Note de lecture sur le papier sur l'Entropie croisée

30 mars 2020

Table des matières

CE = Cross-entropy

La CE est un modèle utilisé pour estimer des probabilités d'événements rares (permet d'avoir moins d'itérations que pour les méthodes classiques d'estimation). Ce modèle peut aussi être utilisé pour résoudre des problèmes d'optimisation combinatoire (COPs). C'est fait en transformant l'optimisation « déterministe » en une optimisation « stochastique » pour ensuite utiliser les méthodes de modélisation d'événements rares.

La méthode de CE est une méthode itérative dont chaque itération peut être décomposée en deux étapes :

1. Génération d'un échantillon aléatoire selon un mécanisme spécifique
2. Mise à jour du paramètre de ce mécanisme en se basant sur les données qui donnent les meilleurs résultats.

FIGURE 1 – Boîte noire utilisée pour décoder le vecteur y

Pour estimer des probabilités d'événements rares, on pourrait penser à la méthode d'*Important sampling*. Un des désavantages de cette méthode est que les paramètres optimaux (*tilting*) sont difficiles à obtenir. Au contraire, l'avantage de la méthode de CE est d'avoir une procédure simple pour estimer les paramètres optimaux.

1 Exemple d'un problème d'optimisation combinatoire

On suppose que l'on a un vecteur binaire $y = (y_1, \ldots, y_n)$ qu'on cherche à deviner. On ne connait pas y mais on a un « oracle » qui pour chaque *input* nous donne une *réponse* ou *performance*. Par exemple : $S(x) = n - \sum_{j=1}^{n} |x_j - y_j|$ qui donne le nombre d'éléments de $x = (x_1, \ldots, x_n)$ égaux à y.
Une méthode naïve est de générer des vecteur $X = (X_1, \ldots, X_n)$ de façon à ce que $X_1, \ldots, X_n$ soient des Bernoulli indépendantes de paramètre $p_1, \ldots, p_n$. Donc $X \sim \mathcal{B}(p)$ et si $p = y$ (loi dégénérée) on a $S(X) = n$ et $X = y$.
L'algorithme de CE consiste à transformer le problème en un événement rares, créer une séquence de paramètres $\hat{p_0}, \hat{p_1}, \ldots$, et des niveaux (*levels*) $\hat{\gamma_1}, \hat{\gamma_2}, \ldots$ de telles sorte que $\hat{\gamma_1}, \hat{\gamma_2}, \ldots$ converge vers la valeur optimale (ici n) et que la suite $\hat{p_0}, \hat{p_1}, \ldots$ converge vers le paramètre optimal (ici y).

Description de l'algorithme :

1. On commence avec un certain $\hat{p}_0$, par exemple $\hat{p_0} = (1/2, \ldots, 1/2)$ et $t := 1$
2. On génére des échantillons $X_1, \ldots, X_N$ selon une loi de bernoulli de paramètre $\hat{p}_{t-1}$. On calcule le score $S(X_i)$ et on trie ces données du plus petit au plus grand : $S_{(1)} \leq \cdots \leq S_{(N)}$. On note $\hat{\gamma}_{t-1}$ le quantile $1 - \rho$ du score : $\hat{\gamma}_{t-1} = S_{\lceil (1-\rho)N \rceil}$.
3. On utilise le même échantillon pour calculer $\hat{p}_t = (\hat{p}_{t,1}, \ldots \hat{p}_{t,n})$ avec la formule :
$$\hat{p}_{t,j} = \frac{\sum_{i=1}^{N} \mathbb{1}_{S(X_i) \geq \hat{\gamma}_t} \mathbb{1}_{X_{i,j}=1}}{\sum_{i=1}^{N} \mathbb{1}_{S(X_i) \geq \hat{\gamma}_t}}$$
Cette condition s'interprète de la façon suivante : pour mettre à jour la j^{e}probabilité, on compte le nombre de vecteurs de X qui ont un score plus grand que $\hat{\gamma}_t$ et dont la j^{e}composante est égale à 1 et on divise ce nombre par le nombre de vecteurs de X qui ont un score plus grand que $\hat{\gamma}_t$.
4. On arrête si on rencontre le critère d'arrêt ($\hat{\gamma}_t$ constant ou $\hat{p}_t$ loi dégénérée), sinon $t := t + 1$.

2 La méthode CE pour les événements rares

Soit $X = (X_1, \ldots, X_n)$ un vecteur aléatoire qui prend ses valeurs dans $\mathcal{X}$, $\{f(\cdot; v),\, v \in \mathcal{V}\}$ une famille de densités de probabilités – *probability density functions* (pdfs) – sur $\mathcal{X}$ associés à une mesure ν(mesure de Lebesgue ou mesure de comptage). Pour toute fonction mesurable H on a :

$$\mathbb{E}H(X) = \int_{\mathcal{X}} H(x) f(x; v) \nu(\mathrm{d}x)$$

Soit S une fonction réelle sur $\mathcal{X}$, on suppose que l'on s'intéresse à la probabilité que $S(x)$ soit supérieure à un certain nombre γ sous la densité $f(\cdot, u)$. Cette probabilité s'écrit :

$$l = \mathbb{P}_u(S(X) \geq \gamma) = \mathbb{E}_u \mathbb{1}_{\{S(x) \geq \gamma\}}$$

Si cette probabilité est petite (plus petite que 10^{-5}), $\{S(X) \geq \gamma\}$ est appelé événement rare (*rare event*). Une façon simple d'estimer l est d'utiliser brutalement une méthode de simulation de Monte-Carlo : on échantillonne $X_1, \ldots, X_N$ de loi $f(\cdot, u)$ et ensuite

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \gamma\}}$$

nous donne un estimateur sans biais de l. Le problème est que $\{S(X) \geq \gamma\}$ est un événement rare : il faut beaucoup de simulations pour estimater avec précisions l (i.e. : avec une faible erreur ou un petit intervalle de confiance).
Une façon alternative est d'utiliser l'important sampling : on tire un échantillon $X_1, \ldots, X_N$ à partir d'une densité g sur $\mathcal{X}$ d'important sampling et on calcule l en utilisation l'estimateur du rapport de vraisemblance – *likelihood ratio* (LR) :

$$\hat{l} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \gamma\}} \frac{f(X_i; u)}{g(X_i)}$$

La meilleure façon d'estimer l est d'utiliser la densité

$$g^*(x) := \frac{\mathbb{1}_{\{S(x) \geq \gamma\}} f(x; u)}{l} \implies \mathbb{1}_{\{S(X_i) \geq \gamma\}} \frac{f(X_i; u)}{g(X_i)} = l$$

Et donc l'estimateur précédent est de variance nulle. Le problème évident est que g^* dépend de l. De plus il est plus pratique de choisir g dans $\{f(\cdot; v),\, v \in \mathcal{V}\}$. L'idée est de choisir un paramètre de référence – *reference parameter* ou *tilting parameter* – v de façon à ce que la distance entre g^* et $f(\cdot, v)$ soit minimale. La distance entre deux densités g et h qui est utilisée est la distance de *Kullback-Leibler*, aussi appelée *cross-entropy* entre g et h :

$$\mathfrak{D}(g, h) = \mathbb{E}_g \ln \frac{g(X)}{h(X)} = \int g(x) \ln g(x) \mathrm{d}x - \int g(x) \ln h(x) \mathrm{d}x$$

Minimiser la distance de Kullback-Leibler entre g^* et $f(\cdot, v)$ est équivalent à choisir v qui minimise $-\int g^*(x) \ln f(x; v) \mathrm{d}x$, soit à résoudre le problème de maximisation :

$$\max_v \int g^*(x) \ln f(x; v) \mathrm{d}x \iff \max_v \int \frac{\mathbb{1}_{\{S(x) \geq \gamma\}} f(x; u)}{l} \ln f(x; v) \mathrm{d}x$$

On obtient le programme :

$$\max_v D(v) = \max_v \mathbb{E}_u \mathbb{1}_{\{S(X) \geq \gamma\}} \ln f(X; v)$$

On peut encore une fois utiliser l'important sampling avec une nouvelle mesure $f(\cdot, w)$ et le problème précédent devient :

$$\max_v D(v) = \max_v \mathbb{E}_w \mathbb{1}_{\{S(X) \geq \gamma\}} W(X; u, w) \ln f(X; v)$$

Avec $W(x; u, w) = \dfrac{f(x; u)}{f(x; w)}$ le rapport de vraisemblance, en x, entre $f(\cdot, u)$ et $f(\cdot, w)$. La solution optimale s'écrit :

$$v^* = \operatorname*{argmax}_v \mathbb{E}_w \mathbb{1}_{\{S(X) \geq \gamma\}} W(X; u, w) \ln f(X; v) \tag{1}$$

On estime v^* à partir du programme stochastique – *stochastic counterpart* – suivant :

$$\max_v \hat{D}(v) = \max_v \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \gamma\}} W(X_i; u, w) \ln f(X_i; v) \tag{2}$$

Avec $X_1, \ldots, X_N$ tirés aléatoirement selon $f(\cdot, w)$. Dans les applications classiques, $\hat{D}$ est différentable et convexe en v, la solution de 2 peut alors être obtenue en résolvant les équations :

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \gamma\}} W(X_i; u, w) \nabla_v \ln f(X_i; v) = 0 \tag{3}$$

Remarque : Pour que le programme de CE 2 soit utile, il faut que la probabilité de l'événement $\{S(X) \geq \gamma\}$ ne soit pas trop petite, sinon les $\mathbb{1}_{\{S(X_i) \geq \gamma\}}$ seront souvent égales à 0.
Pour surmonter cette difficulté, on utilise un algorithme multi-niveau – *multi-level algorithm.* L'idée est de construire une suite de paramètres de références $(v_t)_{t \geq 0}$ et une suite de niveaux $(\gamma_t)_{t \geq 1}$, l'algorithme consiste à construire itérativement cette suite. L'idée est de choisir $\hat{v}_0 = u$ et de prendre $\hat{\gamma}_1$ plus petit que γ : $\hat{v}_1$ rendra l'événement $\{S(X) \geq \gamma\}$ un peu moins rare, donc $\hat{\gamma}_2$ peut être plus proche de γ.

Description de l'algorithme de CE pour la simulation d'événements rares :

1. On pose $\hat{v}_0 = u$, $t = 1$.
2. On génère un échantillon $X_1, \ldots, X_N$ de loi $f(\cdot, v_{t-1})$, on calcule le quantile $(1-\rho)$ de la fonction score qui donne $\hat{\gamma}_t$:

$$\hat{\gamma}_t = S_{\lceil (1-\rho)N \rceil}$$

3. On utilise le **même** échantillon $X_1, \ldots, X_N$ pour résoudre le problème stochastique :

$$\hat{v}_t = \underset{v}{\operatorname{argmax}}\, \hat{D}(v) = \underset{v}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} W(X_i; u, \hat{v}_{t-1}) \ln f(X_i; v) \tag{4}$$

La solution nous donne $\hat{v}_t$.

4. Si $\hat{\gamma}_t < \gamma$, $t = t + 1$ et on recommence à l'étape 2. Sinon on estime la probabilité l en utilisant le rapport de vraisemblance

$$\hat{l} = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbb{1}_{\{S(X_i) \geq \gamma_t\}} W(X_i; u, \hat{v}_T)$$

Avec T le nombre total d'itération.

3 La méthode de CE pour l'optimisation combinatoire

3.1 Le problème d'optimisation combinatoire

On considère le problème de maximisation suivant : soit $\mathcal{X}$ un ensemble fini d'états (*states*) et S une fonction de performance. On cherche à trouver le maximum de S sur $\mathcal{X}$ et les points pour lesquels ce maximum est atteint. Si on note γ^* ce maximum, on cherche donc :

$$S(x^*) = \gamma^* = \max_{x \in \mathcal{X}} S(x) \tag{5}$$

Le point de départ de la méthode de CE est d'associer un problème d'estimation avec un problème d'optimisation précédent. Pour cela on définit un ensemble d'indécatrices $\mathbb{1}_{\{S(x) \geq \gamma\}}$ sur $\mathcal{X}$ pour plusieurs niveaux $\gamma \in \mathbb{R}$.
On note $\{f(\cdot; v),\, v \in \mathcal{V}\}$ une famille discrète de probabilités sur $\mathcal{X}$, paramétrée par un paramètre vectoriel v.
Pour un certain $u \in \mathcal{V}$ on associer le problème de maximisation 5, le problème d'estimation du nombre

$$l(\gamma) = \mathbb{P}_u(S(X) \geq \gamma) = \sum_x \mathbb{1}_{S(x) \geq \gamma} f(x; u) = \mathbb{E}_u \mathbb{1}_{S(x) \geq \gamma} \tag{6}$$

C'est le problème stochastique associé – *associated stochastic problem* (ASP). Cela permet de voire l'analogie avec un problème d'estimation d'une probabilité d'événement rare. En effet, si $f(\cdot; u)$ est la densité uniforme sur $\mathcal{X}$ alors $l(\gamma^*) = f(x^*; u) = \frac{1}{|\mathcal{X}|}$ et donc si $\mathcal{X}$ contient beaucoup d'événements, $S(X) \geq \gamma^*$est un événement rare.

Description de l'algorithme de CE pour l'optimisation :
Il faut définir en avance $\hat{v}_0$, la taille de l'échantillon N et le nombre ρ (10% dans notre projet sur Master Mind)

1. On prend $\hat{v}_0$ fixé arbitrairement , $t = 1$.
2. On génère un échantillon $X_1, \ldots, X_N$ de loi $f(\cdot, v_{t-1})$, on calcule le quantile $(1-\rho)$ de la fonction score qui donne $\hat{\gamma}_t$:

   $$\hat{\gamma}_t = S_{\lceil (1-\rho)N \rceil}$$

   Si $\hat{\gamma}_t \geq \gamma$ on prend $\hat{\gamma}_t = \gamma$.
3. On utilise le **même** échantillon $X_1, \ldots, X_N$ pour résoudre le problème stochastique (avec $W = 1$) :

   $$\hat{v}_t = \underset{v}{\operatorname{argmax}}\, \hat{D}(v) = \underset{v}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \ln f(X_i; v) \qquad (7)$$

4. Si pour un certain $t \geq d$, (par exemple $d = 5$), on a :

   $$\hat{\gamma}_t = \hat{\gamma}_{t-1} = \cdots = \hat{\gamma}_{t-d}$$

   alors on arrête l'algorithme.

3.2 Smoothed updating

Plutôt que de mettre à jour directement $\hat{v}_{t-1}$ à partir de la formule du problème 7, on réalise souvent une mise à jour lissée – *smoothed updating* :

$$\hat{v}_t = \alpha \hat{w}_t + (1-\alpha)\hat{v}_{t-1} \qquad (8)$$

avec $\hat{w}_t$ la valeur obtenue en résolvant le problème 7 ($\hat{w}_t$ correspondait donc à la précédente valeur qu'on utilisait avant pour $\hat{v}_t$). C'est en particulier pertinent pour les problèmes d'optimisaiton avec des variables discrètes : cela permet d'éviter l'occurence de 0 ou de 1 dans les paramètres des vecteurs. En effet, le problème est que dès lors qu'on a un 0 ou un 1, il le reste souvent pour toujours, ce qui a des effets indésirables. On trouve empiriquement que $0,4 \leq \alpha \leq 0,9$ donne des meilleurs résultats.
Si $\alpha = 1$ on retrouve l'algorithme précédent.
Dans beaucoup d'applications on observe une convergence numérique de $f(\cdot; \hat{v}_t)$ vers une mesure dégénérée (une mesure de Dirac), c'est-à-dire que l'on affecte toute la masse de probabilité à un seul état x_T pour lequel, par définition, on a $S(x_T) \geq \hat{\gamma}_T$.

3.3 Estimation du maximum de vraisemblance

Le problème 7 est étroitement lié au problème de maximisation de la vraisemblance. En effet, le problème de maximisasion de vraisemblance revient à a

$$\hat{v}_t = \underset{v}{\operatorname{argmax}} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \ln f(X_i; v) \tag{9}$$

La seule différence avec le problème 7 est la présence de $\mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}}$. Le problème d'optimisation combinatoire peut être réécrit en :

$$\hat{v}_t = \underset{v}{\operatorname{argmax}} \sum_{X_i \,:\, S(X_i) \geq \hat{\gamma}_t} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \ln f(X_i; v) \tag{10}$$

En d'autres termes, $\hat{v}_t$ est égal à l'estimateur de maximum de vraisemblance de $\hat{v}_{t-1}$ calculé uniquement à partir des vecteurs X_i qui ont une performance supérieure ou égale à $\hat{\gamma}_t$.

3.4 Paramètres

Le choix de la taille de l'échantillon N et du paramètre ρ dépendent de la taille du problème et du nombre de paramètres dans le problème de maximisation.
En particulier dans les problème de type s*tochastic node networks* (SNN), on prend généralement $N = cn$ avec n le nombre de *nodes* et c une constante ($c > 1$), souvent $5 \leq c \leq 10$.
Dans les problème de type *stochastic edge networks* (SEN) on prend généralement $N = cn^2$ avec n^2 le nombre de *edges* dans le réseau. La taille de l'échantillon est donc liée au nombre de paramètres à estimer (n et n^2).
Pour estimer k paramètres, il faut prendre une taille d'échantillon **au minimum** égale à $N = ck$ avec $c > 1$.
Pour ρ il est souvent conseillé de prendre ρ autour de 0,01 si n est grand ($n \geq 100$) et de prendre un ρ plus grand, $\rho \simeq \frac{\ln(n)}{n}$ pour $n < 100$.
Les paramètres N et ρ peuvent aussi être déterminés de façon itérative : c'est ce qui est fait dans l'algorithme FACE ou dans Homem-de-Mello, T. and Rubinstein, R. (2002). Rare event estimation for static models via cross-entropy and importance sampling. Submitted for publication.

Dans notre cas, il faut donc prend $N = c \times m \times n$ avec $c > 1$.

3.5 Application au mastermind

Dans notre projet, on a n boules et m couleurs. On numérote les couleurs de 1 à m.
On a donc $\mathcal{X} = \{1, 2, \ldots, m\}^n$.
Les composantes du vecteur $X = (X_1, \ldots, X_n) \in \mathcal{X}$ sont tirées aléatoirement de façons à ce que leur distribution soit déterminée par une suite $p_1, \ldots, p_n$ de

vecteurs de probabilités, la j^{e}composante de p_i étant égale à $p_{ij} = \mathbb{P}(X_i = j)$. On peut représenter cette loi par une matrice

$$P = (p_{i,j})_{i,j} = (\mathbb{P}(X_i = j))_{i,j} \in \{\text{matrices stochastiques de } \mathcal{M}_{n,m}(\mathbb{R})\}$$

La densité s'écrit :

$$f(X;p) = \prod_{i=1}^{n}\prod_{j=1}^{m} p_{i,j}^{\mathbb{1}_{\{X_i=j\}}}$$

Il vient :

$$\ln f(X;p) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbb{1}_{\{X_i=j\}} \ln p_{i,j}$$

On suppose dans un premier temps qu'il n'y a pas de lien entre les $p_{i,j}$ (comme si on n'avait pas $\sum_{j=1}^{m} p_{i,j} = 1$) :

$$\frac{\partial}{\partial p_{k,l}} \ln f(X;p) = \frac{\partial}{\partial p_{k,l}} \sum_{j=1}^{m} \mathbb{1}_{\{X_k=j\}} \ln p_{k,j} = \frac{\mathbb{1}_{\{X_k=l\}}}{p_{k,l}}$$

On prend maintenant un échantillon de taille N, $X_1, \ldots, X_N \overset{i.i.d}{\sim} X$ et on note $X_i = (X_{i,1}, \ldots, X_{i,n}) \in \mathcal{X}$. Le programme de maximisation peut s'écrire comme un problème de maximisation sous contrainte :

$$\begin{cases} \max\limits_{p_{i,j}} & \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i)\geq\hat{\gamma}_t\}} \ln f(X_i;v) \\ s.c & \forall i \ : \ \sum_{j=1}^{m} p_{i,j} = 1 \end{cases}$$

Le Lagrangien s'écrit :

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i)\geq\hat{\gamma}_t\}} \ln f(X_i;v) + \sum_{i=1}^{n} \lambda_i \left(\sum_{j=1}^{m} p_{i,j} - 1 \right)$$

La condition d'optimalité en $p_{k,l}$:

$$\frac{\partial}{\partial p_{k,l}} \mathcal{L} = 0 \implies \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i)\geq\hat{\gamma}_t\}} \frac{\mathbb{1}_{\{X_{i,k}=l\}}}{p_{k,l}} - \lambda_k = 0$$

Soit :

$$\sum_{i=1}^{N} \mathbb{1}_{\{S(X_i)\geq\hat{\gamma}_t\}} \mathbb{1}_{\{X_{i,k}=l\}} = \lambda_k p_{k,l}$$

En sommant sur l et en utilisant la condition sur les $p_{i,j}$, il vient :

$$\lambda_k = \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i)\geq\hat{\gamma}_t\}} \underbrace{\sum_{l=1}^{m} \mathbb{1}_{\{X_{i,k}=l\}}}_{=1}$$

D'où la formule de mise à jour :

$$p_{k,l} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \mathbb{1}_{\{X_{i,k}=l\}}}{\sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}}}$$

4 Algorithme de Metropolis

4.1 Intuition de l'algorithme

Le but de l'algorithme de Metropolis-Hastings est de générer des échantillons d'une distribution $P(x)$, sachant que l'on connait une fonction $f(x)$ proportionnelle à la densité de P. Remarquons qu'il suffit que $f(x)$ soit proportionnelle à la densité de P, ce n'est pas utile qu'il y ait égalité, c'est notamment utile parce que le calcul du facteur de normalisation est souvent difficile.
Pour générer $P(x)$ on va construire itérativement une chaîne de Markov qui converge une asymptotiquement vers une unique loi stationnaire $\pi(x) = P(x)$. Une chaîne de Markov est uniquement définie par sa probabilité de transition $P(x'|x)$ qui représente la probabilité d'arriver à l'état x sachant que l'on est à l'état x'. On a une unique distribution stationnaire si on a les deux conditions suivantes :

1. Existence : une condition suffisante est que la condition de balance soit vérifiée : $\pi(x)P(x'|x) = \pi(x')P(x|x')$
2. Unicité : garantie par l'ergodicité des processus markovien vérifiée lorsque le processus est aperiodique (on ne retourne pas au même état à des intervalles fixes) et positive récurrente (on retourne au même étant en un temps fini).

4.2 Algorithme

L'algorithme de Metropolis-Hastings est le suivant :

1. Initialisation :
   (a) On choisit un état initial x_0
   (b) On fixe $t = 0$
2. Itération :
   (a) On prend une densité de probabilité aléatoire arbitraire $g(x'|x_t)$ qui suggère un nouveau candidat pour le prochain échantillon x' sachant que l'on est à l'état x_t. La fonction g appelée *proposal density* ou *jumping distribution.*
   (b) On calcule la probabilité d'acceptation :

$$A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t|x')}{g(x'|x_t)}\right)$$

Remarquons que si g est symétrique (i.e. $g(x|y) = g(y|x)$) alors le rapport précédent se simplifie. Un cas clasique est de choisir $g(x|y)$ comme étant une distribution gaussienne centrée en y : les points proches de y ont plus de chances d'être visités.

(c) Acceptation ou rejet :

i. Un génére une loi uniforme $u \in [0, 1]$

ii. Si $u \leq A(x', x_t)$ alors on accepte le nouvel état et $x_{t+1} = x'$

iii. Si $u > A(x', x_t)$ on rejette le nouvel état et on garde l'ancien état $x_{t+1} = x_t$

(d) Incrémentation : $t = t + 1$

4.3 Interprétation dans le cas g est symétrique

Le principe de l'algorithme est d'essayer de se déplacer aléatoirement dans l'espace, parfois en acceptant de se déplacer et d'autres fois en restant sur place (sur le même état). Le rapport d'acceptation $A(x', x_t)$ indique la probabilité du nouvel échantillon par rapport à la situation actuelle, sous la distribution $P(x)$. Si on essaye de se déplacer dans un point qui est plus probable que le point actuel (i.e. : un point qui a une plus grande densité sur $P(x)$) alors on va toujours accepter de ce déplacer ($A(x', x_t) = 1$). Toutefois, si on essaye de de se déplacer vers un point moins probable, on va parfois rejeter le déplace, et plus la baisse relative de probabilité est grande, plus on est susceptible de rejeter le nouveau point.
Ainsi, on a tendance à rester dans des régions à haute densité de $P(x)$ (et à renvoyer un grand nombre d'échantillons dans ces régions), tout en visitant occasionnellement les régions à faible densité. Intuitivement, ça explique pourquoi cet algorithme marche et renvoie des échantillons qui suivent la distribution souhaitée $P(x)$.

4.4 Désavantages de l'algorithme Metropolis–Hastings

Par rapport à d'autres algorithme du type adaptative rejection sampling qui génère directement des échantillons indépendants, Metropolis–Hastings et les autres algorithmes MCMC ont plusieurs désavantages :

- Les échantillons sont **corrélés**. Même si à long terme il suivent bien $P(x)$, un ensemble d'échantillons proches sera corrélé entre eux et ne reflétera pas correctement la distribution. Cela signifie que si on souhaite générer des échantillons indépendants, il faut rejeter la majorité d'entre eux et par exemple ne garder que le n^{e}échantillon, pour un certain n fixé (habituellement déterminé en examinant l'autocorrélation entre les échantillons adjacents). L'autocorrélation peut aussi être atténuée en augmentant le *jumping width* (la taille moyenne du saut, qui est liée à la variance de la distribution de la *jumping distribution* g), mais cela va aussi augmenter la probabilité de rejet du saut. Une taille de saut trop grande ou trop petite va mener à une slow-mixing chaîne de Marking, i.e ; : un

ensemble d'échantillons hautement corrélés, de sorte qu'un très grand nombre d'échantillons sera nécessaire pour obtenir une estimation raisonnable des propriétés de la distribution.
- Même si la chaîne de Markov converge vers la distribution souhaitée, les échantillons initiaux peuvent suivre une distribution très différente, notamment si le point de départ se trouve dans une région à faible densité. En conséquence, une période de *burn-in* est nécessaire, où les échantillons initiaux sont rejetés (par exemple les 1000 premiers).

Toutefois, la plupart des méthodes d'échantillon de rejet souffre du fléau de la dimension, où la probabilité de rejet augmente de façon exponentielle en fonction du nombre de dimensions. Metropolis – Hastings, ainsi que d'autres méthodes MCMC, n'ont pas ce problème à un tel degré et sont souvent les seules solutions disponibles lorsque le nombre de dimensions de la distribution à échantillonner est élevée.

4.5 Application à notre projet

4.5.1 Application de l'algorithme de Metropolis–Hastings

Dans notre projet, on prend $P(x) = \exp(-\lambda d(x, x^*))$ avec $d(x, x^*)$ la distance de Hamming entre x et x^* (i.e. : le nombre de positions où les deux permutations x et x^* sont différentes).
Le mécanisme de proposition suggérer est d'inverser deux éléments d'une permutation : si on est à l'état x_t, la proposition x' est donnée en inversant deux éléments de x_t. On a donc g symétrique : la probabilité de proposer x sachant que l'on est en y est égale à la probabilité de proposer y sachant que l'on est en x.
Pour l'initialisation x_0 il suffit de prendre une permutation aléatoire.
La probabilité d'acceptation est donc égale à

$$A(x', x_t) = \min\left(1,\, \mathrm{e}^{-\lambda(\mathrm{d}(\mathrm{x}', \mathrm{x}^*) - \mathrm{d}(\mathrm{x}_t, \mathrm{x}^*))}\right)$$

Remarquons (mais je ne sais pas si c'est utile) que :

$$|d(x', x^*) - d(x_t, x^*)| \in [\![0, 2]\!]$$

Comme $e^0 = 1$, si $d(x', x^*) - d(x_t, x^*) \leq 0$, i.e. : dans la nouvelle proposition il y a au moins un élément mieux placé par rapport à l'état actuel, alors on va forcément accepter ce nouvel état.
Pour lutter contre l'autocorrélation on peut garder un échantillon sur 7 (la corrélation entre X_t et X_{t-7} est nulle). Pour éviter le *burn-in* on peut aussi enlever les 1 000 premiers échantillons.

4.5.2 Application de la méthode de cross-entropy

Qu'en est-il pour l'application de la méthode de cross-entropy avec cette loi ?

Dans cette nouvelle application on a toujours n boules et m couleurs, mais $m \geq n$ et on considère qu'une couleur ne peut apparaitre qu'une fois. La loi est permutation est :

$$\pi(x) \propto \exp\left(-\lambda d(x, x^*)\right)$$

Avec λ et x^* à déterminer.
On a $\mathcal{X} = \{(\sigma_1, \ldots, \sigma_n) \mid \sigma_i \in [\![0, m]\!] \text{ deux à deux différents}\} \subset \{1, 2, \ldots, m\}^n$
Soit $X \in \mathcal{X}$, on a :

$$f_X(x) \propto \exp\left(-\lambda d(x, x^*)\right)$$

On prend maintenant un échantillon de taille N, $X_1, \ldots, X_N \overset{i.i.d}{\sim} X$. La vraisemblance s'écrit

$$f(X_1, \ldots, X_N; \lambda, x^*) \propto \prod_{i=1}^{N} \exp\left(-\lambda d(X_i, x^*)\right)$$

Le programme de maximisation s'écrit de la façon suivante :

$$\max_{\lambda, x^*} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \ln f(X_i; \lambda, x^*) = \min_{\lambda, x^*} \sum_{i=1}^{N} \mathbb{1}_{\{S(X_i) \geq \hat{\gamma}_t\}} \lambda d(X_i, x^*)$$

La distance de Hamming n'étant pas différentiable, on ne peut trouver une réponse analytique. On va donc utiliser une méthode d'approximation pour estimer les paramètres. Comme souligné dans la partie , cela revient à calculer l'estimateur de maximum de vraisemblance sur les observations qui ont un score plus grand que $\hat{\gamma}_t$.