A Discrete Investigation into Markov Chain Monte Carlo

Introduction

Markov chain Monte Carlo (MCMC) is a technique for estimating probabilistic quantities that are too hard to compute directly. The first MCMC application [MRRTT] studied molecules in motion under the physical constraint that two molecules cannot occupy the same position. The constraint made computing the normalizing constant of a probability distribution intractable.

Suppose that we have a distribution q that is not necessarily normalized. That is, $q(y)/q(x)$ is the ratio of the probabilities of x and y but q may not sum to one. MCMC is an algorithm that samples from the distribution q. MCMC first constructs a Markov chain (the base Markov chain) on the state space of x and y. Then MCMC modifies the Markov chain's transition probabilities p_{xy} according to a certain function of p_{xy}, p_{yx}, $q(x)$, and $q(y)$. The result is a new Markov chain whose stationary distribution is q. If we run this new Markov chain for a long time, its path constitutes samples (not independent) from the distribution q.

Although I could verify that MCMC worked, I could not at first see the motivation behind the form of MCMC. This note is my attempt to explain why MCMC has the form it does. To keep things simple, I will work only with a Markov chain on a finite state space $S = \{0, 1, \dots, d-1\}$, where we can represent distributions and transition probabilities by vectors and matrices. S is the integers modulo d as a set, and all our examples will go one step further and treat state $d-1$ as immediately to the left of 0. Thus, a step to the right of $d-1$ lands at 0.

Accordingly, from now on, write q_x for $q(x)$ and think of vectors such as q as *row vectors*. Let $P = (p_{xy})$, the matrix of transition probabilities of the base Markov chain..

MCMC Works

The base Markov chain X on S is defined in terms of its transition probabilities

$$p_{xy} = P(X_{t+1} = y | X_t = x)$$

We need to make several technical assumptions about X:

1. X is homogeneous in time: p_{xy} does not depend on t.
2. X is irreducible, which means that there is a positive probability of visiting any state (possibly after more than one jump) from a given state. If X were to move around in a proper subset of S, it could not sample from q unless q had probability only on S.
3. X is not periodic. A state x is periodic if there is some integer t such that X returns to x with positive probability at times that are multiples of t and at no other times [Feller, XV.5].

The MCMC algorithm changes the transition probabilities p_{xy} to new transition probabilities p'_{xy} according to the following recipe when $\neq y$:

$$p'_{xy} = p_{xy} \min\left(1, \frac{q_y p_{yx}}{q_x p_{xy}}\right) = \frac{\min(q_x p_{xy}, q_y p_{yx})}{q_x} \quad (1)$$

MCMC then adjusts the diagonal terms p'_{xx} of the matrix $P' = (p'_{xy})$ to make the rows sum to one.

Notice that the numerator $k(x,y) = \min(q_x p_{xy}, q_y p_{yx})$ of the right-hand side of (1) is a symmetric function of x and y and that $k(x,y) = q_x p'_{xy}$. Since $k(y,x) = q_y p'_{yx}$ symmetry implies that

$$q_y p'_{yx} = q_x p'_{xy} \quad (2)$$

Equation (2) is called the *local balance equation*; summing both sides over y leads to the matrix equation

$$qP' = q$$

which says that q is the stationary distribution of P'.

Why MCMC Works

The preceding section shows that, with P' defined as in (1), q is its stationary distribution. But where did (1) come from? Our goal is to derive this recipe from some basic principles.

We again start with the base Markov chain X as in the preceding section. Suppose we build X with q as its initial distribution. That is, $P(X(0) = x) = q_x$. If we were extremely lucky, then X itself would satisfy local balance with q, $q_y p_{yx} = q_x p_{xy}$, and q would be its stationary distribution. But if we are not so lucky, then sometimes we will find $q_y p_{yx} \neq q_x p_{xy}$ and we will have to tweak the p_{xy} to eliminate such differences.

What does a difference $q_y p_{yx} \neq q_x p_{xy}$ mean? Writing out $q_x p_{xy}$, we have

$$q_x p_{xy} = P(X_1 = y | X_0 = x)\, P(X_0 = x) = P(X_1 = y, X_0 = x)$$

Similarly,

$$q_y p_{yx} = P(X_1 = x, X_0 = y)$$

One name for $f_{xy} = P(X(1) = y, X(0) = x) = q_x p_{xy}$ is the "flux" from x to y. Think of each state x initially having an amount q_x of a substance. In one time step, a fraction p_{xy} of that substance moves from x to y. The flux f_{xy} is the amount that moves. When $f_{xy} = f_{yx}$ two equal transfers cancel one another out and the amount q in each state does not change. When the fluxes in the two directions are unequal, $f_{xy} \neq f_{yx}$, and $P(X_t = x)$ changes in each time step.

We have to adjust the transition rates to equalize the fluxes and make local balance hold. Suppose, for example, that $f_{xy} < f_{yx}$. To make the two fluxes equal, we can increase p_{xy}, decrease p_{yx}, or do a combination of both.

If we increase transition probabilities, then some row sums in the transition matrix may exceed one and we may have to renormalize them to one by division. On the other hand, if we only decrease (off-diagonal) transition probabilities, the altered rows sum to less than one, a situation we can correct by adding the deficit to the diagonal term. The latter is the approach MCMC uses and we shall follow it.

Do not increase any p_{xy}. If $f_{xy} < f_{yx}$ set $p'_{xy} = p_{xy}$ and decrease p_{yx} to equalize the fluxes.

As a consequence of this policy, $p'_{xy} \leq p_{xy}$ for all $x \neq y$. Multiplying this inequality by q_x we get

$$q_x p'_{xy} \leq q_x p_{xy} = f_{xy} \tag{3}$$

Swapping x and y leads to

$$q_y p'_{yx} \leq q_y p_{yx} = f_{yx} \tag{4}$$

Since our intent is to have p'_{xy} satisfy local balance, the left hand sides of the two preceding equations will be equal.

Now suppose $f_{xy} \leq f_{yx}$. Then, by our policy, $p'_{xy} = p_{xy}$. From local balance and this equality, p'_{yx} satisfies

$$q_y p'_{yx} = q_x p'_{xy} = q_x p_{xy}$$

Therefore,

$$p'_{yx} = \frac{q_x p_{xy}}{q_y} = \frac{q_x p_{xy}}{q_y p_{yx}} = p_{yx} \frac{f_{xy}}{f_{yx}}.$$

This argument shows that $p'_{xy} = p_{xy} \min(1, \frac{f_{yx}}{f_{xy}})$, which is the same as (1).

Python Illustrations

The accompanying code "MCMC Investigation.ipynb" illustrates MCMC and its behaviors. To start, choose `q = t.geometric()` in Target_selector. This makes the target distribution q a geometric distribution whose probability $q(x)$ doubles each time x increases by 1.

1. The original MCMC algorithm assumed that the base Markov chain was symmetric, i.e., $p_{xy} = p_{yx}$. [Hastings] generalized MCMC beyond the symmetric case. However, some conditions on p_{xy} and p_{yx} are still necessary. For example, If $p_{xy} > 0$ but $p_{yx} = 0$, then the rightmost member of (1) shows that $p'_{xy} = p'_{yx} = 0$. As an extreme example, choose

`P = s.make_P_matrix('clock_tick')` in the Python code to make the base Markov chain move one step to the right in each time slot (like the second hand on a clock). The transition probabilities are $p_{xy} = 1$ when $y = x + 1$ modulo d and zero otherwise. The resulting MCMC chain's transition matrix is the identity matrix! MCMC cannot decrease p_{yx} because $p_{yx} = 0$ already when $y > x$. As an aside, note that the clock tick chain is periodic, but we could modify with variable size jumps instead of constant size jumps to the right.

2. Now choose `P = s.make_P_matrix('uniform')`. Compare the forms of P and P_mcmc. The above diagonal entries in both matrices are the same, in accordance with our policy that we do not change p_{xy} if $f_{xy} < f_{yx}$ (we chose q so that $q(x) < q(y)$ if $x < y$ and P is symmetric, so $f_{xy} < f_{yx}$ is the same as $q(x) < q(y)$, which is the same as $x < y$). Thus, MCMC leaves the above diagonal entries unchanged and decreases the below diagonal entries. The probabilities of transition below the diagonal are smaller in P_mcmc than in P because MCMC shifts probability from positive below diagonal terms to the diagonal. MCMC reduces flux from some states to other states by increasing flux to themselves.
3. The changes above and below the diagonal are more apparent if one chooses
    a. `q = t.factorial()` in Target_selector and
    b. `P = s.make_P_matrix('random_walk)` in P.selector.
4. The function `def P_mcmc(P,q)` is the matrix version of MCMC. Steps a and b below implement (1) and step c adds to the diagonal terms of D to make the row sums of P' equal 1.
    a. $Q = \text{diag}(q)$
    b. $D = Q^{-1}\min(QP, (QP)^T)$
    c. $P' = D + I - \text{diag}(De)$, e a vector of all ones.
5. Again choose `q = t.geometric()` and compare
    a. `P = s.make_P_matrix('uniform')` to
    b. `P = s.make_P_matrix('random_walk)`

Convergence is much faster with a than with b because uniform mixes much faster than random walk.

[MRRTT] by N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys*., Vol. 21, (1953), pp. 1087-92.

[Feller] W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley and Sons, 1968.

[Hastings] W. K. Hastings, Monte Carlo Sampling Methods Using Markov Chains and Their Applications, *Biometrika*, Vol. 57, No. 1. (Apr., 1970), pp. 97-109.