

Programme de colle n°11

Semaine du 05/01

Réduction des matrices

Application aux chaînes de Markov et aux systèmes différentiels

Pour cette colle, ce qui tient lieu d'« exercice étoilé » est de savoir écrire le graphe probabiliste et la matrice de transition d'une chaîne de Markov, et d'en donner les états stables.

Reprise du programme de réduction des matrices.

Chaînes de Markov

- Rappels d'ECG1 sur les graphes : sommets, arêtes, graphes orientés, graphes pondérés.
- Matrice d'adjacence d'un graphe (orienté ou non).
- Graphe probabiliste : c'est un graphe
 - orienté et pondéré ;
 - pour tous $(i, j) \in \llbracket 1, r \rrbracket^2$, on a au plus une arête $i \rightarrow j$;
 - pour tout $i \in \llbracket 1, r \rrbracket$, la somme des poids des arêtes sortant du sommet i est 1.

On notera r le nombre de sommets du graphe probabiliste considéré.

- Matrice de transition d'un graphe probabiliste.
- Définition : matrice stochastique. Une matrice est stochastique ssi c'est la matrice de transition d'un graphe probabiliste. Si M est stochastique, $\mathbf{1} \in \text{Sp}(M)$ et $\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in E_1(M)$.
- Chaîne de Markov associée à un graphe probabiliste de matrice de transition M : c'est une suite de variables aléatoires $(X_n)_{n \in \mathbb{N}}$, à valeurs dans $\llbracket 1, r \rrbracket$, telles que

$$\forall (i, j) \in \llbracket 1, r \rrbracket^2, \mathbb{P}_{(X_n=i)}(X_{n+1}=j) = m_{i,j}$$

où $m_{i,j}$ est le coefficient (i, j) de M (c'est donc le poids de l'arête $i \rightarrow j$).

Interprétation : position au temps n d'un système sans mémoire « se déplaçant aléatoirement » sur le graphe en temps discret.

- On range la loi de X_n dans un vecteur de $\mathcal{M}_{1,r}(\mathbb{R})$:

$$V_n = (\mathbb{P}(X_n=1) \quad \dots \quad \mathbb{P}(X_n=r))$$

On a alors la relation $V_{n+1} = V_n M$. V_n est appelé *n-ième état probabiliste de la chaîne*.

- $\forall n \in \mathbb{N}, V_n = V_0 M^n$.
- État probabiliste stable : c'est un état probabiliste $V = (p_1 \quad \dots \quad p_r)$ (les p_i sont donc positifs et de somme 1) tel que $VM = V$.

Un état $V = (p_1 \quad \dots \quad p_r)$ est stable ssi $\begin{pmatrix} p_1 \\ \vdots \\ p_r \end{pmatrix}$ est vecteur propre de ${}^t M$ pour la valeur propre 1.

Systèmes différentiels linéaires

- Rappels d'ECG1 : résolution, pour $(a, b) \in \mathbb{R}^2$, de $y' + ay = 0$, $y'' + ay' + by = 0$ (lorsque l'équation caractéristique a au moins une racine réelle).
- Résolution d'équations non homogènes (la recherche d'une solution particulière doit être guidée).
- Prise en compte d'une condition initiale.
- Systèmes différentiels $X' = AX$ où $A \in \mathcal{M}_n(\mathbb{R})$ est diagonalisable.

Résolution par diagonalisation de A, en posant $X = PY$; ou en écrivant $X(t) = \sum_{i=1}^n K_i e^{\lambda_i t} U_i$ où $(U_i)_{1 \leq i \leq n}$ est une base de vecteurs propres de A, et K_i des constantes réelles.

- Théorème de Cauchy : existence et unicité d'une solution à un problème de Cauchy (cas scalaire d'ordre 1 ou d'ordre 2 ; cas vectoriel)
- Quelques aspects qualitatifs :
 - trajectoire (l'ensemble des $(x_1(t), \dots, x_n(t))$ pour $t \in \mathbb{R}$)
 - trajectoire convergente (les x_i ont une limite finie pour $t \rightarrow +\infty$)
 - point d'équilibre (ce sont les solutions constantes).

Résultats associés : pour le système (S) : $X' = AX$:

- Les points d'équilibre de (S) sont les éléments de $\text{Ker}(A)$.
- Si $\text{Sp}(A) \subset \mathbb{R}_-$, toute trajectoire converge vers un point d'équilibre.
- Si $\text{Sp}(A) \subset \mathbb{R}_+^*$, toute trajectoire converge vers 0.
- Si A admet une $\text{vap} > 0$ il existe des solutions divergentes.

Python

- Simulation d'expériences aléatoires : pas de commandes spécifiques aux chaînes de Markov mais on peut demander de simuler des expériences aléatoires simples avec `rd.random`, `rd.randint` par exemple.
- Algèbre linéaire : on doit savoir utiliser les outils de `numpy.linalg`, notamment sur la matrice de transition. Ci-dessous les principales fonctions qui peuvent être utilisées :
 - `al.eig` : recherche d'éléments propres.
 - `al.rank` : rang d'une matrice.
 - `al.matrix_power` : mise à la puissance n d'une matrice.
 - `np.dot` : produit matriciel.
 - `np.eye` : matrice identité.
 - `np.transpose` : transposition
- Systèmes différentiels : rien n'a été vu en Python à ce sujet pour l'instant.