The Propp-Wilson Algorithm

Background

Last lecture we talked about fast convergence for MCMC algorithms, and gave a result regarding the systematic sweep Gibbs sampler for graph q-colorings.

Another way to address the problem that we in general don't know how far from the target distribution our sampling distribition is is to use perfect sampling, or perfect simulation. In perfect simulation we get samples from the correct distribution, our asmpling distribution and the target distribution are the same. There are at least two simulation schemes for generating perfect samples, and one of them is the Propp Wilson algorithm, the other one is Fills algorithm.

Perfect simulation solves both problems accounted for during the last lecture, regarding convergence rates and total variation distance to the stationary distribution.

Basic idea

The basic idea behind the algorithm is to use coupling in a certin way. We start many different Markov chains, one in every possible state of the state space, and run them for a fixed time. If we run the chain long enough finally all chains has emerged into the same and will stay the same (if we couple them properly). Now consider another chain starting in parallell coupled like the other chain, but this one started according to the stationary distribution. This one will still have the stationary distribution after they all emerged, making that remaining chain also running according to the stationary distribution.

The algorithm

Given a distribution π on a finite set $S = \{s_1, ..., s_k\}$ we do the following.

- Construct an irreducible and aperiodic Markov chain with stationary distribution π and transition matrix P.
- Construct an update function, $\phi : S \times [0, 1] \to S$, according to chapter 3 in the course book.
- Get a sequence $(U_0, U_{-1}, U_{-2}, ...)$ of i.i.d. $U[0, 1]$ random numbers.
- Get an increasing sequence of integers $(N_1, N_2, ...)$.
- Run the following procedure
  1. Let $m = 1$
  2. For each $s \in \{s_1, ..., s_k\}$ simulate a Markov chain starting at time $-N_m$ in state s and running up to time 0 using random numbers $U_{-N_m+1}, U_{-N_m+2}, U_{-1}, U_0$ (these are the same for all k chains).
  3. If all chains in step 2 end up in the same state s' at time 0, then stop and output s', otherwise continue with step 4.
  4. Increase m by one, and continue with step 2.

It's important that we reuse the old random numbers, that is, when running the chain from time $-N_{m+1}$ up to 0 we must use *the same random numbers* at times $-N_m, -N_m + 1, ..., 1, 0$ as we did in the last run. This means that we have to save more and more used random numbers as we start the chain further back into the past, a bit cumbersome but necessary to get the algorithm to generate samples without bias.

An example

Consider an example with times $(N_1, N_2, ..) = (1, 2, 4, ...)$ and state space $S = \{s_1, s_2, s_3\}$. We start by letting $m = 1$ and run the chain for one step to time 0. Since

$$\begin{aligned}\phi(s_1, U_0) &= s_1\\ \phi(s_2, U_0) &= s_2\\ \phi(s_3, U_0) &= s_1\end{aligned}$$

the chains has not coalesced we start all over again and this time let $m = 2$. This time we does not get coalescence either.

$$\begin{aligned}\phi(\ \phi(s_1, U_0),\ U_{-1}) &= \phi(s_2, U_0) = s_2\\ \phi(\ \phi(s_2, U_0),\ U_{-1}) &= \phi(s_3, U_0) = s_1\\ \phi(\ \phi(s_3, U_0),\ U_{-1}) &= \phi(s_2, U_0) = s_2\end{aligned}$$

We let $m = 4$ and start all over again.

$$\begin{aligned}\phi(\ \phi(\ \phi(\ \phi(s_1,\ U_{-3}),\ U_{-2}),\ U_{-1}),\ U_0) &= s_2\\ \phi(\ \phi(\ \phi(\ \phi(s_2,\ U_{-3}),\ U_{-2}),\ U_{-1}),\ U_0) &= s_2\\ \phi(\ \phi(\ \phi(\ \phi(s_3,\ U_{-3}),\ U_{-2}),\ U_{-1}),\ U_0) &= s_2\end{aligned}$$

This time we get coalescence. We stop and output s_2 as our sample from the stationary distribution.

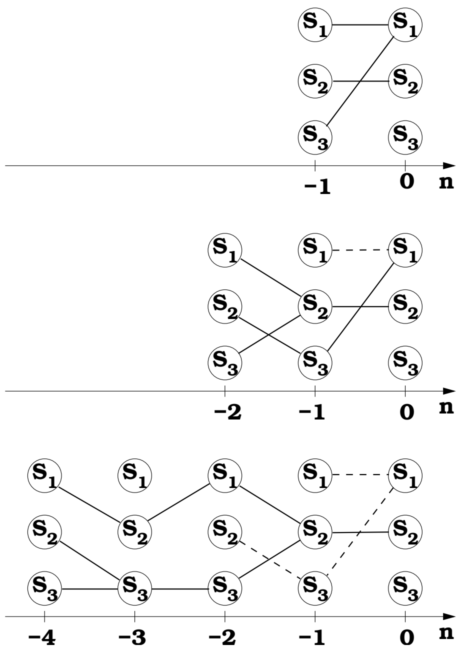

Figure 10.1: An example of a the Propp-Wilson algorithm on s small state space.

Two comments regarding the distribution of the output . . .

1. If we start these chains further back into the pas according to the algorithm they will, regardless of the starting tim, be in state s_2 at time 0. Thus we can regard s_2 as the final state of a chain simulated from time $-\infty$ up to time 0, and that will be distributed according to the stationary distribution.

2. Consider another chain stating in parallell with these three chains, but where the initial state, X'_{-4}, is generated according to the stationary distribution. They of course $X'_0 = s_2$ will also be according to the stationary distribution.

Problems

Why not couple into the future? Instead of running the chains from times further and further back in the past we could run it from time 0 until the all coalesce and then output the result ? Suppose that two chains each starting in s_1 and s_2

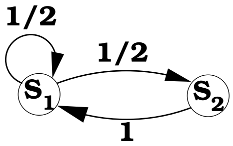

Figure 10.2: In general we cannot use "coupling into the fuiture".

respectively, coalesce at time n. Consider where they were at time $n-1$. Since they have not yet met they are in different states. The one in state s_2 at time $n-1$ must however be in state s_1 at time n since the transition matrix has the form following form.

$$P = \begin{pmatrix} 1/2 & 1/2 \\ 1 & 0 \end{pmatrix}$$

So this algorithm will output s_1 with probability 1. However the chain has a reversible and stationary distribution

$$\pi = \left(\frac{2}{3}, \frac{1}{3}\right)$$

This is a counter example illustrating what can go wrong whenever we couple into the future.

Large state spaces the algorithm prescribes that we simulate a chain for every state in the state space. This is a problem when the state space is very large as in the q-coloring case for graphs. There are techniques around this problem, on is presented in chapter 11 in the course book, and is also the topic for the next lecture.

A lot of random number to save If we are forced to run simulate the chain from a very distant past the saving of old random numbers could be a problem. There is however a version of the Propp-Wilson algorithm that only uses every random number once, thus avoid saving a lot of numbers. It is a bit more complicated that the ordinary algorithm. The course book treart this algorithm in chapter 12 but we do not look futher into this in the course.