Éléments de processus stochastiques
Méthodes de Monte Carlo par chaines de Markov

Prof. Pierre Geurts

Encadrants : Laurine Duchesne, Vân Anh Huynh-Thu, Yann Claes

Année académique 2019-2020

Le travail est à réaliser par groupe de **trois** étudiants. Il sera encadré au moyen de séances de questions/réponses qui auront lieu les mardis de 10h30 à 12h30 au local A2 (B7a). 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. Tout source utilisée devra évidemment être citée dans le rapport.

Le rapport et le code source sont à remettre via la plateforme de soumission de Montefiore pour **le 5 mai 2020 à 22h00** au plus tard.

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

Soit la chaine de Markov à 4 états définie par la matrice de transition Q suivante :

$$Q = \begin{pmatrix} 0 & 0.1 & 0.1 & 0.8 \\ 1 & 0 & 0 & 0 \\ 0.6 & 0 & 0.1 & 0.3 \\ 0.4 & 0.1 & 0.5 & 0 \end{pmatrix},$$

où $Q_{ij} = \mathbb{P}(X_{t+1} = j | X_t = i)$.

Questions :

1. Calculez (numériquement en utilisant Matlab) 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 (en traçant une courbe par état). Discutez et expliquez les résultats obtenus sur base de la théorie.
2. En déduire la distribution stationnaire π_∞ de la chaine de Markov définie par $[\pi_\infty]_j = \lim_{t\to\infty} \mathbb{P}(X_t = j)$.
3. 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.
4. 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.

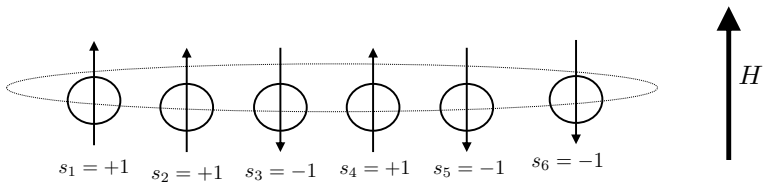

FIGURE 1 – Illustration d'un système d'Ising à une dimension constitué de 6 particules. Chaque particule peut prendre une orientation parmi deux représentées par le sens des flèches. La constante H représente un champ extérieur. L'énergie du système est d'autant plus faible (le système est stable) que deux particules voisines sont orientées dans la même direction et que les particules sont orientées dans la même direction que le champ extérieur. On suppose que le système est circulaire : la particule 1 est voisine de la particule 6.

1.3 Méthode MCMC : illustration sur le modèle d'Ising unidimensionel

En physique statistique, le modèle d'Ising est un modèle qui régit les interactions entre particules physiques à deux états. Ce modèle est principalement utilisé pour étudier les matériaux ferromagnétique. On se propose dans cette section d'utiliser le modèle d'Ising à une seule dimension pour illustrer le fonctionnement de l'algorithme de Metropolis-Hastings.

Le système d'Ising à une dimension (voir la figure 1 pour une illustration) consiste en une série de N particules dont l'état est représenté par N variables aléatoires $\{S_1, \ldots, S_N\}$ prenant chacune leur valeur dans $\{-1, 1\}$, représentant des aimants/dipoles pouvant être orientés dans deux directions. L'énergie totale associée à une configuration particulière $\{s_1, \ldots, s_N\}$ du système est calculée par l'expression suivante :

$$E(s_1, \ldots, s_N) = -J \left(\sum_{i=1}^{N-1} s_i s_{i+1} + s_1 s_N \right) - H \sum_{i=1}^{N} s_i,$$

où $J \geq 0$ est une constante de couplage et $H \in \mathbb{R}$ exprime l'existence d'un champ magnétique extérieur. Etant donné le premier terme, l'énergie du système est d'autant plus faible que des paires de particules consécutives [1] dans l'ordre sont orientées dans la même direction. La probabilité pour le système de se trouver dans un état donné vaut :

$$P_{S^N}(s_1, \ldots, s_N) = \frac{1}{Z} \exp(-\beta E(s_1, \ldots, s_N)),$$

où β est un paramètre mesurant l'inverse d'une température et Z est la constante de normalisation (aussi appelée fonction de partition) définie par :

$$Z = \sum_{(s_1, \ldots, s_N) \in \{-1,1\}^N} \exp(-\beta E(s_1, \ldots, s_N))$$

L'étude de ce type de système demande généralement de pouvoir calculer la valeur moyenne de certaines grandeurs d'intérêt telles que par exemple la *magnétisation* du système définie par la moyenne des valeurs des particules :

$$M(s_1, \ldots, s_N) = \frac{1}{N} \sum_{i=1}^{N} s_i.$$

1. Le terme $s_1 s_N$ exprime le fait que la séquence est fermée circulairement.

Le calcul exact de ces moyennes est cependant complexe, voire impossible, dans le cas où N devient trop grand.

On se propose dans cette section d'estimer ces moyennes via l'algorithme de Metropolis-Hastings, en s'appuyant sur la distribution de proposition suivante :

$$q(S'_1 = s'_1, \ldots, S'_N = s'_N | S_1 = s_1, \ldots, S_N = s_N) = \begin{cases} \frac{1}{N} & \text{si } \exists i \in \{1, \ldots, N\} : s'_i = -s_i, s'_j = s_j \forall j \neq i \\ 0 & \text{sinon} \end{cases},$$

qui consiste à générer un nouvel état du système à partir du précédent simplement en changeant la valeur d'une particule tirée au hasard.

Afin d'évaluer la qualité des estimations obtenues, une fonction Matlab vous est fournie dans le fichier `ising1Dexact.m` calculant les valeurs exactes des probabilités des états, de la magnétisation moyenne et de la constante de normalisation Z. Cette fonction n'est cependant utilisable que pour des valeurs de N faibles.

Questions :

1. Sur base de l'analyse de la section précédente, montrez qu'étant donné le choix de la distribution de proposition q ci-dessus, l'algorithme de Metropolis-Hastings permettra bien de générer des échantillons selon la distribution P_{S^N}.
2. Soit i l'index de la particule tirée au hasard dont la valeur est modifiée et soit $(s_1, \ldots, s_N)$ l'état du système avant modification. Montrez que la probabilité α d'acceptation du nouvel état peut se calculer comme suit :

   $$\alpha = \min\left\{1, \exp(-2\beta s_i (J(s_{l(i)} + s_{r(i)}) + H))\right\}$$

   où $l(i)$ (resp. $r(i)$) est la particule à gauche (resp. à droite) de la particule i, c'est-à-dire la particule $i-1$ si $i > 1$, N si $i = 1$ (resp. $i+1$ si $i < N$, 1 si $i = N$).
3. Implémentez l'algorithme de Metropolis-Hasting avec la distribution de proposition q ci-dessus pour estimer la magnétisation moyenne. Générez une réalisation de la chaine de Markov de longueur 10000 en fixant les paramètres N, J, H, et β respectivement à 10, 1, 3, et 0.2 et tracez l'évolution de la magnétisation moyenne estimée sur base des t premiers échantillons extraits de cette réalisation pour t allant de 1 à 10000. Comparez la courbe obtenue à la valeur exacte de la magnétisation moyenne calculée par la fonction `ising1Dexact` fournie. Votre courbe confirme-t-elle bien la théorie ?
4. Répétez 100 fois l'expérience précédente et tracez pour chaque valeur de t la moyenne des magnétisations moyennes sur les 100 expériences ainsi que les 5ème et 95ème centiles. Qu'en concluez-vous quant à la convergence de l'estimation de la magnétisation moyenne ?
5. En choisissant une longueur de réalisation suffisamment élevée (sur base de l'expérience du point précédent), tracez l'évolution de la magnétisation moyenne en fonction de H (qui variera entre $[-20, 20]$ par pas de 1) en fixant les valeurs des autres paramètres comme aux points précédents. Tracez ensuite la même courbe mais pour une valeur $\beta = 0.02$ (c'est-à-dire une température plus élevée). Comparez les deux courbes ainsi obtenues avec les courbes exactes calculée via la fonction `ising1Dexact`.

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 parmi les suggestions proposées ci-dessous.

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 maximale 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 [2] 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.

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

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.

Choix du problème : Le problème d'optimisation combinatoire que vous aborderez est à choisir dans la liste des cinq problèmes ci-dessous. Il est permis de traiter un problème qui ne serait pas dans cette liste si vous avez une idée originale mais ce problème devra impérativement être validé par les encadrants (au plus tard avant le début de la quatrième séance) :

1. Le problème du voyageur de commerce [3] : 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 obtenues par la distance euclidienne).
2. Le problème du sac à dos [4] : Etant donné un ensemble de N objets ayant chacun un poids $w_i \in I\!R_+^*$ et une utilité $u_i \in I\!R_+^*$, ce problème consiste à trouver la combinaison d'objets d'utilité totale maximale dont le poids total n'excède pas une valeur W fixée. Votre algorithme pourra être testé sur des instances de problèmes générées aléatoirement [5].
3. Débruitage d'images binaires : Soit une image I en noir et blanc représentée par une matrice $N \times M$ de valeurs binaires. Supposons qu'un bruit ait été appliqué à cette image, par exemple en changeant la valeur de certains pixels choisis aléatoirement. Soit I' l'image bruitée. On aimerait reconstruire I à partir de I'. Une manière de procéder est de définir une distribution de probabilité conditionnelle $P(I|I')$ et d'utiliser l'algorithme de Metropolis-Hastings pour rechercher l'image I qui maximise cette probabilité. Pour définir $P(I|I')$, on peut utiliser la formule de Bayes : $P(I|I') \propto P(I'|I)P(I)$ où $P(I'|I)$

3. `https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce`
4. `https://fr.wikipedia.org/wiki/Problème_du_sac_à_dos`
5. Il existe des générateurs, par exemple ici : `http://hjemmesider.diku.dk/~pisinger/codes.html`.

peut être calculé sur base de la connaissance du processus de bruit et $P(I)$ devra exprimer une contrainte a priori sur ce qui constitue une image plausible. On peut par exemple raisonnablement postuler qu'une image standard est généralement telle qu'une majorité de pixels voisins sont identiques. Cette condition pourra être modélisée en utilisant une distribution $P(I)$ proche de la distribution du modèle de Ising.

4. 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.
5. 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.

Les trois premiers problèmes sont des problèmes d'optimisation à proprement parler pour lesquels 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 (exactes ou approchées) 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.

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. La longueur attendue du rapport est entre 15 et 30 pages (figures et bibliographie incluse, police de taille 11, pas de code dans le rapport).

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.