Monte Carlo Methods

732A90
Computational Statistics

Maryna Prus
(maryna.prus@liu.se)

Slides originally by Krzysztof Bartoszek

November 19, 2021
Department of Computer and Information Science
Linköping University

Monte Carlo Methods: Outline

- **Monte Carlo methods**

  - class of computational algorithms that use repeated random sampling

- Monte Carlo methods for random number generation
  - Metropolis–Hastings algorithm
  - Gibbs sampler

- Monte Carlo methods for statistical inference
  - Estimate integrals
  - Variance estimation, variance reduction

Markov Chain Monte Carlo (MCMC): Motivation

Previous lecture:

Generate

- Univariate distributions

  simple transformations of $\mathrm{Unif}(0, 1)$

  inverse CDF

  acceptance/rejection

- Multivariate normal

General multivariate distribution

$$\rightarrow \qquad \textbf{MCMC}$$

Bayesian Inference

D - dataset obtained by sampling from a distribution $p(D|\theta)$

Estimation of θ - ?

- Consider θ as unknown but *fixed* parameter
  $\rightarrow$ estimate θ using ML, etc.
- *Bayesians approach*: θ is a *random* variable with
  - *Prior* distribution $p(\theta)$ (*before* observing D)
  - *Posterio* distribution $p(\theta|D)$ (*after* observing D)

  Bayes' Theorem:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)\mathrm{d}\theta}$$

  p - density / probability functions

Bayesian Inference

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)\mathrm{d}\theta}$$

We know: $p(D|\theta)$ (*model*) and $p(\theta)$ (*prior*)

We need: simulate from $p(\theta|D)$ (*posterior*)

1. General multivariate distribution
2. Integral can be impossible to compute

1. MCMC solves this problem
2. Not needed (given D it is constant)

Markov Chains (Time Discrete)

- Markov chain (MC)

  - sequence X_0, X_1, X_2, $\ldots$ of random variables

  - distribution of X_{t+1} depends only on X_t (and parameters)

    $\rightarrow X_{t+1}$ independent of X_{t-1}, X_{t-2}, ...

- *Time homogeneous* MC:
  - $p(X_{t+1}|X_t)$ independent of t

    $\rightarrow P(X_{t+1} = x|X_t = y) = P(X_t = x|X_{t-1} = y)$ for all t

- *Stationary* MC:

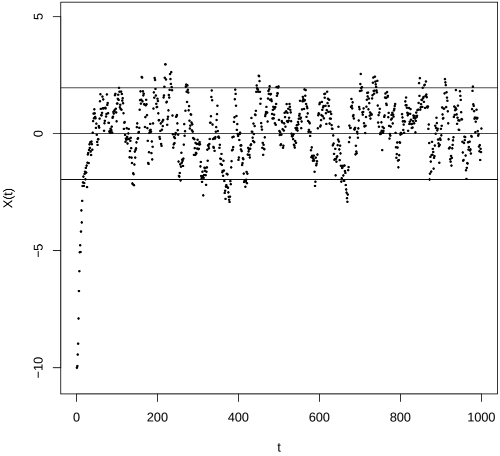
  $X_i \sim \Phi$ for all $i \geq k$, Φ - *stationary distribution*

- Under certain conditions MC converge to stationary distribution

- First $k - 1$ samples normally discarded $\rightarrow$ *burn-in period*

Markov Chains: Example

$$X_{t+1} = e^{-1} X_t + \epsilon\ , \epsilon \sim \mathrm{N}(0, \tfrac{5}{2} \cdot (1 - e^{-2})), \quad X_0 = -10$$



Metropolis-Hastings Algorithm

We have

- Distribution $\pi(\cdot)$ that we want to sample from
- *Proposal* distribution $q(\cdot|X_t)$
  For ex. $q(\cdot|X_t)$ is normal with mean X_t and given variance
- $q(\cdot|X_t)$ has *regular* form w.r.t. to $\pi(\cdot)$
- *Regular* form:
  suffices that $q(\cdot|X_t)$ has the same support as $\pi(\cdot)$

Metropolis-Hastings Algorithm

$$\alpha(X_t, Y) = \min\left\{1, \frac{\pi(Y)q(X_t|Y)}{\pi(X_t)q(Y|X_t)}\right\}$$

Initialize chain to X_0, $t = 0$
while $t < t_{\max}$ **do**
    Generate a candidate point Y from $q(\cdot|X_t)$
    Generate U from $\mathrm{Unif}(0, 1)$
    if $U < \alpha(X_t, Y)$ **then**
        $X_{t+1} = Y$
    else
        $X_{t+1} = X_t$
    end if
    $t = t + 1$
end while

Metropolis-Hastings Algorithm: Properties

- Informally: "The chain $(X_t)_{t=0}^{\infty}$ converges to $\pi(\cdot)$"
- The chain might not move sometimes
- The values of the chain are dependent
- If $q(X_t|Y) = q(Y|X_t)$ (*symmetric* proposal) we get *Random-walk Monte Carlo*:

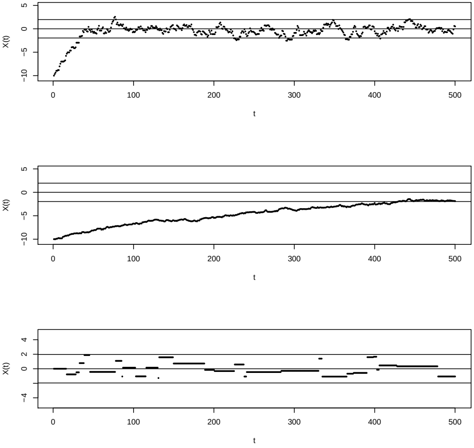
$$\alpha(X_t, Y) = \min\left\{1, \frac{\pi(Y)}{\pi(X_t)}\right\}$$

Example: $\pi(\cdot) = N(0,1)$ - Target Distribution

```
f.MCMC.MH<-function(nstep,X0,props){
    vN<-1:nstep
    vX<-rep(X0,nstep);
    for (i in 2:nstep){
        X<-vX[i-1]
        Y<-rnorm(1,mean=X,sd=props)
        u<-runif(1)
        a<-min(c(1,(dnorm(Y)*dnorm(X,mean=Y,sd=
            props))/(dnorm(X)*dnorm(Y,mean=X,sd=
            props))))
        if (u <=a){vX[i]<-Y}else{vX[i]<-X}
    }
    plot(vN,vX,pch=19,cex=0.3,col="black",xlab="t",
        ylab="X(t)",main="",ylim=c(min(X0-0.5,-5),
       max(5,X0+0.5)))
    abline(h=0)
    abline(h=1.96)
    abline(h=-1.96)
}
```

Example: $\pi(\cdot) = \mathrm{N}(0,1)$

q normal with sd: `props=` 0.5, 0.1 and 20, $X_0 = 0$



Gibbs Sampler

We want to generate from multivariate distribution:

$$p(X_1, X_2, \ldots, X_d)$$

Initialize chain to $X_0 = (X_{0,1}, \ldots, X_{0,d})$, $t = 0$

while $t < t_{\max}$ **do**

  for $i = 1, \ldots, d$ **do**

    Generate
    $X_{t+1,i}$ from $p(\cdot | X_{t+1,1}, \ldots, \mathbf{X_{t+1,i-1}}, \mathbf{X_{t,i+1}}, \ldots, X_{t,d})$

  end for

  $t = t + 1$

end while

Gibbs Sampler

- At each iteration inside the `for` loop univariate random numbers are generated
- Only one element is updated
- We need to know conditional distributions
- Convergence may be slow
- Can be useful in high dimensions (i.e. proposal density may be difficult to find in another way)

Example: d-dim $\mathrm{N}(\mu, \mathbf{\Sigma})$

$X = (X_1, X_2)^\top \sim \mathrm{N}(\mu, \mathbf{\Sigma})$

- $\mu = (\mu_1, \mu_2)^\top$
- $\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$

For $Z_1 = (X_1 | X_2 = x_2)$:

$$Z_1 \sim \mathrm{N}(\mu_1 + \frac{\sigma_1}{\sigma_2}\rho(x_2 - \mu_2), (1 - \rho^2)\sigma_1^2)$$

For $Z_2 = (X_2 | X_1 = x_1)$:

$$Z_2 \sim \mathrm{N}(\mu_2 + \frac{\sigma_2}{\sigma_1}\rho(x_1 - \mu_1), (1 - \rho^2)\sigma_2^2)$$

General d:

more complicated, but closed formulas

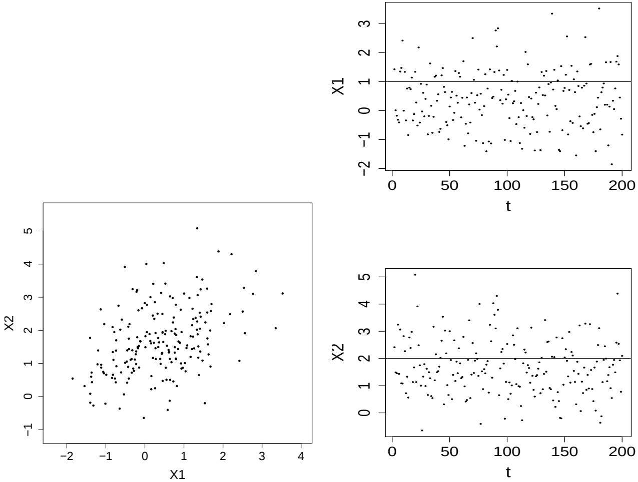
Code for d-dim $N(\mu, \Sigma)$

```
f.MCMC.Gibbs<-function(nstep,X0,vmean,mVar){
    d<-length(vmean); mX<-matrix(0,nrow=nstep,ncol=d);
        mX[1,]<-X0
    for (i in 2:nstep){
        X<-mX[i-1,];Y<-rep(0,d)
        Y[1]<-rnorm(1,mean=vmean[1]+(mVar[1,-1]%*%solve
            (mVar[-1,-1]))%*%(X[2:d]-vmean[-1]),sd=sqrt
            (mVar[1,1]-mVar[1,-1]%*%solve(mVar[-1,-1])%
            *%mVar[-1,1]))
         for (j in 2:(d-1)){Y[j]<-rnorm(1,mean=vmean[j
            ]+(mVar[j,-j]%*%solve(mVar[-j,-j]))%*%(c(Y
            [1:(j-1)],X[(j+1):d])-vmean[-j]),sd=sqrt(
            mVar[j,j]-mVar[j,-j]%*%solve(mVar[-j,-j])%*
            %mVar[-j,j]))}
        Y[d]<-rnorm(1,mean=vmean[d]+(mVar[d,-d]%*%solve
            (mVar[-d,-d]))%*%(Y[1:(d-1)]-vmean[-d]),sd=
            sqrt(mVar[d,d]-mVar[d,-d]%*%solve(mVar[-d,-
            d])%*%mVar[-d,d]))
        mX[i,]<-Y
    };mX}
```

Example: 2-dim $\mathrm{N}(\mu, \boldsymbol{\Sigma})$

Generate from

$$\mathrm{N}([1\ 2]^T, \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix})$$



Convergence Monitoring

- When should we stop the chain?

  When are we (nearly) at the stationary distribution?

- Typically such a sample is generated to make further inference

Convergence Monitoring: Gelman-Rubin Method

We want to estimate $v(\theta)$.

1. Generate k sequences of length n with different starting points.
2. Compute between– and within– sequence variances:

$$B = \frac{n}{k-1}\sum_{i=1}^{k}(\overline{v}_{i\cdot} - \overline{v}_{\cdot\cdot})^2 \quad W = \sum_{i=1}^{k}\frac{s_i^2}{k} \quad s_i^2 = \sum_{j=1}^{n}\frac{(\overline{v}_{ij} - \overline{v}_{i\cdot})^2}{n-1}$$

3. Overall variance estimate: $\hat{\mathrm{Var}}[v] = \frac{n-1}{n}W + \frac{1}{n}B$
4. Gelman–Rubin factor:

$$\sqrt{R} = \sqrt{\frac{\hat{\mathrm{Var}}[v]}{W}}$$

5. Value close to 1 (≤ 1.2) $\rightarrow$ convergence achieved
6. See `?coda::gelman.diag`

Gelman-Rubin Method

```
library(coda)
f1<-mcmc.list();f2<-mcmc.list();n<-100;k<-20
X1<-matrix(rnorm(n*k),ncol=k,nrow=n)
X2<-X1+(apply(X1,2,cumsum)*(matrix(rep(1:n,k),ncol=
    k)^2))
for (i in 1:k){f1[[i]]<-as.mcmc(X1[,i]);f2[[i]]<-as
    .mcmc(X2[,i])}
print(gelman.diag(f1))
# Potential scale reduction factors:
#      Point est. Upper C.I.
#[1,]       0.999       1.01

print(gelman.diag(f2))
# Potential scale reduction factors:
#      Point est. Upper C.I.
#[1,]        1.82       2.38
```

MC for Inference

- Estimation of a definite integral

$$\theta = \int_D f(x) \mathrm{d}x$$

- Decompose into:

$$f(x) = g(x)p(x) \quad \text{where} \quad \int_D p(x) \mathrm{d}x = 1$$

- Then, if $X \sim p(\cdot)$

$$\theta = \mathrm{E}\left[g(X)\right] = \int_D g(x)p(x) \mathrm{d}x$$

- $$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} g(x_i), \quad \forall_i x_i \; from \; p(\cdot)$$

Summary

1. Generating data from a general multivariate distribution
2. Markov Chain Monte Carlo:
   - Metropolis–Hastings algorithm
   - Gibbs sampling
3. Convergence:
   - Gelman–Rubin method
4. Estimation of integral