Feuille de TP n°9 – AR(1), Algorithme EM, Markov caché

1 Processus auto-régressif d'ordre 1 AR(1)

On considère le processus $(X_n)_{n\in\mathbb{N}}$ défini par la donnée de X_0 de loi $\mathcal{N}(m,\alpha^2)$ et la relation de récurrence

$$X_{n+1} = aX_n + \sigma Y_{n+1}, \tag{1}$$

où les v.a. $(Y_n)_{n\geq 1}$ sont i.i.d. de loi gaussienne centrée réduite et indépendantes de X_0.

Remarque 1. Ce processus permet entre autres de modéliser :
- l'écart entre trajectoires théorique et réelle pour le filtre de Kalman (on peut grâce au GPS mesurer très précisément X et vouloir estimer les paramètres a et σ qui quantifient les propriétés de l'avion et du pilote d'une part et les conditions de vol d'autre part) ;
- la vitesse d'une particule de pollen subissant les chocs des molécules d'eau et une force de frottement ;
- l'évolution de prix d'actions (ou le logarithme du rapport des prix entre deux dates).

Proposition 2. *Les estimateurs obtenus par la méthode du maximum de vraisemblance sont*

$$\begin{aligned}\hat{a}_n &= \frac{\sum_{k=1}^n X_{k-1}X_k}{\sum_{k=1}^n X_{k-1}^2},\\ \hat{\sigma}_n^2 &= \frac{1}{n}\sum_{k=1}^n (X_k - \hat{a}_n X_{k-1})^2.\end{aligned}$$

►► Montrer que la loi de X_n est une v.a. gaussienne et que X_n converge vers une loi que l'on précisera.
►► Illustrer la convergence presque sûre de ces estimateurs.
►► Illustrer le fait que $\sqrt{n}(\hat{a}_n - a)$ converge en loi vers une variable aléatoire gaussienne dont on précisera les paramètres en fonction de a et σ.
►► Illustrer le fait que $\sqrt{n}(\hat{\sigma}_n^2 - \sigma^2)$ converge en loi vers une variable aléatoire gaussienne dont on précisera les paramètres en fonction de a et σ.

Étant donnée une suite $(Y_n)_{n\in\mathbb{Z}}$ de v.a. i.i.d. de loi gaussienne centrée réduite, on peut définir le processus

$$X_n = \sigma \sum_{k=-\infty}^{n} a^{n-k} Y_k.$$

►► Montrer que la loi de X_n ne dépend pas de n et que la suite $(X_n)_{n\in\mathbb{Z}}$ vérifie (1). On parle de processus stationnaire[1].

[1] On pourra se repporter à [DCD83].

2 Algorithme EM

On dispose d'un échantillon $(X_i)_{1\leq i\leq n}$ de v.a. de densité

$$\sum_{j=1}^{J}\alpha(j)\gamma_{m(j),\sigma(j)^2}(x)=\sum_{j=1}^{J}\frac{\alpha(j)}{\sqrt{2\pi\sigma(j)^2}}\exp\left[-\frac{(x-m(j))^2}{2\sigma(j)^2}\right],$$

avec, pour tout $j=1,\ldots,J$, $\alpha_j\geq 0$, et $\alpha_1+\cdots+\alpha_J=1$ et γ_{m,σ^2} désigne la densité de la loi $\mathcal{N}(m,\sigma^2)$.

On ne peut pas calculer explicitement un estimateur du maximum de vraisemblance. On utilise un algorithme appelé algorithme EM qui s'apparente à une méthode de descente (ou de montée) pour déterminer un extremum de la vraisemblance[2].

Étant données les observations $X_1,\ldots,X_n$ et des valeurs initiales des paramètres, l'algorithme consiste à répéter les calculs suivants.

– Étant donnés à l'étape k, les trois vecteurs ligne

$$\alpha_k=(\alpha_k(1),\ldots,\alpha_k(J)),\quad m_k=(m_k(1),\ldots,m_k(J))\quad\text{et}\quad v_k=(v_k(1),\ldots,v_k(J)),$$

(v pour variance), on définit la matrice $H^{(k)}$ de taille $n\times j$ par

$$H_{ij}^{(k)}=\frac{\alpha_k(j)\gamma_{m_k(j),v_k(j)}(X_i)}{\sum_{l=1}^{J}\alpha_k(l)\gamma_{m_k(l),v_k(l)}(X_i)}.$$

– Étant donnée $H^{(k)}$, on obtient α_{k+1}, m_{k+1} et v_{k+1} par les relations suivantes :

$$\begin{aligned}\alpha_{k+1}(j) &= \frac{1}{n}\sum_{i=1}^{n}H_{ij}^{(k)}\\ m_{k+1}(j) &= \frac{\sum_{i=1}^{n}X_iH_{ij}^{(k)}}{\sum_{i=1}^{n}H_{ij}^{(k)}}\\ v_{k+1}(j) &= \frac{\sum_{i=1}^{n}(X_i-m_j)^2H_{ij}^{(k)}}{\sum_{i=1}^{n}H_{ij}^{(k)}}.\end{aligned}$$

Remarque 3. Contrairement au cas où l'on observe en même temps X et Z, il peut exister des maxima locaux qui vont piéger l'algorithme. Cette procédure peut s'avérer très sensible au point de départ choisi.

►► En utilisant la fonction `em-aide.sci`, implémenter l'algorithme EM (il ne manque que le coeur du programme à implémenter).

[2] Voir le texte *Algorithme EM.*

3 Chaîne de Markov cachée

On considère la chaîne de Markov (X_n, U_n) sur $\mathcal{A} \times \mathcal{U} = \{1,2,3,4\} \times \{1,2\}$ définie par les relations suivantes :

$$\begin{aligned}
&\mathbb{P}(X_1 = x_1, \dots, X_l = x_l, U_1 = u_1, \dots, U_l = u_l) \\
&\quad = \mathbb{P}(U_1 = u_1, \dots, U_l = u_l)\mathbb{P}(X_1 = x_1, \dots, X_l = x_l | U_1 = u_1, \dots, U_l = u_l) \\
&\quad = \nu(u_1) \prod_{i=2}^{l} \rho(u_{i-1}, u_i) \mu_{u_1}(x_1) \prod_{i=2}^{l} \pi_{u_i}(x_{i-1}, x_i),
\end{aligned}$$

où $(\rho(u,v))_{u,v\in\mathcal{U}}$ est la matrice de transition de U, ν sa mesure initiale, pour tout $u \in \mathcal{U}$, $(\pi_u(i,j))_{i,j\in\mathcal{A}}$ est une matrice de transition sur $\mathcal{A}$ et μ_u est la mesure initiale de X sachant que $U_1 = u$. On prend les valeurs suivantes pour les paramètres du modèle :

$$\rho = \begin{pmatrix} .99 & .01 \\ .02 & .98 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} .3 & .3 & .3 & .1 \\ .3 & .3 & .1 & .3 \\ .3 & .1 & .3 & .3 \\ .1 & .3 & .3 & .3 \end{pmatrix} \quad \text{et} \quad \pi_2 = \begin{pmatrix} .5 & .3 & .1 & .1 \\ .4 & .4 & .1 & .1 \\ .4 & .1 & .4 & .1 \\ .5 & .3 & .1 & .1 \end{pmatrix}$$

Il s'agit donc de calculer, connaissant les matrices de transition, les probabilités[3]

$$\forall v \in \mathcal{U}, \quad \mathbb{P}(U_i = v | X_1 = x_1, \dots, X_l = x_l).$$

On note :
- $P^i(v) := \mathbb{P}(U_i = v | X_1 = x_1, \dots, X_{i-1} = x_{i-1})$ les probabilités de prédiction,
- $F^i(v) := \mathbb{P}(U_i = v | X_1 = x_1, \dots, X_i = x_i)$ les probabilités de filtrage,
- $L^i(v) := \mathbb{P}(U_i = v | X_1 = x_1, \dots, X_l = x_l)$ les probabilités de lissage.

Proposition 4. *On a*

$$\forall i = 1, \dots, l \quad F^i(v) = \frac{\pi_v(x_{i-1}, x_i) P^i(v)}{\sum_{u\in\mathcal{U}} \pi_u(x_{i-1}, x_i) P^i(u)}, \tag{2}$$

$$\forall i = 2, \dots, l \quad P^i(v) = \sum_{u\in\mathcal{U}} \rho(u,v) F^{i-1}(u), \tag{3}$$

$$\forall i = 1, \dots, l \quad L^{i-1}(u) = F^{i-1}(u) \sum_{v\in\mathcal{U}} \rho(u,v) \frac{L^i(v)}{P^i(v)}. \tag{4}$$

L'algorithme est dit *forward-backward* :
- on initialise l'algorithme en choisissant pour $P^1(u)$ la loi initiale,
- on calcule de proche en proche L^1, P^2, L^2,…,P^l et F^l,
- on calcule par récurrence descendante L^l,…,L^1.

►► À partir de la fonction `adn-aide.sci`, générer une trajectoire de la chaîne et implémenter l'algorithme. Il faut combler les trous matérialisés par le code *A faire*. Pour l'évaluation, on pourra afficher la fréquence de bonnes réponses : une réponse sera bonne lorsque la probabilité $L^i(v)$ sera supérieure à .5 et $U_i = v$.

On pourra consulter [RRS03].

[3] Voir le texte *Recherche de zones codantes dans un brin d'ADN.*

Références

[DCD83] D. Dacunha-Castelle et M. Duflo – *Probabilités et statistiques. Tome 2*, Masson, Paris, 1983, Problèmes à temps mobile.

[RRS03] S. Robin, F. Rodolphe et S. Schbath – *Adn, mots et modèles*, Belin, 2003.