Slice Sampling

Denny Dittmar
ddittmar@rbg.informatik.tu-darmstadt.de

Abstract

In many applications of artificial intelligence it is of interest to be able to generate samples from probability distributions. Since it is in most cases not possible to sample from a target distribution directly, especially if it has a very complex structure, special techniques are required that allow sampling from such distributions. One of these sampling techniques are called are Markov chain Monte Carlo(MCMC) methods that form one of the most important tools for such sampling purposes. The goal of this paper is to present one of these MCMC methods called slice sampling [1] and evaluate it in comparison to a modified version of it called elliptical Slice sampling [4] and the more popular Metropolis-Hastings method.

1 Introduction

In artificial intelligence it is often required is to evaluate the expecation of a function f that is given by

$$E[f] = \int f(x)p(x)dz. \tag{1}$$

In general the evaluation of such an integral is intractable, but one can make use of a Monte Carlo method approach and approximate the expectation value by a sum of samples, e.i.

$$E[f] \approx \sum f(x^i), \tag{2}$$

where x^i is are samples independently drawn from the distribution $p(x)$. The accuracy of such an approximation increases with the number of samples and for a sufficiently large number of samples from $p(x)$ this approximation can be seen as valid representation of the expectation value.

So the problem of approximating expecations reduces to be able to draw samples from distribution. In general this is a very challenging task, but there exist several samples techniques, which can be used to face this problem. Such techniques include popular ones like rejection sampling, importance sampling and Markov Chain Monte Carlo. Both rejection sampling and importance Sampling make use of a proposal distribution, from which one can directly draw samples. The disadvantage of these 2 approaches is that their performance depends highly on similarity of both the target and proposal distribution. The more these distributions differ, the less the effectivity of these samples techniques will be. In general this difference will increase with dimensionality of the distribution one wants to sample from.
Another approach of sampling is represented by so called Markov Chain Monte Carlo methods. These methods can achieve a good performance even for complex distributions in high-dimensional spaces. The general idea behind them is to drop the constraint that the samples have to be independent and drawing from a sequence of correlated samples x. More exactly each new sample x^i is drawn from a transition distribution $T(x^{i-1}, x^i) = p(x^i \mid x^{i-1})$, which is conditioned only on the last sample. It follows, that a new sample x^i is independent on all other samples given only the last sample, e.i.

$$p(x^i \mid x^1, \ldots, x^{i-1}) = p(x^i \mid x^{i-1}). \tag{3}$$

So only the last sample must be kept in order to draw the next sample.
This transitions have to be chosen such that the desired target distribution $p(x)$ is invariant for the Markov Chain, e.i.

$$p(x) = \sum_{x'} T(x', x)p(x'). \tag{4}$$

To make sure that the desired target distribution $p(x)$ one wants to sample from is invariant under this Markov chain, one can check, if it satisfies sufficient conditions that imply invariance. An often used condition to check, whether a distribution $p(x)$ is invariant is called detailed balance or reversibility and defined by

$$p(x)T(x, x') = p(x')T(x', x). \tag{5}$$

The fact that this implies invariance for $p(x)$ can be shown as follows:

$$\sum_{x'} p(x')T(x', x) = \sum_{x'} p(x)T(x, x') = p(x) \sum_{x'} T(x, x') = p(x). \tag{6}$$

If the transitions are chosen such that the Markov chain is reversible for $p(x)$, then $p(x)$ is invariant under this Markov chain. For invariance this condition is suffient, but not necessary.
Depending on the initial distribution and the transition probabilities a Markov chain needs different numbers of steps (burn-in) in order to converge to the invariant distribution, if there is a convergence. A necessary condition, that the convergence to a desired distribution is independent of the distribution of the initial state is that each state can be reached from each initial state in finite steps. This property is called ergodicity.
However, the fact that the Markov Chain converges to the desired distribution does not make any statement about how fast this convergence is.

2 Markov Chain Monte Carlo Algorithms

There exist several Markov chain Monte Carlo algorithms. In this chapter 3 of them are introduced. The first tow of them are the Metropolis-Hastings and Gibbs sampling algorithms. The third one is the slice sampling algorithm, which is a newer one, and a modified version of it called Elliptical Slice Sampling.

2.1 Metropolis-Hastings

To draw samples from a distribution $p(x)$ in Metropolis-Hastings a proposal distribution, which depends on the current state x^i of the Markov chain, is used to draw a new sample x' from it and accept the sample as the next state x^{i+1} with the probabilty

$$A(x', x^i) = min(1, \frac{\hat{p}(x')q(x^i \mid x')}{\hat{p}(x^i)q(x' \mid x^i)}) = min(1, \frac{p(x')q(x^i \mid x')}{p(x^i)q(x' \mid x^i)}), \tag{7}$$

where $\hat{p}(x)$ can be unnormalized, that is $\frac{\hat{p}(x)}{Z_{\hat{p}}} = p(x)$ for some unknown normalization constant $Z_p = \int \hat{p}(x)dx$.
If this condition is fullfilled the sample x' is accepted and forms the next state x^{i+1}, otherwise the old state is kept, that is $x^{i+1} = x^i$.
The fact, that the stationary distribution of such a Markov Chain is the desired one, can be shown as follows:

$$\begin{aligned} & p(x)q(x' \mid x)A(x' \mid x) \\ = & \, min(p(x)q(x' \mid x), p(x')q(x \mid x')) \\ = & \, min(p(x')q(x \mid x'), p(x)q(x' \mid x)) \\ = & \, p(x')q(x \mid x')A(x \mid x') \end{aligned} \tag{8}$$

If the proposal distribution is symmetric, that is

$$q(x^i \mid x') = q(x' \mid x^i), \tag{9}$$

which is often the case, this equation simplifies to

$$A(x', x^i) = min(1, \frac{p(x')}{p(x^i)}) \tag{10}$$

and the algorithm results in a more specific version called the Metropolis algorithm.
Like in rejection sampling and importance sampling a good choice of the proposal distribution is crucial in order to achieve a good performance of the algorithm. Such proposal distribution often comes with step-size parameters to adjust the size of the steps. Such step-size parameters have to be chosen carefully, since a smaller steps results in a slow random walk behaviour and a larger steps in a high rejection rate.
For Metropolis Hastings a common choice of this proposal distribution is a Gaussian distribution with mean at the current state. In this case the step-size parameter is given by the variance of this Gaussian distribution.

Algorithm 1 - Metropolis-Hastings:
Input : x^i, q ▷ last state and proposal distribution
Output : x^{i+1} ▷ next state
$x' \sim q(x' \mid x)$ ▷ draw a sample from the proposal given the current state
$A(x', x^i) = min(1, \frac{\hat{p}(x')q(x^i|x')}{\hat{p}(x^i)q(x'|x^i)})$ ▷ calculate the acceptance probability
$u \sim U[0, 1]$
if $u \leq A$ then ▷ accept x' with probability given by A
$x^{i+1} = x'$
else
$x^{i+1} = x^i$
fi

2.2 Gibbs Sampling

Gibbs sampling is another Markov chain Monte Carlo algorithm, but can also be seen as an instance of the Metropolis-Hastings algorithm. It is very simple and can be used for many applications, in which sampling from multivariate distributions is necessary. The core of the algorithm is to resample a new state at each step only for one variable x_j for a given multivariate target distribution $p(x_1, \ldots, x_n)$ and fix the current states of all other variables, that is $x_j^{'} \sim p(x_j \mid x_{k \neq j})$. Doing this for all variables at each sampling step results into Gibbs sampling given in **Algorithm 2**. A difficulty of Gibbs sampling is that it may be hard to find good conditional proposal distributions with respect to the mixing times. For ergodicity it is sufficient to make sure that all conditional distributions have a probability greater than zero for each state, such that each state becomes reachable from each state in a finite number of steps. **Algorithm 3** shows the procedure. It can be shown, that Gibbs sampling satisfies detailed balance and thus leaves the target joint distribution invariant.

Algorithm 2 - Gibbs Sampling:
Input : x^i ▷ last state
Output : x^{i+1} ▷ next state
for $j := 1$ to dim do ▷ update each variable given the current state of the other variables
$x_j^{i+1} \sim p(x_j \mid x_{k<j}^{i+1}, x_{k>j}^i)$
od

2.3 Slice Sampling

A disadvantage of the Metropolis-Hastings algorithm is the dependence of finding a good proposal distribution that involves setting step-size parameters of such a distribution to appropriate values. Such a parameter tuning requires to make a good trade-off between too low step-sizes leading to a random walk and too high step-sizes leading to high rejection rates. Slice sampling is an MCMC technique that tries to overcome this problem by adjusting the step-size more automatically.
Slice sampling makes use of the fact that drawing a sample from a distribution $p(x)$ is the same as uniformly sampling from the points underneath the curve of such a distribution, that is all points

(x, u), for which $0 \leq u \leq p(x)$ holds and then dropping the u value. The condition $0 \leq u \leq p(x)$ can be weakend to $0 \leq u \leq \hat{p}(x)$, where $\hat{p}(x)$ is unnormalized and $\frac{\hat{p}(x)}{Z_{\hat{p}}} = p(x)$ holds for some most likely unknown normalization constant $Z_{\hat{p}} = \int \hat{p}(x)dx$.
In order to obtain such a uniformly distributed sample (x, u) from this area in slice sampling the distibution $p(x)$ is augmented with a new auxiliary variable u and then samples are drawn from the joint distribution $p(x, u)$ given by

$$p(x, u) = \begin{cases} \frac{1}{Z_{\hat{p}}}, & \text{if } 0 \leq u \leq \hat{p}(x) \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

After a sample has been drawn from $p(x, u)$ one can obtain a sample from the marginal $p(x)$ by just dropping the u value. The validity of this statement can be shown as follows:

$$\int p(x, u)dx = \int_0^{\hat{p}(x)} \frac{1}{Z_{\hat{p}}} du = \frac{\hat{p}(x)}{Z_{\hat{p}}} = p(x) \tag{12}$$

In order to sample from $p(x, u)$ Gibbs sampling can be applied by alternatly sample from x and u given the value of the other variable. Drawing a new sample u from $p(u \mid x)$ can easily be done by uniformly sampling from $u \sim U[0, \hat{p}(x)]$. In order to draw a new sample x from $p(x \mid u)$ one has to sample uniformly from the points given by $\{x : u < \hat{p}(x)\}$, which define a 'slice'.
Uniformly sampling from such a slice can be very difficult in practice. If the distribution $p(x)$ is unimodal then similar to adaptive rejection sampling this can be done efficiently by first enclosing the slice in an interval such that the boundaries of the interval are outside the slice and then drawing uniformly samples from this interval until a sample is found for which $u < \hat{p}(x)$ holds. This can be done exponentially fast. If a sample is rejected either the left or right boundary of the current interval is set to this state such that the interval shrinks but still contains the old state. Since all the points of the slice in the unimodal case must lie between 2 points that are not contained in the slice this method is valid, that is it samples uniformly from the slice.
If $p(x)$ is not unimodal as it is mostly the case, more generally one can use a transition operator that leaves the uniform distribution on the slice invariant. One possibilitiy for such a transition was proposed by [1] and works similar to the described method for the unimodal case but still lets the uniform distribution under the slice invariant. As in the unimodal case it tries to find an interval in the univariate case, which encloses the current state and has its boundaries outside the slice. Therefore a step-size parameter is used in order to successively enlarge this interval by a width given by the step-size parameter, until the boundaries are outside the slice. Thus the transition operator depends additionally on the current state, now. The shrinking procedure is done like described for the unimodal case a thus very fast. In order to apply this method to the multivariate case the interval that encloses the slice can be generalized to a hyperractangle, where the expanding and shrinking processes are performed in each dimension.
The resulting slice sampling procedure using this transition and later being used in the experiments is given by **Algorithm 3**.

Algorithm 3 - Slice Sampling:
Input : x^i, w ▷ last state and step-size parameter
Output : x^{i+1} ▷ next state
$u \sim U[0, \hat{p}(x)]$ ▷ set the threshold for the next state
$u' \sim U[0, 1]$
$x_{min} = x^i - u'w$ ▷ initialize the upper and lower bounds of the interval to capture the slice
$x_{max} = x^i_{min} + w$
while $u < \hat{p}(x_{min})$ do
    for $i := 1$ to dim do
        $x_{min}(i) = x_{min}(i) - w(i)$;
    od
od
while $u < \hat{p}(x_{max})$ do
    for $i := 1$ to dim do
        $x_{max}(i) = x_{max}(i) + w(i)$;

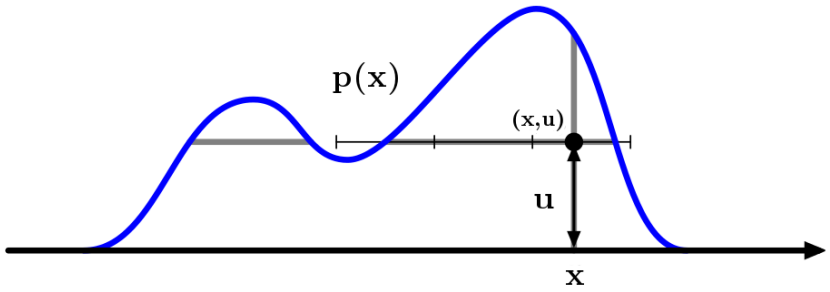

Figure 1: Sampling from $p(x)$ can be done by drawing samples (x, u) uniformly from the area underneath the curve and dropping u. In slice sampling this is done by alternately drawing a new sample u^{i+1} from $p(u^{i+1} \mid x^i)$ and then drawing x^{i+1} from $p(x^{i+1} \mid u^{i+1})$ in Gibbs sampling manner. (graphics taken from [3])

```
        od
od
repeat do
        for i := 1 to dim do
            u' ~ U[0,1]
            x'(i) = x_min(i) + u'(x_max(i) - x_min(i))     ▷ draw a new uniformly from the slice
        od
        if u < p̂(x') then                                  ▷ check for the threshold
                        x^{i+1} = x'
                        break                               ▷ return x' as next state if u < p̂(x')
        fi
        for i := 1 to dim do
            if x'(i) < x(i) then                            ▷ adjust either the lower or upper bound
                            x_min(i) = x'(i)
                        else
                            x_max(i) = x'(i)
            fi
        od
od
```

2.4 Elliptical Slice Sampling

Elliptical Slice Sampling [2] is a MCMC algorithm for sampling from multivariate distributions of the form $p(x) = \mathcal{N}(x;\ \mu, \Sigma)L(x)$, where $\mathcal{N}(x;\ \mu, \Sigma)$ denotes a Gaussian prior and L denotes a likelihood function.
The idea of the algorithm is to define a two-dimensional ellipse and then draw a new sample x' from it in a similar manner to slice sampling. Since such an elliptical slice captures much of the structural properties of a Gaussian prior, this method leads to a fast mixing.
The ellipse of the current step x is defined by

$$x'(\theta) = (x - \mu)cos\theta + (\nu - \mu)sin\theta + \mu \tag{13}$$

with $\nu \sim \mathcal{N}(x;\ \mu, \Sigma)$ as a Gaussian distributed random vector and $\theta \in [0, 2\pi]$. This defines an ellipse centred at μ and goes through the last state x and ν.
Similar to standard slice sampling the algorithm first draws a sample u uniformly from $[0, 1]$ and gets a new state from the ellipse by choosing θ such that $L(x'(\theta)) > uL(x)$ holds. As in slice sampling this can be done by repetitively choosing a θ uniformly from an interval, which always covers the last state, until this condition is fullfilled and otherwise shrinking the interval everytime $x'(\theta)$ was rejected. One convenient difference is that the initial interval $(\theta_{min}, \theta_{max})$ of θ always covers the whole elliptical slice. Thus there is no need for additional step-size parameters like an explorative width or procedures in order to define an initial sampling interval within this elliptical

slice.
One possibility to apply this algorithm to arbitrary distributions $p(x)$ the algorithm can be generalized [5] by setting $L(x)$ to

$$L(x) = \frac{p(x)}{N(x;\ \mu, \Sigma)}. \tag{14}$$

$L(x)$ now is not a likelihood distribution any further, since $\int L(x)dx$ can even be infinite in the case that $p(x)$ has a heavier tail than $N(x;\ \mu, \Sigma)$. The result of this is, that the Markov chain can easily get stuck away from μ, since $L(x)$ increases fast in this case. This problem was already reported by [5].
The resulting elliptical slice sampling procedure using this kind of generalization is given in **Algorithm 4**. It can be shown, that this method respects detailed balance for the desired distribution $p(x) = \mathcal{N}(x;\ \mu, \Sigma)L(x)$ and thus it is correct.

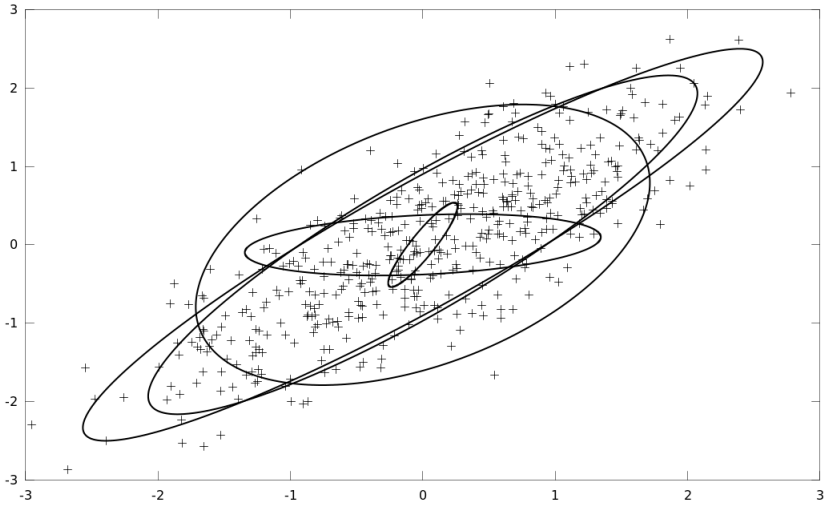

Figure 2: Several ellipses generated by the generalized version of elliptical slice sampling during sampling from a distribution represented by the points.

Algorithm 4 - Elliptical Slice Sampling:
Input : x^i, μ, Σ ▷ last state and parameters for sampling an ellipse
Output : x^{i+1} ▷ next state
$\nu \sim \mathcal{N}(x;\ \mu, \Sigma)$ ▷ fix the ellipse by choosing ν
$u \sim U[0, 1]$
$\log L(x) = \log(p(x)) - log(N(x;\ \mu, \Sigma))$ ▷ set $\log L(x)$ with $L(x) = \frac{p(x)}{N(x;\ \mu, \Sigma)}$ using log-domain
$\log y = \log L(x) + log(u)$ ▷ set next threshold with $y = L(x)u$ using log-domain
$\theta \sim [0, 2 * \pi]$ ▷ initialize θ
$(\theta_{min}, \theta_{max}) = (\theta - 2\pi, \theta)$ ▷ initialize the interval of θ
$x'(\theta) = (x - \mu)cos\theta + (\nu - \mu)sin\theta + \mu$ ▷ get a new state from the ellipse
repeat do
    if $y < L(x')$ then ▷ check threshold holds given by $L(x') = \frac{p(x')}{N(x';\ \mu, \Sigma)}$
        $x^{i+1} = x'$
        break ▷ return the next state
    fi
    if $\theta < 0$ then
        $\theta_{min} = theta$
    else
        $\theta_{max} = theta$

fi
$\theta \sim (\theta_{min}, \theta_{max})$
od

3 Experiments

In this section the 3 previously described MCMC alogrihms Metropolis-Hasting, slice sampling and the elliptical slice sampling version generalized as described are compared with respect to their convergence and mixing behaviour as well as their rejection respective function evaluations per iteration with increasing number of iterations, where number of iterations means the size of the chain at the current step, which starts at zero, that is no burnin-phase is dropped.
The experiments are splitted into 2 parts. In the first part the algorithms are compared on a simple 1-dimensional Gaussian mixture to get an initial guess of the their mixing behaviour. In the second part the algorithms are compared on randomized Gaussian mixtures with increasing dimensionality.

3.1 Comparing for 1-dimensional case

The target distribution used for comparison of the methods is a 1-dimensional Gaussian mixture distribution with 2 equally weighted components at 0 and 10 und unit variances.
The Metropolis-Hastings method was used with Gaussian proposal distribution with the mean as current state and different variances σ^2. This is shown in Figure 4. For low variances there results a slow random walk behaviour without exploring. Thus the algorithm gets stuck at the mode at 0 and moves slowly there with only small steps. The rejections rates are small at this point. With higher variances the Metropolis-Hasting method makes larger steps and is able to discover the other mode faster, but the rejection rate increases in this case.
Slice sampling was tested with different widths (Figure 5). Similar to Metropolis-Hastings with higher variances for larger widths it is more likely for this method to reach the other mode. But for slice sampling the mixing is much better as for Metropolis-Hastings. Once the slice sampling method is able to reach a region, it is able to mix fast between that region in contrast to Metropolis-Hastings. The disadvantage is, that slice sampling needs a high number of function evaluations in order to achieve this.
The generalized elliptical slice sampling uses $\mu = 0$ and different variances σ^2 and is shown in Figure 6. It can be seen there, that for a low variance it stays at one mode and gets sometimes stuck at a distant from the mean of that mode, because $L(x)$ increases fast, if going away from μ and thus many points near μ with much lower $L(x)$ become very unlikely to be taken then. This problem disappears as the variance increases, since $L(x)$ does increase much slower for larger distances from μ. This leads to a better mixing in the mode and in addition it is more likely now to reach the other mode, since larger ellipses are created. For very high variances it results in a good mixing with respect to both nodes. The number of function evaluations is lower than in the normal slice sampling algorithm.

3.2 Comparing for higher-dimensional case

The methods are compared here on multi-dimenional Gaussian mixture distributions consisting of 25 equally weighted components each. The chosen dimensions are $2, 4, 6$ and 8.
The means and variances of the Gaussian mixture were chosen randomly, but the means were limited to be in $[0, 10]$ in each dimension. Each variance was chosen to be the identity matrix multiplied with a random variance $\sigma^2 = exp(v)$ with $v \sim N(0, 1)$. Once these values have been chosen once they are fixed during the rest of the test.
For Metropolis-Hasting an isotropic Gaussian has been used as proposal distribution with different variances as step-sizes(0.1 and 100).
For slice sampling the intervals are expanded with the same widths in each dimension. This was tested for several widths(0.1 and 25). For the generalized version of elliptical slice sampling there was used $\mu_i = 5, \forall i$, that is μ was set to the middle of the space, where the components of the Gaussian mixture were set. Σ was the Identity matrix multiplied with some different variances σ^2(0.1 and 100).

The algorithms were compared on their convergence behaviour with increasing dimensions and for 2 different parameter settings. For this purpose 20 chains were generated by each method with randomly chosen starting points in $[0, 10]$, where the startpoints were chosen once and then used by every method. After that their potential scaling reduction factor(PSRF)[6] was checked in each iteration. PSRF indicates, how similar the disitributions are the chains come from by comparing the between and within variation of the chains. The quality of this value is given by the distance to 1, that is calues near to 1 are better. The value in the plots for each iteration is the average of the PSRF in each dimension. Furthermore, for Metropolis-Hastings the rejected states as well as the function iterations for slice sampling and the generalized elliptical slice sampling were tested. The length of the chains was 200 in the case of Metropolis-Hastings and 100 for the slice sampling methods.

In the plots it can be seen the that Metropolis-Hastings(Figures 6 and 7) has better convergence for the higher proposal variance, since it can move with larger steps. The number of rejections per ieration increases for both variances with the dimensions, but with lower proposal variance it is lowers. In the case of high variance the convergence is worse in higher dimensions.
Slice sampling(Figures 8 and 9) converges better for a larger width than for a lower width, since it can make larger steps like Metropolis-Hastings for higher variances. For a larger width the number of function evaluations per iteration is lower than for a smaller width. The convergence seems to be slightly better than in Metropolis-Hastings. In higher dimensions the convergence is worse. Interestingly for a lower width the number of function evaluations per iteration is higher in low-dimensional spaces.
The generalized elliptical slice sampling(Figures 10 and 11) converges significantly better for the higher variance. For the high variance it seems to have an very good convergence that outperforms the convergence of the other algorithms with respect to the given parameter settings. The number of function evaluations per iteration increases slightly higher in higher dimensions.

4 Summary/Conclusion

In this paper 3 different MCMC methods were introduced and compared. These 3 methods are slice sampling, a generalized version of elliptical slice sampling and Metropolis-Hastings. For all methods there was left a certain space how to use these methods. In slice sampling there had to be defined a transition operator and in Metropolis-Hastings a proposal had to be chosen. The elliptical slice sampling had to be generalized such that it could be applied to arbitrary distributions. All these methods were compared with respect to their mixing/convergence behaviour. For this purpose PSRF had been used as indicator for convergence. It turned out, that the convergence gets mostly significantly worse in higher dimensions for all 3 methods, except for Metropolis-Hastings with lower variance. The convergence was for all 3 methods better for a parameter settings that allowed larger exploration steps. In Metropolis-Hastings larger steps led to higher rejection rates. In slice sampling the function evaluations per iteration decreased for higher widths and for generalized elliptical slice sampling it stayed nearly constant. This shows, that increasing the parameters of these 2 leads to much less trouble than in Metropolis-Hastings an. Thus the slice sampling approaches are more robust for too high parameter values. The convergence of slice sampling seemed to be better than for Metropolis-Hastings and the convergence of the generalized elliptical slice sampling seemed to be better than for slice sampling given the best parameter settings used in the experiments and the the assumption of reliability of the used PSRF indicator for convergence and experiment settings. In conclusion the slice sampling methods performed better than Metropolis-Hastings and elliptical slice sampling performed better than slice sampling.

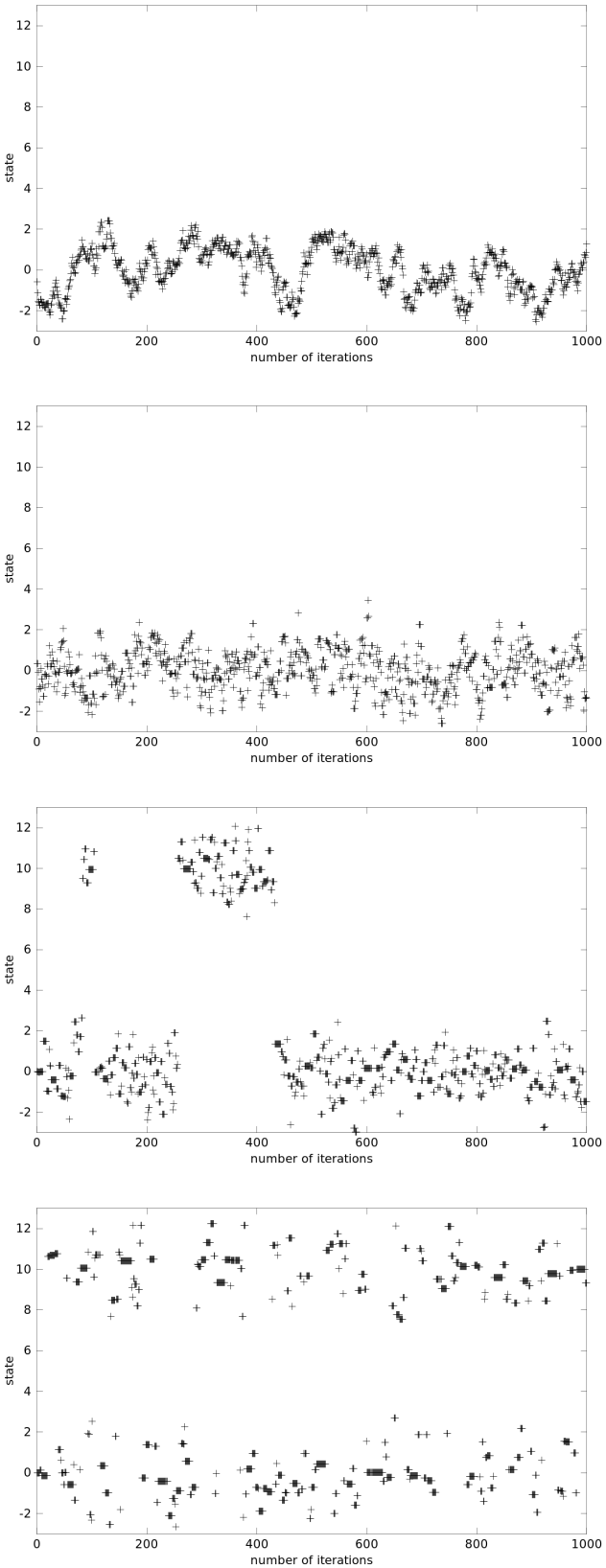

Figure 3: Sampling from a 1-dimensional Gaussian mixture using Hastings-Metroplis with proposal variances $\sigma^2 \in \{0.1, 1, 10, 100\}$(in this order) and starting point at 0. Rejections: 104, 301, 611, 815

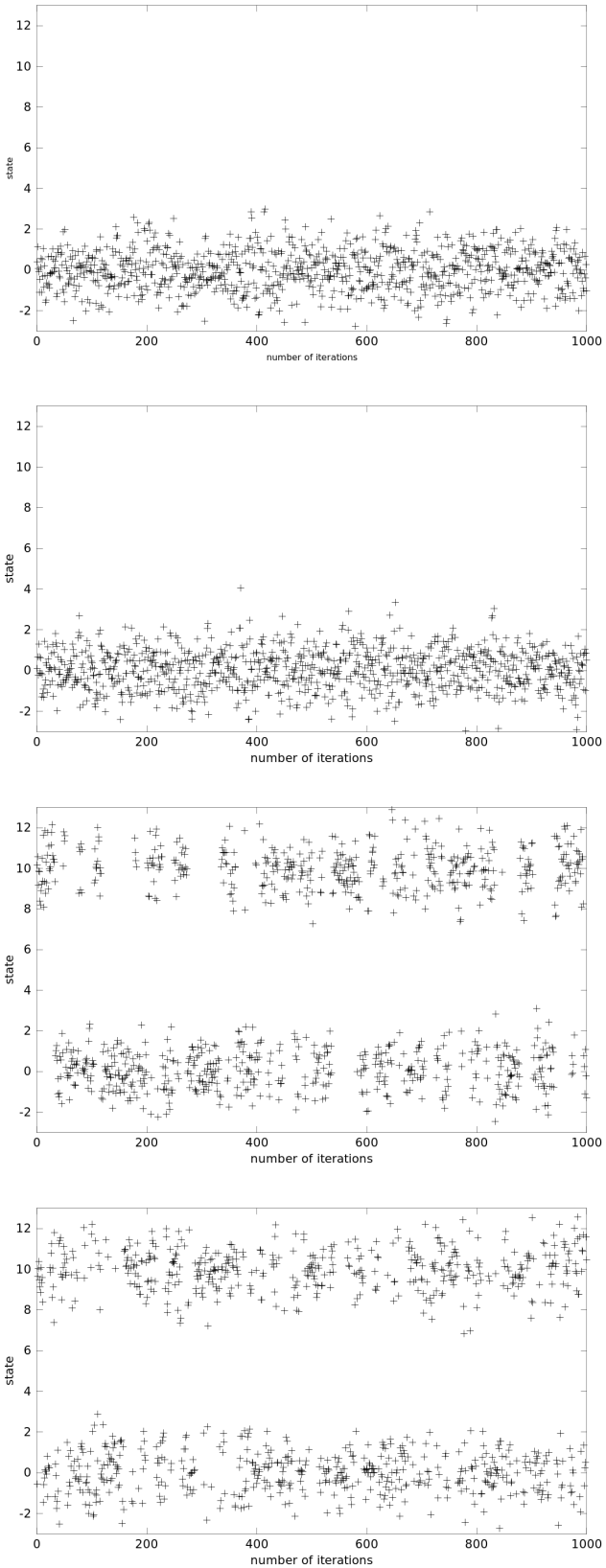

Figure 4: Sampling from a 1-dimensional Gaussian mixture using slice sampling with widths $w \in \{0.1, 1, 10, 25\}$(in this order) and starting point at 0. Function evaluations: 3490, 6493, 5487, 7796

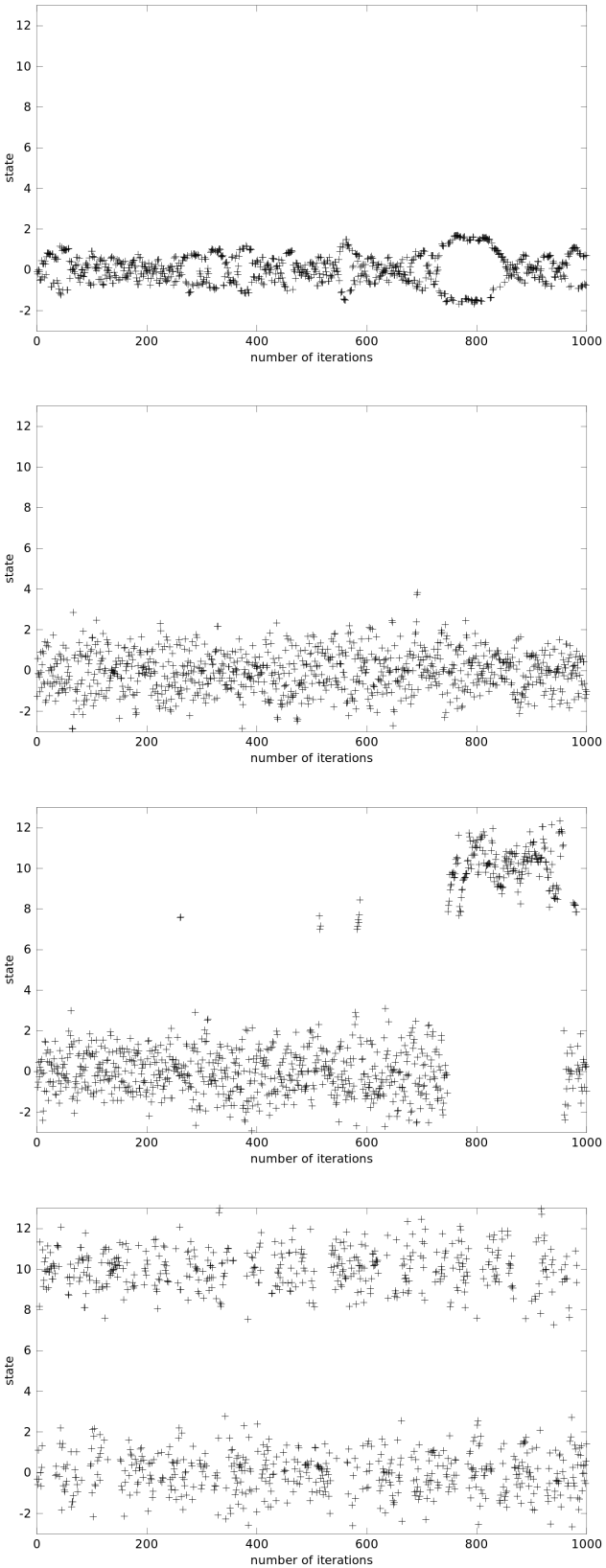

Figure 5: Sampling from a 1-dimensional Gaussian mixture using the generalized elliptical slice sampling with $\mu = 0$, $\Sigma = \sigma^2 I \in \{0.1, 1, 10, 100\}$(in this order) and starting point at 0. Function evaluations: 2050, 1000, 2970, 3874

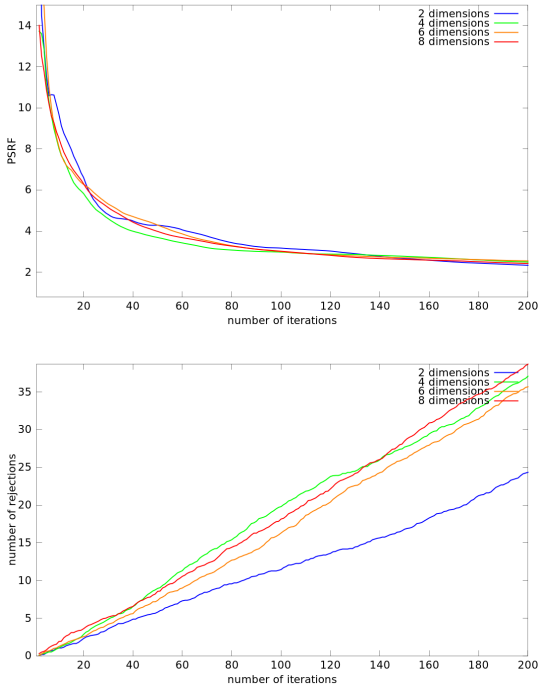

Figure 6: PSRF and rejections for Metropolis-Hastings with 0.1 proposal variance

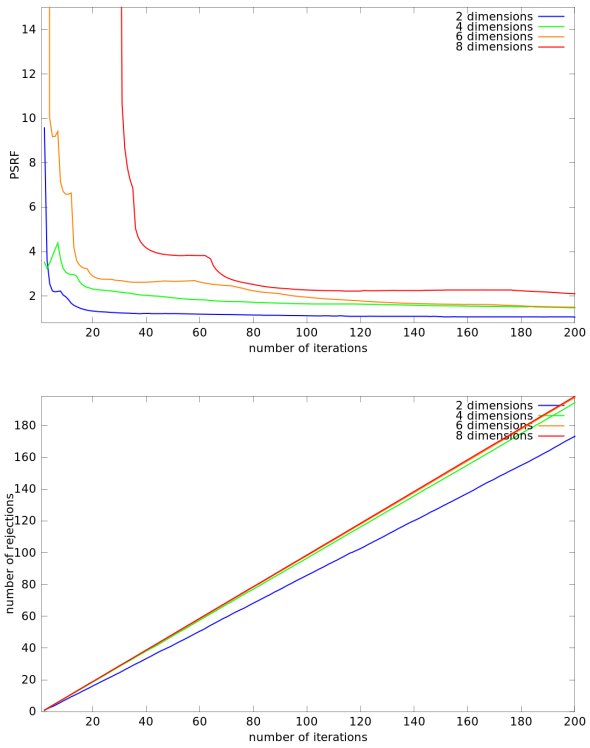

Figure 7: PSRF and rejections for Metropolis-Hastings with 100 proposal variance

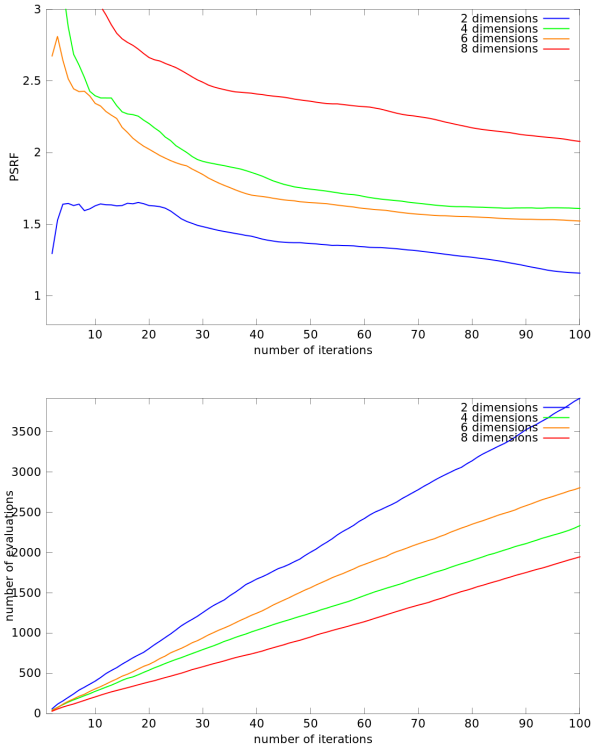

Figure 8: PSRF and function evaluations for slice sampling with 0.1 width

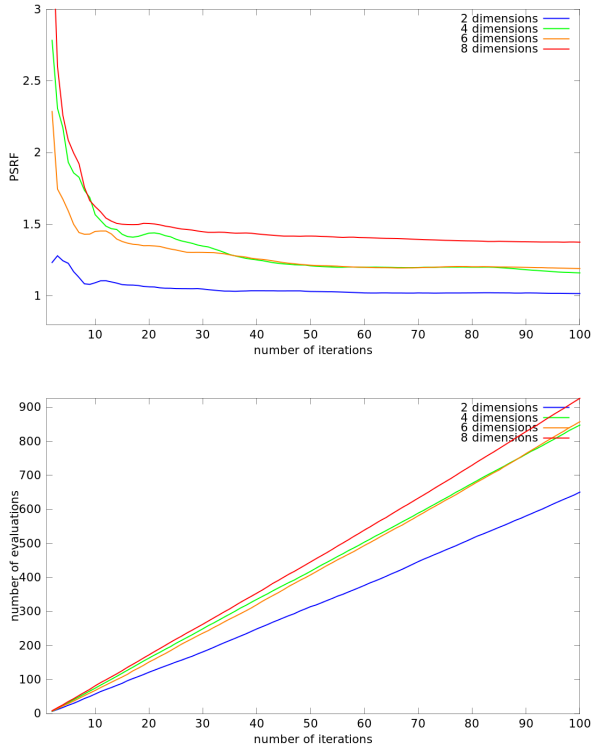

Figure 9: PSRF and function evaluations for slice sampling with 25 width

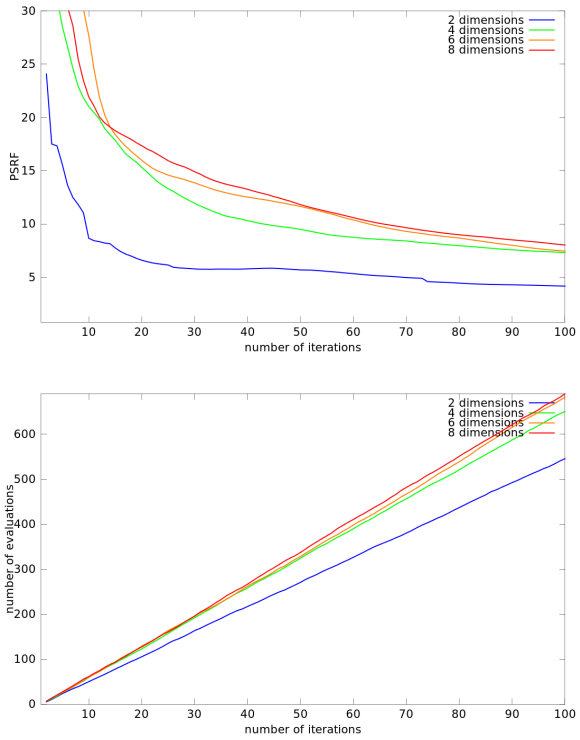

Figure 10: PSRF and function evaluations for generalized version of elliptical slice sampling with $\Sigma = 0.1I$

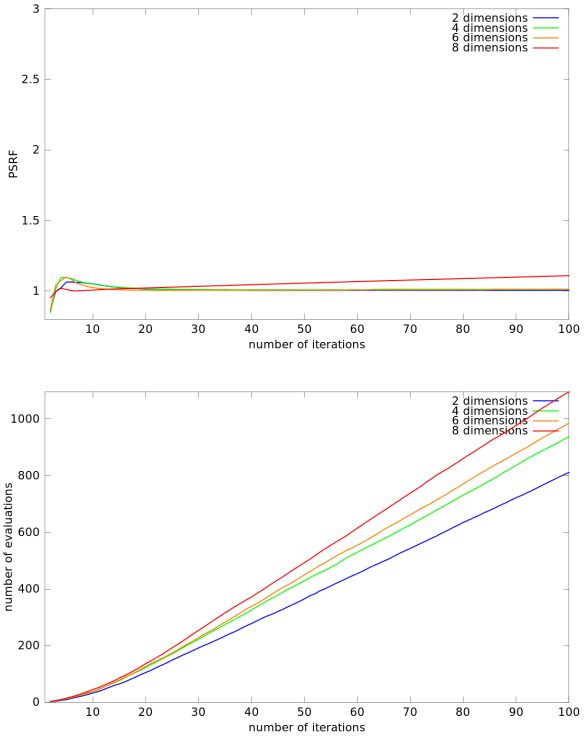

Figure 11: PSRF and function evaluations for generalized version of elliptical slice sampling with $\Sigma = 100I$

References

[1] R. M. Neal. Slice sampling. Annals of Statistics, 31(3):705767, 2003.

[2] C. M. Bishop. Pattern recognition and machine learning. Springer-Verlag, New York, 2006.

[3] Iain Murray, Advances in Markov chain Monte Carlo methods,M.A., M.Sci., Natural Sciences (Physics), University of Cambridge, UK (2002), Gatsby Computational Neuroscience Unit University College London

[4] Murray, R. P. Adams, and D. J. MacKay. Elliptical slice sampling. Journal of Machine Learning Research: W&CP, 9:541548, 2010.

[5] PARALLEL MCMC WITH GENERALIZED ELLIPTICAL SLICE SAMPLING, By Robert Nishihara, Iain Murray and Ryan P. Adams, Harvard University and University of Edinburgh, 2012

[6] Brooks, S.P. and Gelman, A. (1998) General methods for monitoring convergence of iterative simulations. Journal of Computational and Graphical Statistics. 7, 434-455. Note that this function returns square-root definiton of R (see Gelman et al (2003), Bayesian Data Analsyis p. 297).