AMS 207
Efficient MCMC

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

April 28, 2016

Problem

- regular MCMC (Gibbs and Metropolis-Hastings) can be inefficient
- this problem is particularly salient in high dimensional spaces.

Slice sampling

- most useful in the one-dimensional case but can be applied within a more complex sampling scheme
- goal: sample from an arbitrary distribution $f(\theta)$ known up to a proportionality constant
- idea:
  - start from an arbitrary point $\theta^{(0)}$
  - sample an auxiliary variable $Y \sim \mathcal{U}\left(0, f\left(\theta^{(0)}\right)\right)$
  - the region $\{\theta : f(\theta) > Y\}$ defines a "slice" with density at least Y
  - get $\theta^{(1)}$ by sampling uniformly from this "slice"
  - repeat

Slice sampling

- in practice if the distribution is multimodal finding the slice is not straightforward
- one option to simplify this process is to use regional expansion-contraction
- choose a width parameter w and expand the interval $\frac{w}{2}$ units to the left (right) from $\theta^{(t)}$ until the endpoint lies outside the slice

Example

- Gaussian process with constant mean function and exponential covariance function
- in this case we wish to sample the scale from $f(\phi \mid \boldsymbol{\theta}, \mu, \tau, \mathbf{y}) \propto |H(\phi)|^{-\frac{1}{2}} \exp\left\{ \frac{-(\boldsymbol{\theta}-\mu\mathbf{1})' H(\phi)^{-1}(\boldsymbol{\theta}-\mu\mathbf{1})}{2\tau^2} \right\} \pi(\phi)$ where $H_{i,j}(\phi) = \exp\{\phi|\mathbf{x}_i - \mathbf{x}_j|\}$ and $\mathbf{X}$ is a set of known covariates
- assume $\pi(\phi) = \mathcal{U}(0,1)$

Example

```
log_phi_full_cond<-function(phi,theta,mu,tausq,X){
        H<-exp(phi*abs(X-t(X)))
        HI<-solve(H) ## Burn in a pot of (olive) oil
        res<-(-1/2)*log(det(H))-(t(theta-mu)%*%HI%*%(theta-mu))/(2*tausq)
        return(res)
}

phi<-NULL
phi[1]<-runif(1)
w<-.02

L<-max(phi[1]-w/2,0)
R<-min(phi[1]+w/2,1)
```

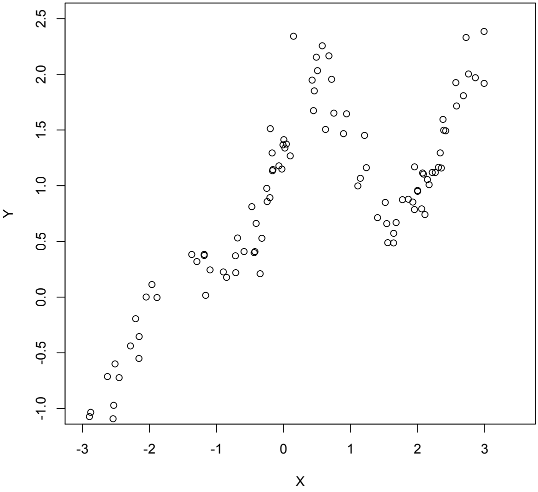
Example

```
for(it in 2:ITER){

        ## [sample theta[it,], mu[it], tausq[it] (GP theory)] ##

        ## calculate density of current sample and endpoints
        yphi<-log_phi_full_cond(phi[it-1],heta[it,], mu[it], tausq[it],X)
        fL<-log_phi_full_cond(L,theta[it,], mu[it], tausq[it],X)
        fR<-log_phi_full_cond(R,theta[it,], mu[it], tausq[it],X)

        ## expand the interval to approximate the slice
        while(fL>yphi){
                L<-max(L-w/2,0)
                fL<-log_phi_full_cond(L,theta[it,], mu[it], tausq[it],X)
        }
        while(fR>yphi){
                R<-min(R+w/2,1)
                fR<-log_phi_full_cond(R,theta[it,], mu[it], tausq[it],X)
        }
```

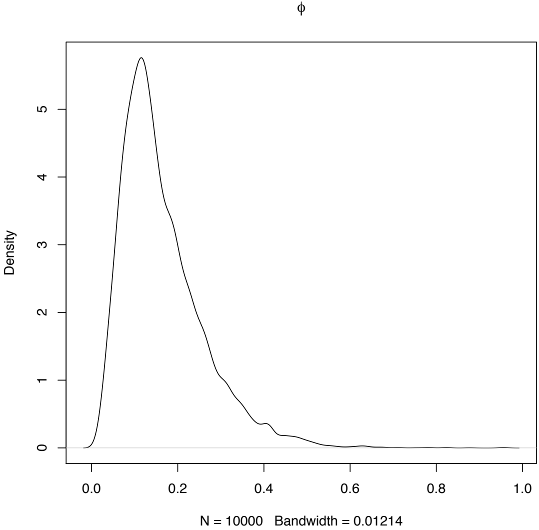
Example

```
        ## sample from the slice
        phihat<-runif(1,L,R)

        ## accept sample only if it comes from the "real slice"
        ## otherwise reject and contract the interval
        fhat<-log_phi_full_cond(phihat,theta[it,], mu[it], tausq[it],X)
        if(fhat>yphi){
                phi[it]<-phihat
                L<-max(phi[it+1]-w/2,0)
                R<-min(phi[it+1]+w/2,bphi)
        }else{
                phi[it]=phi[it-1]
                if(phihat<phi[it-1]){
                        L<-phihat
                }else{
                        R<-phihat
                }
        }
}
```

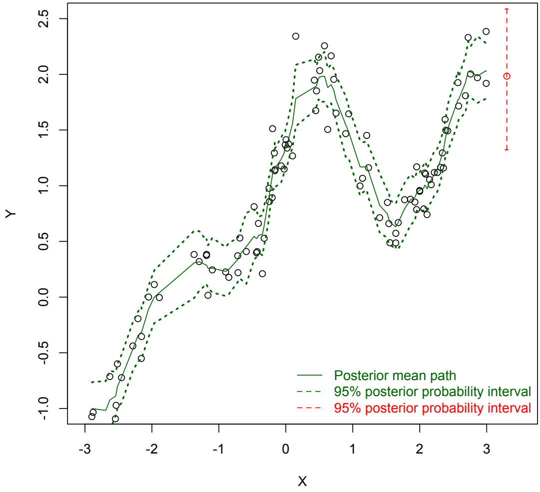
Example: simulated data



Example



Example


Posterior mean path
95% posterior probability interval
95% posterior probability interval
Y
X
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3

Sequential Monte Carlo

- mostly used for hidden Markov models
- much less computationally expensive than MCMC
- allows for on-line inference

Monte Carlo

- let the target be the n-dimensional distribution $f_n(\boldsymbol{\theta})$
- if f_n is available to sample from, its Monte Carlo estimate is given by

$$\hat{f}_n(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{\Theta}^i}(\boldsymbol{\theta})$$

  where N is the sample size, δ is the Dirac delta function and $\{\boldsymbol{\Theta}^i\}$ are samples from f_n
- similarly, any marginal is approximated by

$$\hat{f}(\theta_k) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\Theta_k^i}(\theta_k)$$

Problems

- f_n may not be available, or it may be complicated to sample from
- As n increases, the complexity of sampling from f_n increases (at best) linearly

Importance sampling

- let $q_n(\boldsymbol{\theta})$ be an importance (proposal) density satisfying $q_n(\boldsymbol{\theta}) = 0 \Rightarrow f_n(\boldsymbol{\theta}) = 0$. Then, is possible to write

$$f_n(\boldsymbol{\theta}) = \frac{w_n(\boldsymbol{\theta}) q_n(\boldsymbol{\theta})}{Z_n}$$

- the unnormalized weight function $w_n(\boldsymbol{\theta})$ is given by

$$w_n(\boldsymbol{\theta}) = \frac{\gamma_n(\boldsymbol{\theta})}{q_n(\boldsymbol{\theta})}$$

- and

$$Z_n = \int w_n(\boldsymbol{\theta}) q_n(\boldsymbol{\theta})\, d\boldsymbol{\theta}$$

Importance sampling

- if $q_n(\boldsymbol{\theta})$ is available to sample from, and $\{\boldsymbol{\theta}^i\}$ are N independent samples from it

$$\hat{Z}_n = \frac{1}{N} \sum_{i=1}^{N} w(\boldsymbol{\theta}^i)$$

- therefore

$$\hat{f}_n(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} W_n^i \delta_{\boldsymbol{\theta}^i}(\boldsymbol{\theta})$$

where

$$W_n^i = \frac{w_n(\boldsymbol{\theta}^i)}{\sum_{j=1}^{N} w_n(\boldsymbol{\theta}^j)}$$

Sequential importance sampling

- if an importance density is selected to have the following structure

$$q_n(\boldsymbol{\theta}) = q_n(\theta_n \mid \boldsymbol{\theta}_{1:n-1}) q_{n-1}(\boldsymbol{\theta}_{1:n-1}) = q_1(\theta_1) \prod_{k=2}^{n} q_k(\theta_k \mid \boldsymbol{\theta}_{1:k-1})$$

- the unnormalized weights satisfy the recursion

$$w_n(\boldsymbol{\theta}) = w_{n-1}(\boldsymbol{\theta}_{1:n-1}) \alpha_n(\boldsymbol{\theta}) = w_1(\theta_1) \prod_{k=2}^{n} \alpha_k(\boldsymbol{\theta}_{1:k})$$

where

$$\alpha_n(\boldsymbol{\theta}) = \frac{\gamma_n(\boldsymbol{\theta})}{\gamma_{n-1}(\boldsymbol{\theta}_{1:n-1}) q_n(\theta_n \mid \boldsymbol{\theta}_{1:n-1})}$$

is referred as the incremental weight function

Simulated tempering

- goal: improve Markov chain simulation performance when posterior $f(\boldsymbol{\theta} \mid \mathbf{y})$ is multimodal
- idea: consider a set of K alternative distributions $f_k(\boldsymbol{\theta} \mid \mathbf{y})$ with the same basic shape as the target but with improved Markov chain mixing properties
- commonly $f_k(\boldsymbol{\theta} \mid \mathbf{y}) \propto (f(\boldsymbol{\theta} \mid \mathbf{y}))^{\frac{1}{T_k}}$ with $T_0 = 1$ so that $f_0(\boldsymbol{\theta} \mid \mathbf{y}) \propto f(\boldsymbol{\theta} \mid \mathbf{y})$
- T_k are the set of *temperature* parameters. As $T_k \to \infty$ $f_k \to$ Uniform $\Rightarrow$ the chain moves more around the space

Simulated tempering

- take the state space to be $(\boldsymbol{\theta}^t, s^t)$ where s^t is an indicator of the chain used to sample $\boldsymbol{\theta}^t$. the algorithm can then be summarized as follows:

  1. sample $\boldsymbol{\theta}^{t+1}$ from the Markov chain with stationary distribution q_{s^t}
  2. propose a jump to an alternative chain j with probability $J_{s^t,j}$ and accept the jump with probability $\min\{1, \rho\}$ where

     $$\rho = \frac{c_j f_j\left(\boldsymbol{\theta}^{t+1} \mid \mathbf{y}\right) J_{s^t,j}}{c_{s^t} f_{s^t}\left(\boldsymbol{\theta}^{t+1} \mid \mathbf{y}\right) J_{j,s^t}}$$

     and c_k are adaptive constants to approximate the normalizing constant
  3. keep only the samples from state 0

Hamiltonian (hybrid) Monte Carlo

- avoids "random walk behavior" of Metropolis
- idea: augment the parameter space introducing a set of momentum variables $\boldsymbol{\phi} = \{\phi_i\}_{i=1}^n$ and explore the joint $f(\boldsymbol{\theta}, \boldsymbol{\phi} \mid \mathbf{y}) = f(\boldsymbol{\phi}) f(\boldsymbol{\theta} \mid \mathbf{y})$
- it's usually assumed that $\phi_j \overset{ind}{\sim} \mathcal{N}(0, M_{j,j})$

Hamiltonian (hybrid) Monte Carlo

- the algorithm can be summarized as follows:
  1. sample $\boldsymbol{\phi} \sim \mathcal{N}_n(\mathbf{0}, M)$ where M is referred to as the mass matrix and is given by $M = \text{diag}\{M_{1,1}, M_{2,2}, \ldots, M_{n,n}\}$
  2. take L "leapfrog" steps of the form

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \frac{1}{2}\epsilon\nabla f(\boldsymbol{\theta} \mid \mathbf{y})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \frac{1}{2}\epsilon M^{-1}\boldsymbol{\phi}$$

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \frac{1}{2}\epsilon\nabla f(\boldsymbol{\theta} \mid \mathbf{y})$$

     label the resulting parameters $(\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star)$
  3. accept $(\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star)$ with probability $\min\{1, \rho\}$ where

$$\rho = \frac{f(\boldsymbol{\theta}^\star \mid \mathbf{y}) f(\boldsymbol{\phi}^\star)}{f(\boldsymbol{\theta}^{t-1} \mid \mathbf{y}) f(\boldsymbol{\phi}^{t-1})}$$

Hamiltonian (hybrid) Monte Carlo

- ϵ and M are used as tuning parameters for the algorithm
- Stan uses HMC to perform Bayesian computations

Example

- recall the stomach cancer mortality data from LearnBayes
- consider the model with the Binomial likelihood and conjugate Beta prior

$$\theta \sim \mathcal{B}eta(\alpha, \beta)$$
$$y_i \sim \mathcal{B}in(\theta; n_i)$$

```
library(LearnBayes)
library(rstan)

data(cancermortality)

N<-nrow(cancermortality)
y<-cancermortality$y
n<-cancermortality$n
```

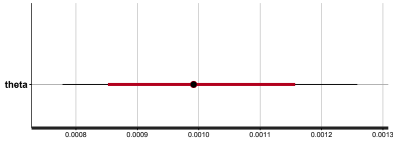
Example

```
//bin.stan

data{
        int<lower=0> N;
        int<lower=0> y[N];
        int<lower=0> n[N];
}
parameters{
        real<lower=0,upper=1> theta;
}
model{
        for(i in 1:N){
                y[i]~binomial(n[i],theta);
        }
        theta~beta(1,1);
}
```

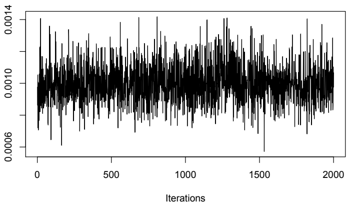
Example

```
bfit<-stan(file="bin.stan",data=c("N","y","n"),iter=1000, chains=4)
```

```
plot(bfit)
```




```
bsim<-extract(bfit, permuted=TRUE)
traceplot(as.mcmc(as.vector(bsim$theta)))
```



Example

- For the Beta-Binamial likelihood

$$y_i \sim \mathcal{B}e\text{-}\mathcal{B}in(\eta, K)$$

$$\pi(\eta, K) \propto \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}$$

```
//betabin.stan

data{
        int<lower=0> N;
        int<lower=0> y[N];
        int<lower=0> n[N];
}
parameters{
        real logK;
        real logiteta;
}
```

Example

```
transformed parameters{
        real<lower=0> K;
        real<lower=0,upper=1> eta;
        K<-exp(logK);
        eta<-inv_logit(logiteta);
}
model{
        real alpha;
        real beta;
        alpha<-K*eta;
        beta<-K*(1-eta);
        for(i in 1:N){
                y[i]~beta_binomial(n[i],alpha,beta);
        }
        // need to add prior because is not in a standard family of distributions
        increment_log_prob(logK-2*log(1+exp(logiteta)));
}
```

Example

```
bbfit<-stan(file="betabin.stan",data=c("N","y","n"),iter=1000,chains=4)
```

```
..
The following numerical problems occured the indicated number of times after warmup on chain 1
Warning messages:
1: There were 476 transitions after warmup that exceeded the maximum treedepth. Increase max_treedepth above
10.
2: Examine the pairs() plot to diagnose sampling problems
```

References

- Albert, J. (2009). *Bayesian computation with R.* Springer Science & Business Media.
- Doucet, A. & Johansen, A. (2011). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. In *Oxford Handbook of Nonlinear Filtering*, pp. 656-704. Oxford University Press.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis (3th ed.).* CRC Press.