CS340 Fall 2006: Homework 6

Out Mon 30 Oct, back Mon 6 Nov

1 Simulated annealing

1. Modify the function `MH.m` so it does simulated annealing instead of sampling from the target distribution. Specifically, the interface to your function should be as follows

```
function [samples, naccept] = SA(target, proposal, xinit, Nsamples,...
                   targetArgs, proposalArgs, proposalProb, temp);
% Simulated annealing algorithm
%
% Inputs (similar to MH.m)
% target returns the unnormalized log posterior, called as 'p = exp(target(x, targetArgs{:}))'
% proposal is a fn, as 'xprime = proposal(x, proposalArgs{:})' where x is a 1xd vector
% xinit is a 1xd vector specifying the initial state
% Nsamples - total number of samples to draw
% targetArgs - cell array passed to target
% proposalArgs - cell array passed to proposal
% proposalProb  - optional fn, called as
%     'p = proposalProb(x,xprime, proposalArgs{:})',
%   computes q(xprime|x). Set to [] if proposal is symmetric.
% temp(s) = temperature at step s
% initTemp - initial temperature, defaults to 1
% coolingFactor - temp(t) = temp(t-1)*coolingFactor, defaults to 0.995
%
% Outputs
% samples(s,:) is the s'th sample (of size d)
% naccept = number of accepted moves
```

2. Use the provided function `SAdemoMOG.m` to test your code. You should get something that looks like Figure 1.

3. Modify the demo so it finds the global optimum of the 2D surface shown in Figure 2. This surface be computed using

```
Z = peaks;
```

and returns a 49×49 matrix, where $Z(i, j)$ is the function value at location i, j. You will need to use a 2D proposal distribution and a suitable cooling schedule.

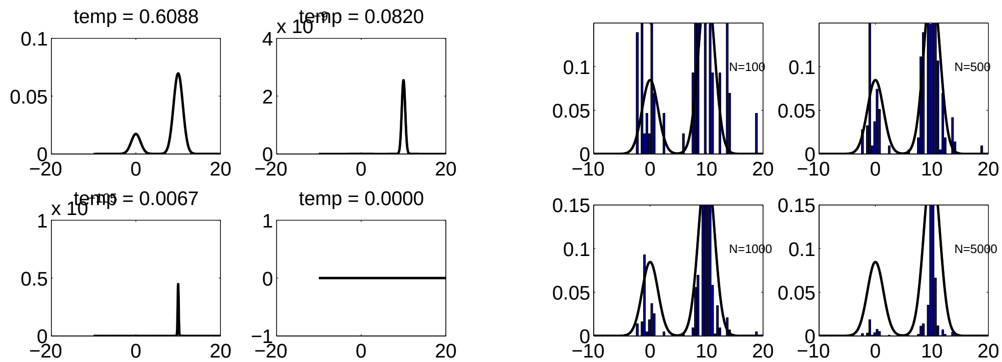

Figure 1: An example of simulated annealing applied to a mixture of two 1D Gaussians. We use the cooling schedule $T_s = 0.995^{s-1}$, starting at $T_1 = 1$. Left: we plot $p(x)^{1/T_s}$ at steps $s = 100, 500, 1000, 5000$. At the lowest temperature the function appears flat, due to numerical underflow. But in the sampling, we just need to compute $[p(x')/p(x)]^{1/T_s}$, which is more numerically stable. Right: we plot samples drawn from this distribution. This figure was produced using `SAdemoMOG.m`.

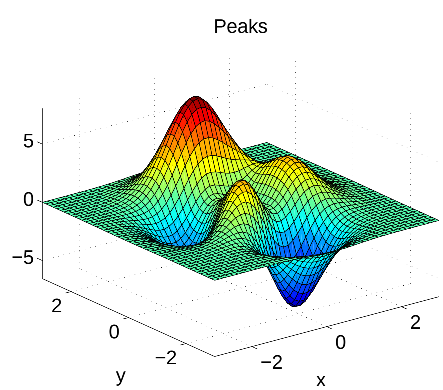

Figure 2: A peaky landscape

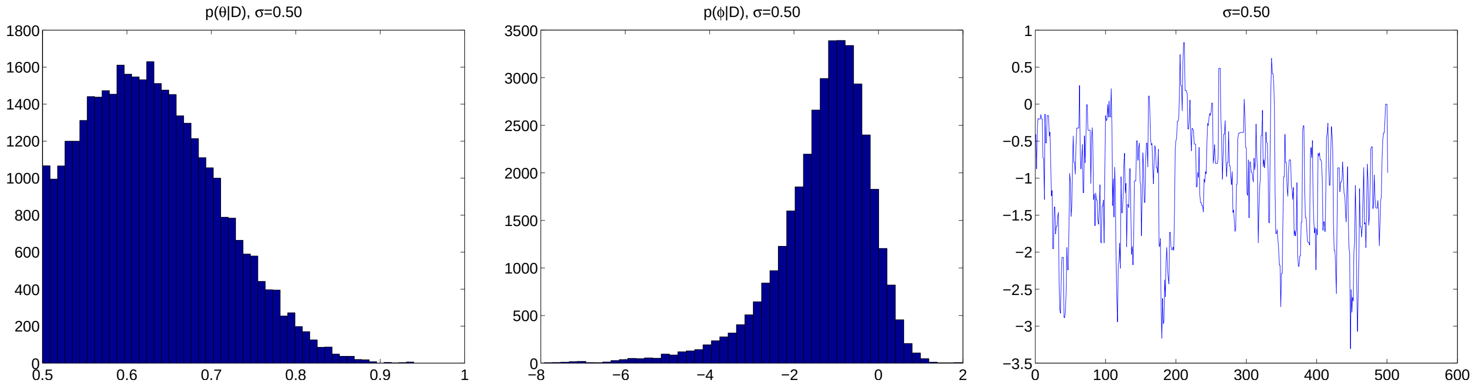

Figure 3: An example of the Metropolis algorithm for sampling from a binomial distribution with uniform prior using a Gaussian proposal with $\sigma = 0.5$. We used 40,000 samples and a burnin of 2000. Left: samples of the original parameter θ. The peak is near the MLE of $\hat{\theta}^{ML} = 0.6$. Middle: samples of the transformed parameter ϕ. Right: plot of the last 500 samples of ϕ. Figure produced using `mhDemoBino.m` (exercise).

2 Metropolis Hastings

Consider again the example of Binomial distribution with non conjugate prior in the MCMC handout. Use the MH algorithm (function MH.m is provided) to draw samples from $p(\phi|X)$, where

$$p(\phi|X) \propto \frac{(0.5 + e^{\phi})^X e^{\phi}}{(1 + e^{\phi})^{N+2}} = \frac{(0.5 + e^{\phi})^{12} e^{\phi}}{(1 + e^{\phi})^{22}} \tag{1}$$

(Note: the MH.m function was the target to compute $\log p(\phi|X)$. You may need to use $\log(p(\phi|X) + \epsilon)$ to avoid log of zero errors.) Use a Gaussian proposal with variance σ^2:

$$q(\phi'|\phi) = \mathcal{N}(\phi'|\phi, \sigma^2) \tag{2}$$

(Hint: use `normrnd` in the statistics toolbox.) Try $\sigma = 0.5$ and $\sigma = 10$. Use the following code snippet to ensure reproducable results

```
seed = 1;
randn('state', seed); rand('state', seed);
xinit = rand(1,1); % initial state
```

Draw 40,000 samples, discarding the first 2000 for burnin (these numbers are somewhat arbitrary). Plot a histogram of all the samples of ϕ post burnin, and also a trace of the last 500 samples of ϕ. Finally, convert the samples of ϕ back to the 0:1 scale using

$$\theta = \frac{0.5 + e^{\phi}}{1 + e^{\phi}} \tag{3}$$

and plot a histogram of these. Your results should look like Figure 3 and Figure 4.

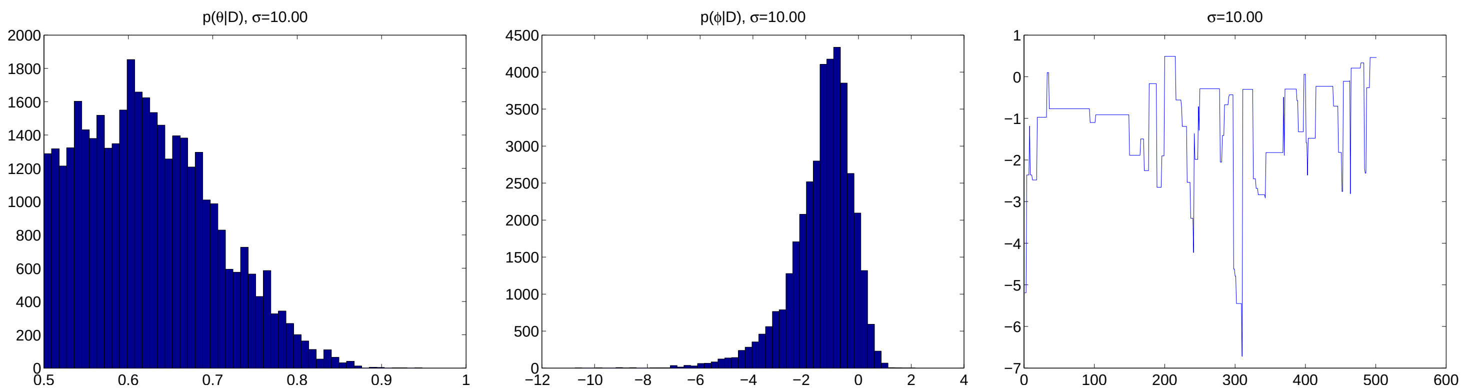

Figure 4: Same as Figure 3, except the Gaussian proposal has $\sigma = 10$. On the right we see the chain is not mixing is well, so the histograms are narrower and more blocky.