Optimization Conscious Econometrics Pset 1

Timothy Schwieg

1 Question One

1.1 Exercise One

Suppose that $q = p$, show that the acceptance rate of the Metropolis-Hastings sampler is 1.

Note that the probability of acceptance is given by:

$$\frac{p(\theta^*)q(\theta^i|\theta^*)}{p(\theta^i)q(\theta^*|\theta^i)}$$

Since it is known that $p = q$, we may substitute to arrive at:

$$\frac{p(\theta^*)p(\theta^i|\theta^*)}{p(\theta^i)p(\theta^*|\theta^i)}$$
$$\frac{p(\theta^* \cap \theta^i)}{p(\theta^* \cap \theta^i)} = 1$$

1.2 Exercise Two

Give the interpretation "in English" of $\mu_t P$ and $P\mu_t$.

We assume throughout this exercise that μ_t is a row vector of size k and that P is a $k \times k$ matrix.

$\mu_t P$ is the distribution after one step in the Markov Chain. It is the probability of being in each of the states after one unit of time.

$P\mu_t$ is not well defined, but for $P\mu_t'$ This is equal to $(\mu_t P')'$. P' is the transition matrix going in reverse direction. $P\mu_t$ gives the probability of being in each state going in the reverse direction as P. This is not the probability of being in each state that resulted in being at μ_t after one step.

1.3 Exercise Three

Show that $\mathbb{E}\left[\tau_z^+\right] = \sum_{x \in X} \bar{\pi}(x)$

$$\begin{aligned}
\mathbb{E}\left[\tau_z^+\right] &= \sum_{t=1}^{\infty} \Pr(\tau_z^+ \geq t) \\
&= \sum_{t=0}^{\infty} \Pr(\tau_z^+ > t) \\
&= \sum_{t=0}^{\infty} \sum_{\omega \neq z} \Pr(\tau_z^+ > t \cap X_t = \omega) \\
&= \text{ Note: } \Pr(\tau_z^+ > t \cap X_t = z) = 0 \\
&= \sum_{x \in X} \sum_{t=0}^{\infty} \Pr(\tau_z^+ > t \cap X_t = x) \\
&= \sum_{x \in X} \bar{\pi}(x)
\end{aligned}$$

1.4 Exercise Four

Show that the Metropolis-Hastings satisfies the detailed balanced equations in the general sense.

Note that we have already shown that it satisfies the detailed balanced equations in the case when $\Pr(\delta_{\theta^i}(\theta^{i+1}) = 1) = 0$. So all that remains to be shown is:

$$\delta_{\theta^i}(\theta^{i+1}) r(\theta^i) \pi(\theta^i) = \delta_{\theta^{i+1}}(\theta^i) r(\theta^{i+1}) \pi(\theta^i)$$

Note that the dirac-delta function is defined as such:

$$\delta_{x_0}(A) = \begin{cases} 1 \text{ if } x_0 \in A \\ 0 \text{ if } x_0 \notin A \end{cases}$$

Thus: $\delta_{\theta^i}(\theta^{i+1}) = \delta_{\theta^{i+1}}(\theta^i)$ as when $\theta \in \{\theta^{i+1}\}$ then $\theta^{i+1} \in \{\theta^i\}$. The same is true when they are not in. Proceed by cases.

Case 1: $\delta_{\theta^i}(\theta^{i+1}) = 0$. then the result is trivially true, as the product of any finite number, which $\pi(\theta^i)$ and $r(\theta^i)$ are, and zero is always zero.

Case 2: $\delta_{\theta^i}(\theta^{i+1}) = 1$. Then $\theta^i = \theta^{i+1}$ and we can see that $\pi(\theta^i) = \pi(\theta^{i+1})$ and $r(\theta^i) = r(\theta^{i+1})$.

2 Question 2

Program and annotate a Metropolis-Hastings sampler. Provide the annotated code with your solution

```
#Start is the value for the initial position of the sampler
#N is the number of iterations to perform
#pFun is the function that evaluates p
#qFun is the function that evaluations q
#qSimFun is a function that simulates from the q-law
#pargs is a vector of the arguments passed to p
#qargs is a vector of the function arguments that are passed to qFun and qSimFun

function Metropolis( startVal, N::Int64, pFun, qFun, qSimFun, pargs, qargs )
    #The uniforms are not conditional on any values of θ
    #So we might as well simulate them all up-front
    coinFlips = rand( Uniform(), N)
    #This allocates the vector for all the thetas.
    #theta = ones(N+1)*startVal
    theta = Vector{typeof(startVal)}(undef,N+1)
    theta[1] = startVal
    for i in 1:N
        #First simulate from q
        thetaStar = qSimFun( theta[i], qargs... )

        #Now go through all the mess of calculating the ratio of p and q's
        quotient = pFun( thetaStar, pargs...)*qFun(theta[i],thetaStar,qargs...) / (
 ↪   pFun( theta[i], pargs...)*qFun(thetaStar,theta[i],qargs...))

        #Decide if we are going to let this guy through or not.
        if( coinFlips[i] < quotient)
            theta[i+1] = thetaStar
        else
            theta[i+1] = theta[i]
        end
    end
```

```
    #Return the entire vector of paths to do with however you please
    return theta
end
```

3 Question 3

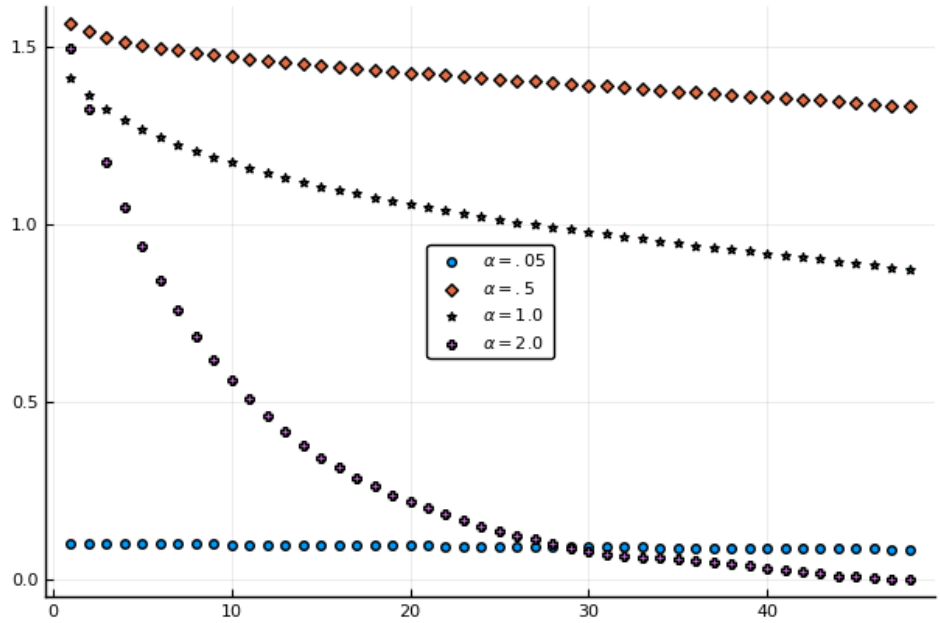
```
function p( x::Float64, ::Nothing)
    a = cos(x)
    b = cos(1.5*x)
    c = pdf( Normal(2,1), x)
    return a*a*b*b*c
end

#This is just uniformly distributed α units away from x
function q( xPrime::Float64, x::Float64, ::Float64)
    return (1/(2*))*( abs(xPrime - x) < )
end

function simQ(  x::Float64, ::Float64)
    return rand( Uniform( x - , x +  ),1)[1]
end


function CalculateAcceptanceRate( theta::Vector{Float64}, N::Int64)
    nDiff = 0.0
    for i in 1:N
        if( theta[i+1] != theta[i])
            nDiff += 1
        end
    end
    #return nDiff
    return nDiff / convert(Float64,N)
end

#This function takes the sample and computes the relevant summaries
function StatWaiter( s::Vector{Float64}, N, )
    a = round(CalculateAcceptanceRate( s, N)*100)/100.0
    b = round((maximum(s) - minimum(s))*100)/100.0
    c = mode( round.( s.*10) ./ 10)
    #plot!(plt, autocov(s),label=@sprintf("%.2f",))

    #d = autocov(s)
    return [a,b,c]
end

function CompareAlphas( alphas::Vector{Float64})
    N = 10000
    #plt = plot( xlims=(0,51), ylims=(-.1,1.6), legend=:topright )#plot(
    ↪  autocov(Metropolis( 2.0, N, p, q, simQ, 4.0)), label="4.0");
    AcceptanceRates = [StatWaiter(Metropolis( rand( Uniform(-1.0, 2.0 ),1)[1], N, p, q,
    ↪  simQ, [nothing], []), N, ) for  in alphas]
    #display(plt)
    #savefig( plt, "acf.png")

```

```
80      return AcceptanceRates
81  end
82
83  alphas = [.01, .05, .1, .2, .35, .5, .75, 1.0, 2.0, 3.0]
84  accepted = mean( [CompareAlphas( alphas) for i in 1:100] )
```

The Table below summarizes the average acceptance rate, range, and mode for 100 chains of length 10000. Each chain was initiated at a point distributed uniform along the interval $[-1, 2]$. The left side of the distribution is chosen to see how often the distribution gets trapped on the false mode occurring on the left tail. The range of the sample path is an effective measure of how much of the support is explored by the sample chain. We can see that the acceptance probability of the chain is decreasing in α, but the mode approaches the true mode, and the range approaches the expected range, equivalent to the range of a normal random variable.

α	Acceptance	Range	Mode
0.01	0.9897	0.7951	0.599
0.05	0.9636	1.7103	0.714
0.1	0.9347	1.961	0.938
0.2	0.8702	3.4003	1.166
0.35	0.7734	5.0604	1.744
0.5	0.6857	5.3093	2.024
0.75	0.5651	5.5536	2.257
1.0	0.4796	5.5905	2.348
2.0	0.3584	6.4453	2.388
3.0	0.3161	6.7637	2.395

Below we generate the auto-covariance function for four different values of α. We see that the auto-correlation is quite low for low values of α, but this comes at the cost of the low exploration of the data set. As α climbs, the auto-correlation increases as the probability of acceptance falls, but the rate it decreases is increasing in α. Note that the scale of the plot is in logarithms.



For these different values of α, we plot the histogram and the pdf of the distribution. As we can see, for the very small value of α, the distribution is stuck in the first mode of the distribution, and the rest of the support remains unexplored, even after 50000 steps, and a burnout period of 10000.

For the slightly larger $\alpha = .5$, we see that most of the distribution is explored, but the starting point of 0.0 is still overstated, even after 50000 steps, and the far side of the distribution is less explored. This indicates that the distribution still has not reached the steady state despite many draws. The same effect is mirrored in $\alpha = 1.0$, indicating that this may be simply caused by noise.

For $\alpha = 2.0$, the histogram almost exactly matches the pdf of the distribution, indicating that good mixing has been achieved.

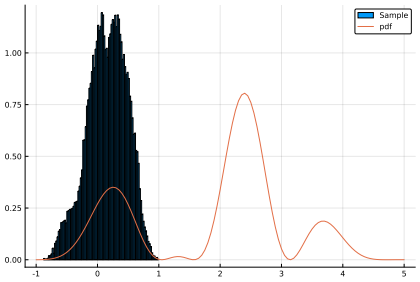


Fig. 1: $\alpha = .05$

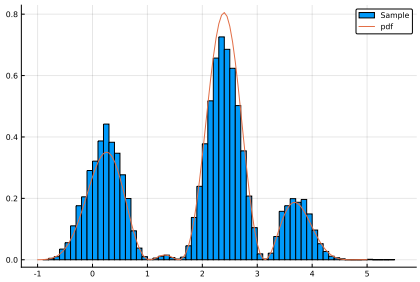


Fig. 2: $\alpha = .5$

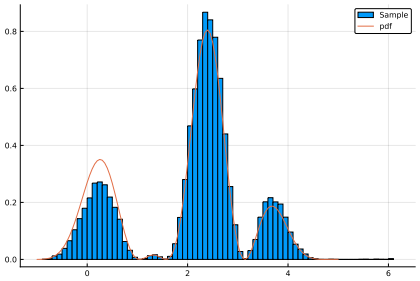


Fig. 3: $\alpha = 1.0$

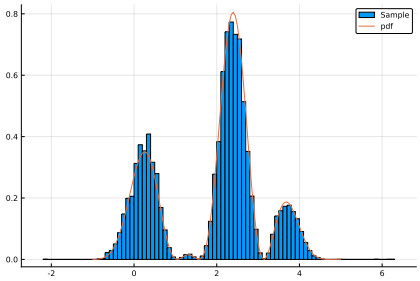


Fig. 4: $\alpha = 2.0$

4 Question 4

4.1 a

Let $\sigma_b^{-2} \sim \Gamma(\alpha_b, \beta_b)$ and $\sigma_\epsilon^{-2} \sim \Gamma(\alpha_\epsilon, \beta_\epsilon)$. $\beta \sim \mathcal{N}(\mu, \Sigma)$. Note that μ is a vector with 4 elements, and that Σ is a 4×4 matrix. This means that there are 24 hyper-parameters.

4.2 b

Conditioned on $X_i, \beta, W_i, b_i, \epsilon_i$,

$$\begin{aligned} \epsilon_i &\sim \mathcal{N}(0, \sigma_\epsilon^2 I) \\ b_i &\sim \mathcal{N}(0, \sigma_b^2) \\ W_i b_i &\sim \mathcal{N}(\vec{0}, W_i \sigma_b^2 W_i') \end{aligned}$$