Éléments de processus stochastiques

Méthodes de Monte Carlo par chaines de Markov

Profs. Pierre Geurts et Louis Wehenkel
Assistante : Laurine Duchesne

Année académique 2018-2019

Le travail sera réalisé par groupe de trois étudiants. Les consignes générales pour réaliser le travail, et celles relatives à la manière de rédiger les rapports, sont les mêmes pour tous les projets et sont communiquées par le Professeur V. Denoël, coordinateur du cours.

Le travail sera encadré au moyen de séances de questions/réponses qui seront programmées au fur et à mesure de l'avancement du projet. Ces séances auront lieu les mardis de 10h00 à 12h00 au local C26 (B6d). Seules les questions posées lors de ces séances donneront lieu à des réponses de la part des encadrants.

Nous encourageons vivement les étudiants à se documenter eux-mêmes sur les sujets abordés dans ce projet.

Contexte général et objectifs

L'objectif général du projet est de développer un algorithme permettant de résoudre un problème d'optimisation combinatoire de votre choix. Pour construire ce système, on se basera sur la méthode de Monte Carlo par chaines de Markov (MCMC pour *Markov chain Monte Carlo* en anglais).

Le projet est divisé en deux parties. La première partie du projet, plus théorique, a pour but de vous familiariser avec la méthode MCMC. La deuxième partie, qui constitue le cœur du projet, vise à mettre en œuvre concrètement la méthode MCMC pour résoudre un problème d'optimisation combinatoire.

Définitions et notations

Dans cette section, nous fournissons les définitions et notations minimales nécessaires pour la première partie du projet. Nous vous encourageons néanmoins à consulter d'autres références pour obtenir plus de détails à propos de la méthode MCMC.

Chaines de Markov. Soit une suite de variables aléatoires $\{X_1, X_2, \ldots, X_t, \ldots\}$. Cette suite définit un modèle (ou une chaine) de Markov d'ordre 1 ssi, pour tout $t \geq 1$, la distribution conjointe de t premières variables peut se factoriser comme suit :

$$P(X_1, X_2, \ldots, X_t) = P(X_1) \prod_{l=2}^{t} P(X_l | X_{l-1}).$$

Dans cette partie, on supposera que le modèle de Markov est discret et on notera sans restriction $\{1, \ldots, N\}$ l'ensemble des valeurs possibles des variables X_i, appelées les états.

Méthode de Monte Carlo. La méthode de Monte Carlo de base permet d'estimer la valeur de l'espérance $E\{h(X)\}$ d'une fonction h d'une variable aléatoire X en utilisant la moyenne empirique

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} h\left(x^{(i)}\right),$$

où les $x^{(i)}$ ont été générés aléatoirement *i.i.d.* selon la densité p_X. Lorsque que la variance de $h(X)$ est finie, $\hat{I}$ converge presque sûrement vers $E\{h(X)\}$, avec une erreur standard qui décroit en $1/\sqrt{n}$ avec la taille de l'échantillon. Des versions plus générales de la loi des grands nombres permettent de garantir la convergence de cette méthode dans certains cas où les $x^{(i)}$ ne sont pas indépendantes ou lorsque la variance de $h(X)$ n'est pas finie.

Méthodes MCMC. Les méthodes MCMC consistent à simuler une chaine de Markov ergodique dont la distribution stationnaire est égale à p_X. Dans ce projet, nous utiliserons l'algorithme de Metropolis-Hastings. Cet algorithme utilise une fonction q appelée *densité de proposition* qui, en pratique, est telle qu'il est possible de générer des échantillons selon cette densité.

Algorithm 1 Algorithme de Metropolis-Hastings.

Set starting state $x^{(0)}$,$t = 1$
while convergence not reached **do**
  Generate $y^{(t)} \sim q(y|x^{(t-1)})$
  $\alpha \leftarrow \min\left\{1, \frac{p_X(y^{(t)})}{p_X(x^{(t-1)})} \frac{q(x^{(t-1)}|y^{(t)})}{q(y^{(t)}|x^{(t-1)})}\right\}$
  Generate u uniformly in $[0, 1[$
  if $u < \alpha$ **then**
    $x^{(t)} \leftarrow y^{(t)}$
  else
    $x^{(t)} \leftarrow x^{(t-1)}$
  end if
  $t \leftarrow t + 1$
end while

1 Première partie : chaines de Markov et algorithme MCMC

La première partie du projet permet de se familiariser avec les chaines de Markov et de comprendre comment l'algorithme MCMC fonctionne.

1.1 Chaines de Markov

Questions :

1. Soit la séquence `seq1` composée uniquement de quatre états et fournie dans le fichier Matlab `seq1.mat`. Utilisez la méthode du maximum de vraisemblance pour construire la matrice de transition Q (avec $[Q]_{i,j} = \mathbb{P}(X_{t+1} = j | X_t = i)$) ainsi que la distribution de probabilité initiale π_0 d'une chaine de Markov modélisant cette séquence. Représentez le diagramme d'états de la chaine de Markov correspondante.

2. Sur base de la matrice de transition Q estimée au point précédent, calculez les quantités suivantes pour des valeurs de t croissantes :
   — $\mathbb{P}(X_t = x)$, où $x = 1, 2, 3, 4$, en supposant que le premier état est choisi au hasard, i.e. l'état initial est tiré dans une loi uniforme discrète.
   — $\mathbb{P}(X_t = x)$, où $x = 1, 2, 3, 4$, en supposant que l'état initial est toujours 3,
   — Q^t, c'est-à-dire la t-ième puissance de la matrice de transition.
   Représentez l'évolution des deux premières grandeurs sur un graphe. Discutez et expliquez les résultats obtenus sur base de la théorie.
3. En déduire la distribution stationnaire π_∞ de la chaine de Markov définie par $[\pi_\infty]_j = \lim_{t\to\infty} \mathbb{P}(X_t = j)$.
4. Générez une réalisation aléatoire de longueur T de la chaine de Markov en démarrant d'un état choisi aléatoirement selon la distribution stationnaire. Calculez pour chaque état le nombre de fois qu'il apparait dans la réalisation rapporté à la longueur de la réalisation. Observez l'évolution de ces valeurs pour chaque état lorsque T croît.
5. Que concluez-vous de cette expérience ? Reliez ce résultat à la théorie.

1.2 Méthode MCMC : analyse théorique dans le cas fini

Les méthodes MCMC sont souvent utilisées lorsqu'il est difficile d'échantillonner directement selon une distribution p_X. Par exemple en inférence bayésienne, on souhaite typiquement générer des échantillons d'un paramètre θ selon sa densité a posteriori $p_{\theta|D}$, alors que celle-ci n'est généralement connue qu'à un facteur de normalisation près.

Les méthodes MCMC peuvent être appliquées aussi bien à des variables continues que discrètes, prenant leurs valeurs dans un ensemble fini ou infini. Cependant, dans le cadre de ce travail nous nous focaliserons sur l'application de cette méthode dans le cas où la variable cible est discrète à valeurs finies, et nous nous limiterons à l'étude de l'algorithme de Metropolis-Hastings.

Questions :

1. Etant donné une matrice de transition Q et une distribution initiale π_0 d'une chaine de Markov invariante dans le temps qui satisfont les équations de balance détaillée (c'est-à-dire $\forall i, j \in \{1, \ldots, N\}$, $\pi_0(i)[Q]_{i,j} = \pi_0(j)[Q]_{j,i}$), montrez que π_0 est une distribution stationnaire de la chaine de Markov. Dans quel(s) cas celle-ci est-elle unique ?
2. Dans le cas où X est discrète, démontrez que l'application de l'algorithme de Metropolis-Hastings en remplaçant p_X par une fonction f telle que $\forall x : f(x) = cp_X(x)$, où c est une constante, génère une chaine de Markov qui satisfait les équations de balance détaillée avec $p_X(x)$ comme distribution stationnaire. Quelles autres conditions la chaine doit-elle respecter pour que l'algorithme de Metropolis-Hastings fonctionne ?
   Indice : commencez par écrire les probabilités de transition en prenant en compte les différents cas possibles (rejet ou acceptation) et vérifiez que les équations de balance détaillée sont satisfaites.

1.3 Méthode MCMC : application sur un exemple simple

Avant d'utiliser la méthode pour résoudre un problème d'optimisation, nous allons l'appliquer sur un problème simple, afin de mieux comprendre comment elle fonctionne et comment l'appliquer.

Questions : Soit une distribution de Poisson tronquée $p_X(x)$ définie sur les entiers $\{0, 1, 2, 3, \ldots, K\}$ par :

$$p_X(x) = Ce^{-\lambda}\frac{\lambda^x}{x!},$$

où C est une constante de normalisation telle que $\sum_{i=0}^{K} p_X(i) = 1$. On souhaite appliquer l'algorithme de Metropolis-Hastings pour échantillonner des valeurs suivant cette loi de Poisson tronquée, avec comme paramètres $\lambda = 2$ et $K = 10$. Pour ce faire, on propose d'utiliser la distribution de proposition suivante :

$$q(y|x) = \begin{cases} 1/2 & \text{si } x = 0 \text{ et } (y = 0 \text{ ou } y = 1), \\ 1/2 & \text{si } x = K \text{ et } (y = K \text{ ou } y = K - 1), \\ 1/2 & \text{si } 0 < x < K \text{ et } (y = x - 1 \text{ ou } y = x + 1), \\ 0 & \text{sinon.} \end{cases}$$

1. Sur base de l'analyse du point précédent, montrez que pour ce choix de distribution de proposition l'algorithme de Metropolis-Hastings permettra bien de générer des échantillons selon la distribution p_X.
2. Générez une réalisation suffisamment longue de la chaine et étudiez la convergence de la moyenne et de la variance des valeurs ainsi générées vers les valeurs théoriques attendues en fonction de la longueur de la réalisation.
3. Tracez un histogramme des fréquences d'apparition de chaque valeur dans la réalisation et comparez cet histogramme avec la distribution théorique.

2 Deuxième partie : résolution d'un problème d'optimisation combinatoire à l'aide de la méthode MCMC

L'objectif de cette seconde partie est d'utiliser la méthode MCMC pour résoudre un problème d'optimisation combinatoire de votre choix.

Un problème d'optimisation combinatoire est un problème d'optimisation où l'on cherche à minimiser (ou maximiser) une fonction objectif dont le domaine de recherche est fini mais potentiellement très large.

Formellement, soit S l'ensemble fini reprenant toutes les solutions possibles du problème d'optimisation combinatoire. Sans perte de généralité, on cherche à obtenir la solution optimale x^* appartenant à S qui minimise la valeur de la fonction objectif $f(x)$, c'est-à-dire telle que :

$$x^* = \arg_{x \in S} \min f(x).$$

Un des exemples les plus connus de problèmes d'optimisation combinatoire est le problème du voyageur de commerce (*Travelling Salesman Problem*, voir les suggestions ci-dessous). Tel que formulé, le problème englobe également des problèmes de recherche combinatoire, où il s'agit de trouver dans S une ou plusieurs solutions x qui satisfont une certaine condition. Il suffit pour cela d'utiliser une fonction $f(x)$ dont la valeur est minimale pour x lorsque que la condition est satisfaite (voir, par exemple, les problèmes de génération de mots croisés et de résolution de sudoku ci-dessous).

Etant donné que S est un ensemble fini, les techniques d'optimisation classiques basées sur le calcul de gradients ne sont pas applicables. La façon la plus directe de résoudre ce type de problème est de calculer la valeur de $f(x)$ de façon exhaustive pour chaque état et de choisir

celui qui minimise $f(x)$. Malheureusement, cette méthode n'est souvent pas utilisable en pratique, le nombre d'états possibles étant en général très élevé. Il est néanmoins possible de trouver une solution approchée au problème en utilisant la méthode MCMC. L'idée générale de cette approche est de définir une distribution de probabilité p_X sur S dont les modes correspondent aux minimums de la fonction $f(x)$ à minimiser et d'ensuite utiliser la méthode MCMC pour construire une chaine de Markov dont la distribution stationnaire correspond à p_X. Cette chaine est ensuite utilisée pour échantillonner des solutions dans S qu'on espère être proche des minimums de $f(x)$.

Une définition potentielle générale de la distribution p_X est la suivante :

$$p_X(x) = Ce^{-\beta f(x)},$$

où β est un paramètre à définir et C est une constante de normalisation. Le paramètre β est un paramètre important de l'approche [1] qui détermine à quel point le distribution p_X est concentrée autour des minimums de $f(x)$ et qui a une influence sur la convergence de la chaine vers sa distribution stationnaire.

L'algorithme de Metropolis-Hastings pour échantillonner selon la distribution p_X mentionnée ci-dessus est appelé l'algorithme de recuit simulé (*simulated annealing*) dans le domaine de l'optimisation. L'objectif de cette partie du travail sera d'utiliser cet algorithme afin de trouver une ou plusieurs solutions optimales au problème choisi.

Questions : Votre rapport pour cette partie est relativement libre mais on vous demande au minimum de répondre aux questions suivantes :

1. Décrivez précisément le problème d'optimisation combinatoire que vous cherchez à résoudre. Définissez l'espace S de solutions et la fonction $f(x)$ à optimiser.
2. Calculez la cardinalité de l'ensemble S de toutes les solutions possibles pour votre problème et concluez sur l'infaisabilité d'une énumération exhaustive de ces solutions.
3. Proposez et argumentez un ou plusieurs choix pour la distribution de proposition q.
4. Implémentez l'algorithme de Metropolis-Hastings et testez votre implémentation sur le problème choisi.
5. Analysez la vitesse de convergence de votre algorithme vers un optimum en fonction de la valeur du paramètre β. Choisissez des instances du problème et des critères appropriés pour étudier cette convergence.
6. Il est souvent conseillé de partir d'un paramètre β de valeur faible et d'ensuite augmenter sa valeur au fur et à mesure des itérations de l'algorithme. Renseignez vous sur les stratégies possibles de mise à jour du β et testez ces stratégies sur votre problème.
7. Commentez la ou les solution(s) obtenue(s) finalement et concluez sur la pertinence de l'algorithme pour résoudre votre problème.

Suggestions : Nous vous encouragons à trouver par vous-mêmes un problème d'optimisation original. Si vous n'avez pas d'inspiration, voici néanmoins trois problèmes que vous pourriez aborder :

1. β est souvent formulé comme l'inverse $\frac{1}{T}$ d'un paramètre T appelé la température.

— Le problème du voyageur de commerce : ce problème consiste, étant donné un ensemble de N villes, à déterminer un tour de ces villes de longueur minimale (c'est-à-dire un chemin partant d'une ville et y revenant en passant une et une seule fois par toutes les villes). Pour tester votre algorithme, nous vous fournissons avec l'énoncé l'ensemble des villes de Belgique avec leurs coordonnées (x, y) (la distance entre deux villes pouvant alors être obtenue par la distance euclidienne).
— Génération de grilles de mots croisés : Soit une grille de taille $N \times N$ où chaque case est remplie par une lettre. Cette grille constitue une grille de mots croisés valide si chaque ligne et chaque colonne correspond à un mot valide du dictionnaire. Le problème est de générer des grilles valides à partir d'une liste de mots de N lettres. Un fichier contenant 2011 mots de 4 lettres (en langue française) vous est fourni avec l'énoncé pour vos expérimentations.
— Résolution de sudoku : une grille de sudoku est une grille de taille 9×9 remplie des chiffres de 1 à 9. Une grille est valide s'il n'y a jamais deux copies du même chiffre dans une ligne, une colonne ou un des 9 carrés 3×3 obtenus en coupant la grille en trois selon les lignes et colonnes. Le problème à résoudre est de compléter une grille partiellement remplie de chiffres de manière à ce qu'elle constitue une grille valide. Dans le cas où aucun chiffre ne serait présent initialement dans la grille, l'approche permettrait également de générer des grilles de sudoku.

Le premier problème est un problème d'optimisation à proprement parler pour lequel une seule solution optimale est recherchée. Pour ce type de problème, il peut être intéressant de tester l'approche sur des problèmes (de petites tailles ou construits artificiellement par exemple) pour lesquels une solution optimale peut être obtenue et de se renseigner sur les solutions algorithmiques efficaces éventuelles qui existeraient pour résoudre le problème. Les deux autres problèmes sont des problèmes de recherche combinatoire. Pour ce type de problème, il faudra déterminer un mécanisme permettant de générer plusieurs solutions valides différentes. Plusieurs possibilités existent pour générer ces solutions (par exemple, générer plusieurs réalisations de la chaine à partir d'un état initial aléatoire en s'arrêtant à chaque réalisation dès qu'une solution valide est trouvée ou encore collecter les solutions valides générées au cours d'une seule réalisation) qu'il serait intéressant de discuter et comparer dans votre rapport. Il pourrait être intéressant également pour ce type de problème de réfléchir à une manière d'utiliser la chaine pour estimer empiriquement le nombre de solutions valides au problème.

3 Rapport et code

Vous devez nous fournir un rapport au format pdf contenant vos réponses, concises mais précises, aux questions posées ainsi que le code Matlab que vous avez utilisés pour y répondre, le tout sous forme d'archive zip. Si vous le souhaitez pour la partie 2, un autre langage de programmation peut être utilisé, avec l'accord des encadrants. Que ce soit avec Matlab ou un autre langage, l'algorithme de Metropolis-Hastings doit cependant être implémenté par vos soins. Aucun outil existant ne peut être utilisé.

4 Références

Toutes les références, les données, et les codes relatifs au projet seront collectés sur la page Web des projets (`http://www.montefiore.ulg.ac.be/~lduchesne/stocha/`). Une foire aux questions sera également ajoutée sur cette page et complétée au fur et à mesure de l'avancement du projet. Veillez à consulter régulièrement cette page.