TP Chaînes de Markov et Modèles de Markov Cachés

Pierre Chambet

November 26, 2025

1 Introduction

2 Markov modelling of rainy and dry periods

In this first part we model the alternation between dry and rainy days by a two–state Markov chain. The goal is to (i) make the link between the mathematical model and a concrete simulation, (ii) study the distribution of dry/rainy spell lengths, and (iii) compare the model with the measured data provided in `RR5MN.mat`.

2.1 Definition of the Markov chain (Questions 1–2)

We consider a discrete–time stochastic process $(X_t)_{t\geq 0}$ taking values in the state space

$$\mathcal{E} = \{E_0, E_1\} = \{\text{dry}, \text{rain}\}.$$

By assumption (X_t) is a homogeneous Markov chain:

$$\mathbb{P}(X_{t+1} = j \mid X_t = i, X_{t-1}, \ldots, X_0) = \mathbb{P}(X_{t+1} = j \mid X_t = i) \quad \text{for all } i, j \in \{0, 1\}.$$

The model is parametrised by three probabilities:

- α: probability to remain in the same dry state, $\mathbb{P}(\text{dry}_{t+1} \mid \text{dry}_t) = \alpha$;
- β: probability to remain in the same rainy state, $\mathbb{P}(\text{rain}_{t+1} \mid \text{rain}_t) = \beta$;
- γ: initial probability of a dry day, $\mathbb{P}(X_0 = \text{dry}) = \gamma$.

With this convention, the transition matrix A of the chain is

$$A = \begin{pmatrix} \mathbb{P}(E_0 \to E_0) & \mathbb{P}(E_0 \to E_1) \\ \mathbb{P}(E_1 \to E_0) & \mathbb{P}(E_1 \to E_1) \end{pmatrix} = \begin{pmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{pmatrix}.$$

The initial distribution is

$$\boldsymbol{\pi}_0 = \big(\mathbb{P}(X_0 = E_0),\, \mathbb{P}(X_0 = E_1)\big) = (\gamma,\, 1-\gamma).$$

In the graphical representation (Question 2), the chain can be drawn as two nodes "dry" and "rain" with four directed edges labelled by the transition probabilities:

$$\text{dry} \xrightarrow[\text{stay dry}]{\alpha} \text{dry}, \qquad \text{dry} \xrightarrow[\text{becomes rainy}]{1-\alpha} \text{rain},$$

$$\text{rain} \xrightarrow[\text{becomes dry}]{1-\beta} \text{dry}, \qquad \text{rain} \xrightarrow[\text{stay rainy}]{\beta} \text{rain}.$$

A zero entry in the matrix A would correspond to a *forbidden transition*: an arrow that does not exist in the graph.

In this first exercise we assume that the observation is equal to the state: on each day we observe $Y_t = X_t$, so no extra emission probabilities are needed.

2.2 Stationary probability of a rainy day (Question 3)

We are interested in the long–term probability of being in the rainy state, that is the second component of the stationary distribution $\boldsymbol{\pi}^\star = (\pi_0^\star, \pi_1^\star)$ satisfying

$$\boldsymbol{\pi}^\star = \boldsymbol{\pi}^\star A, \qquad \pi_0^\star + \pi_1^\star = 1.$$

Writing the first equation componentwise gives

$$\pi_0^\star = \pi_0^\star \alpha + \pi_1^\star (1-\beta), \qquad \pi_1^\star = \pi_0^\star (1-\alpha) + \pi_1^\star \beta.$$

Using the constraint $\pi_0^\star = 1 - \pi_1^\star$ and substituting into the second equation yields

$$\begin{aligned} \pi_1^\star &= (1-\pi_1^\star)(1-\alpha) + \pi_1^\star \beta \\ &= (1-\alpha) - (1-\alpha)\,\pi_1^\star + \beta \pi_1^\star. \end{aligned}$$

We regroup all terms containing $\pi_1^\star$ on the left:

$$\pi_1^\star + (1-\alpha)\pi_1^\star - \beta \pi_1^\star = 1 - \alpha,$$

so that

$$\pi_1^\star \big(1 + 1 - \alpha - \beta\big) = 1 - \alpha \quad \Longrightarrow \quad \pi_1^\star = \frac{1-\alpha}{2-\alpha-\beta}.$$

If we instead parameterise the chain using the transition probabilities $p = \mathbb{P}(\text{dry} \to \text{rain})$ and $q = \mathbb{P}(\text{rain} \to \text{dry})$, that is

$$A = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix},$$

then the same computation gives the well–known formula

$$\mathbb{P}(X_t = \text{rain in stationarity}) = \pi_1^\star = \frac{p}{p+q}.$$

2.3 Simulation of the chain (Question 4)

To check the theoretical value of the stationary rainfall probability, we simulate a long trajectory of the Markov chain and estimate the empirical frequency of rainy days.

We use a pure NumPy implementation of the Markov chain; because the observation is equal to the state, we only simulate (X_t).

For a large number of samples (`Nsamples` in the order of 10^4–10^5), the empirical frequency of rainy days `freq_rain_emp` is very close to the theoretical stationary probability $\pi_1^\star$ computed in the previous subsection.

2.4 Distribution of dry and rainy spell lengths (Question 5)

A key advantage of the Markov model is that it gives a simple description of the *spell lengths*, that is, the duration of consecutive dry or rainy periods.

Let D_{rain} denote the length, in time steps, of a rainy spell. As soon as the chain enters state "rain", it will remain in that state until it transitions back to "dry". If we use the parametrisation

$$A = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix},$$

then a rainy spell ends when the transition rain→dry occurs, which happens with probability q at each step. It follows that D_{rain} has a geometric distribution:

$$\mathbb{P}(D_{\text{rain}} = k) = (1-q)^{k-1}\, q, \qquad k \in \{1, 2, 3, \dots\}.$$

Similarly the dry spell length D_{dry} is geometrically distributed with parameter p:

$$\mathbb{P}(D_{\text{dry}} = k) = (1-p)^{k-1}\, p.$$

The corresponding expectations are

$$\mathbb{E}[D_{\text{rain}}] = \frac{1}{q}, \qquad \mathbb{E}[D_{\text{dry}}] = \frac{1}{p}.$$

On the simulated sequence `obs_sim` we use the provided function `duree` to extract the durations of dry and rainy spells.

We can compare the empirical mean and variance of D_{rain} and D_{dry} to the theoretical values $1/q$ and $1/p$. We can also compare the full empirical distributions to the geometric law on a semi–log plot.

To plot the empirical probability mass functions, the helper `duree` returns normalised histograms `pdfSec` and `pdfRain` together with the bin edges. Since the edges have one more element than the pdf, we use bin centres on the x–axis.

2.5 Comparison with measured data (Question 6)

Finally we repeat the spell–length analysis on the measured data contained in `RR5MN.mat`. The file provides a vector `Support` of zeros and ones indicating dry and rainy days. We load it and apply the same `duree` function.

We then superimpose the simulated and measured spell–length distributions. Using bin centres for both histograms.

On the plots we observe that the simple two–state Markov chain with fixed parameters does not perfectly reproduce the empirical distributions: for instance, in our experiments the model tends to produce longer rainy spells than those observed in the real data. This suggests that, while the Markov chain is a useful first approximation, a more refined model (for example with parameters estimated from the data, or with additional hidden states) would be needed for a more accurate fit.

3 Partie 2 : HMM à 3 états pour la pluie

3.1 Définition et Graphe du modèle (Question 1)

Dans ce modèle, nous introduisons une couche cachée représentant l'état du ciel, qui ne peut pas être observé directement, mais qui influence la probabilité de pluie (l'observation). Les états sont : E_0 (Ciel Clair), E_1 (Nuageux), E_2 (Très Nuageux).

Les matrices données sont :

$$T = \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0.5 & 0.4 & 0.1 \\ 0 & 0.35 & 0.65 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0.8 & 0.2 \\ 0 & 1 \end{pmatrix}$$

Interprétation physique des zéros :

- **Dans la matrice de transition T :** La valeur $T_{0,2} = 0$ (1ère ligne, 3ème colonne) signifie qu'il est impossible de passer directement d'un "Ciel Clair" (E_0) à un ciel "Très Nuageux" (E_2) en un seul pas de temps (5 minutes). Le système doit obligatoirement transiter par l'état intermédiaire "Nuageux" (E_1). Cela traduit une inertie physique de la météorologie.
- **Dans la matrice d'émission B :**
  - $B_{0,1} = 0$: Un ciel clair ne produit jamais de pluie.
  - $B_{2,0} = 0$: Un ciel très nuageux produit toujours de la pluie (dans ce modèle simplifié).

3.2 Apprentissage des paramètres (Baum-Welch)

L'algorithme de Baum-Welch permet d'ajuster les matrices T et B pour maximiser la vraisemblance des observations réelles. [Insérer ici les nouvelles matrices obtenues après l'exécution du code 'model.fit()'].

On constate généralement que l'apprentissage tend à ajuster les probabilités de transition pour coller à la rareté des épisodes pluvieux réels.

3.3 Définition du modèle

On considère un HMM avec $K = 3$ états cachés E_0, E_1, E_2 et des observations binaires $Y_t \in \{0, 1\}$ (0 = sec, 1 = pluie). Le modèle est entièrement caractérisé par :

- la matrice de transition entre états cachés T (taille 3
$times3$), - la matrice d'émission B (taille 3
$times2$) donnant $P(Y = 0|E_i)$ et $P(Y = 1|E_i)$ pour chaque état i, - la distribution initiale
pi_0 sur les états cachés.

En notation matricielle :

$$T = (T_{ij})_{0lei,j<3}, qquadB = (B_{i,y})_{0lei<3,;yin0,1},$$

avec
$sum_j T_{ij} = 1$ et
$sum_y B_{i,y} = 1$ pour tout i.

3.4 Estimation par Baum–Welch (EM)

Le HMM est entraîné sur la série mesurée par l'algorithme de Baum–Welch (une variante EM pour HMM). En pratique, on choisit une initialisation de T, B et
pi_0 et on itère les étapes E (calcul des postérieurs) et M (mise à jour des paramètres) jusqu'à convergence.

Remarque pratique : selon la version de la bibliothèque, il peut être nécessaire d'assurer la compatibilité des types (float32 vs float64) pour éviter des erreurs numériques lors du calcul forward/backward.

3.5 Extraction des paramètres appris

Après apprentissage on récupère les paramètres estimés T^*, B^*,
pi_0^* à partir des attributs du modèle (en adaptant la conversion si la librairie utilise des tenseurs internes). Dans le notebook nous avons utilisé une extraction robuste et obtenu, à titre indicatif, les estimations suivantes (approximatives) :

```
T^*  [[0.97236, 0.02756, 0.000085],
      [0.47757, 0.51236, 0.01008],
      [0.01533, 0.06998, 0.91469]]

B^*  [[1.0,      0.0     ],
      [0.94864,  0.05136 ],
      [0.04245,  0.95755 ]]

_0^*  [0.96168, 0.03832, ~0]
```

Ces résultats s'interprètent ainsi : l'état 0 ("Clear") émet presque toujours des observations 0 (sec), l'état 2 ("VeryCloudy") émet presque toujours pluie, et l'état 1 est intermédiaire.

3.6 Interprétation et pistes d'amélioration

L'écart observé entre modèle et données peut venir de plusieurs sources :

- l'initialisation des paramètres et la convergence vers un minimum local (Baum–Welch est sensible aux conditions initiales) ; - le choix de trois états peut être insuffisant pour capter toutes les régimes météorologiques pertinents ; - le modèle HMM avec émissions conditionnelles indépendantes peut ne pas capturer certaines dépendances temporelles fines.

Actions recommandées :

- effectuer plusieurs ré-initialisations aléatoires de Baum–Welch et garder le modèle avec la plus grande log-vraisemblance ; - essayer un HMM avec plus d'états ou des lois d'émission plus riches (par ex. un modèle autorégressif ou des émissions conditionnées par des covariables) ; - calibrer le modèle sur des sous-périodes et comparer la stabilité des paramètres.

Dans le notebook j'ai laissé les cellules nécessaires pour reproduire les étapes : construction du modèle, ajustement, extraction des paramètres, simulation et tracés comparatifs des durées.

4 Partie 3 : Reconnaissance de mots par HMM gaussiens

Cette partie presente les idees et la procedure mises en oeuvre pour explorer la reconnaissance de mots courts a partir de caracteristiques audio, en utilisant des modeles de Markov caches a emissions gaussiennes (Gaussian HMM). L'objectif est d'expliquer clairement le pipeline, les choix methodologiques et les mesures d'evaluation, de facon a rendre reproductible l'experience meme si l'ensemble des essais n'est pas complet dans le depot.

4.1 Pipeline general

Le pipeline experimental typique du notebook est le suivant :

- Acquisition des fichiers audio et etiquetage (un dossier par mot). - Pretraitement: normalisation, suppression des silences, fenetrage (frame), calcul de caracteristiques (MFCC, energies, deltas). - Agrégation: chaque enregistrement est represente par une sequence de vecteurs de dimension d (ex. d = 13 MFCC + deltas). - Modele: pour chaque mot on entraine un HMM gaussien (etats cachés K), chaque etat ayant une loi normale multivariate (moyenne, covariance). - Classification: pour une sequence de test, calculer la log-vraisemblance sous chaque modele mot et choisir le mot de score maximum. - Evaluation: matrice de confusion, accuracy, precision/recall et F1, eventuellement cross-validation en K-fold.

4.2 Caracteristiques audio et pretraitement

Les caracteristiques utilisees dans le notebook sont les MFCC (Mel Frequency Cepstral Coefficients), eventuellement enrichies par leurs derivees (delta, delta-delta). Points pratiques :

- echantillonnage: resampler a une frequence commune si necessaire; - normalisation par enregistrement (zero-mean, unit-variance) avant l'entraînement des HMM ; - trimming / gate du silence pour eliminer les segments tres courts contenant du bruit ; - parametrage du fenetrage: taille de frame (ex. 25 ms), pas (10 ms).

Ces etapes reduisent la variabilite non pertinente et aident les HMM a apprendre des transitions d'etats representant la structure temporelle du mot.

4.3 Modele: HMM gaussien par mot

Chaque mot est represente par un HMM ou chaque etat a une densite gaussienne multivariate. Le nombre d'etats K est un hyper-parametre choisi par l'experience (par ex. K entre 3 et 8 selon la duree du mot). L'entrainement est effectue par l'algorithme Baum–Welch (EM) sur les sequences MFCC du mot. Dans le notebook la logique d'entraînement est similaire a :

```
# pseudo-code: entrainement d'un HMM gaussien pour un mot
for word in vocabulary:
    X_word = load_sequences(word)  # list of (T_i x d) arrays
    model = GaussianHMM(n_states=K, cov_type='diag')
    model.fit(X_word)
    save_model(word, model)
```

Remarques pratiques :

- l'initialisation (k-means sur les frames) ameliore souvent la convergence du EM ; - utiliser des covariances diagonales est plus robuste sur peu de donnees; - normaliser les caracteristiques avant l'apprentissage permet d'eviter que certaines dimensions dominent le calcul de log-probabilites.

4.4 Classification et mesure de performance

Pour evaluer un modele supervise par mot on procede comme suit :

- Pour chaque sequence de test, calculer la log-vraisemblance sous chaque modele mot : $\text{score}_word = model_word.score(X_test).-Predirelemotayantlescorelepluseleve.-Construirelamatricedeconfusionetcalculerlesmetriques : accuracy, precision, recall, F1-score(macroetmicrosiplusieursclasses).Lenotebookcontientdesfonctionspourafficherlamatricedec$

Pour obtenir des estimations robustes, on evalue aussi le modele par K-fold cross-validation : partitionner les enregistrements par mot en K folds, re-entrainer sur K-1 folds et tester sur le fold restant, repetee K fois.

4.5 Resultats attendus et limites

Avec un jeu de donnees restreint et des HMM simples on attend :

- des performances correctes sur des mots bien distincts (ex. "apple" vs "banana") ; - confusions entre mots proches acoustiquement ou tres courts ; - sensibilite a la quantite de donnees par classe et a la qualite du pretraitement (silence, bruit).

Pistes d'amelioration :

- utiliser des GMM-HMM (mixture de gaussiennes par etat) si plus de flexibilite est necessaire ; - augmenter les donnees (data augmentation : bruit, decalage temporel) ; - tester des architectures alternatives (HMM conditionnels, modele hybride HMM+NN pour emissions) ; - selection d'hyperparametres (nombre d'etats K, type de covariance) via validation croisee.

4.6 Outils et reproductibilite

Le notebook 'notebooks/part3$_hmm_audio.ipynb'contientlescellulespourextrairelesMFCC, entrainer$

Si vous voulez, je peux : (A) lancer maintenant les cellules d'entrainement et d'evaluation (long selon taille du dataset), ou (B) executer une compilation LaTeX rapide du rapport pour verifier l'insertion. Dites ce que vous preferez.

5 Conclusion