Chapter 9

Markov chain Monte Carlo methods

In this chapter we will continue with the Monte Carlo methods, but with a particulate family of MC metohds, that is, Monte Carlo Markov chain (MCMC). The MCMC methods have become very popular over the few recent decades with the improved power of computers, and because they offer a quite generic tool that is especially suitable for Bayesian estimation problems with no closed-form solution.

The material in this chapter is based mainly on two references, first, the book by Robert & Casella, *Monte Carlo statistical methods* (1999, Springer texts in statistics), and second, the lecture notes by Solonen, Haario, and Laine, for the course *Statistical Analysis in Modeling* (2014, Lappeenranta University of Technology).

Note that in this chapter we deal quite often with some non-specified distribution, which I try to mark with the letter 'f', so the distribution is $\mathcal{F}$ and its probability density function is f. Furthermore, in most of the methods we need an auxiliary (often called instrumental/proposal) distribution marked with 'g', so distribution $\mathcal{G}$ and pdf g. And, because these distributions appear in the algorithms a lot, it is sometimes more convenient to write shortly something like $\mathrm{f}(x|y)$, which should be understood to mean the pdf of the conditional distribution, $\mathrm{f}_{X|Y}(x,y)$, or in some cases even the distribution $X|Y \sim \mathcal{F}(X,Y)$.

Before actually going to MCMC, I will introduce two 'regular' MC methods that have lead the way towards the MCMC and the Metropolis-Hastings algorithm.

9.1 Monte Carlo towards MCMC

Importance sampling is a technique which allows us to sample from a distribution $\mathcal{F}$ without actually being able to simulate $\mathcal{F}$ directly. The second example, the simulated annealing algorithm, is leading us to the Metropolis-Hastings algorithm in MCMC.

9.2 Importance sampling

Importance sampling is related to the accept-reject method of creating random numbers from a distribution $\mathcal{F}$ with the help of an envelope distribution $\mathcal{G}$ (see Sec. 8.1.3), but in this technique, all the proposed random numbers are accepted, only with different weights.

The algorithm is based on the fact, that for any function h of the random variable X, the expected value can be computed as

$$\mathrm{E}_f[\mathrm{h}(x)] = \int \mathrm{h}(x)\mathrm{f}(x)dx = \int \mathrm{h}(x)\frac{\mathrm{f}(x)}{\mathrm{g}(x)}\mathrm{g}(x)dx. \tag{9.1}$$

In the last form we have simply multiplied and divided with the pdf g of an instrumental distribution $\mathcal{G}$. For the equality to hold, the support of g (where $\mathrm{g} > 0$) has to be at least the support of f.

Now the last form in the previous equation means that we can simulate a sample $x_1, \ldots, x_n$ from distribution $\mathcal{G}$, and estimate

$$\mathrm{E}_f[\mathrm{h}(x)] \approx \frac{1}{n}\sum_{i=1}^{n}\mathrm{h}(x_i)\frac{\mathrm{f}(x_i)}{\mathrm{g}(x_i)}. \tag{9.2}$$

Especially if the $\mathrm{h}(x) = x$, we are estimating the mean of the distribution $\mathcal{F}$.

The importance sampling is useful when the distribution $\mathcal{F}$ is impossible or very costly to simulate from, but the distribution $\mathcal{G}$ is not. There is a practical limitation to $\mathcal{G}$, the pdf g should have 'heavier' tails than f, that is, the ratio f/g should not approach to ∞. If the tails of g would be 'lighter' than of f, then with rare cases of large $|x_i|$, the weights $\mathrm{f}(x_i)/\mathrm{g}(x_i)$ would be very large, and the variance of the mean estimator would behave badly.

An example of the importance sampling is shown here with the distribution $\mathcal{F}$ being the Fisher's z-distribution, which is the the distribution of a logarithm-transformed F-distribution (the distribution of the ratio of two independent χ^2-variables) variable. The pdf of Firher's z-distribution is already a bit complicated:

$$\mathrm{f}(x; n, m) = \frac{2\, n^{n/2}\, m^{m/2}}{\mathrm{B}(n/2, m/2)}\frac{e^{nx}}{(ne^{2x} + m)^{(n+m)/2}}, \tag{9.3}$$

where B is the Beta function. The inverse cdf needed for the general method for creating random numbers involves the inverse of the regularized Beta function, which is not a closed-form function. Even the expected value of the distribution involves special functions.

To approximate the mean and pdf of $\mathcal{F}$, we use the Cauchy distribution as the instrumental distribution $\mathcal{G}$. The inverse of the Cauchy cdf with parameters a, b is $\mathrm{F}^{-1}(u; a, b) = a + b\tan(\pi(u - 1/2))$, so it is easy to simulate. Furthermore, the tails of the Cauchy distribution are heavy, so we can trust that $\mathrm{f}(x_i)/\mathrm{g}(x_i)$ is limited to a finite value.

The target pdf and the instrumental pdf are shown in Fig. 9.1 for Fisher's z with $n = 2, m = 10$, and for Cauchy with $a = 0, b = 1/2$. In the same figure, the sample mean $\frac{1}{n}\sum x_i \frac{\mathrm{f}(x_i)}{\mathrm{g}(x_i)}$ with Cauchy instrumental distribution is computed for five chains of simulations with n from 10 to 10,000.

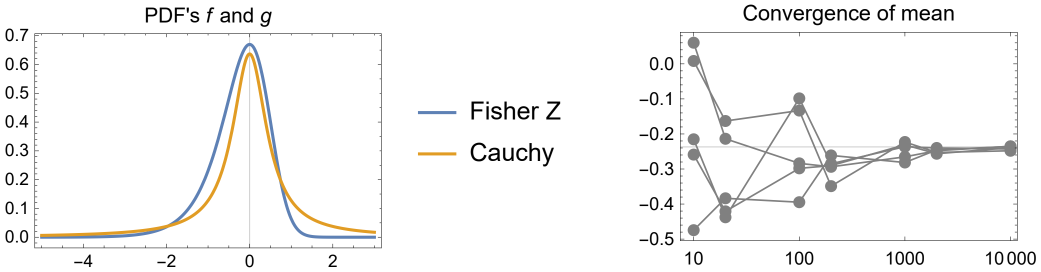

Figure 9.1: In the left, the target distribution Fisher z with $n = 2, m = 10$ with the instrumental distribution Cauchy with $a = 0, b = 1/2$. In the right, the convergence of the mean of the Fisher z-distribution estimated with the importance sampling algorithm. The correct mean value (-0.237) is marked with the horizontal gray line.

By choosing different functions h, quite many different properties can be estimated with importance sampling. For example, probabilities such as $\mathrm{P}(X < c)$ (so, cfd values) can be estimated by choosing $\mathrm{h}(x) = \mathbb{I}_{x<c}$ (indicator function).

The usability of importance sampling is, however, a bit hindered by the fact that the full f needs to be known, up to any normalizing constant. And, the instrumental distribution g should be chosen so that the tails are heavier compared to f, which is not always a straightforward task.

9.3 Simulated annealing

The importance sampling can be though as Monte Carlo integration, but the simulated annealing is a method for Monte Carlo optimization. The algorithm is interesting for MCMC, since the Metropolis-Hastings MCMC algorithm was developed from the simulated annealing algorithm.

Simulated annealing optimization fits well for large-dimensional global optimization problems, and does not require any information about the derivatives of the function to be optimized. The algorithm is the following:

- With the target function h to be minimized, choose x_0
- At round $i, i = 1, \ldots$

    1. generate x from symmetric instrumental distribution $\mathcal{G}(x_{i-1})$

2. Take

$$x_i = \begin{cases} x & \text{with probability } \min\left(\exp(\Delta h_i/T_i), 1\right) \\ x_{i-1} & \text{otherwise} \end{cases} \tag{9.4}$$

- Repeat

The symmetric instrumental distribution $\mathcal{G}(x_{i-1})$ will give us a random movement from the previous chain value x_{i-1}. The symmetry means that both the directions x and $-x$ are equally probable, e.g., $\mathrm{g}(x; x_{i-1}) = \mathrm{g}(-x; x_{i-1})$. A simple choice is the uniform distribution around the previous value, $[x_{i-1} - c, x_{i-1} + c]$. The width of the instrumental distribution must be large enough for the chain to easily move around the realistic search area, but not too large so that it will not 'shoot off' too often, which would lead to inefficiency of the algorithm.

The function $T_i := T(i)$ is the 'temperature' of the system. It must be decreasing as the simulation goes forward. It has been shown that a choice of $T(i) = k/\log(1+i)$ with sufficiently large k will guarantee good properties for the algorithm.

Finally, as the 'time' i goes forward and the 'temperature' T_i is decreasing, the probability of 'bad' moves x_i away from the minimum of h will be more and more difficult, because the $\Delta h_i = \mathrm{h}(x_{i-1}) - \mathrm{h}(x)$ are scaled with T_i, so the probability limit $\exp(\Delta h_i/T_i) \to 0$.

The global optimization is achieved by the ability of the algorithm to make 'bad' moves from time to time, making sure it can escape local minima and converge into global minimum. However, one can try to run the algorithm a few times with different starting points x_0 to make sure that the global minimum is really found.

In the example shown in Fig. 9.2 there is a quite nasty function

$$\mathrm{h}(x,y) = (y\sin(20x) + x\sin(20y))^2\cosh(x\sin(10x)) + \cosh(y\cos(20y))(x\cos(10y) - y\sin(10x))^2 \tag{9.5}$$

with several local minima is optimized with the simulated annealing algorithm. A step size of $c = 0.1$ for the instrumental uniform distribution is chosen, and the temperature is lowered as $T(i) = 10/\log(1+i)$. We can see how the chain of successive values in the optimization are mainly exploring the large plain in the middle, and also the narrow valleys.

9.4 The Metropolis-Hastings algorithm for MCMC

The MCMC method is quite generally defined in Robert & Casella (1999): "A Markov chain Monte Carlo method for the simulation of a distribution $\mathcal{F}$ is any method producing an ergodic Markov chain $(X^{(t)})$ whose stationary distribution is $\mathcal{F}$." Before moving to the most popular MCMC algorithm, the Metropolis-Hastings (M-H), we will shortly introduce what is Markov chain so that we can understand a bit the definition above.

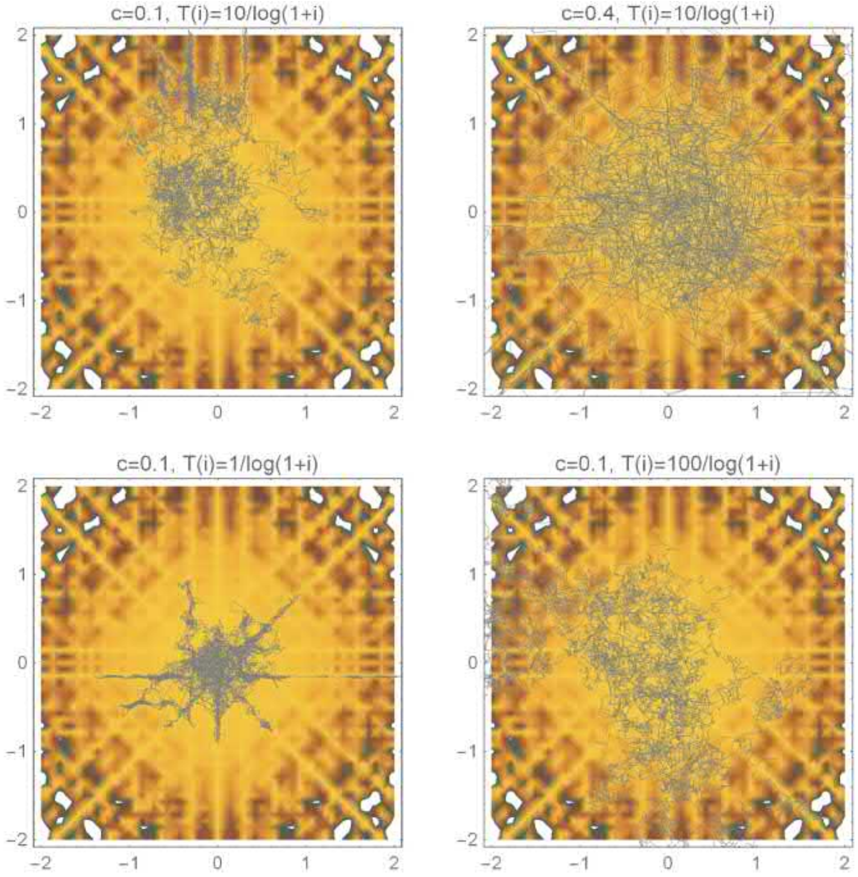

Figure 9.2: A complex function and the simulated annealing minimization chain with four different step size c and temperature decrease functions $T(i)$.

9.4.1 Markov chain

The mathematics behind Markov chains are quite involved, so to our purposes they just work, like magic. But, for them to work, they need to have some properties such as ergodicy, so let us try to list the required properties. First, Markov chains are, in MCMC, discrete random processes $(X^{(t)}), t = 1, \ldots$. The markovian property of the chain is that the next value of the chain, $(X^{(t+1)})$ depends on the previous values only by the current value $(X^{(t)})$.

The chain is often associated with a *transition kernel* K where $\mathrm{K}(x, y)$ marks the probability density to move from x to y in the chain. In another words, the kernel is the conditional probability distribution $\mathcal{G}$ so that $\mathrm{K}(x, y) = \mathrm{g}(y|x)$.

The ergodicy of the chain means that is will converge to a stationary distribution $\mathcal{F}$. It can be proven that this requires the chain to be Harris recurrent, i.e., the expected number of visits of the chain to a arbitrary set A is infinite. In practice, this means that every possible value of $\mathcal{F}$ is accessible by the chain, regardless of the starting value of the chain.

If we can construct a (Markov) chain such that the limiting stationary distribution of $(X^{(t)})$ is $\mathcal{F}$, then we can estimate for any function h

$$\int \mathrm{h}(x)\mathrm{f}(x)dx = \mathrm{E_f}[\mathrm{h}(x)] \approx \frac{1}{T}\sum_{1}^{T}\mathrm{h}(x^{(t)}). \tag{9.6}$$

Note that the $'x'$ here will usually be the unknown parameter (vector) $\boldsymbol{\theta}$ of our probability model or, in Bayesian analysis, of our posterior density $\mathrm{f}(\boldsymbol{\theta}|\boldsymbol{y}) \propto \mathrm{f}(\boldsymbol{\theta})\mathrm{L}(\boldsymbol{\theta};\boldsymbol{y})$. From now on, we will use the symbol θ, not x.

9.4.2 General Metropolis-Hastings algorithm

To complete the magic in the previous section we need a way to construct a Markov chain that has the required properties and, most of all, has the ability to have an arbitrary distribution $\mathcal{F}$ as the limiting distribution. Such a magic can be done with the Metropolis-Hastings (M-H) algorithm. The general version of that is the following:

```
Algorithm General Metropolis-Hastings
  • Choose $\theta^{(0)}$
  • At round $t, t = 1, \ldots, T$
      1. Generate $\theta$ from distribution $\mathcal{G}_{\theta|\theta^{(t-1)}}$
      2. Take
         $\theta^{(t)} = \begin{cases} \theta & \text{with probability } \min\left(\frac{\mathrm{f}(\theta)}{\mathrm{f}(\theta^{(t-1)})}\frac{\mathrm{g}(\theta^{(t-1)}|\theta)}{\mathrm{g}(\theta|\theta^{(t-1)})}, 1\right) \\ \theta^{(t-1)} & \text{otherwise} \end{cases}$
  • Repeat
```

The popularity of the M-H algorithm, especially with Bayesian analysis, lies now in the ratios of f's and g's, because any common normalizing factors not depending on the parameter θ can be canceled from $\mathrm{f}(x)$'s and from $\mathrm{g}(x|y)$'s. So, in the Bayesian framework, the pdf 'f' of the target distribution can be the proportional part of the posterior distribution, $\mathrm{f}(\theta)\mathrm{L}(\theta;\boldsymbol{y})$, and the transition kernel pdf g can be anything that produces an ergodic chain.

From this M-H algorithm description one can notice the relationship to the simulated annealing algorithm. In both the algorithms, 'bad' moves can be accepted with a positive probability. In M-H, this probability value is given by the ratio $\mathrm{f}(\theta)/\mathrm{f}(\theta^{(t-1)})$. If the proposed new value θ is 'more probable', then the ratio is above one and the accepted (well, the transition probability ratio $\mathrm{g}(\theta^{(t-1)}|\theta)/\mathrm{g}(\theta|\theta^{(t-1)})$ has to be taken also into account). But even if the ratio is below one and the proposed value is 'less probable', it can be accepted with a positive probability. The reason why the simulated annealing is not strictly a MCMC algorithm is in the decreasing temperature of the system, which makes the chain non-homogeneous, which is needed for the ergodicy.

9.4.3 Independent Metropolis-Hastings algorithm

The MCMC in general form is presented above. It seems quite straightforward with the only ambiguous part being the choice of distribution $\mathcal{G}_{x|y}$. One particular choice, leading to the so-called *independent Metropolis-Hastings* is to have $\mathcal{G}$ so that it is not conditional, i.e., $\mathcal{G}_x$.

Algorithm *Independent Metropolis-Hastings*

- As general M-H, but select $\mathrm{g}(x|y) = \mathrm{g}(x)$

This algorithm is in particular quite similar to the importance sampling (see Sec. 9.2). The proposal (i.e., instrumental) distribution is the same, but the weights of the accepted samples are a different. In independent M-H, there are no weights in such, but some chain values are repeated (giving them larger weights) if the proposed chain movement is not accepted.

The similar considerations for the proposal distribution $\mathcal{G}$ hold with independent M-H — the g should be able to visit all the values from the support of f, and preferably with a reasonable probability (compare with tail weights f/g in importance sampling).

As an example, we repeat the sampling from Fisher's z-distribution with $n = 2$, $m = 10$ as in Sec. 9.2, and use the Cauchy distribution with $a = 0$, $b = 1/2$ as the proposal distribution. One important benefit of MCMC is, that now the chain $(\theta^{(t)})$ should have the distribution $\mathcal{F}$ without any weights. This means that we can not only estimate any function $\mathrm{h}(\theta)$ with $\theta \sim \mathcal{F}$, but also the distribution $\mathcal{F}$ itself.

To speed up the computation, we can clean the $\mathrm{f}(x)$ for the Fisher z-distribution from anything not depending on x, so we can use $\mathrm{f}(\theta; n, m) \propto \exp(n\theta)\ (m{+}\exp(2\theta)\ n)^{\frac{1}{2}(-n-m)}$.

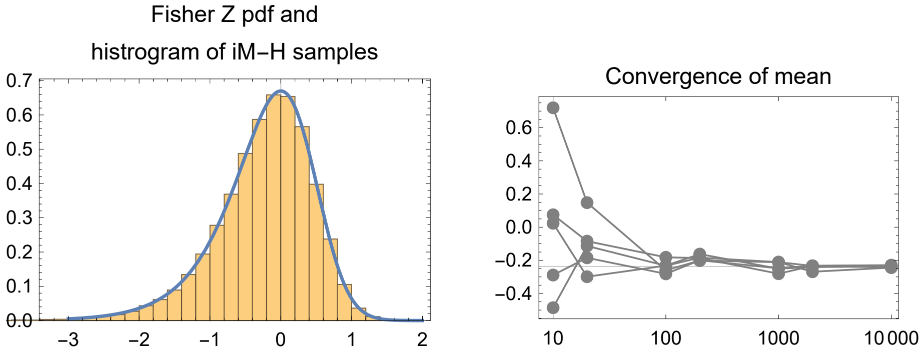

Figure 9.3: In the left, the target distribution Fisher z with $n = 2, m = 10$ and a histogram of one 100,000 sample independent M-H chain using the Cauchy$(0, 1/2)$ as the proposal distribution. In the right, the convergence of the mean of the Fisher z-distribution estimated with the independent M-H. The correct mean value (-0.237) is marked with the horizontal gray line.

9.4.4 Random walk Metropolis-Hastings

Probably the most used version of the M-H algorithms is the random walk Metropolis-Hastings. This is because the construction of the algorithm does not require detailed knowledge of the target distribution $\mathcal{F}$. In fact, one basically needs only to know the support of $\mathcal{F}$, and the proportional form $\mathrm{f}(x) \propto \mathrm{f}^*(x)$.

Algorithm *Random walk Metropolis-Hastings*

- As general M-H, but with symmetric random-walk proposal distribution g:

$$\mathrm{g}(\theta|\theta^{(t-1)}) = \mathrm{g}(\theta - \theta^{(t-1)}) = \mathrm{g}(\theta^{(t-1)} - \theta) = \mathrm{g}(\theta^{(t-1)}|\theta)$$

- The acceptance probability simplifies to

$$\min\left(\frac{\mathrm{f}(\theta)}{\mathrm{f}(\theta^{(t-1)})}\frac{\mathrm{g}(\theta^{(t-1)}|\theta)}{\mathrm{g}(\theta|\theta^{(t-1)})}, 1\right) = \min\left(\frac{\mathrm{f}(\theta)}{\mathrm{f}(\theta^{(t-1)})}, 1\right)$$

In practice, the symmetric g can be implemented as a random movement from the previous chain value, i.e., random walk. With a small random number (or vector of numbers) ξ, the random-walk g can such that

$$\theta = \theta^{(t-1)} + \xi. \tag{9.7}$$

The random number ξ can be drawn from, e.g., uniform distribution over $[\theta^{(t-1)} - c, \theta^{(t-1)} + c]$, or Gaussian distribution $\mathcal{N}(\theta^{(t-1)}, \sigma^2)$.

Let us go through the use of random walk M-H in a Bayesian framework. Let us model the photon count in a detector with an exponential distribution $\mathcal{E}xp(\lambda)$. The parameter λ describes the number of events (i.e., photons arriving) on a unit time interval. Furthermore, we have some previous knowledge saying that the distribution of λ could be modeled with the log-normal distribution $\mathcal{LN}(\alpha, \beta)$ with $\alpha = 1.5$ and $\beta = 0.75$.

The posterior distribution $f(\lambda|\alpha, \beta) \propto \mathrm{f}_p(\lambda; \alpha, \beta)\mathrm{L}(\lambda; \boldsymbol{x})$, where f_p is the prior distribution (log-normal), and L is the likelihood function of exponential-distributed data vector $\boldsymbol{x}$. Now the posterior can be solved analytically, but it is not in the form of any 'common' distribution. In some other case, the posterior could not be solved in closed form. Also, the estimates such as the posterior mean cannot be solved analytically.

With data $\boldsymbol{x} = (0.254, 0.360, 0.0372, 0.340, 0.252, 0.105, 0.111, 0.222, 0.162, 0.0307)$ the prior distribution and the likelihood-function (correctly scaled to a proper pdf) are shown in Fig. 9.4. Now, using the random-walk M-H algorithm we can create a chain of values $(\lambda^{(t)})$ that should have the correct posterior distribution. One example chain is shown in Fig. 9.5.

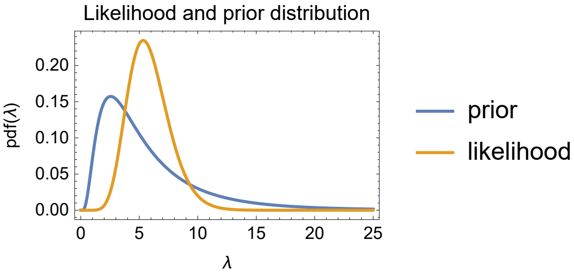

Figure 9.4: The likelihood function for the exponential model, and the a priori pdf with log-normal distribution for the parameter λ.

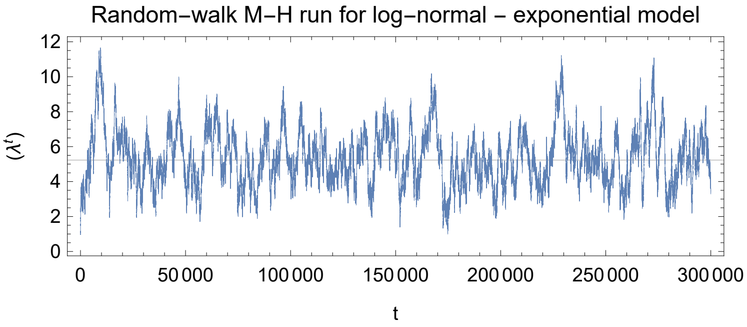

Figure 9.5: The random-walk M-H chain for the parameter λ with 300,000 samples. The correct mean of 5.23 is shown in the figure with the gray horizontal line.

9.4.5 MCMC diagnostics

The M-H algorithm with suitable proposal distribution should converge to the target distribution $\mathcal{F}$ if you have infinite time. However, with finite (computer) time, you should somehow make sure that your results have already converged. There is no definite proof for that, but some steps of checks that you should at least make.

First of all, you should try different starting values for the chain, i.e., run several chains. There is a so-called *burn-in period* with the random chains when the chain start from an initial point and finds its way towards the target distribution. This burn-in period should be discarded from the data when doing analysis based on the chain values. In Fig. 9.6 there are three chains with different starting points. One can see that the chains approach to same distribution only after some, say 3,000, steps.

The mixing of values is one interesting property to be followed with the convergence. It means that the chain should visit different values and ranges often enough. This also means that the *acceptance ratio*, the ratio of accepted movements to all the proposed movements, should not be too low (chain stuck) or too high (chain still converging). In Fig. 9.7 one can see three chains with different random walk step sizes c. The acceptance ratios are 99 % ($c = 0.1$), 97 % ($c = 0.25$), and 87 % ($c = 1$). The smallest step size causes too slow mixing and too high acceptance ratio, while

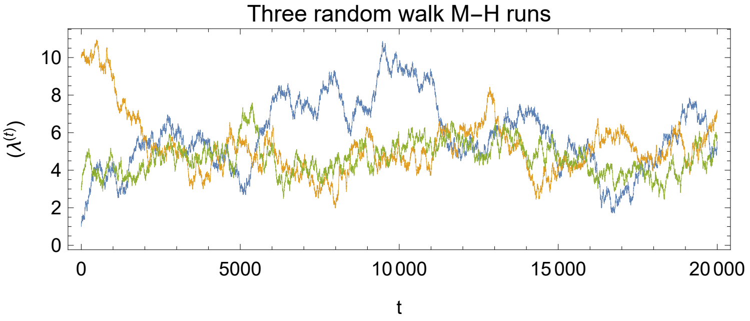

Figure 9.6: Three chains with different starting points for random walk M-H.

the largest step size is inefficient with too small acceptance ratio.

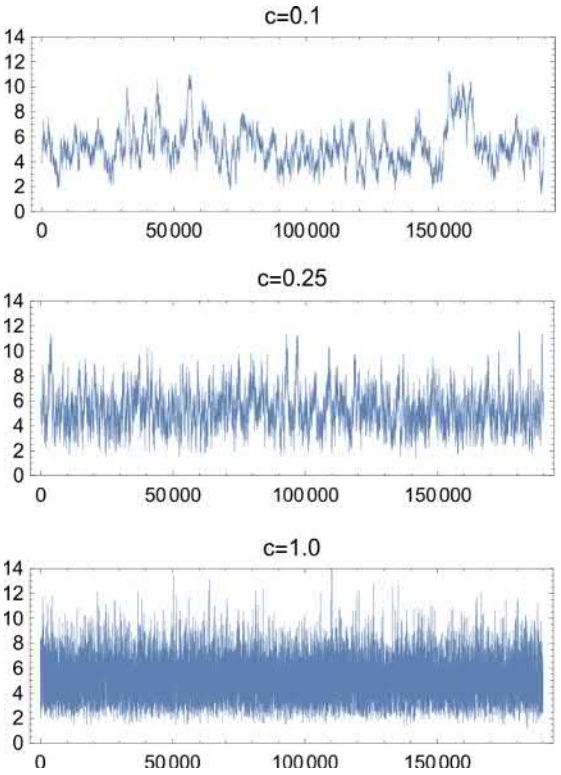

Figure 9.7: Three different step sizes for random walk M-H.

Other properties to follow include the chain autocorrelation, which should not be too long. The autocorrelation for distance ρ_t can be computed as

$$\rho_t = \frac{(\sum_{i=1}^{T-t}(\theta^{(i)} - \overline{\theta})(\theta^{(i+t)} - \overline{\theta})/(T-t)}{(\sum_{i=1}^{T}(\theta^{(i)} - \overline{\theta})^2)/T} \tag{9.8}$$

and should approach 0 when the distance grows. The length of the chain should be remarkably larger than the length where its autocorrelation approaches 0. For the three chains (see above), the autocorrelation length are shown in Fig. 9.8. For step size $c = 0.1$ the autocorrelation is above 0 for up to 8,000 samples, so the autocorrelation length is quite large. The chain with $c = 1.0$ has a very short autocorrelation length, but also small acceptance ratio. The chain with $c = 0.25$ has quite reasonable autocorrelation length of ∼1,000 samples and a large acceptance ratio.

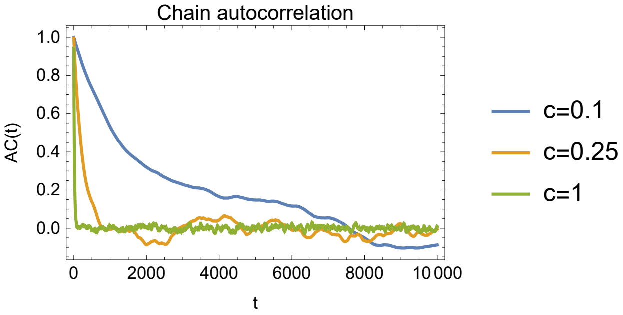

Figure 9.8: Chain autocorrelation with three different step sizes for random walk M-H.

9.4.6 Metropolis-Hastings with regression models

In regression models we have the systematic part of the regression model, let us call that $\mathrm{r}(\boldsymbol{x};\boldsymbol{\beta})$ to distinguish from target distribution f. In linear regression the function r is the linear matrix equation $\mathbf{X}\boldsymbol{\beta}$, and in nonlinear regression it is any function r.

The random part for regression comes with the residuals $\boldsymbol{\epsilon}$. If we assume these residuals to follow a normal distribution, our probability model is then

$$\boldsymbol{Y} \sim \mathcal{N}_n(\mathrm{r}(\boldsymbol{x};\boldsymbol{\beta}), \boldsymbol{\Sigma}), \tag{9.9}$$

where $\boldsymbol{\Sigma}$ is the (unknown) covariance matrix, usually assumed to be $\sigma^2\mathbf{I}$. The likelihood function for the model parameters $\boldsymbol{\beta}$ and σ^2 is now

$$\mathrm{L}(\boldsymbol{\beta},\sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}\sum_i^n (y_i - \mathrm{r}(\boldsymbol{x}_i;\boldsymbol{\beta}))^2\right) \tag{9.10}$$

The prior distributions for $\boldsymbol{\beta}$ and σ^2 can be usually thought to be independent of each other, so $\mathrm{f}_p(\boldsymbol{\beta},\sigma^2) = \mathrm{f}_p(\boldsymbol{\beta})\mathrm{f}_p(\sigma^2)$. With this information we can write the posterior distribution as

$$\mathrm{f}(\boldsymbol{\beta},\sigma^2|\boldsymbol{y}) \propto \mathrm{f}_p(\boldsymbol{\beta})\mathrm{f}_p(\sigma^2)\mathrm{L}(\boldsymbol{\beta},\sigma^2). \tag{9.11}$$

In practice, the M-H update round for the vector $(\boldsymbol{\beta},\sigma^2)$ can be done *component-wise*, updating only one coordinate at time. This increases the overall acceptance ratio in problems with many coefficients β_i.

Algorithm for *component-wise* regression random walk Metropolis-Hastings:

- Choose $\boldsymbol{\beta}^{(0)}$ and $(\sigma^2)^{(0)}$
- At round $t, t = 1, \ldots, T$

  1. Initialize vector $\boldsymbol{\beta} = \boldsymbol{\beta}^{(t-1)}$
  2. Generate σ^2 from proposal distribution
  3. Take

     $$(\sigma^2)^{(t)} = \begin{cases} \sigma^2 & \text{with probability } \min\left(\frac{\mathrm{f}_p(\sigma^2)\,\mathrm{L}(\boldsymbol{\beta},\sigma^2)}{\mathrm{f}_p((\sigma^2)^{(t-1)})\,\mathrm{L}(\boldsymbol{\beta},(\sigma^2)^{(t-1)})}, 1\right) \\ (\sigma^2)^{(t-1)} & \text{otherwise} \end{cases}$$

  4. For $i = 1, \ldots, k$
     (a) Copy current $\boldsymbol{\beta}$ to $\boldsymbol{\beta}'$. Generate component β_i' from proposal distribution
     (b) Take

     $$\beta_i = \begin{cases} \beta_i' & \text{with probability } \min\left(\frac{\mathrm{f}_p(\boldsymbol{\beta}')\,\mathrm{L}(\boldsymbol{\beta}',(\sigma^2)^{(t)})}{\mathrm{f}_p(\boldsymbol{\beta})\,\mathrm{L}(\boldsymbol{\beta},(\sigma^2)^{(t)})}, 1\right) \\ \beta_i & \text{otherwise} \end{cases}$$

  5. Repeat
  6. Update $\boldsymbol{\beta}^{(t)} = \boldsymbol{\beta}$

- Repeat

You may note that the ratio of the likelihoods in step 4.b can be written, by using notation $\mathrm{S}(\boldsymbol{\beta}) = \sum_i^n (y_i - \mathrm{r}(\boldsymbol{x}_i;\boldsymbol{\beta}))^2$, as $\exp\left(-\frac{1}{2\sigma^2}\left(\mathrm{S}(\boldsymbol{\beta}') - \mathrm{S}(\boldsymbol{\beta})\right)\right)$. With uninformative prior distribution for $\boldsymbol{\beta}$, the algorithm uses only this for testing new chain values for β_i.

Advanced MCMC

There is a vast collection of different small improvements to the random walk M-H algorithm for cases where the convergence is poor using the basic form of the algorithm. *Adaptive MCMC* uses multidimensional normal distribution as the proposal distribution. The adaptation is achieved by estimating the covariance matrix of the proposal distribution from the previous values of the chain.

Other small tweaks to the proposal distribution include using a population of possible parameter values, and computing ’typical’ movements somehow from them.

9.5 Gibbs sampling for MCMC

Gibbs sampling (GS) can be treated as a special case of M-H algorithm. GS is suitable for MCMC simulation of multidimensional parameter vector $\boldsymbol{\theta}$ in a stepwise manner, similar to component-wise M-H. The distinct features of GS are, that the proposed values are always accepted, and that the full conditional distributions of $\mathcal{F}$ needs to be known. By full conditional distribution we mean the pdf's

$$\theta_i|\theta_1,\ldots,\theta_{i-1},\theta_{i+1},\ldots,\theta_p \sim \mathrm{f}(\theta_i|\theta_1,\ldots,\theta_{i-1},\theta_{i+1},\ldots,\theta_p) = \mathrm{f}(\theta_i|\boldsymbol{\theta}\backslash\theta_i). \tag{9.12}$$

If the full conditionals are known, the *Gibbs sampling* algorithm is:

```
• Choose multivariate $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)},\ldots,\theta_p^{(0)})$
• At round $t, t = 1,\ldots,T$
    – Update $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$
    – Loop over $i = 1,\ldots,p$
        * Generate $\theta_i^{(t)}$ from $\mathrm{f}(\theta_i|\boldsymbol{\theta}^{(t)}\backslash\theta)$
    – Repeat
• Repeat
```

Gibbs sampling is quite suitable for Bayesian multivariate regression problems if suitable prior distributions are selected for the parameters. For example, by selecting (multivariate) normal prior to $\boldsymbol{\theta}$ the conditional posterior distributions are (1D) normal distributions. Another typical application are the so-called *hierarchical models*.

9.5.1 BUGS software

BUGS (Bayesian interface Using Gibbs Sampling) and its versions WinBUGS and OpenBugs are software packages for Bayesian analysis with Gibbs sampling. The BUGS software is quite unique in its user interface and internal logic, but once one learns it, it can be a powerful tool for Bayesian model analysis.

It is somewhat hard to explain the software here in text. Instead, we will take a quick look at its usage in the lecture.