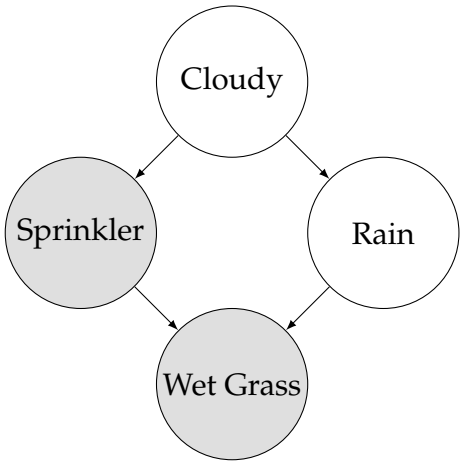
Sampling-based Inference 8

8.1 Introduction 79
8.2 Approximate sampling . . 80
8.2.1 Rejection sampling 80
8.2.2 Importance sampling . . . 81
8.3 Markov chain 82
8.3.1 Introduction 82
8.3.2 Detailed Balance 83
8.4 Markov Chain Monte Carlo 84
8.4.1 Metropolis Hastings . . . 84
8.4.2 Gibbs Sampling 85
8.4.3 Sampling based inference in PGM 87
8.4.4 Convergence Diagnostics 88
8.4.5 Effective sample size . . . 89
8.5 Hamiltonian Monte Carlo 91

8.1 Introduction

Our focus in this chapter is on solving inference problems by sampling strategies. This is needed because in the context of Bayesian Networks we are only able to perform inference if our network satisfies certain structural properties; in particular it has to be a tree or a poli-tree.

Even a simple example like



does not satisfy the property that makes our belief propagation algorithm work.

Therefore we need to change our strategy. One possibility is to rely on sampling to perform approximate inference of the probability distributions we are interested in.

The question then is how to sample from $p(x|y = \bar{y})$?

To do this we notice that we can generally compute the unnormalized probability distribution $\tilde{p}(x)$ such that

$$p(x) = \frac{1}{Z}\tilde{p}(x) \tag{8.1}$$

The typical scenario where this happens is given by Bayes Theorem

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{8.2}$$

where $p(y)$ is hard to compute (can be a complicated high dimensional integral)

Our sampling problem than boils down to generate samples $x_1, \ldots, x_n$ from $p(x)$ knowing $\tilde{p}(x)$, as independent as possible among them. Once

we have this set of samples we are also able to estimate quantities like

$$\mathbb{E}_x[f(x)] = \int f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{8.3}$$

We are going to see two main ways to perform this:

- Sample from $q(x) \approx p(x)$ and then correct. These are very common methods but they may suffer from efficiency issues.
- Markov Chain Monte Carlo (MCMC), which allows us to generate a set of samples from an unnormalized distribution.

Remark. Sampling is a computational intensive tasks and vanilla methods are not scalable to high dimensional probability distributions.

8.2 Approximate sampling

We are going to explore methods that allow us to sample from a surrogate $q(x) \approx p(x)$.

8.2.1 Rejection sampling

Suppose to have a distribution $p(x)$ and another distribution $g(x)$ from which it is easy to sample, called the **proposal distribution**, such that $\exists M > 0 : Mg(x) \geq p(x), \forall x$, which of course implies that $\frac{p(x)}{Mg(x)} \leq 1$

Rejection sampling consists in the following algorithm:

1. Sample $\hat{x}$ from $g(x)$
2. Accept $\hat{x}$ with probability $\frac{p(\hat{x})}{Mg(\hat{x})}$
3. If reject, then repeat the algorithm until acceptance

Graphically, once we samples $\hat{x}$, the acceptance mechanism corresponds to sample uniformly a number α between 0 and 1, multiply it by $Mg(x)$ and accept if and only if $\alpha Mg(x)$ falls below the original distribution at the point $\hat{x}$. We reject if the opposite happens.

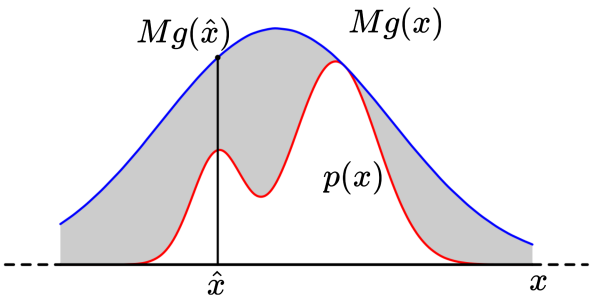


Figure 8.1: In the rejection sampling method, samples are drawn from a simple distribution $g(x)$ and rejected if they fall in the grey area between the distribution $p(x)$ and the scaled distribution $Mg(x)$. The resulting samples are distributed according to $p(x)$. (Bishop)

Remark. We can actually compute the expected number of samples from g to accept a single sample on p, and this expectation is in fact M. Therefore, if M is large, this method becomes inefficient. Rejection

sampling is especially inefficient in high dimension (you need a large M to "cover" p).

Remark. If we consider the unnormalized distribution $\tilde{p}(x)$ in place of $p(x)$ in the rejection sampling scheme, then we can use the fraction of rejected points to estimate the normalization constant Z. In fact,

$$Z = \int \tilde{p}(x)dx = M \int \frac{\tilde{p}(x)}{Mg(x)} g(x)dx = M \cdot p(\text{accept})$$

8.2.2 Importance sampling

The goal in importance sampling is to evaluate

$$\mathbb{E}_p[f(x)] = \int f(x)\frac{1}{Z}\tilde{p}(x)dx = \frac{\int f(x)\tilde{p}(x)dx}{\int \tilde{p}(x)dx} \tag{8.4}$$

where f is an integrable function.

We consider a proposal distribution $g(x)$ from which we know how to sample from, and we turn the expectation on p into an expectation on g.

$$\frac{\int f(x)\tilde{p}(x)dx}{\int \tilde{p}(x)} = \frac{\int f(x)\tilde{p}(x)\frac{g(x)}{g(x)}}{\int \tilde{p}(x)\frac{g(x)}{g(x)}} = \frac{\mathbb{E}_g[f\frac{\tilde{p}}{g}]}{\mathbb{E}_g[\frac{\tilde{p}}{g}]} \tag{8.5}$$

Then we sample $x_1, \dots, x_n$ from $g(x)$ and replace the expectation with sums

$$\frac{\mathbb{E}_g[f(x)\frac{p}{g}]}{\mathbb{E}_g[\frac{p}{g}]} = \frac{\frac{1}{N}\sum_{i=1}^N f(x_i)w(x_i)}{\frac{1}{N}\sum_{i=1}^N w(x_i)} \tag{8.6}$$

where

$$w(x_i) := \frac{\tilde{p}(x_i)}{g(x_i)} \tag{8.7}$$

are known as the **importance weights**.

If $\frac{f\tilde{p}}{g}$ is approximately constant, then estimates can be very good. If weights vary a lot instead, we have a large variance and consequently a poor estimate. So, the quality of the result depends from the choice of g(x).

Notice that $\frac{1}{N}\sum_{i=1}^N w(x_i) \approx Z$, so we also have an approximation of the partition function.

Remark. This technique can be used to estimate functions of rare events, increasing the probability that the event happens and then correcting it.

Example. If we want to compute $\mathbb{E}_p[f(x)]$ where $p(x) = \mathcal{N}(x; 0, 1)$ and $f(x) = x^{20}$, using samples from $p(x)$ would be inefficient, as the expected value is mostly influenced by extreme values of x. One can then use

importance sampling with a Gaussian with larger variance as proposal distribution.

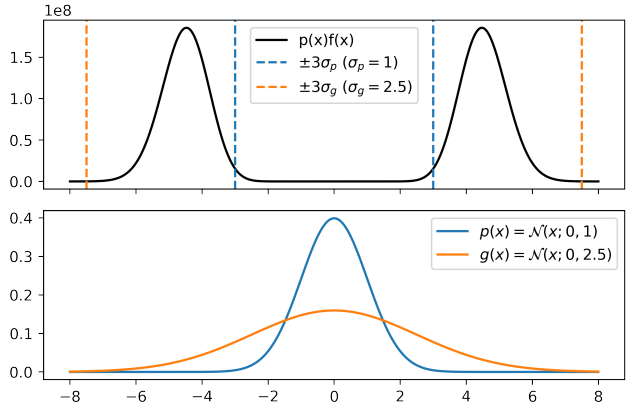


Figure 8.2: Importance sampling example. Sampling from $p(x) = \mathcal{N}(x; 0, 1)$ to compute $\mathbb{E}_p[f(x)]$ with $f(x) = x^{20}$ would be inefficient, as most of the contribution to the integral is given by values of x that are rare ($|x| > 3\sigma_p$). Using as proposal distribution a Gaussian with larger variance allows us to sample from the region that contributes to the expected value.

8.3 Markov chain

8.3.1 Introduction

A Markov Chain is a stochastic process, denoted as

$$\{X_t\}_{t \geq 0} \tag{8.8}$$

with $t \in \mathbb{N}$, $X_0, X_1, \ldots, X_t \in \mathcal{X}$ where $\mathcal{X}$ can be discrete or continuous.

It can be described as a *dynamical system*, i.e. a system in which we start from a certain state x_0 with a given probability $p_0(x)$ and that changes state according to a dynamic described by the *transition kernel* $p(x_n|x_{n-1})$.

Remark. Markov Chains satisfy the *memoryless property*, which corresponds to the assumptions encoded into the following probabilistic graphical model:

Therefore we have that

$$p(x_n|x_{n-1}, \ldots, x_0) = p(x_n|x_{n-1}) \tag{8.9}$$

which is known as the memoryless or Markov property.

We also require the **time homogeneity** property that states that

$$p(x_n|x_{n-1}) = p(x_1|x_0), \ \forall n \geq 1 \tag{8.10}$$

which means that the probability to jump from a certain state to another state stays the same for every time step.

Definition 8.3.1 *$p(x_n|x_{n-1})$ satisfying the time homogeneity property is called the (one step)* ***transition kernel****.*

For a discrete state space we would have a *transition matrix*, while for a continuous state space, $p(x_n|x_{n-1})$ is a probability density on x_n depending continuously on x_{n-1}.

Since we are interested at the behaviour of the Markov Chain as the index n progresses, i.e. for large times, we need to define few more concepts.

Definition 8.3.2 *A Markov Chain is* ***ergodic*** *iff* $\forall x, y \in \mathcal{X}, \exists t \geq 0 : p(x_t = y|x_0 = x) > 0$.

If a Markov Chain is ergodic, it means that there is always the possibility of going from a given state x to another given state y if we are patient enough. This means that the entire state space is reachable, no matter where we start exploring.

Definition 8.3.3 *A* ***stationary distribution*** $\Pi(y)$ *is such that*

1. $$\Pi(y) = \int p(y|x)\Pi(x)dx \tag{8.11}$$
   which means that Π is an ***invariant measure*** *with respect to the Markov Chain dynamics defined by the Transition Kernel $p(y|x)$.*
2. $p_n(x) = p(x_n|x_0) \xrightarrow[n\to\infty]{} \Pi(x)$
3. Π *is unique*

8.3.2 Detailed Balance

The following condition is sufficient to guarantee that $\Pi(x)$ is a stationary distribution.

Definition 8.3.4 *We say that a Markov Chain is* ***reversible*** *(or it satisfies the* ***balance condition****) iff $\exists \Pi(x)$, probability distribution such that*

$$p(x|y)\Pi(y) = p(y|x)\Pi(x) \tag{8.12}$$

Proposition 8.3.1 *If an ergodic Markov Chain is reversible with respect to the distribution $\Pi(x)$, then $\Pi(x)$ is a stationary distribution.*

Proof. If the Markov Chain is reversible then we can write

$$\int p(y|x)\Pi(x)dx = \int p(x|y)\Pi(y)dx = \Pi(y)\int p(x|y)dx = \Pi(y) \tag{8.13}$$

□

Remark It is not true that the existence of a stationary distribution implies that our Markov Chain is reversible.

We are now ready to tackle Markov Chain Monte Carlo: by requiring that the Markov Chain we are going to define satisfies the detailed balance condition for a given distribution $\Pi(x)$, we will be eventually ($t \to \infty$) able to sample from distribution of interest, i.e. $\Pi(x)$.

8.4 Markov Chain Monte Carlo

8.4.1 Metropolis Hastings

Intuitively, the idea is that we want to sample from a distribution and we build an ergodic Markov Chain with a transition kernel such that the stationary distribution coincides with the one we want to sample from.

Assume that we want to sample from $p(x) = \frac{1}{Z}\tilde{p}(x)$, where we know the unnormalized distribution $\tilde{p}(x)$ but the normalization constant is too complicated to compute. We fix $q(x|y)$, the *proposal kernel* of our Markov Chain, such that

1. It is easy to sample from $q(x|y)$
2. It makes our Markov Chain ergodic.

A typical choice for our proposal kernel is to use a Gaussian distribution, in which the variance is chosen in such a way that makes it quick to reach the stationary distribution.

Suppose that we start from a certain state x at the time step t.

The algorithm works as follows:

1. We sample y from $q(y|x)$
2. Borrowing from rejection sampling, we are going to accept/reject the new sampled point based on the following rule

$$x_{t+1} = \begin{cases} y, \text{ with probability } \alpha(y|x) = min\left\{1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right\} \\ x, \text{ otherwise} \end{cases} \tag{8.14}$$

The criterion for defining

$$\alpha(y|x) = min\left\{1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right\} \tag{8.15}$$

is called the **Metropolis-Hastings** criterion; for symmetric transition kernels $q(x|y) = q(y|x)$, it becomes the Metropolis criterion.

Observe that

$$\frac{\tilde{p}(y)}{\tilde{p}(x)} = \frac{p(y)}{p(x)} \tag{8.16}$$

and so the regions with higher probability $p(y) > p(x)$ are more likely to be visited.

Notice that we haven't defined in a full mathematical way the transition kernel of our Markov Chain here, but it can be derived by the operational procedure given above.

Lemma 8.4.1 *The Metropolis-Hastings acceptance criterion satisfies the detailed balance condition.*

Proof. Let's start from the full transition kernel in an implicit form and suppose that $y \neq x$. Then we need to prove that

$$p(y|x)p(x) = p(x|y)p(y) \tag{8.17}$$

If $y \neq x$, the first term reduces to the product of the probability to sample y given x, which is given by $q(y|x)$, times the probability to accept y, which is given by $\alpha(y|x)$. Therefore

$$p(y|x)p(x) = q(y|x)\alpha(y|x)p(x) = \min\left\{1, \frac{p(y)q(x|y)}{p(x)q(y|x)}\right\} \cdot q(y|x)p(x) \tag{8.18}$$

Notice that

$$\frac{p(y)}{p(x)} = \frac{\tilde{p}(y)}{\tilde{p}(x)} \tag{8.19}$$

Performing some calculations we obtain

$$\min\left\{1, \frac{p(y)q(x|y)}{p(x)q(y|x)}\right\} \cdot q(y|x)p(x) = \min\{q(y|x)p(x), p(y)q(x|y)\} = \tag{8.20}$$

$$\min\left\{\frac{p(x)q(y|x)}{p(y)q(x|y)}, 1\right\} \cdot q(x|y)p(y) = \alpha(x|y)q(x|y)p(y) = p(x|y)p(y) \tag{8.21}$$

□

There might still be issues: if we pick a bad q the algorithm may take a lot of steps before reaching the stationary distribution. We say that the **mixing time** is high. This time can be estimated by some statistics on the chain that is being sampled, that form a sort of *diagnostics tools*. However, there is not a way to determine the exact moment in which the steady state is reached. Moreover, another issue is that the samples x_n, x_{n+1} are not independent.

Another good property that we may want to have in our proposal kernel is to have a good balance in between exploration and exploitation, which becomes especially important in multimodal distribution.

8.4.2 Gibbs Sampling

This time, let's write the probability distribution from which we want to sample from as $p(x) = p(x_1, \ldots, x_n)$ and suppose that we know how to sample from the 1-dimensional conditionals, i.e. from $p(x_i|x_{-i})$ where $x_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$.

The main loop of Gibbs sampling algorithm is the following:

1. Pick $k \in \{1, \dots, n\}$
2. Set $x_j^{(t+1)} = x_j^{(t)}$ for $j \neq k$ (the j-th component of the t+1 state)
3. Sample $x_k^{(t+1)}$ from $\tilde{p}(x_k | x_{-k}^{(t)})$

Therefore we are choosing a coordinate and sample along that coordinate, keeping everything else fixed. We have different ways to choose k:

1. Round-Robin strategy, i.e. sample starting from 1, go up to k on the first k steps and then repeat.
2. Choose uniformly at random.

Lemma 8.4.2 *The transition kernel of Gibbs Sampling is exactly the same as Metropolis-Hastings algorithm*

Proof. In fact, for Gibbs Sampling we have

$$q_k(y|x) = \begin{cases} p(y_k|x_{-k}), & \text{when } y_{-k} = x_{-k} \\ 0 & \text{otherwise} \end{cases} \tag{8.22}$$

Which means that $\alpha_k(y|x) = 1$ because

$$\alpha_k(y|x) = \frac{p(y)q(x|y)}{p(x)q(y|x)} = \frac{p(y_k|y_{-k})p(y_{-k})}{p(x_k|x_{-k})p(x_{-k})} \cdot \frac{p(x_k|y_{-k})}{p(y_k|x_{-k})} \tag{8.23}$$

Where the equality is given by the standard conditional probability expansion. Also notice that $x_{-k} = y_{-k}$, otherwise there would be no jump according to the definition of our proposal kernel. Then

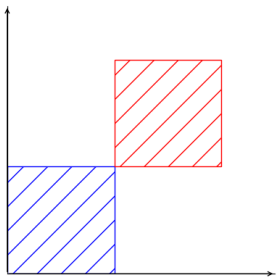
$$\frac{p(y_k|y_{-k})p(y_{-k})}{p(x_k|x_{-k})p(x_{-k})} \cdot \frac{p(x_k|y_{-k})}{p(y_k|x_{-k})} = 1 \tag{8.24}$$

□

Gibbs Sampling algorithm can be generalized, for example, by sampling blocks instead of a single variable, i.e. we can sample $x_j \dots, x_k \subseteq x$. Moreover, we could apply this even if we don't know explicitly how to sample from $p(x_i|x_{-i})$, for example by applying rejection sampling, or even a Metropolis-Hastings MCMC to sample our conditional distribution (the latter strategy is called "Metropolis within Gibbs").

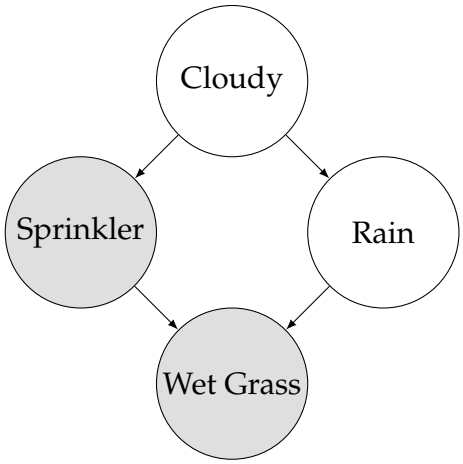
Remark.

- We may not satisfy ergodicity in Gibbs sampling, since it may not always be possible to find a path between two states which is only made of "single component" steps.
  Example: Consider the bivariate distribution $p(x, y) = 2$ if $(x \in [0, 0.5] \wedge y \in [0, 0.5]) \vee (x \in [0.5, 1] \wedge y \in [0.5, 1), 0$ otherwise. In this case, we cannot sample from an "island" different from the one where we start sampling.



- If variables are strongly correlated, then our mixing time can be very high.

8.4.3 Sampling based inference in PGM



Suppose that we observe the variables "Sprinkler" and "Wet grass", but we don't observe "Cloudy" and "Rain".

More generally, suppose to have a set of variables (could be vector of random variables), x and y that are described by a PGM and we know that y is observed, $y = \hat{y}$. We want to make inference on x, which typically means computing $p(x|y = \hat{y})$. We aim at sampling from this probability, but we do not have access to such a conditional distribution. We only know $\tilde{p}(x) = p(x, y = \hat{y})$, but not the normalization constant $Z = p(y = \hat{y})$.

Here are some strategies then to generate samples from this distribution:

- Rejection sampling: sample from the full joint $p(x, y)$ which can be done by ancestral sampling (which is fast) and then reject if $y \neq \hat{y}$. If we observe a lot of variables then this becomes very inefficient.
- A better strategy is to use MCMC. We can sample from $p(x, y = \hat{y})$ using some proposal distribution (which will depend on the variables that we are trying to sample). Although the state of the art of MCMC (Hamiltonian Monte Carlo, explained later on) is generally good, in the framework of Bayesian Networks we can do better.

- If we can compute efficiently the one dimensional conditional distribution, i.e. $p(x_i|x_{-i}, y) = p(x_i|MB_i)$, where MB_i is the Markov Blanket of x_i, we can use *Gibbs sampling*. This is especially efficient with discrete, and relatively small state spaces.

8.4.4 Convergence Diagnostics

How can we check that our Markov Chain has reached the steady state? We will put forward a set of tools that can monitor one or more trajectories and roughly tell us whether we reached it or not. Notice that since we are interested in sampling the stationary probability distributions, we shall start keeping the samples *only* when we actually reached the steady state.

Let's define some notation for the rest of the section. We are going to denote a general function over a state of the MCMC trajectory $(X_t)_{t\geq 0}$ as $\Psi : \mathcal{X} \to \mathbb{R}$. This function can be a lot of different things, depending on what we are interested in computing, e.g. a projection on single coordinate.

We will assume that Ψ has values in $\mathbb{R}$, and not just in a subset of it, and, if it does, we transform the function Ψ to make it compliant to this assumption (taking the logarithms of quantities in between $(0, \infty)$ for example). Let's fix some further notation:

- $x_1, \ldots, x_n$ is our sampled trajectory
- $\Psi_j := \Psi(x_j)$
- $\bar{\Psi} := \frac{1}{N} \sum_j \Psi_j$ is the estimate of $\mathbb{E}[\Psi] = \int \Psi(x)p(x)dx$

On a high level, the idea is to look at more than one chain and compare the distribution of the samples that we obtain and see if they look more or less the same.

Practically,

1. we sample $\frac{m}{2} \geq 1$ trajectories from overdispersed initial points. We try to start from different states that are far away in our state space
2. Sample for $4n$ steps
3. we throw away the first half of every trajectory so that we have only $2n$ points left. This phase is known as the **burn-in** or **warm-up** phase. We do this because it takes time to reach the steady state (notice this is an heuristic: convergence to the stationary distribution can happen faster or slower than $2n$ steps).
4. Then we split in 2 parts the remain trajectories, so that we are left with m different trajectories each of length n.

From now on we will denote each sample as x_{ij} where $i \in [1, n]$ and $j \in [1, m]$, where this notation describes the i_{th} sample of the j_{th} trajectory. We will also denote $\Psi(x_{ij}) = \Psi_{ij}$ and define

$$\begin{cases} \bar{\Psi}_j := \frac{1}{n} \sum_{i=1}^{n} \Psi_{ij} \\ \bar{\Psi} := \frac{1}{m} \sum_{j=1}^{m} \bar{\Psi}_j \end{cases} \tag{8.25}$$

Hence, $\bar{\Psi}_j$ is the average within the trajectory j and $\bar{\Psi}$ the average over all the trajectories, respectively. We are also interested in the variance of $\bar{\Psi}$, but since our samples are not independent, we don't have that

$\text{VAR}[\bar{\Psi}] = \frac{1}{n}\text{VAR}[\Psi]$, i.e. the variance of the estimator in this case is not just the variance of our random variable divided by the number of samples. If you think about two consecutive points in the chain, there is a high chance that the correlation between them is higher than that of two points which are sampled distantly in time from one another.

Let's define these two quantities:

$$W \coloneqq \frac{1}{m}\sum_{j=1}^{m} s_j^2, \qquad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\Psi_{ij} - \bar{\Psi}_j)^2 \tag{8.26}$$

$$B \coloneqq \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\Psi}_j - \bar{\Psi})^2 \tag{8.27}$$

W is called the **within variance** while B is called the **between variance**.

We know by definition that

$$W \leq \text{VAR}[\Psi],$$

because when sampling single trajectories we have not necessarily explored and visited the full space. Increasing the number of samples we will converge to the true variance. Moreover, as long as the initial states are overdispersed, it can be shown that

$$\text{VAR}[\Psi] \leq \text{VAR}^+[\Psi] \coloneqq \frac{n-1}{m}W + \frac{1}{n}B$$

Then we have both a lower and an upper bound for our variance, both converging to the true variance. Therefore we can compute the statistics

$$\hat{R} \coloneqq \sqrt{\frac{\text{VAR}^+[\Psi]}{W}} \tag{8.28}$$

which can be monitor while running the MCMC simulations. Notice that $\hat{R} > 1$ and that $\hat{R} \xrightarrow[n\to\infty]{} 1$. Heuristically, we can say that when $\hat{R} \leq 1.1$ we have converged to our stationary distribution.

8.4.5 Effective sample size

We may want to have some measure of efficiency of our statistics, i.e. we may want to compute $\text{VAR}[\bar{\Psi}]$. Notice the following: if $m \cdot n$ samples are independent then we know that

$$\text{VAR}[\bar{\Psi}] = \frac{\text{VAR}[\Psi]}{n \cdot m} \tag{8.29}$$

but unfortunately this is not the case as we have already seen.

The correlations among nearby points (when positive as typically the case) are increasing the variance of the estimator $\bar{\Psi}$, in a way which can

be expressed by this formula:

$$nm\text{VAR}[\bar{\Psi}] \approx \left(1 + 2\sum_{k=1}^{\infty} \rho_k\right) \text{VAR}[\Psi] \tag{8.30}$$

Where ρ_k is the **autocorrelation** of lag k, which is, by definition

$$\rho_k \coloneqq \text{Corr}[\Psi(x_i), \Psi(x_{i+k})] \tag{8.31}$$

So it is the correlation in between two points in the same chain which are k steps apart.

If we compare the formula above (8.31) with what we would have in case of independence (8.30), we can find the **effective number of samples** produced by our chain

$$n_{\text{eff}} = \frac{nm}{(1 + 2\sum_{k=1}^{\infty} \rho_k)} < nm \tag{8.32}$$

And we can also write

$$\text{VAR}[\bar{\Psi}] = \frac{\text{VAR}[\Psi]}{n_{\text{eff}}} \tag{8.33}$$

A typical rule of thumb here is to reach at least $n_{\text{eff}} = 100$ effective samples.

The question now is: how do we compute the autocorrelation? We know that this identity holds (no proof given)

$$\mathbb{E}[(\Psi_i - \Psi_{i-k}^2)] = 2(1 - \rho_k)\text{VAR}[\Psi] \tag{8.34}$$

and since we can estimate both the left hand side term and VAR$[\Psi]$ from our data, we can also estimate ρ_k by inverting the formula.

The expectation above is named **Variogram at lag k** and can be estimated as

$$V_k = \frac{1}{m(n-k)} \sum_{j=1}^{m} \sum_{i=k+1}^{n} (\Psi_{i,j} - \Psi_{i-k,j})^2 \tag{8.35}$$

So that

$$\hat{\rho}_k = 1 - \frac{V_k}{2\text{VAR}^+[\Psi]} \tag{8.36}$$

We still have problems in the limit of large k because we would have few samples and very noisy estimates. Therefore we will stop the sum over k in (8.30) when the following condition is satisfied

$$T = \min\{k | k \text{ is odd}, \ \hat{\rho}_{k+1} + \hat{\rho}_{k+2} < 0\} \tag{8.37}$$

When this condition is satisfied we are in a regime where our summation is not relevant any more.

Therefore, we approximate the sum as

$$\sum_{k=1}^{\infty} \rho_k \approx \sum_{k=1}^{T} \hat{\rho}_k \tag{8.38}$$

In conclusion, we have two different diagnostics tool to detect convergence: either we look at an estimate of the variance and compute the factor $\hat{R}$ or we look at the number of effective samples and have an estimate of how decent our approximation is going to be.

8.5 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo can be considered as the state of the art method for doing Markov Chain Monte Carlo. It falls into the category of *augmented variables* Monte Carlo methods.

The idea is to turn the problem into an Hamiltonian, augmenting the state space with momentum variables that provide a sort of "kinetic energy" that allows the algorithm to move along the surface of the energy corresponding to our probability distribution. This scheme improves a lot the mixing time, hence the efficiency of MCMC algorithms. Moreover, it can be used in the case of high-dimensional multimodal distributions because it does not remain stuck in a single mode.

As usual, we start in a situation in which we want to sample from $p(x) = \frac{1}{Z}\tilde{p}(x)$, and now we express our distribution as

$$\frac{1}{Z}\tilde{p}(x) = \frac{1}{Z_x}\exp(H_x(x)) \tag{8.39}$$

This procedure is called the *Boltzmann Trick* and it is always possible (just take the logarithm of the distribution) and turns our probability distribution in the form of an energy.

Then, we introduce momentum variables y, as many as the number of x variables that we have, and we assign to them the probability distribution $p(y) = \frac{1}{Z_y}exp(H_y(y))$, where is typical to make the assumption

$$H_y(y) = \frac{1}{2}y^T y \tag{8.40}$$

i.e. to consider $p(y)$ a standard Gaussian.

We are going to sample from the joint distribution, exploiting the independence of our variables

$$p(x,y) = p(x)p(y) = \frac{1}{Z_x Z_y}\exp(H_x(x) + H_y(y)) = \frac{1}{Z}\exp(H(x,y)) \tag{8.41}$$

The idea of the algorithm is to sample from $p(x,y)$ and then forget about y. But how do we sample? We are going to sample according to the force field defined by the Hamiltonian, i.e. along lines which keep the energy constant. This means that, given some velocity, we follow a trajectory on the probability distribution space accordingly to the equations of motion in order to explore the space without losing energy. In fact, we would like to preserve energy because we want to move between regions with high probability.

Hence we have the following algorithm:

1. We start from point x_i
2. We sample $y \sim p(y)$, i.e. we randomize the momentum
3. We choose a random direction in time, i.e. we sample from $\{-1, 1\}$ uniformly. This makes our problem reversible and provides ergodicity.
4. We move according to Hamiltonian dynamics from (x_i, y) to a candidate (x', y') doing L steps
5. We introduce the following rejection criterion: we accept if $H(x', y') > H(x, y)$, otherwise we accept with probability $exp(H(x', y') - H(x, y))$

Moving accordingly to Hamiltonian dynamics means to move from (x, y) to (x', y') keeping $H(x', y') = H(x, y)$. At each step we set $(x', y') = (x + \Delta x, y + \Delta y)$ and Taylor expand the Hamiltonian

$$H(x + \Delta x, y + \Delta y) \approx H(x, y) + \nabla_x H_x(x)^T \Delta x + \nabla_y H_y(y)^T \Delta y \quad (8.42)$$

Then, enforcing the condition $H(x', y') = H(x, y)$, we can derive the equations for our movement:

$$\Delta x = \epsilon \nabla_y H_y(y) \quad (8.43)$$

$$\Delta y = -\epsilon \nabla_x H_x(x) \quad (8.44)$$

and we do L steps of this dynamics to find (x', y')

Notice that, even if in principle we require the Hamiltonian to stay constant (and in that case we would always accept because $exp(H(x', y') - H(x, y)) = exp(0) = 1$), the are still numerical integration errors introduced by the approximation above which actually let us change the value of the Hamiltonian, hence we need to define the acceptance criterion as above to ensure that the dynamics compensate this error.

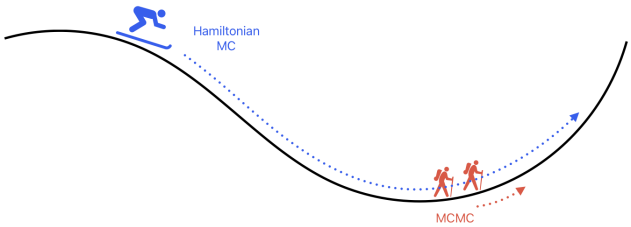
Remark Since we are required to compute gradients, Hamiltonian Monte Carlo works only for continuous variables. The advantage of exploiting gradient information is similar to the one in gradient based optimizers (gradient descent).

Remark Our acceptance criterion is, in fact, Metropolis acceptance criterion

$$\alpha = \min\left\{1, \frac{p(x', y')}{p(x, y)}\right\} = \min\{1, \exp(H(x', y') - H(x, y))\} \quad (8.45)$$

Additional heuristics, like the *no-u-turn* Hamiltonian Monte Carlo, are the state of the art of Monte Carlo methods.

Remark The number of effective samples is higher than standard MCMC, because with Hamiltonian Monte Carlo we take larger steps


Hamiltonian
MC
MCMC