Université Clermont Auvergne Année 2020-2021

Master 1 MAS - Algorithmes Stochastiques

TP 5 - Applications d'algorithme de Metropolis-Hastings

---

1 Recuit simulé

Les algorithmes de recuit simulé sont destinés à des problèmes de minimisation difficiles, où

– une recherche exhaustive est rendu impossible par la taille de l'espace,

– la fonction à minimiser admet un grand nombre de minima locaux que l'on souhaite éviter.

Le mot *recuit* désigne un procédé métallurgique qui consistant à fondre, puis à laisser refroidir *très* lentement, un métal pour améliorer ses qualités.

L'objectif de recuit simulé est de minimiser une fonction f dans un domaine. La première idée est de suivre l'algorithme de Metropolis-Hastings, sachant qu'il passe le plus de temps dans les régions où une densité de probabilité donnée est le plus grande. Ainsi on définit la densité $p(x)$ comme *proportionnelle* à $\exp(-f(x))$. Attention : la vrai exploit de l'algorithme M-H consiste à nous débarrasser de la constante qui normalise la densité ! En effet, il suffit de connaître les probabilités *relatives* $p(x_1)/p(x_2)$ ce qui se traduit directement par $\exp(-(f(x_1) - f(x_2)))$.

La deuxième composante du recuit simulé est de faire baisser la probabilité d'acceptation très lentement dans le temps. Ainsi cette probabilité de l'acceptation vaudra donc $\exp(-(f(x_1) - f(x_2))/T)$ où T est un paramètre, appelé *température*, qui décroit dans le temps.

Ainsi quand la température est grande, l'algorithme accepte facilement des états moins optimales et donc explore tout l'espace. Quand T diminue, l'algorithme commence à préférer des améliorations *locales* (ce qu'on appelle l'exploitation d'une niche).

Le point délicat est de trouver une équilibre entre ces deux stratégies. Dans le procédé du recuit simulé, en s'inspirant de la métallurgie, on fait baisser la température lentement, typiquement comme $1/\log(t)$ où t est le temps.

Exercice 1. *Voyageur de commerce*

Appliquer l'algorithme du recuit simulé pour le problème du voyageur de commerce.

Exercice 2. *Optimisation continue* Dans plusieurs domaines des mathématiques appliqués, l'IA on rencontre des problèmes qui se résument à trouver un maximum ou minimum d'une fonction compliquée (souvent en grande dimension).

A l'aide de recuit simulé adapté à l'espace d'état E non-discret, trouver une approximation du maximum des fonctions suivantes :

1. $f(x) = 10\sin((0.3x) * \sin(1.3x^2 + 0.00001x^4 + 0.2x + 80))$ sur $E = [-10, 10]$,
2. $f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos(2\pi x_1) + \cos(2\pi x_2))$ sur $E = [-10, 10]^2$.

2 Algorithmes génétiques

Algorithmes génétiques sont des cousins du recuit simulé bien que l'idée de l'algorithme génétique, inspirée par une analogie avec le processus de sélection naturelle est apparue bien avant le recuit simulé. L'idée de base est la même : coupler une recherche systematique de l'optimum avec une exploration stochastique de l'espace d'états. Il s'agit de maximiser une fonction (appelée adaptation, ang. fitness) définie sur $E = 0, 1^d$, à valeurs dans R^+,

$$f : E \mapsto \mathbb{R}^+.$$

On définit une chaîne de Markov $X_n = (X_n^1, X_n^2, \ldots, X_n^M)$ à valeurs dans E^M. La chaîne X_n est donc M-uplet de mots binaires de longueur d, appelés aussi *individus* ou encore *chromosomes*.

L'algorithme de passage de X_n à X_{n+1} se décompose en trois étapes :

$$X_n \overset{\text{mutation}}{\longrightarrow} Y \overset{\text{croisement}}{\longrightarrow} Z \overset{\text{selection}}{\longrightarrow} X_{n+1}$$

Il s'agit de 3 stratégies de modification de la population inspirées par l'évolution naturelle. Mais les implementations concrètes peuvent varier.

Exercice 3. *Problème du sac à dos (version binaire)*

On est donné un ensemble d'objets ayant chacun un poids et une valeur. L'objectif est de remplir un sac à dos de façon à maximiser la valeur totale des objets sans dépasser le poids maximum.

Plus formellement, soit $n \in \mathbb{N}$ et $x = (x_1, \ldots, x_n)$ un vecteur 0-1 qui indique les items à mettre dans le sac à dos.

Soit $v = (v_1, \ldots, v_n)$ un vecteur des valeurs des items.

Soit $p = (p_1, \ldots, p_n)$ un vecteur des poids.

On veut alors maximiser

$$f(x) = \sum_{i=1}^{n} x_i v_i$$

sous contrainte

$$\sum_{i=1}^{n} x_i p_i < M$$

où $M \in \mathbb{R}$ est le poids maximum.,