MCMC - Samplingstrategien

Volker Schmid

29. Mai 2017

Optimal scaling bzw. Tuning

Random Walk Proposal

$$\theta^* = \theta^{i-1} + u_i$$

- u_i aus einer symmetrischen Verteilung (Metropolis-Algorithmus)
- Metropolis (1953) benutzt $u_i \sim U[-\alpha, +\alpha]$
- oft $u_i \sim N(0, \sigma^2)$
- Ist θ beschränkt, evtl. problematisch an Grenzen, dann Transformation sinnvoll
- Vorteil: leicht zu implementieren, funktioniert praktisch immer
- Nachteil: ineffizient, Wahl von α, σ^2?

Theoretische Ergebnisse

Roberts, Gelman und Gilks (1997). Zieldichte sei $f(\theta) = \prod_{i=1}^{d} f(\theta_i)$. RW-Proposal mit

$$u \sim N_d(0, I_d)$$

und $d \to \infty$. Für eine normale Random Walk Vorschlagsverteilung gilt: Die optimale Akzeptanzrate ist 0.234.

Optimale Standardabweichung

$$\sigma = 2.38/\sqrt{Id}$$

mit $I = \int \left(\frac{f'(\theta)}{f(\theta)}\right)^2 f(\theta) d\theta$.

Praktisch sind die Ergebnisse von Roberts et al. nicht anwendbar. Für mehr generelle Situationen existieren weitere Studien mit ähnlichen Ergebnissen.

Anwendung der Theorie, Tuning

- Bei Abhängigkeit eher noch kleinere Akzeptanzraten optimal * Für kleine Dimensionen kann Rate höher sein (0.44 bei $d = 1$); gilt auch für eindimensionalen Metropolis (!)
- Hohe Effizienz bei Akzeptanzraten zwischen ca. 0.1 und 0.6
- Tuning:
  - starte MCMC, berechne Akzeptanzrate
  - passe σ bzw.α entsprechend an
  - Kovarianz mit anderen Parametern wichtig, Tuning etwa zeitgleich mit burn-in der anderen Parameter

Block-Algorithmen

Parameter-Blöcke

Sind Parameter stark voneinander abhängig, verschlechtert sich bei eindimensionalem Ziehen die Effizienz

Beispiel:

$$x \sim N(\mu, 1); \mu \sim N(\nu, 1); \nu \sim N(0, 1)$$

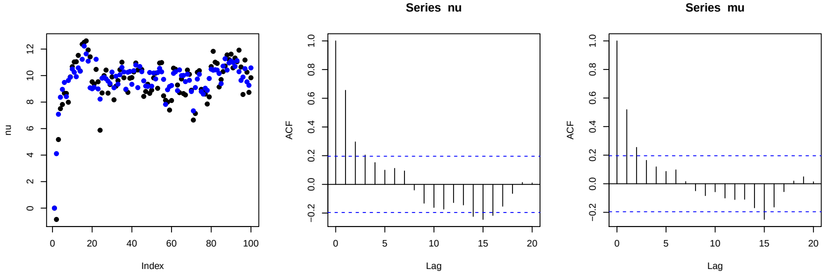
$$f(\mu, \nu | x) \propto \exp\left(-\frac{1}{2}(2\mu^2 - 2x\mu - 2\mu\nu + \nu^2)\right)$$

Parameter-Blöcke

```
mu<-rep(0,100)
nu<-rep(0,100)
x<-10
for(i in 2:100) {
nu[i]<-rnorm(1,mu[i-1],1)
mu[i]<-rnorm(1,(x+nu[i])/2,1/2)
}
```

Parameter-Blöcke

```r
par(mfrow=c(2,3))
plot(nu, pch=19)
points(mu,col="blue", pch=19)
acf(nu)
acf(mu)
```



Block-Update

Effizienter: mehrere abhängige Parameter gleichzeitig ziehen. Im Beispiel:

$$\mu, \nu | x \sim N_2(Q^{-1}m, Q^{-1})$$

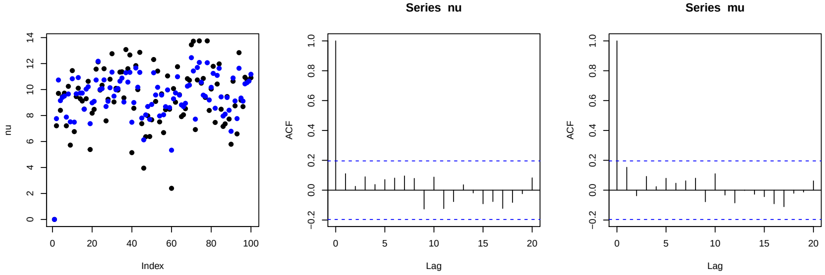
$$m = (x, 0)^T; Q = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

Block-Update

```
mu<-rep(0,100)
nu<-rep(0,100)
x<-10
Q<-matrix(c(2,-1,-1,1),nrow=2)
Q1<-solve(Q)
for(i in 2:100)
{
  temp<-rnorm(2)+c(x,0)
  temp<-Q1%*%temp
  nu[i]<-temp[2]
  mu[i]<-temp[1]
}
```

Block-Update

```
par(mfrow=c(2,3))
plot(nu, pch=19)
points(mu,col="blue", pch=19)
acf(nu)
acf(mu)
```



Block-Update bei Metropolis-Hastings

- Klassischer Metropolis-Algorithmus (heute Metropolis-Within-Hastings genannt) zieht Parameter einzeln
- Effizienter ist gemeinsames Ziehen, wenn möglich unter Berücksichtigung der Abhängigkeit (z.B. Random Walk mit getunter Kovarianzmatrix)
- Tuning bzw. Proposal-Wahl muss sorgfältig erfolgen, da Posterioriwerte im mehrdimensionalen kleiner

Blockupdates in hierarchischen Modellen

In hierarchischen Modellen können Parameter θ und Hyperparameter κ gleichzeitig gezogen werden. Sei z.B. $x|\theta \sim f(x|\theta); \theta \sim p(\theta|\tau); \kappa \sim p(\kappa)$. Sei $\kappa|\theta$ Dichte einer bekannte Verteilung; $\theta|\kappa, x$ nur bis auf Konstante bekannt und q eine Vorschlagsdichte. Blockupdate wie folgt:

Ziehe $\kappa^* \sim \kappa|\theta^{(i-1)}$

Ziehe θ^* aus $q(\theta^*|\kappa^*, x, \theta^{i-1})$

Akzeptiere oder verwerfe (θ^*, κ^*) gemeinsam

Bei der Akzeptanzwahrscheinlichkeit kürzen sich viele Terme, da

$$q(\theta^*, \kappa^*|\theta^{(i-1)}) = q(\theta^*|\kappa^*)p(\kappa^*|\theta^{(i-1)})$$

Blockupdates in hierarchischen Modellen

Beispiel aus Knorr-Held, Rue, 2000.

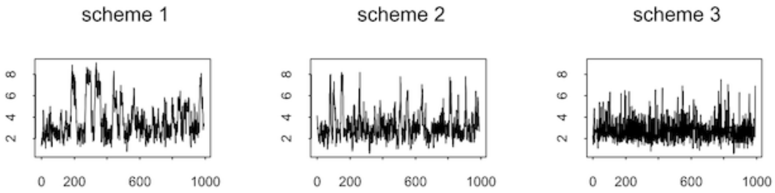


Figure 5: Trace plots of $\log \kappa$ for the three different schemes.

Figure 1:

Single-site update - Blockupdate θ - Blockupdate (θ, κ)

Weitere Strategien

Approximation der Full Conditional

- Benutze Vorschlagverteilung, welche die Full conditional gut approximiert
- Ziel: $\alpha \approx 1$ – im Gegensatz zum allgemeinen Random Walk! Entspricht "fast" Gibbs-Sampler
- Beispiele:
  - IWLS proposal (Gamermann 1997): Proposal-Dichte erzeugt aus einer Iteration des Iterative Weighted Least Squares-Algorithmus (GLM)
  - Normalverteilungsapproximation durch z.B. Taylor-Reihe, auch mehrdimensional möglich

Hilfsvariablenansätze ("Data augmentation")

Manchmal lassen sich Probleme vereinfachen, wenn man latente Variablen einführt.

Beispiel: *Binärregression nach Holmes und Held* (2006):

$$y_i \sim B(1, g^{-1}(\eta_i)); \eta_i = x_i\beta; \beta \sim p(\beta)$$

Ist g der Probit-Link, ist das Modell identisch mit

$$y_i = \begin{cases} 1 & \text{wenn } z_i > 0 \\ 0 & \text{sonst} \end{cases}$$

$$z_i = x_i\beta + \epsilon_i; \epsilon_i \sim N(0, 1); \beta \sim p(\beta)$$

Hilfsvariablenansätze ("Data augmentation")

Ist g der Logit-Link, dann ist das Modell identisch mit

$$\epsilon_i \sim N(0, \lambda_i); \lambda_i = 4\phi^2; \phi \sim KS$$

wobei KS die Kolmogorov-Smirnov-Verteilung ist. Auch hier sind alle F.C. bekannte Verteilungen.

Beispiel: *Bayesian Lasso* (Park und Casella, 2008)

Lineares Modell

$$y_i = x_i\beta + \epsilon_i$$

Ridge-Penalisierung entspricht

$$\beta \sim N(0, \tau); \tau \sim IG(\lambda^2/2, \lambda^2/2)$$

Hilfsvariablenansätze ("Data augmentation")

Lasso-Penaliserung entspricht

$$\beta \sim \mathit{Laplace}$$

Die Laplace-Verteilung lässt sich als skalierte Mischung von Normalverteilungen darstellen. Daraus ergibt sich:

$$\beta \sim N(0, \sigma^2\tau_i); \tau_i \sim IG(\lambda^2/2, \lambda^2/2); p(\sigma^2) \propto 1/\sigma^2$$

Alle Full Conditionals sind dabei bekannte Verteilungen.