## Sampling for Bayesian Neural Networks

Another common approach to dealing with intractable posteriors is to sample from them and treat the samples as representative of the entire distribution. Formally, this procedure makes the following approximation:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, p(\mathbf{w} \mid \mathcal{D})\, \mathrm{d}\mathbf{w} \approx \frac{1}{S} \sum_{s=1}^{S} p(y^* \mid \mathbf{x}^*, \mathbf{w}_s)$$

where $\mathbf{w}_s \sim p(\mathbf{w} \mid \mathcal{D})$ are samples from the intractable posterior. One interpretation of this method is that instead of using just a single setting of the weights e.g., $\mathbf{w}^*$, we instead consider an ensemble of networks, each one with a different set of weights, $\mathbf{w}_s$.

### Rejection Sampling

Unfortunately, sampling from the distribution $p(\mathbf{w} \mid \mathcal{D})$ is nontrivial in many settings, including Bayesian neural networks. Consider, for example, rejection sampling, one common algorithm for drawing exact samples from an intractable distribution. This algorithm, detailed in Algorithm 1 and depicted pictorially in Figure 1, samples a point from some *proposal* distribution $q$ and then "accepts" that point with probability proportional to its likelihood under the target distribution. Crucially, this algorithm works even if the target distribution is only known up to normalization i.e., it suffices to be able to compute $p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w}) \propto p(\mathbf{w} \mid \mathcal{D})$, which is incredibly important given that $p(\mathbf{w} \mid \mathcal{D})$ is assumed to be intractable. We do require that the proposal distribution $q$ dominates or is greater than $p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})\ \forall\ \mathbf{w}$, which can be achieved via scaling.

**Algorithm 1:** Rejection Sampling

**input :** simple proposal distribution $q(\mathbf{w})$ e.g., s a standard multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, scaling constant $k$ s.t. $kq(\mathbf{w}) \geq \tilde{p}(\mathbf{w} \mid \mathcal{D}) = p(\mathcal{D} \mid \mathbf{w})p(\mathbf{w}) \propto p(\mathbf{w} \mid \mathcal{D})\ \forall\ \mathbf{w}$

**output :** sample from $p(\mathbf{w} \mid \mathcal{D})$, $\mathbf{w}_s$

**while** *TRUE* **do**
  Draw a sample from $q$, $\mathbf{w}_s$
  Draw a value uniformly at random from $[0, kq(\mathbf{w}_s)]$, $u_s$
  **if** $u_s \leq \tilde{p}(\mathbf{w}_s \mid \mathcal{D})$ **then**
    BREAK
  **end**
**end**
RETURN $\mathbf{w}_s$

In his Ph.D. thesis, Radford Neal (1993) conducted a small experiment where he drew 10 samples from the posterior of a *very* simple Bayesian neural network (one hidden layer with 16 nodes): using rejection sampling to draw these 10 samples, even after tailoring the proposal distribution and scaling constant, required 2.6 million samples from the proposal distribution!

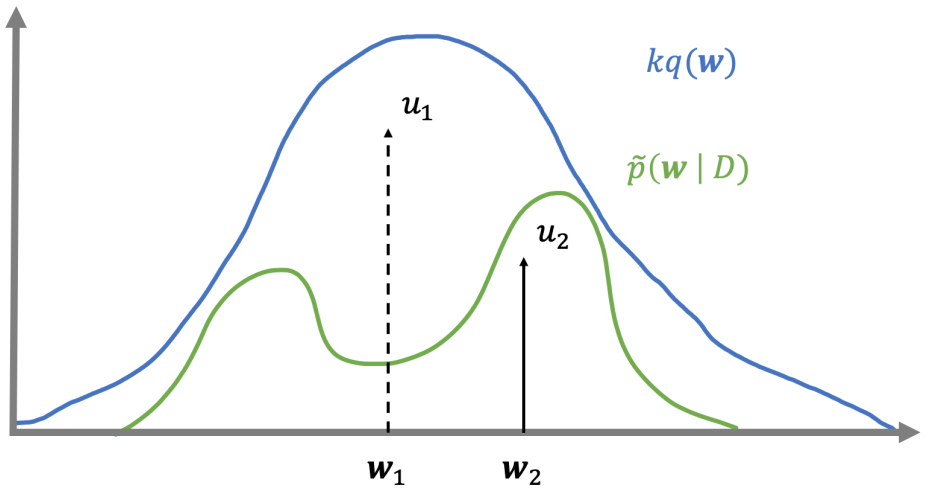


Figure 1: A depiction of two iterations of rejection sampling. The blue curve is the proposal distribution multiplied by a scaling constant and the green curve is proportional to the target distribution. The first sampled point $\mathbf{w}_1$ would be "rejected" while the second sampled point $\mathbf{w}_2$ would be "accepted".

**Importance Sampling**

An alternative to rejection sampling, which can waste or "reject" a lot of samples, is importance sampling. Like rejection sampling, importance sampling also leverages an easy to sample from proposal distribution; however, instead of directly rejecting "bad" or unlikely samples, importance sampling reweights samples according to their likelihood under the intractable posterior. Formally, given a proposal distribution $q$, importance sampling leverages the following approximation:

$$\begin{aligned}\int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, p(\mathbf{w} \mid \mathcal{D})\, \mathrm{d}\mathbf{w} &= \int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, \frac{p(\mathbf{w} \mid \mathcal{D})}{q(\mathbf{w})}\, q(\mathbf{w})\, \mathrm{d}\mathbf{w} \\ &\approx \frac{1}{S}\sum_{s=1}^{S} p(y^* \mid \mathbf{x}^*, \mathbf{w}_s)\frac{p(\mathbf{w}_s \mid \mathcal{D})}{q(\mathbf{w}_s)} = \frac{1}{S}\sum_{s=1}^{S} c_s p(y^* \mid \mathbf{x}^*, \mathbf{w}_s)\end{aligned}$$

where $\mathbf{w}_s \sim q(\mathbf{w})$ and $c_s = {}^{p(\mathbf{w}_s \mid \mathcal{D})}/_{q(\mathbf{w}_s)}$; written in this form, we can interpret importance sampling as approximating the posterior as a weighted average of samples drawn from $q$ where the weights are determined by plausible each sample is under $p(\mathbf{w} \mid \mathcal{D})$.

In the form above, importance sampling requires the ability to compute the true but intractable posterior $p(\mathbf{w}_s \mid \mathcal{D})$. However, much like rejection sampling, we can still use importance sampling even when the posterior is only known up to normalization i.e., using just $p(\mathcal{D} \mid \mathbf{w}_s)\, p(\mathbf{w}_s)$:

$$\begin{aligned}\int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, \frac{p(\mathbf{w} \mid \mathcal{D})}{q(\mathbf{w})}\, q(\mathbf{w})\, \mathrm{d}\mathbf{w} &= \frac{1}{Z}\int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, \frac{p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})}{q(\mathbf{w})}\, q(\mathbf{w})\, \mathrm{d}\mathbf{w} \\ &\approx \frac{1}{SZ}\sum_{s=1}^{S} p(y^* \mid \mathbf{x}^*, \mathbf{w}_s)\frac{p(\mathcal{D} \mid \mathbf{w}_s)\, p(\mathbf{w}_s)}{q(\mathbf{w}_s)} = \frac{1}{SZ}\sum_{s=1}^{S} \tilde{c}_s p(y^* \mid \mathbf{x}^*, \mathbf{w}_s)\end{aligned}$$

where

$$Z = \int p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})\, \mathrm{d}\mathbf{w} = \int \frac{p(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})}{q(\mathbf{w})}\, q(\mathbf{w})\, \mathrm{d}\mathbf{w} \approx \frac{1}{S} \sum_{s=1}^{S} \frac{p(\mathcal{D} \mid \mathbf{w}_s)\, p(\mathbf{w}_s)}{q(\mathbf{w}_s)} = \frac{1}{S} \sum_{s=1}^{S} \tilde{c}_s.$$

Substituting this result into the equation above gives the desired result:

$$\int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, \frac{p(\mathbf{w} \mid \mathcal{D})}{q(\mathbf{w})}\, q(\mathbf{w})\, \mathrm{d}\mathbf{w} \approx \sum_{s=1}^{S} \frac{\tilde{c}_s}{\sum_{s=1}^{S} \tilde{c}_s}\, p(y^* \mid \mathbf{x}^*, \mathbf{w}_s) \text{ where } \tilde{c}_s = \frac{p(\mathcal{D} \mid \mathbf{w}_s)\, p(\mathbf{w}_s)}{q(\mathbf{w}_s)}.$$

While less wasteful than rejection sampling, importance sampling can also struggle or be inaccurate if the proposal distribution $q$ is very different from the true, intractable posterior. In particular, what tends to occur is that the weight on almost all of the samples is very small and the approximation becomes a function of just a small number of samples; in effect, it is as if we "rejected" a bunch of samples by simply assigning them a weight near zero! In such cases, we often turn towards approximate sampling methods as opposed to trying to draw exact samples.

**Markov Chain Monte Carlo**

A popular class of approximate sampling techniques are known as Markov chain Monte Carlo (MCMC) methods. The idea behind these methods is that we maintain a Markov chain or sequence of samples $\{\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_t, \ldots\}$ defined by a stochastic *transition function* from the current sample, $\mathbf{w}_t$, to the next, $\mathbf{w}_{t+1}$. If the transition function meets certain properties related to the target distribution then eventually (i.e., after sufficiently many steps or transitions), the Markov chain will converge to a "stationary" or "equilibrium" distribution and samples from this stationary distribution are approximate samples from the target distribution, $p(\mathbf{w} \mid \mathcal{D})$.

As a simple example, suppose we are trying to approximate a discrete distribution over three possible values or states:

$$p^* = \begin{bmatrix} \frac{3}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}.$$

Consider the transition matrix

$$T = \begin{bmatrix} \frac{2}{3} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & 0 & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{2} & 0 \end{bmatrix} \text{ where } T_{ij} = \text{ the probability of transitioning from state } i \text{ to state } j.$$

We will say that $p^*$ is an *invariant* distribution of $T$ as $Tp^* = p^*$. In addition, $p^*$ is the equilibrium distribution of $T$ as

$$T^N \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \approx T^N \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \approx T^N \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \approx p^* \text{ for sufficiently large } N.$$

Our goal in this context will be to develop transition functions such that a) the equilibrium distribution is a good approximation for the true, intractable posterior and b) the number of steps needed to converge to the equilibrium distribution is low.

The various MCMC methods differ in how they compute the transitions from one sample to the next; we will explore a few of the commonly-used MCMC algorithms below.

### Metropolis-Hastings

One of the simplest MCMC methods is known as the Metropolis-Hastings algorithm. At a high-level, it looks and behaves a lot like rejection sampling: we use a simple proposal distribution to generate potential next samples and then, with some probability related to the likelihood under the true, intractable posterior, we either transition to the proposed next sample or "reject" it and remain at our current sample. The full algorithm is detailed in Algorithm 2.

**Algorithm 2:** Metropolis-Hastings

**input :** initial value $\mathbf{w}_0$, proposal variance $\sigma^2$,
burn-in or number of samples the Markov chain needs to converge, $T$
**output :** sample from $p(\mathbf{w} \mid \mathcal{D})$, $\mathbf{w}_T$

**for** $t = 1, \ldots, T$ **do**
  Draw a sample from a simple conditional proposal distribution $q(\mathbf{w} \mid \mathbf{w}_{t-1})$ e.g., $\mathcal{N}(\mathbf{w}; \mathbf{w}_{t-1}, \sigma^2\mathbf{I})$, $\mathbf{w}_s$
  Calculate the acceptance ratio

$$\alpha = \frac{\tilde{p}(\mathbf{w}_s \mid \mathcal{D})}{\tilde{p}(\mathbf{w}_{t-1} \mid \mathcal{D})} = \frac{p(\mathcal{D} \mid \mathbf{w}_s)p(\mathbf{w}_s)}{p(\mathcal{D} \mid \mathbf{w}_{t-1})p(\mathbf{w}_{t-1})}$$

  Draw a value uniformly at random from $[0, 1]$, $u_t$
  **if** $u_t \leq \alpha$ **then**
    $\mathbf{w}_t = \mathbf{w}_s$
  **else**
    $\mathbf{w}_t = \mathbf{w}_{t-1}$
  **end**
**end**
RETURN $\mathbf{w}_T$

Under certain assumptions about the proposal distribution $q$, it can be shown that the posterior distribution, $p(\mathbf{w} \mid \mathcal{D})$, is the invariant distribution of this transition function. Metropolis-Hastings is predictably quite sensitive to the choice of proposal distribution: for the simple Gaussian distribution used in Algorithm 2, if $\sigma^2$ is too large, we can end up rejecting almost all of the proposed samples and if $\sigma^2$ is too small, it can take a long time to converge to the true posterior.

### Gibbs Sampling

An alternative to Metropolis-Hastings that does not waste samples is Gibbs sampling. In Gibbs sampling, given a joint distribution over a set of variables, $p(\mathbf{w} \mid \mathcal{D}) = p(w_1, w_2, \ldots, w_D \mid \mathcal{D})$, we generate subsequent samples one dimension or one variable at a time, keeping the other $D - 1$ fixed. The full algorithm is detailed in Algorithm 3.

Of course, the key issue in Gibbs sampling is drawing samples from the nightmarish conditional distribution $p\left(w_d \mid \mathcal{D}, w_1, w_2, \ldots, w_{d-1}, w_{d+1}, \ldots, w_D\right)$, which is almost certainly intractable. Fortunately, we can leverage the fact that this is inherently a univariate distribution and thus, is (potentially) easier to sample from than the full joint. If necessary, we can nest something like rejection sampling or Metropolis-Hastings within Gibbs sampling to draw exact or approximate samples from this conditional distribution respectively; the latter is known as "Metropolis-within-Gibbs" sampling.

**Algorithm 3:** Gibbs Sampling

**input** : initial value $\mathbf{w}^{(0)} = \begin{bmatrix} w_1^{(0)} & w_2^{(0)} & \cdots & w_D^{(0)} \end{bmatrix}^T$,
burn-in or number of samples the Markov chain needs to converge, $TD$
**output**: sample from $p(\mathbf{w} \mid \mathcal{D})$, $\mathbf{w}_T$

**for** $t = 1, \ldots, T$ **do**
  **for** $d = 1, \ldots, D$ **do**
    Draw a sample from the true conditional distribution over $w_d$, conditioned on the current values for all other variables in $\mathbf{w}$:
    $w_d^{(t)} \sim p\left(w_d \mid \mathcal{D}, w_1^{(t)}, w_2^{(t)}, \ldots, w_{d-1}^{(t)}, w_{d+1}^{(t-1)}, \ldots, w_D^{(t-1)}\right)$
    Update the corresponding entry of $\mathbf{w}^{(t)}$ with the new sample
  **end**
**end**
RETURN $\mathbf{w}_T$

### Hamiltonian Monte Carlo

The problem with many of the simpler MCMC algorithms is that it can take them a long time to reach a true stationary distribution. Furthermore, even once they have reached the equilibrium distribution, consecutive samples (while still technically approximating the target distribution) tend to be highly correlated, so it may be necessary to draw and discard many intermediate samples to get independent samples from the intractable posterior.

One powerful MCMC method that tends to converge quickly is known as Hamiltonian Monte Carlo (HMC) (this is also sometimes referred to as "hybrid" Monte Carlo), which was proposed by Neal, again in his Ph.D. thesis. HMC uses the gradient of the target distribution (again, easily computable for Bayesian neural networks): the resulting Markov chains tend to converge faster and explore more of the domain instead of getting stuck in local maxima of the posterior, which some other MCMC methods have a tendency of doing.

At a high-level, HMC methods apply ideas from Hamiltonian dynamics for physical systems to simulate a path or trajectory through the posterior distribution. The next sample is determined based on the "potential energy" or gravitational pull on a ball rolling through the negated, log posterior.