Seminar on Statistics, FS 2008

Bayesian Statistics: Computational Aspects

Mattia Bergomi & Cinzia Pedrazzoli *(D-MATH)*

May 6, 2008

Contents

1 Introduction

Given the data x which are realizations of $X \sim f(x|\theta)$, a prior distribution $\pi(\theta)$ and a loss function L, for a Bayes decision we need to minimize the Bayes risk

$$\int \int L(\theta, \delta) f(x|\theta) \pi(\theta)\, dx\, d\theta.$$

This is equivalent to choosing the estimator δ that minimizes the posterior expected loss for each x:

$$d(x) = \arg\min_{\delta} E[L(\theta, \delta)|x] = \arg\min_{\delta} \int_{\Theta} L(\theta, \delta) \pi(\theta|x)\, d\theta$$

The calculation of d has two hindrances:

1. The explicit computation of $\pi(\theta|x)$ may be impossible because of the normalizing constant.
2. In many cases the above integration cannot be computed analytically, even if $\pi(\theta|x)$ is known.

To avoid the usage of simple prior distributions (namely conjugate priors) and simple losses, we need approximation methods, such as numerical methods or simulations.

2 Generation of Random Variables with an arbitrary Distribution in low Dimensions

In this paper we deal with the following problem: given a distribution π on a space $\mathbb{X}$ (with a $\sigma-$algebra $\mathcal{F}$) and the possibility to generate infinitely-many uniform random variables $U_1, U_2, ...$ (e.g. by a pseudo-random number generator on a PC), we want to generate random variables $X_1, X_2, ...$ on $\mathbb{X}$ such that they are all i.i.d. π-distributed. In other words, we would like to simulate from π!

If $\mathbb{X}$ is only $1-$dimensional (w.l.o.g. let us assume $\mathbb{X} = \mathbb{R}$), then a lot of algorithms exist for our purpose (accept/reject, ratio of uniforms, polar coordinates relations, importance sampling, etc.: we will present the *accept-reject method*, since we will also need it later), but if $\mathbb{X}$ is high-dimensional (again let us assume $\mathbb{X} = \mathbb{R}^n$ with n large), then things start to become difficult, and we will need to use Markov Chains for this purpose.

For the rest of this chapter we will hence assume $\mathbb{X} = \mathbb{R}$ (i.e. $1-$dimensional) as well as the target distribution π having density f and cumulative distribution function F (we remark again that for our purposes on Bayesian statistics, π represents the posterior density $\pi(\theta|y)$).

2.1 Direct Quantile Simulation

Definition 1 *The function* $F^{-1}(u)$=$\inf\{x|F(x) \geq u\}, u \in (0,1)$ *is called* quantile function.

Not every distribution admits an analytical computation of F^{-1} (e.g. it is impossible for the normal distribution), but assuming it to be calculable (directly, if F is explicitely invertible, or by numerical approximations otherwise) we immediately obtain the following algorithm/result:

Theorem 1 *If $U \sim Unif(0,1)$, then $X := F^{-1}(U) \sim F$.*

Proof: We assume F to be invertible (otherwise the proof becomes too technical).
Then: $P[X \leq k] = P[F^{-1}(U) \leq k] \overset{F\ invertible}{=} P[U \leq F(k)] \overset{U \sim Unif(0,1)}{=} F(k)$
∎

This is the simplest and easiest way to simulate from F, but for some distributions (normal-, beta-, gamma-) numerical approximations are needed, hence the method is not appropriate.

2.2 Accept-Reject Method

A smarter way to get rid of this obstacles is the *Accept-Reject Method* (ARM for short, *Verwerfungsmethode* in German), which already introduces a background idea that will also be used in the Metropolis-Hastings algorithm for higher-dimensional $\mathbb{X}$'s. The main idea is to firstly simulate from another distribution τ, called the *proposal*, and then "correct" the result such that it fits well for the distribution π. We may assume that π and τ have densities f, g with respect to a common measure μ (often, in the discrete case it is the counting-measure and in the continuous case the Lebesgue-measure). The proposal τ must be chosen such that it is easy for us to simulate from it (otherwise the problem remains the same) and such that it is defined for all x with $f(x) > 0$.

Theorem 2 *If the quotient $f(x)/g(x)$ (called* importance ratio*) is bounded by a constant $M < \infty$, i.e.*

$$a(x) := \frac{f(x)}{g(x)} \leq M < \infty, \forall x$$

then for $X \sim \tau$ and $U \sim Unif(0,1)$ we have that $X|(a(X) \geq U)$ is π-distributed, i.e. we can simulate from π by using $P[X \in A|U \leq a(X)] = \pi(A)$.

$a(x)$ is called the *"acceptance probability"*, since it represents the probability to accept a generated X and not to reject it. We can hence formulate the last theorem as an algorithm for simulating from a target π.

Algorithm 2.1 Accept-Reject(π)

---

1. Choose an appropriate proposal distribution τ s.t. simulating from τ is easy. Define f = density(π), g = density(τ).
2. Simulate $X \sim \tau, U \sim Unif(0,1)$.
3. If $U \leq \frac{f(X)}{M \cdot g(X)}$ accept X as result, otherwise reject it and repeat from step 2.

Proof:

- It holds:

$$P[X \in A, U \leq a(X)] = E[P[X \in A, U \leq a(X)|X]] = \int 1_A(x) \cdot a(x) \tau(dx) =$$

$$= \int 1_A(x) \cdot \frac{f(x)}{M \cdot g(x)} g(x) \mu(dx) = \frac{1}{M} \int_A f(x) \mu(dx) = \frac{\pi(A)}{M}.$$

- Setting $A = \mathbb{X}$ we obtain: $P[U \leq a(X)] = \frac{1}{M} \cdot 1$.
- Therefore: $P[X \in A | U \leq a(X)] = \frac{P[X \in A, U \leq a(X)]}{P[U \leq a(X)]} = \frac{\pi(A)}{M} \cdot M = \pi(A)$.

■

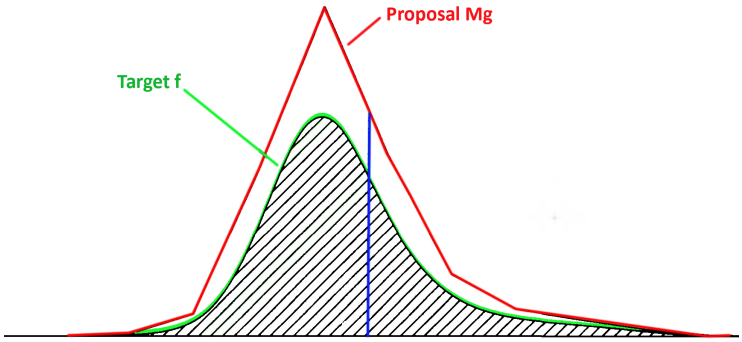

Figure 1: Illustration of the Accept-Reject Method: the top curve is the approximation proposal $Mg(x)$ of the lower curve f. As required, $Mg \geq f$ holds everywhere. The blue vertical line represents a sample from g, accepted only with probability equal to the ratio of the heights of the two curves.

A good approximate density g should be *roughly proportional to* f: the ideal situation would therefore be $g \propto f$ (where, with a suitable M, we accept almost every draw), while when g is not nearly proportional to f, the bound M must be set so large that almost every sample is rejected, turning the method to be (even if still working in theory) practically useless (see Figure 1 to understand why). In fact we have that the expected number of trials before a value is accepted is exactly M, since the acceptance probability $P[U \leq a(X)]$ equals $1/M$.

This happens in particular in high dimensions, where two probability distributions π, τ are typically "almost singular", i.e. there exists an E with $\pi(E) \approx 1, \tau(E) \approx 0$.

3 Monte Carlo Methods

The Monte Carlo methods are numerical methods originally developed in the fifties by physicists (Metropolis, Ulam and Von Neuman) to use random number generation for computing integrals.

To compute an integral $\int_a^b h(x)dx$ we decompose $h(x)$ in a function $f(x)$ and a density $p(x)$ defined over (a, b), then the integral can be expressed as an expectation of $f(x)$ over $p(x)$:

$$\int_a^b h(x)dx = \int_a^b f(x)p(x)dx = E_{p(x)}[f(x)]$$

Thus, if we draw a large number of random variables $x_1, \cdots, x_n$ from $p(x)$, then

$$\int_a^b h(x)dx = E_{p(x)}[f(x)] \simeq \frac{1}{n}\sum_{i=1}^{n} f(x_i).$$

This method is called *Monte Carlo Integration*.

In particular, to approximate the integral $\int_\Theta g(\theta)f(x|\theta)\pi(\theta)d\theta$ we generate random variables $\theta_1, \cdots, \theta_n$ from $\pi(\theta)$, then

$$\frac{1}{n}\sum_{i=1}^{n} g(\theta_i)f(x|\theta_i) \xrightarrow[n\to\infty]{a.s.} \int_\Theta g(\theta)f(x|\theta)\pi(\theta)d\theta$$

according to the law of large numbers. In a similar way, the generation of an i.i.d. sample of θ_i's from $\pi(\theta|x)$ leads to the convergence

$$\frac{1}{n}\sum_{i=1}^{n} g(\theta_i) \xrightarrow[n\to\infty]{} \frac{\int_\Theta g(\theta)f(x|\theta)\pi(\theta)d\theta}{\int_\Theta f(x|\theta)\pi(\theta)d\theta} = E[g(\theta)|x].$$

If the posterior variance $var(g(\theta)|x)$ is finite then, thanks to the central limit theorem, the distribution of $\frac{1}{n}\sum_{i=1}^{n} g(\theta_i)$ is approximately normal with mean $E[g(\theta)|x]$ and variance $var(g(\theta)|x)/n$ for n large, i.e.

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} g(\theta_i) - E[g(\theta)|x]\right) \xrightarrow[n\to\infty]{d} N(0, var(g(\theta)|x)).$$

From this normal distribution we can build confidence regions with an error of order $1/\sqrt{n}$.

We therefore need a method to simulate the θ_i's $\sim \pi(\cdot|x)$.

4 Markov Chains

Let $\mathbb{X}$ be a space with $\sigma - Algebra\ \mathcal{F}$.

Definition 2 *A* Transition Kernel *(or* Transition Probability*) is a function* $K : \mathbb{X}\times\mathcal{F} \to [0,1]$ *such that*

1. $K(x,\cdot)$ *is a probability measure on* $(\mathbb{X},\mathcal{F})$*,*
2. $K(\cdot, A)$ *is a measurable function for each* $A \in \mathcal{F}$*.*

A kernel defines a function on the set of the probabilities on $(\mathbb{X},\mathcal{F})$ given by:

$$\nu K(A) := \int K(x,A)\nu(dx)$$

as well as a function on the set of the bounded, measurable functions on $(\mathbb{X},\mathcal{F})$ given by:

$$Kf(x) := \int f(y)K(x,dy).$$

Further, two kernels can be combined to obtain a new kernel using the construction:

$$K_1K_2(x,A) := \int K_2(y,A)K_1(x,dy).$$

Remark: If $\mathbb{X}$ is finite, K is a matrix $(K(i,j))_{i,j}$ with $K(i,j) \geq 0,\ \forall\ i,j$ and $\sum_j K(i,j) = 1$, following the rules of matrix operations.

The transition kernel also defines a probability measure on the state space, and we assume that the density k exists, i.e. $K(x,A) = K(A|X_n = x) = \int_A k(x,y)dy$.

Definition 3 *A* Markov Chain *on* $(\mathbb{X}, \mathcal{F})$ *with initial distribution* ν_0 *and transition probability* K *is a sequence of random variables* $(X_n)_{n\geq 0}$ *on* $\mathbb{X}$ *such that*

1. $P[X_0 \in A] = \nu_0(A)$
2. $P[X_{t+1} \in A | X_t = x_t, \ldots, X_0 = x_0] = P[X_{t+1} \in A | X_t = x_t] = K(x_t, A)$

That means: given the present state, future states are independent of the past.

Simple consequences are:

- $P[X_{t+k} \in A | X_t = x_t, \ldots, X_0 = x_0] = K^k(x_t, A)$
- $E[f(X_{t+k}) | X_t = x_t] = K^k f(x_t)$
- $P[X_t \in A] = \nu_0 K^t(A)$
- The joint distribution of $(X_0, \cdots, X_n)$ is $\nu_0(dx_0) \prod_{s=1}^{t} K(x_{s-1}, dx_s)$.

For our purposes, we would like the distribution of the chain to be $p(\theta|y)$, but since we don't know this distribution, we want to generate a chain which converges to the distribution $p(\theta|y)$.

We consider a probability distribution Π, assuming its densitiy π to exist: $\Pi(A) = \int_A \pi(x)dx$.

Definition 4 *A probability distribution* Π *on* $(\mathbb{X}, \mathcal{F})$ *is* invariant *or* stationary *for a transition kernel* K *if*

$$\Pi K(A) = \int K(x, A)\Pi(dx) = \Pi(A)$$

Definition 5 *A probability distribution* Π *on* $(\mathbb{X}, \mathcal{F})$ *is* reversible *for a transition kernel* K *if*

$$k(x, y)\pi(x) = k(y, x)\pi(y)$$

Theorem 3 *If* Π *is reversible, then* Π *is stationary.*

Proof: By integration:

$$\int k(y, x)\pi(y)dy = \int k(x, y)\pi(x)dy = \pi(x) \int k(x, y)dy = \pi(x)$$

∎

Thus, since we want $p(\theta|y)$ to be the stationary distribution of our chain, using the above theorem we can consider a chain for which $p(\theta|y)$ is reversible. The reversibility condition will be helpful in the Metropolis-Hastings algorithm, which will produce the requested chain.

Definition 6 *A Markov chain is* irreducible *if for each* A *with* $\Pi(A) > 0$ *and for each* $x \in \mathbb{X}$ *there exists an* n *such that* $K^n(x, A) > 0$.

In other words, an irreducible chain can reach every state starting from every other state. We will see that the construction in the Metropolis-Hastings altorithm introduces dependence between the θ_i's by using the current value as new starting value. This could be a problem, because the law of large numbers holds only if the random variables are independent and identically distributed. However, the ergodic theorem, which states the convergence of $\frac{1}{N}\sum_{n=1}^{N} g(\theta_n)$ against $E_\pi[g(\theta)|x]$ (if $E_\pi[|g(\theta)||x] < \infty$), removes this problem. Ergodicity vanishes the influence of the starting value. Our chain is then convergent to a unique stationary distribution, and the average converges to the corresponding expectation, as the following theorem states:

Theorem 4 *Suppose that $\{X_t\}$ is irreducible with transition kernel K and invariant distribution π. Then π is the only invariant distribution. Moreover, if $\int |g(x)|\pi(dx) < \infty$, then*

$$P\left[\frac{1}{N+1}\sum_{t=0}^{N} g(X_t) \xrightarrow[N\to\infty]{} \int g(x)\pi(dx) \mid X_0 = x\right] = 1$$

for π-almost all starting values x (under some additional conditions the "π-almost" even disappears...).

5 MCMC Methods: Metropolis-Hastings Algorithm & Gibbs Sampler

Markov Chain Monte Carlo Method (MCMC for short) is a general (and nowadays standard) way of simulating draws from distributions in high dimensions. The idea is to draw values of θ from an approximate distribution (which plays an analogue role to the proposal distribution seen in the Accept-Reject Method) and then correct those draws to better approximate the target posterior distribution $p(\theta|y)$. The samples $(\theta_t)_{t=1,2,...}$ are drawn sequentially, where each draw only depends on the last value drawn; hence, the draws form a Markov Chain. Our target is to construct a Markov Chain P such that for values of t large enough, θ_t has approximatively the right distribution $p(\theta|y)$.

MCMC is obviously used only when it is not possible to sample θ directly from $p(\theta|y)$; instead, we sample it iteratively in such a way that at every step of our algorithm the distribution of the last draw becomes always closer to the target $p(\theta|y)$.

Since this method is much more general and also has a lot of applications in non-Bayesian statistics, we describe the method in a general way by letting again the target distribution be indicated by π (for our purposes we should therefore simply set $\pi := p(\theta|y)$...).

We also notice that, after the process converged, not every draw computed during the steps has an approximate distribution π, since we started from an arbitrary distribution and "moved" towards π. To diminish the effect of the starting distribution, we discard the early iterations of the simulation runs: this process is known as the *burn-in*. A usual convention says to discard the first half of the iterations and to concentrate on the second half, however this is not a general rule and other strategies can be considered.

Our job is therefore now to construct a Markov Chain (i.e. its kernel P) that satisfies the following properties:

- π is invariant (stationary) for P, i.e. $\pi P = \pi$.
- π is the unique stationary distribution of P.
- We are able to simulate from P (at least easier than from π).

And then simply generate a Markov Chain X_t with transition P.

There are also some interesting questions as:

- Will the process converge for any starting distribution?
- How long do we have to wait until the process has converged "enough" (burn-in strategy)?
- ...

In the following we denote the density of P by $p(y|x)$.

5.1 The Metropolis-Hastings Algorithm (MHA)

The Metropolis-Hastings Algorithm (that has been considered one of the top 10 algorithms of the millennium!) was invented by Nicholas Constantine Metropolis (in 1953, with some limitations in it) and later generalized by W. Keith Hastings (in 1970) to the actual form. It is nowadays (together with the Gibbs Sampler that we will see later) the standard way to sample from high-dimensional probability distributions. The original form of the algorithm had some limitations (the most important was that the proposal distribution must be symmetric) that have been fixed in the Metropolis-Hastings form: we will therefore directly talk about the definitive version of the algorithm.

From the conditions of the last section, we want to construct a kernel P such that π is its unique stationary distribution and we are able to simulate from P. Using the properties seen on the Chapter about Markov Chains, we can restate these requests as follows:

- P is irreducible *($\Rightarrow$ unique invariant distribution)*.
- π is reversible for P *($\Rightarrow$ π is stationary)*.
- We can construct an easy method for simulating from $P(x, A)$.
- Eventually, P is aperiodic *(which ensures that $P^n(x, \cdot) \to \pi(\cdot)$ for π-almost all x)*.

The condition of reversibility states:

$$p(y|x)\pi(x) = p(x|y)\pi(y).$$

If we make a proposal q (admissible, i.e. $q(\cdot|x)$ is defined where $\pi(x) > 0$), then this will not generally satisfy the above needed equation (assume w.l.o.g. $q(y|x)\pi(x) > q(x|y)\pi(y)$), but we can find a factor $0 \leq a(x, y) \leq 1$ s.t.

$$q(y|x)\pi(x) \cdot a(x, y) = q(x|y)\pi(y)$$

i.e.

$$a(x,y) = \min\left\{\frac{q(x|y)\pi(y)}{q(y|x)\pi(x)}, 1\right\}$$

The transition kernel corresponding to the modified equation is then:

$$p(A|x) = \underbrace{\int_A q(y|x)a(x,y)dy}_{A\ proposed\ \&\ accepted} + \underbrace{1_{x\in A}\left(1 - \int_{\mathbb{X}} q(y|x)a(x,y)dy\right)}_{staying}$$

Algorithm 5.1 Metropolis-Hastings(π)

1. Choose an admissible proposal kernel $q(\cdot|\cdot)$, from which it is easy to simulate.
2. Choose a starting point X_0 (e.g. uniformly at random).
3. *For t=1...N*
   (a) Simulate $Y \sim q(Y|X_{t-1})$ and $U \sim Unif(0,1)$.
   (b) Compute $a(X_{t-1}, Y) = \min\left\{\frac{q(X_{t-1}|Y)\pi(Y)}{q(Y|X_{t-1})\pi(X_{t-1})}, 1\right\}$
   If $U \leq a(X_{t-1}, Y)$ **then** set $X_t = Y$ **else** set $X_t = X_{t-1}$

   End For
4. Burn-in: Delete the first $X_{1\ldots r}$, since possibly meaningless.

In order to understand why this works, we would better consider accurately the discrete case (where the kernel being a matrix helps a lot in understanding concretely what's going on...):

5.1.1 The Metropolis-Hastings Algorithm - Discrete Case

The condition of reversibility now simply states:

$$\pi(i)P(i,j) = \pi(j)P(j,i)$$

We see that, once a component $P(a,b)$ is determined, the other one $P(b,a)$ is already uniquely defined. The components $P(a,a)$ are instead free for this condition. Since the Kernel must be a stochastic matrix, we also need $\sum_j P(i,j) = 1$, that cannot be achieved by simply choosing the components $P(i,j), i \leq j$ at random. Starting from the observation that, if the sum $\sum_{j\neq i} P(i,j)$ is < 1, we can easily "correct" by setting $P(i,i) = 1 - \sum_{j\neq i} P(i,j)$, we arrive at the following construction:

Start with any transition matrix $Q(i,j)$ (i.e. stochastic matrix, the *proposal distribution*) that only satisfies the (not too strong) condition $Q(i,j) > 0 \Leftrightarrow Q(j,i) > 0$ (we will need this when defining the minimum later...).
Set $P(i,j) = Q(i,j)$ OR $P(j,i) = Q(j,i)$ (and the other value according to the rule above) in such a way that both $P(i,j) \leq Q(i,j)$ AND $P(j,i) \leq Q(j,i)$. Therefore:

$$\sum_{j\neq i} P(i,j) \leq \sum_{j\neq i} Q(i,j) \leq 1$$

and with $P(i,i) := 1 - \sum_{j \neq i} P(i,j)$ we get the desired stochastic condition.

The choice of $P(i,j) = Q(i,j)$ OR $P(j,i) = Q(j,i)$ is easy to achieve, since:

$$P(i,j) = Q(i,j) \Rightarrow P(j,i) = \frac{\pi(i)Q(i,j)}{\pi(j)} \overset{!}{\leq} Q(j,i) \Leftrightarrow \pi(i)Q(i,j) \overset{!}{\leq} \pi(j)Q(j,i)$$

We can write this more compactly (and in a known way) as:

$$P(i,j) := \min\left\{Q(i,j), \frac{\pi(j)}{\pi(i)}Q(j,i)\right\} = Q(i,j) \cdot \overbrace{\min\left\{1, \frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}\right\}}^{=:a(i,j)\leq 1}$$

$$P(i,i) := 1 - \sum_{j \neq i} P(i,j)$$

As we will see immediately, $a(i,j)$ represents again an *acceptance probability*, that plays almost the same role as in the Accept-Reject Algorithm.

Theorem 5 *Simulating from P is now simple by using the following algorithm:*

Algorithm 5.2 Metropolis-Hastings-Discrete(π)

---

1. Choose proposal kernel $Q(i,j)$ that satisfies $Q(i,j) > 0 \Leftrightarrow Q(j,i) > 0$, from which it is easy to simulate.
2. Choose a starting point X_0 (e.g. uniformly at random).
3. *For t=1...N*
   (a) Simulate $Y \sim Q(X_{t-1}, \cdot)$ and $U \sim Unif(0,1)$.
   (b) **If** $U \leq a(X_{t-1}, Y)$ **then** set $X_t = Y$ **else** set $X_t = X_{t-1}$

   End For
4. Burn-in: Delete the first $X_{1...r}$, since possibly meaningless.

Proof: Having defined P as $P(i,j) := Q(i,j) \cdot a(i,j)$ and $P(i,i) := 1 - \sum_{j \neq i} P(i,j)$ it is easy to see that in fact we are really simulating from P:

$$P[X_t = j | X_{t-1} = i] = \overbrace{Q_{i,j}}^{proposed} \cdot \overbrace{a_{i,j}}^{accepted} = P_{i,j}, \forall j \neq i$$

$$P[X_t = i | X_{t-1} = i] = \underbrace{Q_{i,i} \cdot a_{i,i}}_{proposed+accepted} + \underbrace{\sum_{j \neq i} Q_{i,j}(1 - a_{i,j})}_{not\ moving} = 1 - \sum_{j \neq i} Q_{i,j} a_{i,j} = P_{i,i}$$

■

5.1.2 Remarks to the Metropolis-Hastings Algorithm

- The role played by $a(i,j)$ is a bit different from the AR Method, since now even if we don't accept a drawn value Y, this counts as a step of the algorithm and the unchanged value $X_t = X_{t-1}$ counts as a new valid draw!
- The target distribution π needs only to be known up to a normalizing constant, since this cancels in the computation of $a(i,j)$. This is very important, since in our examples we will always only know $\pi \propto$ ....
- The possibility of using non-symmetric proposals helps in increasing the speed of the convergence.
- The ideal situation would be to sample proposals directly from the target distribution π, where the ratio would always be $= 1 \Rightarrow$ every draw is accepted (as it should be).
- An empirical way for checking convergence is to let 2 or more different chains run in parallel (see Example 7.2) and see if they are concentrating "on the same place".
- A good proposal should be chosen such that:
  - The ratio is easy to compute.
  - Each jump moves "reasonably much" (otherwise the chain moves to slowly).
  - The jumps are not rejected too often (otherwise we always stay at the same places).
- Often we allow the proposal to change during the iteration (i.e. to depend on the number t of the iteration: $q(\cdot,\cdot) = q_t(\cdot,\cdot)$), using for example parameters computed during the previous steps (see the proof of "Gibbs Sampler = special case of Metropolis-Hastings" for an example).
- Some common choices for q are:
  - Symmetric: $q(x|y) = q(y|x)$, since the ratio simplifies a lot (but convergence could be slow). If q only depends on $|y - x|$, then it is called *Metropolis Random Walk*.
  - Independent: $q(y|x) = q(y)$, and it is called *Independence Metropolis*.

5.2 The Gibbs Sampler

The *Gibbs Sampler* (also called *Alternating Conditional Sampling*) is a special case of the Metropolis-Hastings Algorithm, where only a part of the vector X is updated at a time, conditioned on the others entries. More concretely, let's consider a subdivision $X = (X_1, ..., X_d)$ of X in d subvectors (usually one can set $d = n$ and consider the single components of X as a subdivision). Each iteration of the Gibbs Sampler *cycles through the subvectors of X, drawing each subset conditional on the value of all the others* (there are thus d steps during iteration t). At each iteration t an order of the d subvectors is chosen (usually simply going straightforward $1, 2, .., d$) and each X_j^t is sampled from the conditional distribution given all the other components of X (using the most recent ones):

$$X_j^t \sim \pi(X_j|X_{-j}^{t-1})$$

where the notation is:

$$X_{-j}^{t-1} := (X_1^t, ..., X_{j-1}^t, X_{j+1}^{t-1}, ..., X_d^{t-1})$$

Thus, each subvector X_j is updated conditional on the latest values of the other components of X.

5.2.1 Remarks to the Gibbs Sampler

- Is the simplest of the MCMC Methods and the first choice for conditionally conjugate models, where we can directly sample from each conditional distribution. Recall for this the last seminar about *Hierarchical Models*!
- A general problem is when the parameters are highly correlated in the target distribution, then the convergence could be very slow (see Example 7.2). Usually this can be fixed by a *reparameterization* of the parameters of interest.
- If some of the conditional posteriors can be sampled and some cannot, then a usual practice is to update the parameters one by one, with Gibbs Sampling where possible and one-dimensional Metropolis Hastings otherwise. This method is therefore a *Combination of Gibbs Sampler and Metropolis-Hastings*: for parameters whose conditional posterior distributions have standard forms, use Gibbs Sampler, otherwise use Metropolis Jumps.
- Gibbs Sampling is indeed a special case of the Metropolis-Hastings Algorithm, where each iteration t consists of d steps (i.e. the whole vector X^{t-1} is updated during 1 iteration), where at step j the proposal kernel is defined as:

  $$q_{j,t}^{Gibbs}(\hat{X}|X^{t-1}) = \begin{cases} \pi(\hat{X}_j|X_{-j}^{t-1}) & \hat{X}_{-j} = X_{-j}^{t-1} \\ 0 & otherwise \end{cases}$$

  i.e. the only possible jumps are between vectors with only the $j-th$ component possibly different. Under this jumping distribution, the ratio is always $= 1$, hence every step is always accepted.
- Sometimes, if the conditional density $\pi(X_j|X_{-j}^{t-1})$ is not computable, one approximates it with another easier function $\tilde{\pi}$.

6 The Ising Model

The Ising Model (that won a nobel price!) is a way to model the behaviour of the spins of the atoms of a ferromagnete. The particles (i.e. atoms) of the ferromagnete are supposed to sit on a d-dimensional lattice (grid) $L = \{1; 2; ...; n\}^d$ (we will consider the 2D case, i.e. $d = 2$) and each particle i has a spin up ($\sigma_i = +1$) or $down$ ($\sigma_i = -1$). A *Configuration* of the system is therefore a matrix $\sigma \in \{-1; +1\}^L$. Every pair of particles has an interaction energy of $J_{i,j}\sigma_i\sigma_j$, i.e. dependent from spins and distance (we remark that, usually, the atoms who interact are the neighbor ones). To a configuration σ we can associate a lot of properties as the total energy $E(\sigma) =$

$-\sum_{i,j} J_{i,j}\sigma_i\sigma_j$, the total magnetization M_n ={arithmetic mean of the σ_i's}, etc. Every configuration σ is modeled to be random (due to thermal noise) with:

$$\pi(\sigma) = \frac{1}{C} e^{\frac{1}{T}E(\sigma)} = \pi(-\sigma)$$

where C is only a normalization constant and T is the absolute temperature of the system. This model is well appropriated for a Gibbs-Sampling simulation for the distribution π, also noting that a single spin modification produces only a small variation of the energy ΔE. A simulation for the configuration σ needs also an input parameter T, the temperature, that will be crucial for the behaviour of the ferromagnete: it exists a critical temperature $T_{critical}$ such that for $T > T_{critical}$ the shape of the spins remains more or less equidistributed (like a noise-pattern), while for $T < T_{critical}$ the shape is completely destroyed having all the spins in a unique direction.

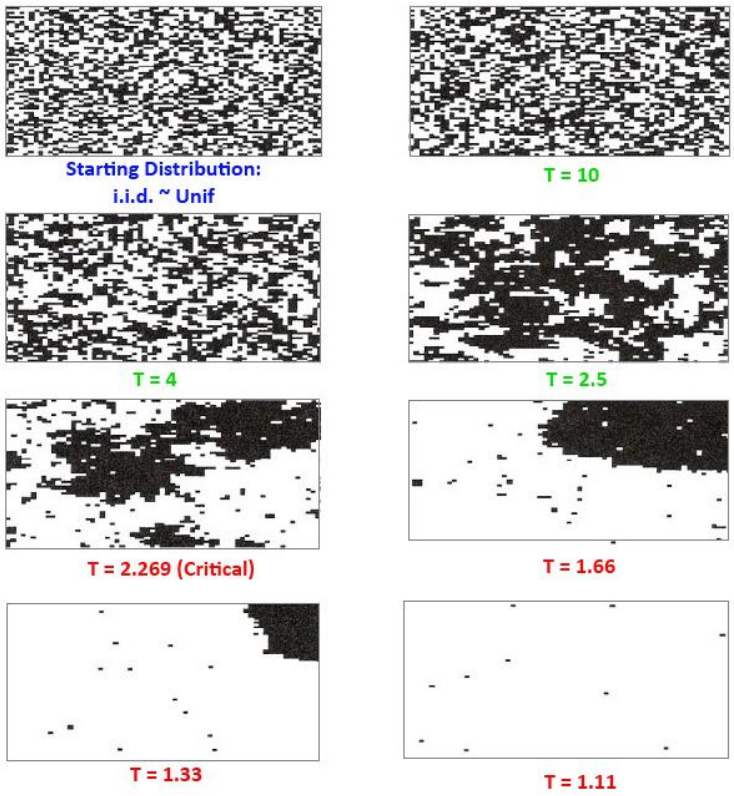

Figure 2: Illustration of the Ising Model with Gibbs Sampling: the critical temperature is $T_{critical} = 2.269$ (Images produced with the R-Code of [3], 1000 Iterations)

Another question could be about the distribution of the total magnetization M_n for n large. The answer is that for $T < T_{critical}$ the shape always remains the one of the bimodal distribution (symmetric around zero).

7 Examples

Let's just recall the main definitions/properties:

Prior Density $\pi(\theta)$

Density of Observations $f(y|\theta)$

Joint Density $f(y,\theta) = f(y|\theta) \cdot \pi(\theta)$

Posterior Density $\pi(\theta|y) = \frac{\pi(\theta)f(y|\theta)}{\int_\Theta \pi(\theta)f(y|\theta)d\theta} \propto \pi(\theta)f(y|\theta)$

Predictive Density $f(y_{future}|y) = \int f(y_{future}|y,\theta)\pi(\theta|y)d\theta$

Our aim is to simulate the posterior density $\pi(\theta|y)$ under some model assumptions.

7.1 Metropolis-Hastings: Example 1

Let's consider the *Weibull Distribution* for our data, with density

$$f_{\alpha,\eta}(y) = \alpha\eta y^{\alpha-1}e^{-y^\alpha\eta}$$

This doesn't belong to the exponential family and explicit posteriors for α, η are not analytically computable. Let's further consider the (independent) priors for the parameters:

$$\begin{array}{rrcl} & \pi(\alpha) & \propto & e^{-\alpha} \\ & \pi(\eta) & \propto & \eta^{\beta-1} \cdot e^{-\beta\eta} \\ \Rightarrow & \pi(\alpha,\eta) & \propto & e^{-\alpha} \cdot \eta^{\beta-1} \cdot e^{-\beta\eta} \end{array}$$

with observations $Y = [0.2, 0.1, 0.25]$.

We want to approximate the posterior $\pi(\alpha,\eta|Y)$ using the Metropolis-Hastings Algorithm, using a proposal distribution

$$q(\alpha',\eta'|\alpha,\eta) = \frac{1}{\alpha\eta}e^{-\left(\frac{\alpha'}{\alpha}+\frac{\eta'}{\eta}\right)}$$

(a product of two independent exponentials with means α and η) (we remark that the kernel q is not symmetric, therefore the original Metropolis Algorithm would not suffice). The ratio $a(\cdot,\cdot)$ used in algorithm becomes therefore (writing $\alpha_{prop} = \alpha', \eta_{prop} = \eta'$):

$$\frac{q(\alpha,\eta|\alpha',\eta')\pi(\alpha',\eta'|Y)}{q(\alpha',\eta'|\alpha,\eta)\pi(\alpha,\eta|Y)} = \frac{q(\alpha,\eta|\alpha',\eta')\pi(\alpha',\eta') \cdot f(Y|\alpha',\eta')}{q(\alpha',\eta'|\alpha,\eta)\pi(\alpha,\eta) \cdot f(Y|\alpha,\eta)} =$$

$$= \left(\frac{\alpha'\eta'}{\alpha\eta}\right)^{n-1} \cdot \left(\frac{\eta'}{\eta}\right)^{\beta-1} \cdot e^{\alpha-\alpha'+\beta(\eta-\eta')+\frac{\alpha'}{\alpha}+\frac{\eta'}{\eta}-\frac{\alpha}{\alpha'}-\frac{\eta}{\eta'}} \cdot (y_1 \cdot \ldots \cdot y_n)^{\alpha'-\alpha} \cdot \left(e^{\eta y_1^\alpha - \eta' y_1^{\alpha'}} \cdot \ldots \cdot e^{\eta y_n^\alpha - \eta' y_n^{\alpha'}}\right)$$

Choosing an arbitrary starting point (for example $\alpha_0 = 2, \eta_0 = 2$) and a parameter β (for example $\beta = 2$) we can therefore use the following Matlab-Code (Algorithm 5.1) to run the simulation.

Some interesting results we can see are:

- The run:
  `[alphas, etas] = MetropolisHastings1([0.2 0.1 0.25], 2, 2, 2, 10000, 5000, 0);`
- The histogram of the generated α_i's and η_i's:
  `hist(alphas,100); hist(etas,100);`
- The plot of the generated points without lines:
  `plot(alphas, etas, '.')`
- The empirical mean of the θ_i's:
  `mean(alphas); mean(etas);`

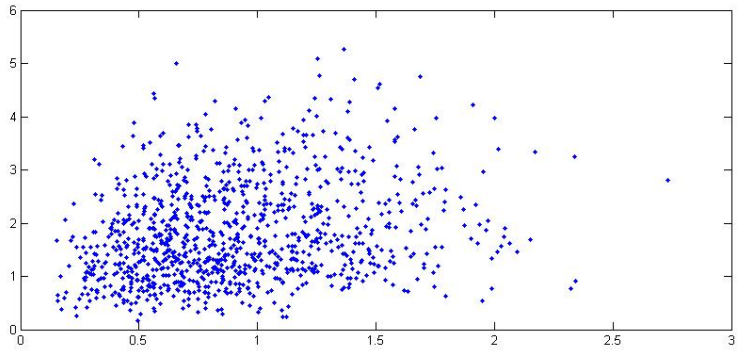

Figure 3: Illustration of Example 7.1 using Metropolis-Hastings on a Weibull Distribution: `plot(alphas, etas, '.')`

7.2 Gibbs Sampler: Example 2

Consider a single observation (y_1, y_2) of a *Bivariate Normal Distribution* (the 2-dimensional case of the normal distribution) with unknown mean $\theta = (\theta_1, \theta_2)$ and known covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$.
Assuming a uniform prior $\theta \sim Unif$ the posterior is:

$$p(\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} |y) \sim N\left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right)$$

From the multivariate normal distribution properties we know that:

$$X = (X_1 ... X_n) \overset{i.i.d}{\sim} N(0,1) \Rightarrow Y := \mu + AX \sim N_n(\mu, AA^T)$$

It would be therefore easy to directly simulate from $p(\theta|y)$ (e.g. by Cholesky decomposition of $\Sigma = AA^T$), but for the purpose of explanation we want to use here the Gibbs Sampler (having therefore a way to control if the computed result is right or not).

Algorithm 7.1 Metropolis-Hastings - Weibull Distribution

```
function [alphas, etas] = MetropolisHastings1(Y, beta,
                          alpha0, eta0, Iter, BurnIn, stop)
clf; hold on;

n      = length(Y);
alpha  = alpha0;
eta    = eta0;
alphas = [alpha];
etas   = [eta];
plot(alpha0,eta0,'s','MarkerSize',10 );
pause;

for k=1:Iter % Metropolis-Hastings

 alpha_prop = -alpha * log(rand()); % Quantile Simulation
 eta_prop   = -eta   * log(rand());

 alpha_old = alpha; % Only for the Picture
 eta_old   = eta;

 ProdExpY = prod(exp(eta*Y.^alpha-eta_prop*Y.^alpha_prop));

 ratio = (alpha_prop*eta_prop/alpha/eta)^(n-1) * ...
         (eta_prop/eta)^(beta-1) * exp(alpha -   ...
         alpha_prop + beta*(eta-eta_prop)+       ...
         alpha_prop/alpha + eta_prop/eta-        ...
         alpha/alpha_prop-eta/eta_prop) *        ...
         prod(Y)^(alpha_prop-alpha)*ProdExpY;

 a = min(ratio, 1);
 if (rand() <= a), alpha = alpha_prop; eta=eta_prop; end;
 alphas = [alphas alpha];
 etas   = [etas eta];

 line([alpha_old,alpha],[eta_old,eta]);
 if stop, pause(); end;

end

% BurnIn:
alphas = alphas(BurnIn:end);
etas   = etas(BurnIn:end);

return;
```

To apply the Gibbs Sampler we only need the conditional posterior distributions $\pi(\theta_i|\theta_j, y)$ which, from the properties of the multivariate normal distribution, are simply given by:

$$\theta_1|\theta_2, y \sim N(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2)$$
$$\theta_2|\theta_1, y \sim N(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2)$$

Choosing an arbitrary starting point, we can therefore use the following Matlab-Code (Algorithm 7.2) to simulate $\theta|y$ using the Gibbs Sampler.

Some interesting results we can show are:

- The run of several chains that converge on the same space by:
  `thetas = GibbsSampler1(0.8,[0;0],[-2.5,-2.5,-2.5,2.5,2.5,-2.5,2.5,2.5],10000,0)`
- The histogram of the generated θ_i's:
  `hist(thetas(1,1:10001),100)`
- The plot of the generated points (without lines):
  `plot(thetas(1,5000:10001),thetas(2,5000:10001),'.')`
- The empirical mean of the θ_i's:
  `mean(thetas(1,5000:10001))`

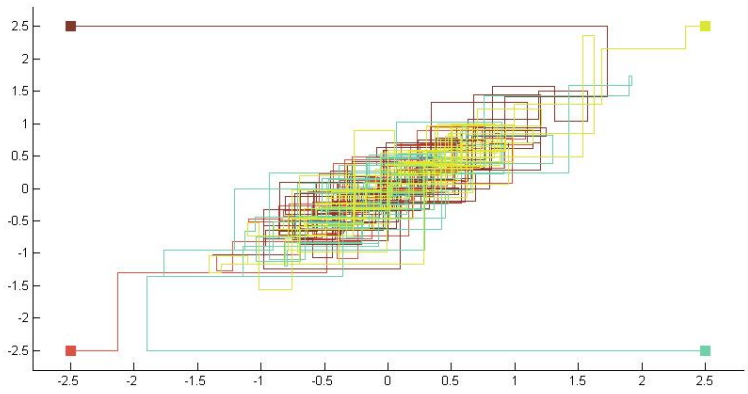

Figure 4: Illustration of Example 7.2 using Gibbs Sampling on a Bivariate Normal Distribution: the figure is obtained by the call `GibbsSampler1(0.8,[0;0],[-2.5,-2.5,-2.5,2.5,2.5,-2.5,2.5,2.5],100,0)`

Algorithm 7.2 GibbsSampler - Bivariate Normal Distribution

```
function thetas = GibbsSampler1(rho, y, theta, Iter, stop)
%(Stop = boolean to stop after each iteration)

clf; hold on;
axis([-2.8, 2.8, -2.8, 2.8]);
n = length(theta)/2; % #of starting points = #of chains
thetas =[];          % Generated Points

for i=1:n  %Different Chains
 rand('state',i+1); Color = [rand() rand() rand()];
 t(1)=theta(2*i-1); %Starting Points
 t(2)=theta(2*i);
 thetas = [thetas, t'];
 plot(t(1),t(2),'s','MarkerFaceColor',Color,
      'MarkerEdgeColor',Color, 'MarkerSize',10 );
 pause;

 for k=1:Iter %Gibbs Sampling
  %Update theta(1)
  old = t;
  t(1)=y(1)+rho*(t(2)-y(2)) + randn()*(1-rho^2);
  line([old(1),t(1)],[old(2),t(2)],'Color',Color);
  if stop, pause(); end;

  %Update theta(2)
  old = t;
  t(2)=y(2)+rho*(t(1)-y(1)) + randn()*(1-rho^2);
  line([old(1),t(1)],[old(2),t(2)],'Color',Color);
  if stop, pause(); end;
  thetas = [thetas, t'];
 end
end

return;
```

7.3 Gibbs Sampler: Example 3

Let's now consider a more complicated example of a sample with independent priors:

$$\begin{array}{rcl} Y_1, ..., Y_n & \overset{i.i.d}{\sim} & N(\mu, 1/\tau) \\ \mu & \sim & N(\xi, 1/\kappa) \\ \tau & \sim & \Gamma(\alpha, \beta) \end{array}$$

The joint distribution is:

$$\begin{array}{rcl} f(Y, \mu, \tau) & = & \left[\prod_{i=1}^n f(Y_i|\mu, \tau)\right] \pi(\mu)\pi(\tau) \\ & = & \left[\prod_{i=1}^n \frac{\sqrt{(\tau)}}{\sqrt{(2\pi)}} e^{\frac{-(Y_i-\mu)^2\tau}{2}}\right] \cdot \frac{\sqrt{\kappa}}{\sqrt{2\pi}} e^{\frac{-(\mu-\xi)^2\kappa}{2}} \cdot \tau^{\alpha-1} \frac{e^{-\frac{\tau}{\beta}}}{\beta^\alpha \Gamma(\alpha)} \\ & = & \frac{\sqrt{\kappa}}{\sqrt{2\pi}^{n+1} \cdot \beta^\alpha \Gamma(\alpha)} \cdot \tau^{\frac{n}{2}+\alpha-1} \cdot e^{-\frac{\tau}{2}} \cdot e^{-\frac{\kappa}{2}(\mu-\xi)^2} \cdot e^{-\frac{\tau}{2}\sum_{i=1}^n (Y_i-\mu)^2} \end{array}$$

To find the full conditional for μ (resp. τ) we select only the terms containing μ (resp. τ) and normalizing:

$$\begin{array}{rcl} \pi(\mu|\tau, Y) & \propto & e^{-\frac{\tau}{2}\sum_{i=1}^n (Y_i-\mu)^2} \cdot e^{-\frac{\kappa}{2}(\mu-\xi)^2} \\ & \propto & e^{-\frac{1}{2}(1+n\tau)\left(\mu - \frac{\tau\sum_{i=1}^n Y_i}{1+n\tau}\right)^2} \\ & \sim & N(\frac{\tau\sum_{i=1}^n Y_i}{1+n\tau}, \frac{1}{1+n\tau}) \\ \\ \pi(\tau|\mu, Y) & \propto & \tau^{\frac{n}{2}+\alpha-1} e^{-\frac{\tau}{2}\sum_{i=1}^n (Y_i-\mu)^2} \cdot e^{-\frac{\tau}{\beta}} \\ & \sim & \Gamma(2+\frac{n}{2}, 1+\frac{1}{2}\sum_{i=1}^n (Y_i-\mu)^2) \end{array}$$

Let's take $\xi = 0, \kappa = 1, \alpha = 2, \beta = 1, n = 30$ and $Y_i \sim N(1, 4^2)$ and again use the Gibbs Sampler for the simulation (Matlab-Code in Algorithm 7.3). We can plot/compute some interesting properties:

- Run: `[mus, taus] = GibbsSampler2(30, 1, 1/16, 0, 2, 10000, 500)`
- The histogram of the generated μ's, τ's:
  `hist(mus,100); hist(taus,100);`
- The plot of some generated points without lines:
  `axis([-2 4 0.02 0.16]);`
  `plot(mus(7500:9500),taus(7500:9500),'.')`
- The empirical means:
  `mean(mus) -> 1.1934; mean(taus); -> 0.0653`
  We notice that the approximations are quite good, since we used $\mu = 1$, $\tau = 1/16 = 0.0625$.

Algorithm 7.3 GibbsSampler - Normal/Gamma Distribution

```
function [mus, taus] = GibbsSampler2(n, mu_r, tau_r, mu_0,
                                     tau_0, Iter, BurnIn)
clf; hold on;
randn('state',10);

Y = mu_r + 1/sqrt(tau_r)*randn(1,n);
SumY = sum(Y);

mus  = [mu_0]; taus = [tau_0];
mu   = mu_0;   tau  = tau_0;
plot(mu_0,tau_0,'s','MarkerSize',10 );

for k=1:Iter %Gibbs Sampling
 %Update mu
  mu_new = (tau*SumY)/(1+n*tau) + randn()* sqrt(1/(1+n*tau));
  line([mu,mu_new],[tau,tau]);
  mus = [mus mu_new];
  mu = mu_new;

  %Update tau
  tau_new = rand_gamma(2+n/2, 1 + sum((Y-mu).^2)/2,1,1);
  line([mu,mu],[tau,tau_new]);
  taus = [taus tau_new];
  tau = tau_new;
end

%Burn-In:
mus  = mus (BurnIn:Iter+1);
taus = taus(BurnIn:Iter+1);

hold off; return;
```

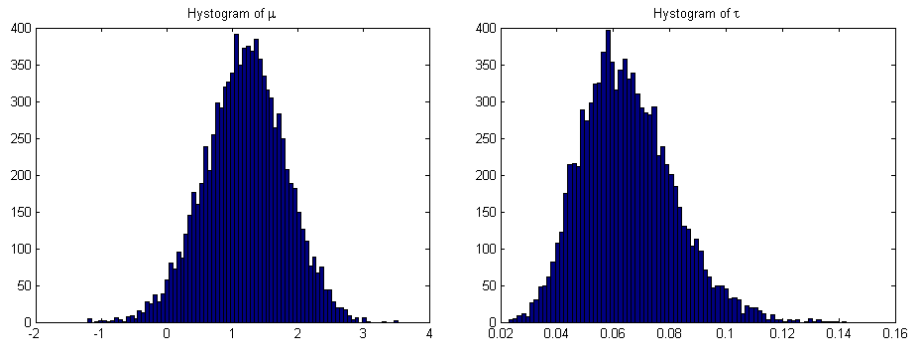

Figure 5: Illustration of Example 7.3 using Gibbs Sampling on a Normal/Gamma Distribution: the figure shows the histograms of the obtained parameters μ, τ.

8 Bibliography

References

[1] Brani Vidakovic, *MCMC: Metropolis and Family of Algorithms, Gibbs Sampler*. Handout in Bayesian Statistics *(http://www2.isye.gatech.edu/∼brani/isyebayes/handouts.html)*.

[2] Brani Vidakovic, *Bayesian Computation, Numerical Algorithms and Monte Carlo Methods; Importance Sampling and ARM*. Handout in Bayesian Statistics *(http://www2.isye.gatech.edu/∼brani/isyebayes/handouts.html)*.

[3] Hansruedi Künsch, *"Stochastische Simulation"*. Lecture Notes of the lecture in HS07/08, ETHZ.

[4] Claudia Spellecchia, *MCMC*. Handout in Top10-Algorithms Seminar, 13.03.2008 ETHZ.

[5] Isabel Beichl, Francis Sullivan, *The Metropolis Algorithm*. Computing in Science & Engineering - The Top10 Algorithms of the Millenium *(http://cise.aip.org/getpdf/servlet/GetPDFServlet?filetype=pdf&id= CSENFA000002000001000065000001&idtype=cvips)*.

[6] Christian P. Robert, *The Bayesian Choice - From Decision-Theoretic Foundations to Computational Implementation (Volume 2)*. Springer Science and Business Media, 2nd edition, 2007.

[7] Andrew Gelman, John B. Carlin, Hal S. Stern, Donald B. Rubin. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall/CRC, 2nd edition, 2003.