MonteCarlo: Algorisme Metròpolis

Josep Planelles

September 11, 2020

1 Integració amb MonteCarlo

Imaginem que volem calcular la integral $I = \int_a^b f(x)dx$, on $f(x)$ és una funció contínua en $[a, b]$. Podem aproximar I per l'expressió:[1]

$$I = \int_a^b f(x)dx \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i) \tag{1}$$

on x_i és una col·lecció de nombres aleatoris uniformement distribuïts en $[a, b]$. La corresponent variància és:

$$\sigma^2 = \frac{(b-a)^2}{N} \sum_{i=1}^{N} f(x_i)^2 - I^2 \tag{2}$$

i l'error en la integració és de l'ordre de:

$$\sigma_I = \sqrt{\frac{\sigma^2}{N}} = \frac{\sigma}{\sqrt{N}} \tag{3}$$

Com la mostra (*sample*) és uniforme, les contribucions rellevants a la integral provenen d'una fracció relativament petita de punts, cosa que origina un error estadístic gran, encara que fem créixer N.

Podem minvar la variància usant l'anomenat mostreig d'importants (*importance sampling*). La idea és la següent: Imaginem que $g(x)$ és una funció de probabilitat (*probability density finction PDF*) definida en $[a, b]$ que tinga *una forma semblant* a $f(x)$, en el sentit que $\frac{f(x)}{g(x)} \approx constant$ en $[a, b]$ i que estiga normalitzada, i.e.,

$$\int_a^b g(x)dx = 1 \tag{4}$$

i que, per ser una densitat de probabilitat, $g(x) \geq 0 \ \forall x \in [a, b]$. Aleshores podem escriure:

$$I = \int_a^b f(x)dx = \int_a^b \left[\frac{f(x)}{g(x)}\right] g(x)dx \tag{5}$$

Per tant, podem calcular la integral com la mitjana de $\frac{f(x)}{g(x)}$ en una sèrie de punts aleatoris **no** uniformes, sinó distribuïts d'acord amb les probabilitats de la PDF $g(x)$.

[1] Si els punts x_i estan equi-distanciats, $h = \frac{b-a}{N}$ és el pas d'integració i la fórmula representa la integració amb rectangles. Per una altra banda, $\frac{1}{N}\sum_{i=1}^{N} f(x_i)$ representa la mitjana d'alçades de la funció, que multiplicada per l'amplada $(b-a)$ proporciona l'àrea, que no és més que la integral.

Aparentment, amb açò no hem avançat gens. Ara bé, si definim:

$$G(x) = \int_a^x g(x')dx' = r \tag{6}$$

tenim que $G(a) = 0$ (perquè no integrem res), $G(b) = 1$ (per la condició de normalitzacio) i per tant, $0 \leq r \leq 1$. Tanmateix, per derivació directa tenim que $dr = g(x)dx$ i, atès que $r = G(x)$, també $x = G^{-1}(r)$. Podem doncs transformar la integral (5) de la forma:

$$I = \int_a^b \left[\frac{f(x)}{g(x)}\right] g(x)dx = \int_0^1 \left[\frac{f(G^{-1}(r))}{g(G^{-1}(r))}\right] dr \tag{7}$$

i aproximar-la per la suma:

$$I \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(G^{-1}(r_i))}{g(G^{-1}(r_i))} \tag{8}$$

sobre una distribució *uniforme* de nombres aleatoris $r_i \in [0, 1]$, amb l'avantatge que com hem triat $g(x)$ de manera que $\frac{f(x)}{g(x)} \approx$ *constant* en $[a, b]$, l'integrand serà molt pla i, per tant, la variància quedarà molt reduïda.

Podem també interpretar aquest canvi a partir del fet que com els punts r_i estan triats d'una distribució uniforme, aleshores tindrem que $dr \approx$ *constant*, és a dir, els punts estaran aproximadament equiespaiats. Per una altra banda, $g(x) = \frac{dr}{dx}$, en conseqüència, allà on $g(x)$ (i és d'esperar que també $f(x)$) siga gran, aleshores, dx serà petit, i.e., els punts $x_i \in [a, b]$ estaran densament distribuïts. D'igual manera, allà on $g(x)$ és petita, els punts x_i estaran molt dispersos. En altres paraules, triem més punts allà on la PDF $g(x)$ és més gran.

1.1 Exemple

Volem calcular la integral,

$$I = \int_0^\pi \frac{1}{x^2 + \cos^2 x} dx \tag{9}$$

el valor *exacte* de la qual és $I = 1.58119$ (calculada amb mètodes estàndard, com el Gauss adaptatiu, inclosos en el comandament NIntegrate de Mathematica). La integració MC amb una distribució uniforme de punts en $[0, \pi]$,

$$I = \int_0^\pi f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \tag{10}$$

es tradueix amb aquest algorisme Mathematica:

```
np = 10 000; int = 0; sigma = 0;
For[i = 1, i ≤ np, i++,
  z = RandomReal[] * Pi;
  int = int + π / (Cos[z]^2 + z^2);
  sigma = sigma + (π / (Cos[z]^2 + z^2))^2];
int = int / np;
sigma = sigma / np;
sigma = (sigma - int^2) / np;
Print[int, "  ", Sqrt[sigma]]
1.58916  0.01088
```

és a dir, $I = 1.59 \pm 0.01$.

La integració MC amb una PDF $g(x) = Ae^{-\lambda x}$, que és correctament definida positiva i que normalitzem:

$$I = \int_0^\pi g(x)dx = 1 \to A = \frac{\lambda}{1 - e^{-\pi\lambda}} \tag{11}$$

A partir de la qual definim $r = G(x)$ i $x = G^{-1}(r)$:

$$r = G(x) = \int_0^x g(x')dx' = \int_0^x \frac{\lambda}{1 - e^{-\pi\lambda}} e^{-\lambda x'} dx' = \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda\pi}} \quad \to \quad x = G^{-1}(r) = -\frac{1}{\lambda}\ln[1 - r(1 - e^{-\lambda\pi})] \tag{12}$$

Amb açò calculem la mitjana

$$\int_0^\pi \frac{f(x)}{g(x)} g(x)dx = \int_0^1 \frac{f(G^{-1}(r))}{g(G^{-1}(r))} dr \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(G^{-1}(r_i))}{g(G^{-1}(r_i))} \tag{13}$$

Calculem també la variància i repetim el càlcul per a diferents valors λ trobant la màxima semblança entre $f(x)$ i $g(x)$ en $[0, \pi]$, i.e. la mínima variància, per a un valor $\lambda = 0.8$. Amb aquest valor calculem la integral amb l'algorisme Mathematica següent:

```
λ = 0.8;
np = 10000; int = 0; sigma = 0;
For[i = 1, i ≤ np, i++,
  r = RandomReal[];
  int = int + f[-1/λ Log[1 - r (1 - e^(-π λ))]] / w[-1/λ Log[1 - r (1 - e^(-π λ))]];
  sigma = sigma + (f[-1/λ Log[1 - r (1 - e^(-π λ))]] / w[-1/λ Log[1 - r (1 - e^(-π λ))]])^2];
int = int / np;
sigma = sigma / np;
sigma = (sigma - int^2) / np;
Print[int, "  ", Sqrt[sigma]]
1.58635  0.00270204
```

és a dir, $I = 1.586 \pm 0.003$.

Veiem que amb el mateix nombre de punts, l'ús de $g(x)$ ens ha pujat la precisió un ordre de magnitud.

Hi ha casos on encara se trauen més avantatges. Per exemple, si volem avaluar, $\int_0^\infty P(x)e^{-ax}dx$, on $P(x)$ és un polinomi o una fracció de polinomis. La integració sobre $x \in [0, \infty]$ obliga a triar un interval d'integració $[0, L]$, amb L molt gran ($L \gg 1$) i aixina i tot cal triar una distribució homogènia de punts en $[0, L]$ prou gran. Tanmateix, si fem ús e.g. de $g(x) = Ae^{-\lambda x}$, la normalització de $g(x)$ condueix a $g = \lambda e^{-\lambda x}$ i des de $r = G(x) = \int_0^x g(x')dx' = 1 - e^{-\lambda x}$ trobem $x = G^{-1}(r) = -\frac{1}{\lambda}\ln(1 - r)$ i aleshores aproximem la integral per la suma,

$$\frac{1}{N}\sum_{i=1}^{N} \frac{f(G^{-1}(r_i))}{g(G^{-1}(r_i))} \tag{14}$$

amb $r_i \in [0, 1]$, que no suposa cap repte especial respecte de la primera de les integrals calculades.

Malauradament, el canvi de variables $x = G^{-1}(r)$ és en general difícil (sinó impossible) en casos multidimensionals (que és on MC trau avantatge als mètodes tradicionals d'integració com el mètode dels trapezis o l'adaptatiu de Gauss etc.). Per tant, cal cercar alternatives.

2 Algorisme Metròpolis

El mètode que hem emprat en la secció anterior comporta dues etapes bàsiques: (a) la generació de nombres aleatoris x_i sobre el volum d'integració (d'acord amb una PDF $g(x)$ determinada) i (b) l'avaluació de $\frac{f(x)}{g(x)}$ en aquests punts.

La segon etapa és immediata mentre que per a la primera havíem trobat una solució cercant en una distribució uniforme de nombres aleatoris $r_i \in [0,1]$ i usant la funció inversa $x = G^{-1}(r)$ per robar la mostra requerida.

Si trobar $G^{-1}(r)$ no està al nostre abast, caldrà cercar una alternativa i una molt bona alternativa és l'algorisme Metròpolis. Descriurem primer l'algorisme i després justificarem que proporciona una mostra de valors aleatoris ajustats a la PDF $g(x)$ triada.

L'algorisme Metròpolis comporta les seguents etapes:

- Triem un pas h.
- A partir d'un punt x_i, amb el pas h, generem x_t amb un random uniforme dins l'interval $[x_i - \frac{h}{2}, x_i + \frac{h}{2}]$.
- Calculem la ratio $r = \frac{g(x_t)}{g(x_i)}$.
- Si $r \geq 1$ acceptem $x_{i+1} = x_t$.
- Si $r < 1$ acceptem $x_{i+1} = x_t$ amb una probabilitat r. És a dir, triem un nombre aleatori d'una distribució uniforme: $\eta \in [0,1]$. Aleshores, si $r > \eta$ acceptem $x_{i+1} = x_t$, però si $r \leq \eta$ el rebutgem, i.e., fem que $x_{i+1} = x_i$,

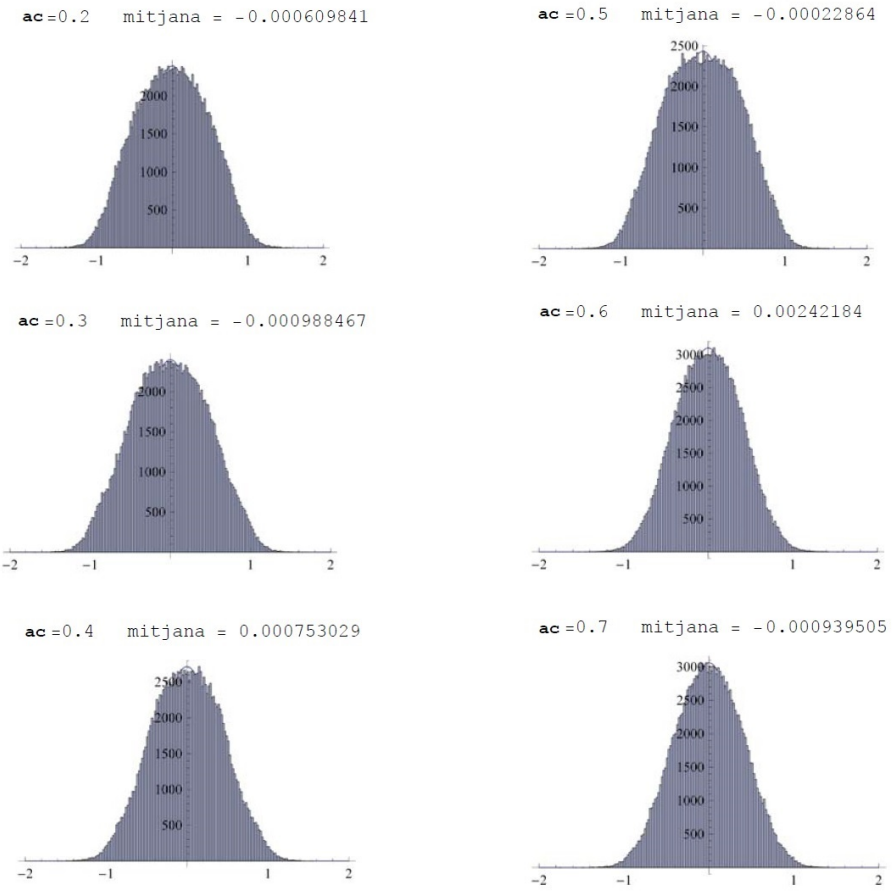
i d'aquesta manera generem una seqüència $\{x_0, x_1, \ldots, x_i, \ldots, x_N\}$ anomenada *random walk*. Demostrarem que aquesta llista es correspon amb una distribució aleatòria de números que presenta una densitat de distribució de probabilitat $g(x)$.

Hi ha dos aspectes clau a tenir en compte. En primer lloc la tria de x_0 (el primer element de la sèrie). En segon lloc la tria del pas h. Respecte de la tria de x_0, un punt de partida raonable seria un punt on la PDF $g(x)$ siga gran (i.e., un punt altament probable). Tanmateix, en problemes multidimensionals, el màxim de $g(x)$ no se coneix o no és fàcil de determinar. Aleshores procedim a la *termalització* del *walker* abans de començar el càlcul. És a dir, partim d'un punt arbitrari i deixem que la successió de probabilitats i tries el facin caminar cap a un punt d'alta probabilitat, a la vegada que li eliminem qualsevol dependència (o biaix) del punt triat de partida.

El càlcul adquireix robustesa si incloem no únicament un *walker*, sinó una llarga sèrie de *walkers* independents i fem el càlcul superposant les diferents distribucions de nombres aleatoris (les diferents sèries de punts dels diferents *random walks*).

Respecte del pas h, aquest cal triar-lo amb compte. Imaginem que $g(x_i)$ siga màxim. Aleshores, si h és excessivament llarg ocurrirà que $g(x_t) \ll g(x_i)$ i la majoria d'etapes seran rebutjades, de manera que fem una mostra molt poc eficient de $g(x)$. Si h se tria massa petit, la majoria de les etapes seran acceptades, però el *walker* a penes es mourà de la regió on està, deixant molt de volum per explorar i generant, per tant, una mostra (*sample*) dolenta.

La proposta més emprada és la tria d'h de manera que la meitat de les propostes s'accepten i la meitat se rebutgen,[1] i.e. una *acceptace ratio* $ac = 0.5$. Així el procés és suficientment eficient a la vegada que permet una exploració raonablement exhaustiva del volum d'integració. Cal dir que hom pot trobar diferents propostes per a ac en la literatura, des de valors menors a 0.5, entre 0.25 a 0.4,[3] a valors superiors a aquest valor central, com ara 0.6.[2] És clar que si el nombre de punts N creix suficientment, les diferents ac convergeixen a la mateixa solució. El que cal fer és explorar la ac que, per al problema que s'està estudiant, se demostre més eficient. A manera d'exemple mostrem el càlcul del valor mitjà de la posició i les distribucions aproximades que l'algorisme Metròpolis atorga a una PDF gaussiana (que podria correspondre a l'estat fonamental de l'oscil·lador harmònic) per a distints valors de l'*acceptance ratio* i a continuació el codi mathematica que les genera:



```
ClearAll["Global`*"];
walkers = 20;
For [ac = 0.2, ac ≤ 0.8, ac += 0.1,
 r0 = 1;
 delr = r0 / 2;
 p[x_] = Exp[-2 * x^2] ^2;
 m = 10 000;
 llista2 = {};
 For[walk = 1, walk ≤ walkers, walk++,
  x0 = r0 * (2 * RandomReal[] - 1);
  p0 = p[x0];
  (*thermalization*)
  cont = 0;
  For[i = 1, i ≤ 0.2 * m, i++,
   cont = cont + 1;
   x1 = x0 + delr * (2 * RandomReal[] - 1);
   p1 = p[x1];
   w = RandomReal[];
   If[p1 / p0 > w > 0, x0 = x1, i = i - 1];
  ];
  delr = delr * 0.2 * m / (ac * cont);
  (*calculation*)
  i2 = i + 1;
  cont = 0;
  For[i = i2, i ≤ m, i++,
   cont = cont + 1;
   x1 = x0 + delr * (2 * RandomReal[] - 1);
   p1 = p[x1];
   w = RandomReal[];
   If[p1 / p0 > w , llista2 = AppendTo[llista2, x1]; x0 = x1, i = i - 1];
  ];
 ];
 height = Max[HistogramList[llista2]];
 f1 = Histogram[llista2];
 f2 = Plot[height / p[0] * p[x], {x, -2, 2}, PlotRange → All];
 Print["ac=", ac, "    ", "mitjana = ", Total[llista2] / Length[llista2]];
 Print[Show[f2, f1]];
]
```

2.1 Justificació de l'algorisme Metròpolis

Per justificar que Metròpolis genera una seqüència de punts distribuïts d'acord amb $g(x)$ considerem un nombre gran de *walkers* començant itineraris en diferents punts i movent-se de manera independent a través de l'espai X. Anomenem $N_n(X)$ al nombre, o millor, a la densitat de *walkers* que han aplegat a X després de n etapes. Aleshores, el nombre net de *walkers* que aniran del punt X al punt Y serà:

$$\Delta N(X) = N_n(X)P(X \to Y) - N_n(Y)P(Y \to X) = N_n(Y)P(X \to Y)\left[\frac{N_n(X)}{N_n(Y)} - \frac{P(Y \to X)}{P(X \to Y)}\right] \tag{15}$$

on $P(X \to Y)$ representa la probabilitat de que el *walker* transite des de X fins a Y en una etapa.

En equilibri $\Delta N(X) = 0$, per tant,

$$\frac{N_e(X)}{N_e(Y)} = \frac{P(Y \to X)}{P(X \to Y)} \tag{16}$$

Si el sistema no està en equilibri, $\Delta N(X) > 0$ vol dir que hi ha massa *walkers* en X. En altres paraules que:

$$\frac{N_n(X)}{N_n(Y)} > \frac{P(Y \to X)}{P(X \to Y)} = \frac{N_e(X)}{N_e(Y)} \tag{17}$$

i que, aleshores, hi ha un trànsit $X \to Y$ que s'aproximarà a la distribució d'equilibri. Un raonament semblant pot fer-se en el cas $\Delta N(X) < 0$.

Queda per provar que la seqüència de tries que fa Metròpolis condueix a la distribució d'equilibri dels *walkers*, és a dir, $N_e(X) \sim g(x)$. Amb aquesta finalitat factoritzem la probabilitat del trànsit $X \to Y$ com el producte de la probabilitat $T(X \to Y)$ de fer la tria de l'etapa $X \to Y$ per la probabilitat $A(X \to Y)$ d'acceptar-la.

Si partint de X podem aplegar a Y en una etapa, la recíproca també serà certa i, aleshores, com no hi ha situacions privilegiades, haurà de succeir que $T(X \to Y) = T(Y \to X)$, de manera que per a la distribució d'equilibri:

$$\frac{N_e(X)}{N_e(Y)} = \frac{P(Y \to X)}{P(X \to Y)} = \frac{A(Y \to X)}{A(X \to Y)} \tag{18}$$

Ara bé, en Metròpolis, si $g(x) > g(y)$ aleshores $A(Y \to X) = 1$ i $A(X \to Y) = \frac{g(y)}{g(x)}$ i si $g(x) < g(y)$ aleshores $A(Y \to X) = \frac{g(x)}{g(y)}$ i $A(X \to Y) = 1$. Per tant, en qualsevol dels dos casos tenim que:

$$\frac{N_e(X)}{N_e(Y)} = \frac{A(Y \to X)}{A(X \to Y)} = \frac{g(x)}{g(y)} \tag{19}$$

que vol dir que els *walkers* se distribueixen d'acord a $g(x)$.

References

[1] S.E. Kooning and D.C. Meredith, *Computational Physics Fortran Version*, Westview Press, USA 1990.

[2] V. Apaja, *Quantum Monte Carlo*, University of Jyväskylä, 2018.

[3] D. Luengo, L. Martino, M. Bugallo, V. Elvira and S. Särkkä *EURASIP Journal on Advances in Signal Processing* (2020) 2020:25