Example 5.8 (The banana density)**.** Consider the following density:

$$p(x, y) \propto \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

This is only available in unnormalised form and it is an excellent test problem for many algorithms to fail. We have

$$\bar{p}_\star(x, y) = \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

and let us choose the proposal

$$q(x', y'|x, y) = \mathcal{N}(x'; x, \sigma_q^2)\mathcal{N}(y'; y, \sigma_q^2).$$

This is a symmetric proposal so the acceptance ratio is

$$\mathsf{r}(x, y, x', y') = \frac{\bar{p}_\star(x', y')}{\bar{p}_\star(x, y)}.$$

Note that it makes sense to only compute log-acceptance ratio here

$$\log \mathsf{r}(x, y, x', y') = \log \bar{p}_\star(x', y') - \log \bar{p}_\star(x, y),$$

and implement the acceptance rate by drawing $U \sim \mathrm{Unif}(0, 1)$ and accepting if $\log U < \log \mathsf{r}(x, y, x', y')$. The result can be seen from Fig. 5.4.

5.4 GIBBS SAMPLING

We will now go into another major class of MCMC samplers, called Gibbs samplers. The idea of Gibbs samplers is that, given a joint distribution of many variables $p(x_1, \ldots, x_d)$, one can build a Markov chain that samples from this distribution by sampling from the conditional distributions. This will also allow us straightforwardly sample from high-dimensional distributions. The downside of this approach is that, one has to derive the conditional distributions, which can be difficult. However, if one can do this, then the Gibbs sampler can be a very efficient method.

In this chapter, we denote our target similarly as $p_\star(x)$ where $x \in \mathbb{R}^d$ and define the *full conditional* distributions as

$$p_{m,\star}(x_m|x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_d) = p_{m,\star}(x_m|x_{1:m-1}, x_{m+1:d}) = p_{k,\star}(x_m|x_{-m}),$$

where $x_{-m} = (x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_d)$ is the vector of all variables except x_m. For a moment, assume that the full conditionals are available. Also assume, we obtain $X_{n-1} \in \mathbb{R}^d$ at the $n-1$'th iteration of the algorithm. To denote individual components, we use $X_{n-1,m}$ to denote the m'th component of X_{n-1}. A Gibbs step is then defined as

$$X_{n,m} \sim p_{m,\star}(X_{n,m}|X_{n-1,-m}),$$

for $m = 1, \ldots, d$. We define this more precisely in Algorithm 11. Of course, the key aspect

Algorithm 11 Pseudocode for the Gibbs sampler

Input: The number of samples N, and starting point $X_0 \in \mathbb{R}^d$.
for $n = 1, \ldots, N$ **do**
    for $m = 1, \ldots, d$ **do**
        Sample

$$X_{n,m} \sim p_{m,\star}(X_{n,m}|X_{n-1,-m}),$$

    end for
end for
Discard first `burnin` samples and return the remaining samples.

of the Gibbs sampler is to derive the full conditional distributions. We will come back to this point, but we will first investigate why the Gibbs sampling approach provides us a valid MCMC kernel, in other words, how the Gibbs sampler satisfies the detailed balance.

Let us denote $x = (x_m, x_{-m})$. It is easy to see from Algorithm 11 that the Gibbs sampler for every iteration (at time n) is defined as d separate operations, each sampling from the conditional distribution. We can first look at what goes on in each of these d updates . It is also easy to see that, the kernel defined in each of these d updates is given as

$$K_m(x'|x) = p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}),$$

where $\delta_{x_{-m}}(x'_{-m})$ is the Dirac delta function. Intuitively, each step samples from the full conditional $p_{m,\star}(\cdot|x_{-m})$ for mth dimension where $m \in \{1, \ldots, d\}$ and leaves others unchanged, which is enforced by the term $\delta_{x_{-m}}(x'_{-m})$. One can then see that the entire Gibbs kernel can be written as

$$K = K_1 K_2 \ldots K_d.$$

Note that each kernel is an *integral operator* – therefore the above equation is almost symbolic, it does not mean multiplication of kernels. We will now show that the Gibbs kernel satisfies the detailed balance.

Proposition 5.3. *The Gibbs kernel K leaves the target distribution $p_\star$ invariant.*

Proof. We first show that each kernel K_m satisfies the detailed balance condition:

$$\begin{aligned}
p_\star(x)K_m(x'|x) &= p_\star(x)p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}) \\
&= p_\star(x_{-m})p_{m,\star}(x_m|x_{-m})p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}) \\
&= p_\star(x'_{-m})p_{m,\star}(x'_m|x'_{-m})p_{m,\star}(x_m|x'_{-m})\delta_{x'_{-m}}(x_{-m}) \\
&= p_\star(x')K_m(x|x').
\end{aligned}$$

The steps of this derivation follows from the fact that the use of Dirac allows us to exchange variables x and x'. This shows that K_m satisfies the detailed balance condition, therefore, we have

$$\int K_m(x'|x)p_\star(x)\mathrm{d}x = p_\star(x'),$$

i.e., K_m leaves $p_\star$ invariant. Let us denote now the integral

$$(K_m, p_\star) = \int K_m(x'|x)p_\star(x)\mathrm{d}x = p_\star.$$

One can see that we have $(K_2, (K_1, p_\star)) = (K_2, p_\star) = p_\star$, which is true for all $m = 1, \ldots, d$. Therefore, we see that application of d kernels $K_1, \ldots, K_d$ will leave $p_\star$ invariant. □

We can also see why Gibbs sampling works by relating it to the Metropolis-Hastings algorithm. Recall that we can see our sampling from the conditional as a proposal, i.e.,

$$q_m(x'|x) = p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}).$$

If we calculate the acceptance ratio for this proposal

$$\begin{aligned}\alpha_m(x'|x) &= \min\left\{1, \frac{p_\star(x')q_m(x|x')}{p_\star(x)q_m(x'|x)}\right\}, \\ &= \min\left\{1, \frac{p_\star(x')K_m(x|x')}{p_\star(x)K_m(x'|x)}\right\}.\end{aligned}$$

We see that this is equal to 1 as the detailed balance is satisfied for q_m (which is K_m – see the proof of Proposition 5.3).

As we noted before, we have shown that the kernel K would leave $p_\star$ invariant, but this would not give us proper convergence guarantees. Note that the version of the algorithm we presented is called *deterministic scan* Gibbs sampler. The reason for this is that the algorithm in Alg. 11 is implemented so that we sample $x_1, \ldots, x_d$ in order, *scanning* the variables deterministically. It turns out, while this sampler's convergence guarantees cannot be established easily, there is an algorithmic fix which results in a procedure that is also guaranteed to converge. Instead of scanning the variables deterministically, we can sample them in a random order. This is called the *random scan* Gibbs sampler. The algorithm is given in Alg. 12. We will now see an example.

Algorithm 12 Random scan Gibbs sampler

1: Input: The number of samples N, and starting point $X_0 \in \mathbb{R}^d$.
2: **for** $n = 1, \ldots, N$ **do**
3: **for** $m = 1, \ldots, d$ **do**
4: Sample $j \sim \{1, \ldots, d\}$

$$X_{n,j} \sim p_{j,\star}(X_{n,j}|X_{n-1,-j}),$$

5: **end for**
6: **end for**

Example 5.9 (Image denoising)**.** A biologist knocked your door as some of the images from the microscope were too noisy. You decide to help and use the Gibbs sampler for this.

Consider a set of random variables X_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. This is a matrix modeling an $m \times n$ image. We assume that we have an image that takes values $X_{ij} \in \{-1, 1\}$ – note that this is an "unusual" image, as the images usually take values

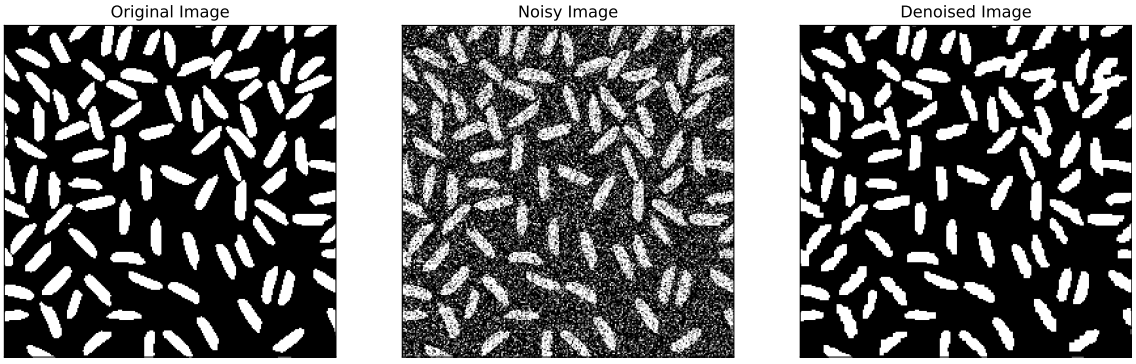

Figure 5.5: Denoising of an image using Gibbs sampler. The left column shows the original image, the middle column shows the noisy image, and the right column shows the denoised image. I used $\sigma = 1$, $J = 4$ for this and the Gibbs sampler scanned the entire image only 10 times.

between $[0, 255]$ (or $[0, 1]$). We assume that the image is corrupted by noise, i.e., we have a noisy image

$$Y_{ij} = X_{ij} + \sigma\epsilon_{ij},$$

where $\epsilon_{ij} \sim \mathcal{N}(0, 1)$ and σ is the standard deviation of the noise. We assume that the noise is independent of the image. We want to recover the image X_{ij} from the noisy image Y_{ij} and utilise Gibbs sampler for this purpose.

Our aim is to obtain (conceptually) $p(X|Y)$, i.e., samples from $p(X|Y)$ given Y. For this, we need to specify a prior $p(X)$. We take this from the literature and place as a prior a smooth Markov random field (MRF) assumption. This is formalised as

$$p(X_{ij}|X_{-ij}) = \frac{1}{Z}\exp(JX_{ij}W_{ij}),$$

where W_{ij} is the sum of the X_{ij}'s in the neighbourhood of X_{ij}, i.e.,

$$W_{ij} = \sum_{kl:\text{neighbourhood of } (i,j)} X_{kl} = X_{i-1,j} + X_{i+1,j} + X_{i,j-1} + X_{i,j+1}.$$

This is an intuitive model of the image, making the current value of the pixel depend on the values of its neighbours.

We aim at using a Gibbs sampler approach from sampling the posterior $p(X|Y)$. Note that now we need to sample from full conditionals, e.g., for each (i, j), we need to sample from $X_{ij} \sim p(X_{ij}|X_{-ij}, Y_{ij})$. We derive the full conditional as

$$p(X_{ij} = k|X_{-ij}, Y_{ij}) = \frac{p(Y_{ij}|X_{ij} = k)p(X_{ij} = k|X_{-ij})}{\sum_{k\in\{-1,1\}} p(Y_{ij}|X_{ij} = k)p(X_{ij} = k|X_{-ij})},$$

where $p(Y_{ij}|X_{ij} = k) = \mathcal{N}(Y_{ij}; k, \sigma^2)$ is the likelihood of the noisy image given the value of the pixel. We can easily compute these probabilities since each term in the Bayes rule is computable (and $1/Z$ cancels). Therefore, we can get explicit expressions for $q = p(X_{ij} = 1|X_{-ij}, Y_{ij})$ and $1 - q = p(X_{ij} = -1|X_{-ij}, Y_{ij})$. We can then sample from the full conditional as

$$X_{ij} \sim \begin{cases} 1 & \text{with probability } q, \\ -1 & \text{with probability } 1 - q. \end{cases}$$

We can now loop over (i, j) (to sample from each full conditional) and sample from the full conditionals. This is the Gibbs sampler algorithm as described above. The results of this procedure can be seen from Fig. 5.5.

5.5 LANGEVIN MCMC METHODS

We briefly introduced Metropolis-adjusted Langevin algorithm (MALA) in Sec. 5.3.3. MALA is just one example of a more general class of samplers, called Langevin MCMC algorithms, which are based on Langevin dynamics. These approaches are at the forefront of modern MCMC methods, used in a variety of settings, including sampling from high-dimensional targets, deep learning, sampling from Bayesian neural networks, and so on. We will now introduce the Langevin MCMC methods and see how they work.

Consider again our target $p_\star$ defined on $\mathbb{R}^d$. It turns out, we can use stochastic differential equations (SDEs) to sample from $p_\star$ (recall that SDEs are differential equations with a stochastic term). Consider the following SDE:

$$\mathrm{d}X_t = -\nabla \log p_\star(X_t)\mathrm{d}t + \mathrm{d}B_t, \tag{5.5}$$

where B_t is a standard Brownian motion. It turns out the marginal distributions of X_t driven by this SDE converge to $p_\star$ as $t \to \infty$. In other words, in many suitable metrics, we can quantify the convergence $d(p_t, p_\star) \to 0$ as $t \to \infty$. This means that all we need to draw samples from $p_\star$ is to numerically solve this SDE which can be done with a variety of numerical methods (akin to ODE solvers) . One caveat in this situation is that, while the SDE would target $p_\star$, its discretisation would incur bias. This is why MALA is "Metropolised".

Let us recall the MALA algorithm. We start with a point X_0 and then define the proposal

$$q(x_n|x_{n-1}) = \mathcal{N}(x_n; x_{n-1} + \gamma\nabla \log p_\star(x_{n-1}), \sqrt{2\gamma}I_d),$$

where $\gamma > 0$ is a step size. We then sample from q to obtain X_n. We then accept X_n with probability

$$\alpha(X_n, X_{n-1}) = \min\left(1, \frac{p_\star(X_n)q(X_{n-1}|X_n)}{p_\star(X_{n-1})q(X_n|X_{n-1})}\right). \tag{5.6}$$

Recall that the MALA proposal would not define a symmetric proposal, therefore, we would need to compute the ratio. Now one can see that, Eq . (5.6) can be equivalenly written as

$$X_n = X_{n-1} + \gamma\nabla \log p_\star(X_{n-1}) + \sqrt{2\gamma}\mathrm{d}W_n, \tag{5.7}$$

where $W_n \sim \mathcal{N}(0, I)$. The relationship of Eq. (5.7) to (5.5) can be seen by noting that the discretisation of the SDE in (5.5) would exactly take the form of Eq. (5.7). Therefore, MALA uses this Langevin SDE as the proposal and then accept/reject its samples. This has a beneficial effect of correcting the bias of the discretisation.

However, the Metropolis step can be computationally infeasible in higher dimensions, just as in the case of rejection sampling. Higher dimensional problems cause the acceptance rate to vanish, which results in slow convergence. To remedy this situation, a common approach is to simply drop the Metropolis step and use the following iteration:

$$X_n = X_{n-1} + \gamma\nabla \log p_\star(X_{n-1}) + \sqrt{2\gamma}\mathrm{d}W_n. \tag{5.8}$$

This is simple the MCMC method - which is called the unadjusted Langevin algorithm (ULA). The ULA is a discretisation of the SDE, and as such, its stationary measure is not $p_\star$. However, under various conditions, it can be shown that the limiting distribution of ULA $p_\star^\gamma$ can be made arbitrarily close to $p_\star$ as $\gamma \to 0$. This means that the ULA can be a viable alternative.

Example 5.10. Consider the target $p_\star(x) = \mathcal{N}(x; \mu, \sigma^2)$, and let us sample from it using the ULA. We will first need

$$\frac{\partial}{\partial x} \log p_\star(x) = -\frac{x-\mu}{\sigma^2}.$$

We can then write the iterates of ULA as

$$\begin{aligned} X_n &= X_{n-1} + \gamma \frac{\partial}{\partial x} \log p_\star(X_{n-1}) + \sqrt{2\gamma} \mathrm{d}W_n, \\ &= X_{n-1} - \gamma \frac{X_{n-1} - \mu}{\sigma^2} + \sqrt{2\gamma} W_n, \\ &= \left(1 - \frac{\gamma}{\sigma^2}\right) X_{n-1} + \frac{\gamma}{\sigma^2}\mu + \sqrt{2\gamma} W_n, \end{aligned}$$

where $W_n \sim \mathcal{N}(0, 1)$. In this simple case, we can compute the stationary distribution of the chain and analyse its relationship to the true target $p_\star$. Let

$$a = 1 - \frac{\gamma}{\sigma^2}, \quad b = \frac{\gamma}{\sigma^2}\mu.$$

We can write now the iterates beginning at x_0 as

$$\begin{aligned} x_1 &= a x_0 + b + \sqrt{2\gamma} W_1, \\ x_2 &= \underbrace{a^2 x_0 + ab + a\sqrt{2\gamma} W_1}_{a x_1} + b + \sqrt{2\gamma} W_2, \\ x_3 &= \underbrace{a^3 x_0 + a^2 b + a^2\sqrt{2\gamma} W_1 + ab + a\sqrt{2\gamma} W_2}_{a x_2} + b + \sqrt{2\gamma} W_3, \\ &\vdots \\ x_n &= a^n x_0 + \sum_{k=0}^{n-1} a^k b + \sum_{k=0}^{n-1} a^k \sqrt{2\gamma} W_k. \end{aligned}$$

As $n \to \infty$, one can compute the mean and the variance of x_∞ iterate. Since W_i are zero-mean, and $0 < a < 1$, the asymptotic mean is given by

$$\mu_\infty = \sum_{k=0}^{n-1} a^k b = \frac{b}{1-a} = \mu.$$

The variance of the iterates as $n \to \infty$ can also be computed. Note that for finite n, we have

$$\begin{aligned} \mathsf{var}(x_n) &= \mathsf{var}\left(\sum_{k=0}^{n-1} a^k \sqrt{2\gamma} W_k\right), \\ &= 2\gamma \sum_{k=0}^{n-1} (a^2)^k, \\ &= 2\gamma \frac{1 - a^{2n}}{1 - a^2}. \end{aligned}$$

Therefore, we obtain the limiting variance as

$$\begin{aligned}\lim_{n\to\infty} \mathsf{var}(x_n) &= 2\gamma \frac{1}{1-a^2} \\ &= 2\gamma \frac{1}{1-\left(1-\frac{\gamma}{\sigma^2}\right)^2} \\ &= 2\gamma \frac{1}{\frac{2\gamma}{\sigma^2}-\frac{\gamma^2}{\sigma^4}}, \\ &= \frac{2\sigma^4}{\sigma^2-\gamma}.\end{aligned}$$

Therefore, we obtained the target measure of ULA as

$$p_\star^\gamma(x) = \mathcal{N}\left(x; \mu, \frac{2\sigma^4}{\sigma^2-\gamma}\right),$$

which is different than $p_\star$. Note that in this particular case, the means of the $p_\star^\gamma$ and $p_\star$ agree. It can be seen that the bias enters the picture through the variance, but this vanishes as $\gamma \to 0$.

We can also derive the ULA for the Banana density.

Example 5.11. Consider the Banana density

$$p(x,y) \propto \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y-x^2)^2\right).$$

This is only available in unnormalised form:

$$\bar{p}_\star(x,y) = \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y-x^2)^2\right).$$

Recall that $\nabla \log \bar{p}_\star(x,y) = \nabla \log p_\star(x,y)$. Therefore, we will directly compute the unnormalised gradients:

$$\nabla \log p_\star(x,y) = \begin{bmatrix} \frac{-2x}{5} + 4x(y-x^2) \\ \frac{-2y^3}{5} - 4(y-x^2) \end{bmatrix}.$$

Therefore, the update in 2D is given by

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \gamma \nabla \log p_\star(x_n, y_n) + \sqrt{2\gamma} V_n$$

where $V_n \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$.

Example 5.12 (Bayesian inference with ULA)**.** We can also straightforwardly perform Bayesian inference in this setting . Recall the target posterior density in this setting

$$\bar{p}_\star(x|y) = p(x) \prod_{k=1}^{n} p(y_k|x),$$

where $p(x)$ is the prior density and $p(y_k|x)$ is the likelihood where observations are conditionally i.i.d given x. We can write the ULA iterates as

$$\begin{aligned} X_n &= X_{n-1} + \gamma \nabla \bar{p}_\star(X_{n-1}|y) + \sqrt{2\gamma} V_n, \\ &= X_{n-1} + \gamma \left(\nabla \log p(x) + \sum_{k=1}^{n} \nabla \log p(y_k|X_{n-1}) \right) + \sqrt{2\gamma} V_n. \end{aligned}$$

A common problem arising in machine learning and statistics is *big data*, where the number of observations n is large. In this case, both ULA and MALA are infeasible as both require the iterates above to be evaluated, e.g., each iteration involves summing n terms. If n is order of millions, this is computationally infeasible. In this case, we can use the *stochastic* gradients. This is only applicable in the setting of ULA as we will see below, which is one reason why ULA-type methods are more popular than MALA-type methods.

5.5.1 STOCHASTIC GRADIENT LANGEVIN DYNAMICS

The problem of large number of data points arise in the setting of ULA as a sum, therefore, we should look for estimating large sums with something cheaper. Consider the following sum of (arbitrary) numbers:

$$g = \frac{1}{n} \sum_{k=1}^{n} g_i.$$

If n is simply too large to compute this sum efficiently, we can instead resort to *unbiased* estimates of it. This can be done by sampling $i_1, \dots, i_K \sim \{1, \dots, n\}$ uniformly and constructing

$$\hat{g} = \frac{1}{K} \sum_{k=1}^{K} g_{i_k}.$$

This estimate is an unbiased estimate of g, i.e.,

$$\mathbb{E}[\hat{g}] = \mathbb{E}\left[\frac{1}{K} \sum_{k=1}^{K} g_{i_k} \right] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[g_{i_k}] = g.$$

This idea can be used to construct stochastic gradients. We provide some examples below.

Example 5.13 (Large scale Bayesian inference)**.** Recall the problem setting in Example 5.12:

$$X_n = X_{n-1} + \gamma \left(\nabla \log p(x) + \sum_{k=1}^{n} \nabla \log p(y_k|X_{n-1}) \right) + \sqrt{2\gamma} V_n.$$

Assume we sample uniformly from $i_1, \ldots, i_K$, we can then approximate the sum

$$\sum_{k=1}^{n} \nabla \log p(y_k|X_{n-1}) \approx \frac{n}{K} \sum_{k=1}^{K} \nabla \log p(y_{i_k}|X_{n-1}).$$

Note that (n/K) factor comes here as the sum itself did not have $(1/n)$ term (as opposed to the sum example above). Therefore, the stochastic gradient Langevin dynamics (SGLD) iterate can be written as

$$X_n = X_{n-1} + \gamma \left(\nabla \log p(x) + \frac{n}{K} \sum_{k=1}^{K} \nabla \log p(y_{i_k}|X_{n-1}) \right) + \sqrt{2\gamma} V_n.$$

This is also called *data subsampling* as one can see that the gradient only uses a subset of the data. Every iteration is cheap and computable as we only need to compute K terms. This is a very popular method in Bayesian inference and is used in many applications.