Mini projet 3 : Chaine de Markov

Fabien Tang
Valentin Colliard

2018

1.1 Introduction

Lors de ce projet, nous avons cherché à réaliser un ensemble de codes permettant de facilement modéliser et analyser une chaine de Markov à temps et à états discrets.
Remarque : seulement une partie des résultats seront affichés pour ne pas rendre le compte rendu trop lourd (cf. notebooks).

1.2 Définition de cadre de modélisation et de visualisation pour les CdM

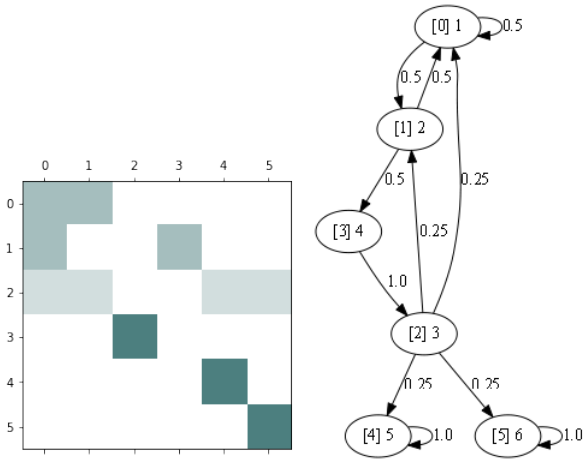
Dans cette première partie, nous avons construit la classe MouseInMaze représentant une CdM modélisant une souris dans un labyrinthe à 6 pièces.
Tout comme la classe FeuRouge (représentant le comportement d'un feu), le code de MouseInMaze implémente la classe abstraite CdM. Cette classe nous permettra par la suite de réaliser des opérations sur une CdM de manière plus simple.

1.2.1 Enrichissement de CdM.py

Afin de rendre plus facilement manipulable une CdM, nous avons défini :
- un attribut stateToIndex traduisant les états à des indices,
- les méthodes distribution_to_vector et vector_to_distribution permettant de passer de la représentation en dictionnaire (distribution) à la représentation vectorielle des probabilités sur les états (et vice-versa),
- la méthode show_distribution affichant une distribution,
- les méthodes get_transition_matrix et show_transition_matrix construisant et affichant la matrice de transition,
- la méthode show_transition_graph dessinant le graphe de transition,
- la méthode get_communication_classes analysant le graphe de transition pour en ressortir les composantes fortement connexes,
- la méthode get_absorbing_classes analysant le graphe de transition pour en ressortir les sous chaines de Markov irréductibles,
- la méthode is_irreducible permettant de savoir si un graphe est irreductible,
- la méthode get_periodicity calculant la périodicité d'une chaîne de Markov,
- la méthode is_aperiodic indiquant si une CdM est apériodique,
- la méthode is_ergodic indiquant si une CdM est ergodique ou non.

1.2.2 Résultats

Mouse In Maze :



Composantes fortement connexes : [5, 6, 1, 2, 3, 4]
Sous-chaines de Markov : [5, 6]
Irréductible : False
Apériodique : True
Périodicité (plus petite période) : 1
Ergodic (Irréductible + Apériodique) : False

1.3 Algorithmes sur les CdM

Dans cette seconde partie, nous avons implémenté 4 algorithmes différents permettant d'étudier le comportement asymptotique d'une CdM.

1.3.1 Simulation d'une CdM

Afin de pouvoir implémenter les différents algorithmes étudiant le comportement d'une CdM, nous devions générer plusieurs séquences d'états suivant les caractéristiques π_0 et $P(X_t|X_{t-1})$.
Nous avons donc mis en place un framework d'échantillonnage de manière générique en se basant sur le design pattern Observer.
Les observers de notre modèle sont des collectors ayant pour but de collecter l'information généré par la CdM et potentiellement d'arrêter la simulation.
Nous avons ainsi défini notre collector :
- CollDistribution calculant la probabilité de chaque état et arrêtant la simulation à la convergence (quand la différence entre 2 distributions de probabilité est inférieure à un epsilon donné)

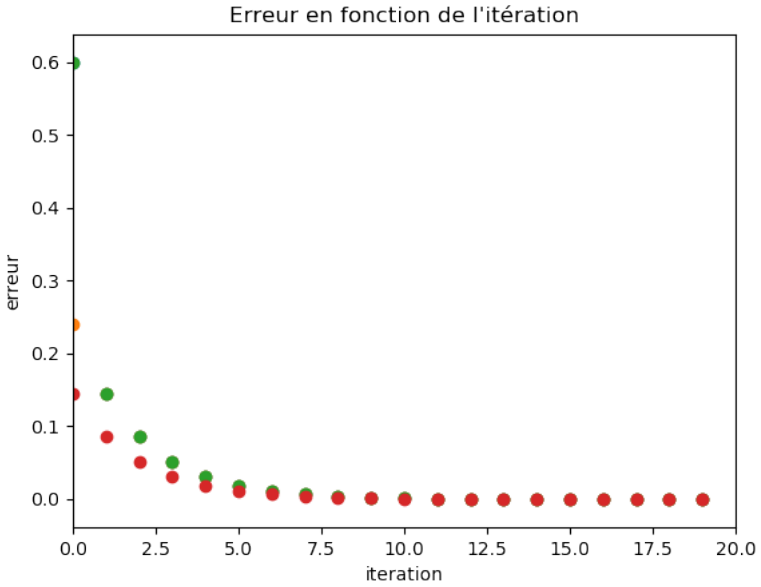
1.3.2 Algorithmes de calcul de $\pi*$

Nous avons défini nos algorithmes sous forme de collector calculant $\pi*$ suivant les différentes méthodes décrites ci-dessous :
- Méthode 1. Convergence de π_n : en itérant l'équation $\pi_{n+1} = \pi_n \cdot M$ et en s'arrêtant quand la distance entre π_n et π_{n+1} est assez faible.
- Méthode 2. Convergence de M^n : la suite des puissances de M.
- Méthode 3. Point fixe : π^* est un point fixe et vérifie $\pi^* = \pi^* \cdot M$.
π^* est donc un vecteur propre de M pour la valeur propre 1.
- Méthode 4. $\pi_{n+1} = \pi_t \cdot M = \pi_0 \cdot M^n$

1.3.3 Résultats

MonoBestiole (15,0.4,0.6)

Méthode	Durée	Nb iteration	Distribution
1 (bleu)	0.17	20	{1 : 0.52, 2 : 0.23, 3 : 0.04, 4 : 0.09, 5 : 0.09}
2 (orange)	0.12	20	{1 : 0.19, 2 : 0.14, 3 : 0.09, 4 : 0.09, 5 : 0.09, 6 : 0.04, 7 : 0.04, 8 : 0.04, 9 : 0.09, 10 : 0.09, 11 : 0.04}
3 (vert)	0.30	1456	{1 : 0.34, 2 : 0.23, 3 : 0.15, 4 : 0.07, 5 : 0.05, 6 : 0.04, 7 : 0.02, 8 : 0.02, 9 : 0.02, 10 : 0.01}
4 (rouge)	0.13	19	{1 : 0.35, 2 : 0.35, 3 : 0.25, 4 : 0.05}



- les points de la méthode 1 sont confondus avec ceux de la méthode 4
- les points de la méthode 2 sont confondus avec ceux de la méthode 3

1.4 Discussion et analyses des modèles et algorithmes basés sur une chaine de Markov complexe

Dans cette dernière partie, nous avons cherché à comparer nos différentes méthodes en fonction des différentes CdM à notre disposition.

1.4.1 Résultats

CdM	Méthode	Durée moyenne	Nb iteration moyen	Variance de la durée
MonoBestiole	1	0.13	20	0.012
MonoBestiole	2	0.12	20	0.013
MonoBestiole	3	0.73	1456	0.027
MonoBestiole	4	0.12	19	0.01
MouseInMaze	1	0.11	16	0.013
MouseInMaze	2	0.11	15	0.013
MouseInMaze	3	0.51	1074	0.020
MouseInMaze	4	0.11	15	0.011
Periodic CdM	1	0.10	1	0.014
Periodic CdM	2	0.11	28	0.014
Periodic CdM	3	0.92	2085	0.40
Periodic CdM	4	0.11	0	0.010

D'après les résultats, nous remarquons que plus la chaine de Markov est longue, plus le temps et le nombre d'itération pour arriver à convergence est important. De plus, nous remarquons que la moyenne de temps et d'itération entre chaque méthode est plus ou moins similaire pour les CDM donné sauf lors de l'utilisation de la méthode 3.

1.4.2 Analyses des algorithmes sur la CdM du Jeu de l'Oie

Afin d'analyser l'efficacité de nos différents algorithmes, nous avons choisi de créér une CdM représentant le jeu de l'Oie où nous pouvons modifier le nombre de cases (donc la taille de la CdM).

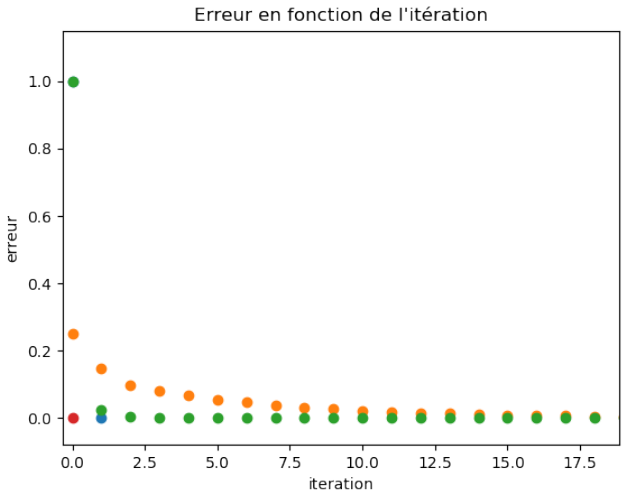
1.4.3 Résultats

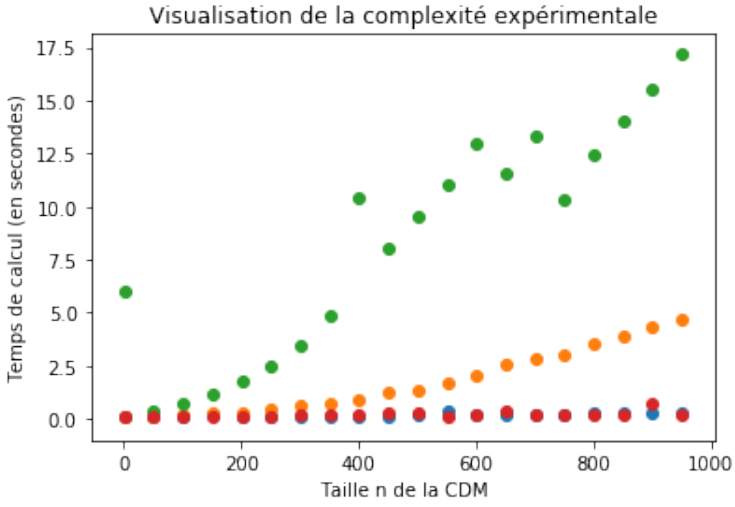
Méthode	Durée	Nb iterations	Distribution
1 (bleu)	0.11	7	{1 : 0.125, 4 : 0.125, 10 : 0.125, 14 : 0.125, 18 : 0.125, 20 : 0.375}
2 (orange)	0.12	53	{1 : 0.018, 7 : 0.018, 11 : 0.018, 16 : 0.018, 20 : 0.92}
3 (vert)	0.21	415	{1 : 0.002, 3 : 0.002, 4 : 0.002, 8 : 0.002,10 : 0.002,14 : 0.002, 20 : 0.985}
4 (rouge)	0.11	6	{1 : 0.142, 5 : 0.142, 7 : 0.142, 13 : 0.142, 17 : 0.142, 20 : 0.285}

Pour le jeu de l'Oie avec 20 cases, nous remarquons que les algorithmes 1 et 4 arrivent à convergence plus rapidement que les algorithmes 2 et 3 nécessitant respectivement 53 et 415 itérations.

Méthode	Durée moyenne	Nb iteration moyen	Variance de la durée	Variance des itérations
1 (bleu)	0.10	1.96	0.012	2.19
2 (orange)	0.12	53	0.014	0
3 (vert)	0.21	465	0.033	125.97
4 (rouge)	0.11	0.84	0.013	2.08

L'analyse sur 50 itérations de chacune des méthodes sur la CdM de l'Oie permet de mettre en avant l'efficacité des algorithmes 1 et 4 face aux algorithmes 2 et 3.





1.5 Conclusion

En conclusion, ce projet nous a permis :
- de définir un cadre de modélisation et de visualisation pour les CdM grâce aux différentes méthodes implémentés dans CdM.py
- de développer des outils d'analyse au travers de différents algorithmes.

Enfin, les analyses réalisées ont mis en évidence l'efficacité de certains algorithmes face à différentes CdM de différentes tailles.