilités de manière équiprobables :

$$\pi = \begin{pmatrix} 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 \end{pmatrix}$$

Concernant l'implémentation de Baum-Welsch, après de nombreux essais d'implémentation, qui ne fonctionnaient pas comme souhaité, je me suis rabbatu sur une implémentation standard de Baum-Welsch disponible en open-source, que j'ai traduit en Python et adapté à mes besoins.

Voici l'implémenation finale de l'algorithme de Baum-Welch. :

```
def fonction_Baum_Welch(V, A, B, distribution_initiale, nb_iter=1000):

    L = len(V)
    Mat = a.shape[0]


    for n in range(n_iter):
        val1 = forward(V, A, B, distribution_initiale)
        val2 = backward(V, A, B)

        xi = np.zeros((Mat, Mat, L - 1))
        for t in range(L - 1):
            den = np.dot(np.dot(val1[t, :].L, A) * B[:, V[t + 1]].L, val2[t + 1, :])
            for i in range(M):
                num = val1[t, i] * A[i, :] * B[:, V[t + 1]].L * val2[t + 1, :].L
                xi[i, :, t] = num / den

        val3 = np.sum(xi, axis=1)
        A = np.sum(xi, 2) / np.sum(val3, axis=1).reshape((-1, 1))

        #On additionne le Lème élément dans val3
        val3 = np.hstack((val3, np.sum(xi[:, :, L - 2], axis=0).reshape((-1, 1))))

        K = B.shape[1]
        den = np.sum(val3, axis=1)
        for l in range(K):
            B[:, l] = np.sum(val3[:, V == l], axis=1)

        B = np.divide(b, den.reshape((-1, 1)))
```

Cette fonction s'appuie sur l'algorithme dit "avant-arrière", donc les fonctions "avant" et "arrière" (nommée forward et backward) sont implémentées à part. C'est un algorithme itératif, le nombre d'itérations étant ici fixé par le paramètre nb-iter. Cet algorithme a notamment pour objectif de préciser les fonctions A (Matrice de transition) et B(Matrice d'émission). Enfin, j'ai utilisé l'algorithme de Viterbi pour la phase de prédiction.

Chapitre 5

Interprétation

Au vu de la complexité du jeu de donnée en entrée, les matrices de transition obtenues sont difficiles à analyser.

Par ailleurs, concernant la prédiction,on imagine graphiquement une corrélation entre les valeurs prédites et les valeurs réelles, mais il reste de l'erreur (quantifiable). Cela peut être du à plusieurs choses :

— La part d'aléatoire dans les données est peut être plus importante que prévue.

— Il peut y avoir des événements inédits dans la partie prédite qui causent une marge d'erreur.

J'ai trouvé ce projet très intéressant, bien qu'ambitieux : en effet, avec du recul, j'aurai plutôt fait le modèle de Markov simple.