Chapitre Hors-Programme

Annexe

Hugo Salou MP2I

Dernière mise à jour le 24 juin 2022

TABLE DES MATIÈRES

I Algorithme de Metropolis-Hastings **2**

II Gradient automatique **5**

Première partie

Algorithme de Metropolis-Hastings

On a un texte à déchiffrer. Il a été codé par substitution : on a remplacé une lettre par une autre (par exemple, "$A \to W$" et "'$B \to U$"). L'algorithme de Metropolis-Hastings permet de déchiffrer ce type de message.

Cet algorithme fonctionne itérativement, et, pour chaque lettre, il tente de la permuter avec une autre et il conserve la meilleur.

Pour comparer deux permutations, il utilise des *bigrammes* : un couple de deux lettres consécutives (où l'on a supprimé les espaces). Avec un entrainement sur un autre texte de la même langue, l'algorithme determine la *plausibilité* de chaque bigramme. Au moment de comparer deux permutations, il compare les plausibilités des bigrammes obtenus à la suite des deux permutations.

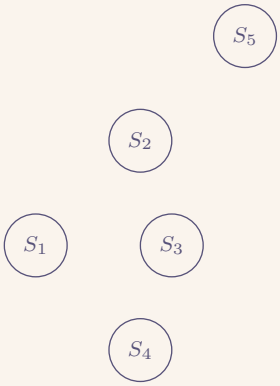
On se place dans le groupe symétrique et la fonction qui donne la plausibilité associée à une permutation est une fonction du groupe symétrique dans les réels : $p : S_n \longrightarrow \mathbb{R}$ (en français, $n = 26$).

L'algorithme ressemble donc à :

> On choisit une permutation.
>
> Si, en l'appliquant au texte, la plausibilité augmente, on conserve cette permutation. Sinon, on ne la conserve pas.
>
> Et on répète.

Cet algorithme utilise donc une marche aléatoire, il ne teste pas *toutes* les permutations (avec $n = 26$, on a $\#S_n \geqslant 10^{61}$).

Définition (Chaîne de Марков)**:** Le système est dans un état E et peut passer dans d'autres états avec des probabilités associées. Ce changement d'état ne dépend **que** de l'état actuel, et n'utilise pas les états passés.



On associe à une chaîne de Марков une matrice de transition :

$$\begin{matrix} & \overset{i}{\downarrow} & \\ \left(\right. & P(S_j \mid S_i) & \left. \right) \leftarrow j. \end{matrix}$$

On souhaite trouver $\max\limits_{x \in S} f(x)$. Pour cela, on considère le changement $x \longrightarrow x'$ et on pose $\alpha = \frac{f(x')}{f(x)}$. On ne conserve x' avec une probabilité de α, sinon, on revient à x.

Mais, la probabilité de transition est proportionnelle à $f(x')$ et donc, pour que ce choix soit marqué, il faut que la fonction f ait une grande variance.

Cette méthode de choix ne peut être appliquée que si la chaîne de Марков est symétrique et connexe. La symétrie de la chaîne de Марков est assurée par le fait qu'une permutation est bijective, on peut donc appliquer sont inverse pour revenir à un état précédent.

Deuxième partie

Gradient automatique

On considère l'ensemble

$$D = \{a + b\varepsilon \mid (a, b) \in \mathbb{R}^2\}$$

où $\varepsilon \notin \mathbb{R}$ donc $\varepsilon \neq 0$ mais $\varepsilon^2 = 0$ (similaire à i dans les complexes).

Cet ensemble peut existe : avec $D = {}^{\mathbb{R}[X]}/_{\sim}$ où

$$\begin{aligned} P \sim Q &\iff P \equiv Q \ \left[X^2\right] \\ &\iff X^2 \mid P - Q. \end{aligned}$$

On définit donc $\varepsilon = \overline{X} \neq 0 : \varepsilon^2 = \overline{X^2} = \overline{0} = 0$. En appliquant la division euclidienne de P par X^2, on a $P = X^2Q + R$ et donc $\overline{P} = \overline{R} = \overline{a + bX} = a + b\varepsilon$.

C'est comme ça que l'on a définit $\mathbb{C}$: en effet, on a $\mathbb{C} = {}^{\mathbb{R}[X]}/_{(X^2+1)}$.

On essaie de multiplier deux *duaux* avec la distributivité du produit :

$$(1 + 2\varepsilon)(3 + 4\varepsilon) = 3 + 4\varepsilon + 6\varepsilon + 8\varepsilon^2 = 3 + 10\varepsilon.$$

On considère maintenant une fonction réelle

$$\begin{aligned} f : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto 3x^2 - 5x^2 + 2x + 6 \end{aligned}$$

et on pose F l'extension de f dans l'anneau des nombres duaux.

$$\begin{aligned} F(x + \varepsilon) &= 3(x + \varepsilon)^3 - 5(x + \varepsilon)^2 + 2(x + \varepsilon) + 6 \\ &= 3(x^3 + 3x^2\varepsilon + \cancel{3x\varepsilon^2} + \cancel{\varepsilon^2}) - 5(x^2 + 2x\varepsilon + \cancel{\varepsilon^2}) + 2x + 2\varepsilon + 6 \\ &= \underbrace{(3x^3 - 5x^2 + 2x + 6)}_{f(x)} + \varepsilon \underbrace{(9x^2 - 10x + 2)}_{f'(x)} \\ &= f(x) + \varepsilon f'(x) \end{aligned}$$

On généralise : soient f et g deux fonctions définies et dérivables sur $I \subset \mathbb{R}$ à valeurs réelles. On pose

$$\forall x \in I, \forall y \in \mathbb{R}, \begin{cases} F(x + y\varepsilon) = f(x) + y\varepsilon f'(x), \\ G(x + y\varepsilon) = g(x) + y\varepsilon g'(x). \end{cases}$$

Soit $x \in I$ et $y \in \mathbb{R}$.

$$\begin{aligned} F(x + \varepsilon) + G(x + \varepsilon) &= \big(f(x) + g(x)\big) + \varepsilon\big(f'(x) + g'(x)\big); \\ F(x + \varepsilon) \times G(x + \varepsilon) &= \big(f(x) + \varepsilon f'(x)\big)\big(g(x) + \varepsilon g'(x)\big) \\ &= f(x)\, g(x) + \varepsilon\big(f'(x)\, g(x) + f(x)\, g'(x)\big) \\ &= (f \times g)(x) + \varepsilon(f \times g)'(x) \\ F\big(G(x + \varepsilon)\big) &= F\big(g(x) + \varepsilon g'(x)\big) \\ &= f\big(g(x)\big) + g'(x)\varepsilon f'\big(g(x)\big) \end{aligned}$$

Ces opérations sur les fonctions dans l'anneau dual sont compatibles avec les règles de dérivation habituelles.

Exemple:
On calcule

$$\frac{\mathrm{d}}{\mathrm{d}x}\big(e^x \cos x - \cos(\ln x)\big).$$

On pose

$$\begin{cases} F(x+y\varepsilon) = e^x + y\varepsilon e^x, \\ G(x+y\varepsilon) = \cos(x) - y\varepsilon \sin x, \\ H(x+y\varepsilon) = \ln x + \dfrac{y\varepsilon}{x}. \end{cases}$$

$$\begin{aligned} & F(x+\varepsilon)G(x+\varepsilon) - G(H(x+\varepsilon)) \\ &= (e^x + \varepsilon e^x)\big(\cos x - \varepsilon \sin x\big) - G\left(\ln x + \frac{\varepsilon}{x}\right) \\ &= e^x \cos x - \varepsilon \sin(x) e^x + \varepsilon \cos(x) e^x - \left(\cos(\ln x) - \frac{\varepsilon}{x} \times \sin(\ln x)\right) \\ &= e^x \cos(x) - \cos(\ln x) + \varepsilon \left(-\sin(x) e^x + \cos(x) e^x + \frac{1}{x}\sin(\ln x)\right) \end{aligned}$$

On peut calculer les dérivées de fonctions. À la main, ça prend autant de temps que d'utiliser les techniques habituelles. Cependant, informatiquement, le calcul de dérivée est simple ; en Python on peut le faire avec :

```
class Dual:
  def __init__(self, x, y):
    self.x = x
    self.y = y

  def __add__(self, other):
    return Dual(self.x + other.x, self.y + other.y)

  def __mul__(self, other):
    return Dual(self.x * other.x, self.x * other.y + self.y * other.x)

  def exp(self):
    return Dual(exp(self.x), self.y * exp(self.x))
```

Par example, pour chercher $\min_{x\in\mathbb{R}} \big(x^4 + 3x^3 - 5x^2 + 2\big)$, on pose $f(x_0) = f(x_0 + \varepsilon f'(x_0)$ et $x_{n+1} = x_n - hf'(x_n)$ où $h > 0$. La suite (x_n) tends vers x^* où x^* est un minimum local.

Ce calcul de dérivée est exact, on ne calcule pas $f'(x) \simeq \dfrac{f(x+h) - f(x)}{h}$.

Pour le calcul de gradient dans le cas de deux dimensions, on considère deux nombres vérifiant les propriétés de "ε" : ε_1 et ε_2.