Optimization Methods III.

The MCMC. Exercises.

Anatoli Iambartsev
IME-USP

[RC] A generic Markov chain Monte Carlo algorithm.

The Metropolis-Hastings algorithm associated with the objective (target) density f and the conditional density q produces a Markov chain $(X(t))$ through the following transition kernel:

Algorithm Metropolis-Hastings Given $x^{(t)}$,

1. Generate $Y_t \sim q(y \mid x^{(t)})$.

2. Take

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \rho(x^{(t)}, Y_t), \\ x^{(t)} & \text{with probability } 1 - \rho(x^{(t)}, Y_t), \end{cases}$$

where

$$\rho(x, y) = \min\left\{ \frac{f(y)}{f(x)} \frac{q(x \mid y)}{q(y \mid x)}, 1 \right\}.$$

[RC] A generic Markov chain Monte Carlo algorithm.

A generic R implementation is straightforward, assuming a generator for $q(y|x)$ is available as `geneq(x)`. If `x[t]` denotes the value of $X(t)$,

```
> y=geneq(x[t])
> if (runif(1)<f(y)*q(y,x[t])/(f(x[t])*q(x[t],y))){
+    x[t+1]=y
+    }else{
+      x[t+1]=x[t]
+      }
```

since the value y is always accepted when the ratio is larger than one.

[RC] Example 6.1.

We can use a Metropolis-Hastings algorithm to simulate a beta distribution, where the target density f is the $Beta(2.7, 6.3)$ density and the candidate q is uniform over $[0, 1]$, which means that it does not depend on the previous value of the chain. A Metropolis-Hastings sample is then generated with the following R code:

```
> a=2.7; b=6.3; c=2.669 # initial values
> Nsim=5000
> X=rep(runif(1),Nsim)  # initialize the chain
> for (i in 2:Nsim){
+   Y=runif(1)
+   rho=dbeta(Y,a,b)/dbeta(X[i-1],a,b)
+   X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
+   }
```

[RC] Example 6.1.

```
> ks.test(jitter(X),rbeta(5000,a,b))

Two-sample Kolmogorov-Smirnov test

data: jitter(X) and rbeta(5000, a, b)

D = 0.031, p-value = 0.01638

alternative hypothesis: two-sided
```

jitter() - Add a small amount of noise to a numeric vector: we need this function because of the presence of repeated values in the vector X, KS test works well when the samples are from continuous r.v. what supposes the different values within both samples, i.e. `ks.test` do not accept ties..

[RC] The independent Metropolis-Hastings algorithm.

The Metropolis-Hastings algorithm allows a candidate distribution q that only depends on the present state of the chain. If we now require the candidate q to be independent of this present state of the chain (that is, $q(y|x) = g(y)$), we do get a special case of the original algorithm:

Algorithm Metropolis-Hastings Given $x^{(t)}$,

1. Generate $Y_t \sim g(y)$.

2. Take

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \min\left\{\frac{f(Y_t)g(x^{(t)})}{f(x^{(t)})g(Y_t)}, 1\right\}, \\ x^{(t)} & \text{otherwise.} \end{cases}$$

[RC] The independent Metropolis-Hastings algorithm.

This method then appears as a straightforward generalization of the Accept-Reject method in the sense that the instrumental distribution is the same density g as in the Accept-Reject method. Thus, the proposed values Y_t are the same, if not the accepted ones.

Metropolis-Hastings algorithms and Accept-Reject methods (Section 2.3), both being generic simulation methods, have similarities between them that allow comparison, even though it is rather rare to consider using a Metropolis-Hastings solution when an Accept-Reject algorithm is available.

[RC] The independent Metropolis-Hastings algorithm.

a. The Accept-Reject sample is iid, while the Metropolis-Hastings sample is not. Although the Y_ts are generated independently, the resulting sample is not iid, if only because the probability of acceptance of Y_t depends on $X(t)$ (except in the trivial case when $f = g$).

b. The Metropolis-Hastings sample will involve repeated occurrences of the same value since rejection of Y_t leads to repetition of $X(t)$ at time $t + 1$. This will have an impact on tests like `ks.test` that do not accept ties.

c. The Accept-Reject acceptance step requires the calculation of the upper bound $M \geq \sup_x f(x)/g(x)$, which is not required by the Metropolis-Hastings algorithm. This is an appealing feature of Metropolis-Hastings if computing M is time-consuming or if the existing M is inaccurate and thus induces a waste of simulations.

[RC] The independent Metropolis-Hastings algorithm. Exercise 6.3.

Compute the acceptance probability $\rho(x, y)$ in the case $q(y|x) = g(y)$. Deduce that, for a given value $x^{(t)}$, the Metropolis-Hastings algorithm associated with the same pair (f, g) as an Accept-Reject algorithm accepts the proposed value Y_t more often than the Accept-Reject algorithm.

Exercise 6.3

When $q(y|x) = g(y)$, we have

$$\begin{aligned}\rho(x,y) &= \min\left(\frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}, 1\right)\\ &= \min\left(\frac{f(y)}{f(x)}\frac{g(x)}{g(y)}, 1\right)\\ &= \min\left(\frac{f(y)}{f(x)}\frac{g(x)}{g(y)}, 1\right).\end{aligned}$$

Since the acceptance probability satisfies

$$\frac{f(y)}{f(x)}\frac{g(x)}{g(y)} \geq \frac{f(y)/g(y)}{\max f(x)/g(x)}$$

it is larger for Metropolis–Hastings than for accept-reject.

[RC] The M-H algorithm. Symmetric conditional.

Here we simulate Y_t according $Y_t = X^{(t)} + \varepsilon_t$, where ε_t is a random perturbation with distribution g independent of $X(t)$, for instance a uniform distribution or a normal distribution, meaning that $Y_t \sim U[X^{(t)} - \delta, X^{(t)} + \delta]$ or $Y_t \sim N(X^{(t)}, \tau^2)$ in unidimensional settings. The proposal density $q(y \mid x)$ is now of the form $g(y - x)$. The Markov chain associated with q is a random walk, when the density g is symmetric around zero; that is, satisfying $g(-t) = g(t)$. But, due to the additional Metropolis-Hastings acceptance step, the Metropolis-Hastings Markov chain $\{X(t)\}$ is not a random walk.

[RC] The M-H algorithm. Random walk.

Algorithm Random Walk Metropolis-Hastings

Given $x^{(t)}$,

1. Generate $Y_t \sim g(y - x^{(t)})$.
2. Take

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \min\left\{\frac{f(Y_t)}{f(x^{(t)})}, 1\right\}, \\ x^{(t)} & \text{otherwise.} \end{cases}$$

[RC] The M-H algorithm. Random walk.

Example 6.4. The historical example of Hastings (1970) considers the formal problem of generating the normal distribution $N(0,1)$ based on a random walk proposal equal to the uniform distribution on $[-\delta, \delta]$. The probability of acceptance is then

$$\rho(x^{(t)}, y_t) = \exp\left\{\frac{(x^{(t)})^2 - y_t^2}{2}\right\} \wedge 1.$$

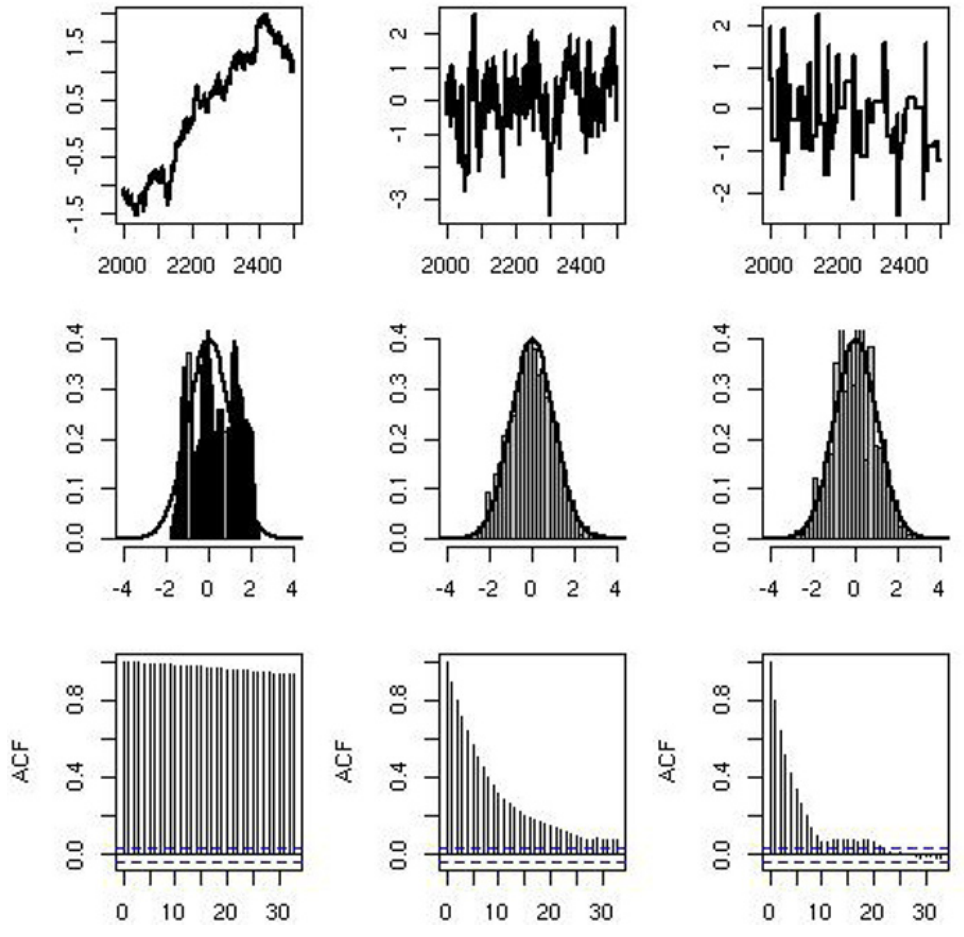

Fig. 6.7. Outcomes of random walk Metropolis–Hastings algorithms for Example 6.4. The left panel has a $\mathcal{U}(-.1, .1)$ candidate, the middle panel has $\mathcal{U}(-1, 1)$, and the right panel has $\mathcal{U}(-10, 10)$. The upper graphs represent the last 500 iterations of the chains, the middle graphs indicate how the histograms fit the target, and the lower graphs give the respective autocovariance functions.

[RC] The M-H algorithm. Random walk.

Figure describes three samples of 5000 points produced by this method for $\delta = 0.1, 1$, and 10 and clearly shows the difference in the produced chains: Too narrow or too wide a candidate (that is, a smaller or a larger value of δ) results in higher auto-covariance and slower convergence. Note the distinct patterns for $\delta = 0.1$ and $\delta = 10$ in the upper graphs: In the former case, the Markov chain moves at each iteration but very slowly, while in the latter it remains constant over long periods of time.

As noted in this formal example, calibrating the scale δ of the random walk is crucial to achieving a good approximation to the target distribution in a reasonable number of iterations.

References.

[RC] Cristian P. Robert and George Casella. *Introducing Monte Carlo Methods with R*. Series "Use R!" . Springer

[RC1] Cristian P. Robert and George Casella. *Introducing Monte Carlo Methods with R Solutions to Odd-Numbered Exercises*.