Bayesian Machine Learning

Blaine William Rogers

November 2023

Abstract

In this report, I describe the Bayesian approach to machine learning, detailing how it treats statistical parameters as random variables and uses Bayes' rule to compute posterior distributions. I explain basic techniques like Maximum Likelihood Estimation (MLE) and introduce complex methods such as Variational Inference and Markov Chain Monte Carlo (MCMC) sampling. Variational Inference is highlighted for its performance and upgrade over Frequentist methods, while MCMC is presented as the gold standard for posterior sampling.

Contents

1 Probability Theory

1.1 Taxonomy of distributions

For two random variables X and Y,

- $P(X \text{ and } Y)$ or $P(X, Y)$ is called the *joint distribution* of X and Y. It gives the probability (density) of (X, Y) taking on a pair of values (x, y).
- $P(X)$ is called the *marginal distribution* of X. It gives the probability of X taking on a value x, given that you know nothing about Y.
- $P(X \text{ given } Y)$ or $P(X \mid Y)$ is called the *conditional distribution* of X given Y. If gives the probability of X taking on a value x given that you know Y takes on a value y.

1.2 Identities

There are three important identities that relate the joint, marginal and conditional distributions:

$$P(X, Y) = P(Y, X) \qquad \text{Reflexivity} \tag{1}$$

$$P(X, Y) = P(X \mid Y)\, P(Y) \qquad \text{Chain Rule} \tag{2}$$

$$P(X) = \int P(X \mid Y)\, P(Y)\ dY \qquad \text{Marginalization} \tag{3}$$

We can use the chain rule to derive another important identity. By reflexivity and the chain rule we have

$$P(X \mid Y)\, P(Y) = P(X, Y) = P(Y, X) = P(Y \mid X)\, P(X)$$

Dividing through by $P(Y)$ we obtain

$$P(X \mid Y) = \frac{P(Y \mid X)\, P(X)}{P(Y)} \qquad \text{Bayes' Rule} \tag{4}$$

We will also sometimes use the expected value of a distribution:

$$\mathbb{E}_{P(X)}[f(X)] = \int f(X) P(X)\ dX \qquad \text{Expected Value} \tag{5}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} f(x_i), \quad x_i \overset{\text{iid}}{\sim} P(X) \qquad \text{Numerical Expectation} \tag{6}$$

Notice that therefore $P(X) = \mathbb{E}_{P(Y)}[P(X \mid Y)]$, and we can estimate $P(X)$ by numerical expectation.

Please use this page as a reference for the rest of the talk.

2 Frequentism vs Bayesianism

2.1 Frequentism

Sometime we are interested in finding out a statistic. The canonical example is finding the average height of people.

The simplest way to do this is to get a big sample of people and measure their heights, giving us a dataset $X = \{x_i\}_{i=1}^{N}$. We can then easily calculate the average height of the people in the sample:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{7}$$

Is this good enough? There are 8 billion people in the world. Probably we don't have all 8 billion in our sample. Even if our sample is representative, it's unlikely our $\bar{x}$ will be exactly equal to the true average height of the population. We can make some assumptions about how height is distributed, and in doing so calculate the expected error between the sample average and the true average. We can also increase our sample size: probability theory tells us that the larger our sample is, the closer our estimate of the mean will be to the true mean.

Is *that* good enough? Suppose we actually do have all 8 billion people in our sample. Is $\bar{x} = \frac{1}{8\times10^9} \sum_{i=1}^{8\times10^9} x_i$ "the average height of a person"?

If you said yes, you are a Frequentist at heart. What would it even mean for the average height of all the people in the world not to be "the average height of a person"?

2.2 Bayesianism

Thought experiment: someone is born. Once they've finished growing, how tall will they be?

Most people would say "probably around average height". But if you define "average height" as the actual average height of all the people in the world, that becomes a circular answer.

The Bayesian says "there is something we are gesturing at when we say 'average height' that is distinct from the actual population average. It's something like 'the height we expect a person to be'." Call this μ to distinguish it from $\bar{x}$, the population average. The Bayesian says that this μ is itself a random variable, which cannot be observed; we can only estimate it through its effects on the observable data variables x_i, the actual heights of people. This gives rise to a huge joint distribution

$$P(\mu, x_1, x_2, \ldots, x_N) = P(\mu, X)\,. \tag{8}$$

We can use Bayes' Rule (4) to break this joint into its component parts:

$$P(\mu \,|\, X) = \frac{P(X \,|\, \mu)\, P(\mu)}{P(X)} \tag{9}$$

- $P(\mu \mid X)$ is called the *posterior distribution* over μ.
- $P(X \mid \mu)$ is called the *conditional likelihood* of the data given μ.
- $P(\mu)$ is called the *prior* on μ.
- $P(X)$ is called the *marginal likelihood* of the data.

$P(\mu \mid X)$ is the distribution of most interest to us: it tells us how tall we should expect a person to be, given how tall the people in our sample are. $P(X \mid \mu)$ and $P(\mu)$ are modelling choices—we are free to pick an appropriate and convenient family of distributions for them.

$P(X)$ is usually thorny. We can define it using (3) as

$$P(X) = \int P(X \mid \mu)\, P(\mu)\ d\mu \tag{10}$$

but that integral is often intractable. Fortunately, notice that $P(X)$ does not depend on μ; it plays the role of a normalizing constant. In situations where we only care about the relative probabilities of different values of μ, we can write

$$\begin{aligned} P(\mu \mid X) &= \frac{1}{P(X)} P(X \mid \mu)\, P(\mu)\,, \\ &\propto P(X \mid \mu)\, P(\mu)\,. \end{aligned} \tag{11}$$

which means we only have to deal with quantities we can easily calculate.

For a suitable choice of likelihood, we recover $\bar{x} = \mathbb{E}_{P(\mu \mid X)}[\mu]$. That is, the Frequentist population average returns as the *expected value of average height given the data*. We also say that the posterior *concentrates at* $\bar{x}$. But with this rich Bayesian framing, we can also talk about other quantities. For instance, the variance $\mathrm{Var}_{P(\mu \mid X)}[\mu]$ measures our uncertainty in μ.

Often we are most interested in, you know, predicting the height of an unseen person. For this we use the *posterior predictive distribution*

$$P(x' \mid X) = \int P(x' \mid \mu)\, P(\mu \mid X)\ d\mu \tag{12}$$

$$= \mathbb{E}_{P(\mu \mid X)}[P(x' \mid \mu)] \tag{13}$$

A common point of confusion is that there is no such thing as the "true value" of μ. This took me four years of undergrad and two years of professional experience in Bayesian machine learning to truly understand.

3 Again, but for Machine Learning

3.1 Frequentism

In a standard machine learning problem, we have a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ and our task is to recover an unknown function $y = f(x)$. We come up with a function class $\hat{y} = g_\theta(x)$, which is usually a neural network. Now our task becomes finding $\theta^\star$ such that $g_{\theta^\star} = f$. By coming up with a loss function $L(y, \hat{y})$, we can turn this into an optimization problem:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^N L(y_i, g_\theta(x_i)) \tag{14}$$

We have many choices for solving this optimization problem, but usually we solve it by gradient descent. Let $\theta_0 \sim N(0, 1)$, then

$$\theta_{n+1} = \theta_n - \lambda \frac{\partial}{\partial \theta} \left[\sum_{i=1}^N L(y_i, g_\theta(x_i)) \right] \Bigg|_{\theta=\theta_n} \tag{15}$$

This is "frequentist" machine learning. We took a sample D of a population, then calculated the exact value of a statistic $\theta^\star$ for that population.

3.2 Bayesianism

When all you have is a hammer, everything looks like a nail. One day, a Bayesian discovered machine learning and relabelled all its parts.

As before, we start by treating the statistic of interest θ as a random variable. Then we use Bayes' rule (4) to decompose the joint distribution:

$$P(\theta \mid D) = \frac{P(D \mid \theta)\, P(\theta)}{P(D)} \tag{16}$$

This requires giving probabilistic interpretations of each component of the machine learning problem.

- The network $\hat{y} = g_\theta(x)$ becomes the *predictive distribution* $P(y \mid x, \theta)$. This distribution is defined in terms of the loss:

  $$P(y \mid x, \theta) \propto e^{-L(y, g_\theta(x))}, \quad L(y, g_\theta(x)) = -\log P(y \mid x, \theta) \tag{17}$$

  For some common loss functions this distribution has a nice form. The L2 loss $(y - \hat{y})^2$ becomes a Gaussian distribution $N(\hat{y}, 1)$. The cross-entropy loss becomes a categorical distribution.

- Since $\log ab = \log a + \log b$, the log likelihood of the dataset is just the

negative sum of the loss plus a term constant in θ

$$P(D \mid \theta) = \prod_{i=1}^{N} P(y_i \mid x_i, \theta)\, P(x_i) \tag{18}$$

$$\log P(D \mid \theta) = \sum_{i=1}^{N} [\log P(y_i \mid x_i, \theta) + \log P(x_i)] \tag{19}$$

$$= -\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \sum_{i=1}^{N} \log P(x_i) \tag{20}$$

- The *prior* $P(\theta)$ is just the distribution from which we draw the initial parameters for gradient descent: if $\theta_0 \sim N(0, 1)$ then $P(\theta) = N(0, 1)$.
- The *marginal likelihood* $P(D)$ we can recover by integrating:

$$P(D) = \int P(D \mid \theta)\, P(\theta)\ d\theta \tag{21}$$

Now our machine learning problem becomes *characterizing the posterior distribution*. This is a powerful reframing that lets you ask a bunch of interesting questions. For instance, there are often multiple functions $y = f(x)$ that are consistent with your dataset. Instead of asking which one is "the true function", you can ask which ones are relatively more or less likely given the data you've observed.

In machine learning, we often don't care about the particular value of θ at all! Instead we care about making good predictions y' on unseen inputs x'. One big benefit of the Bayesian framing is that by characterising the posterior, we get *confidence* estimates for free! Compare the Frequentist approach

$$\hat{y}' = g_{\theta^\star}(x') \tag{22}$$

to the Bayesian approach

$$P(y' \mid x', D) = \mathbb{E}_{P(\theta \mid D)}[P(y' \mid x', \theta)] \tag{23}$$

Immediately, we can ask not just for the *expected prediction* $\hat{y} = \mathbb{E}_{P(y' \mid x', D)}[y']$ but also the *confidence* $P(y' = \hat{y} \mid x', D)$ and the *variance* $\mathrm{Var}_{P(y' \mid x', D)}[y']$.

4 The 2023 Bayesian ML Meta

How do we actually characterize the posterior? The dream would be to find a simple closed form for $P(\theta \mid D)$ or $P(y' \mid x', D)$. Then we could easily calculate the expected prediction, confidence and variance.

Unfortunately, the distribution

$$P(\theta \mid D) = \frac{1}{Z} e^{-\sum_{i=1}^{N} L(y_i, g_\theta(x_i))} \prod_{i=1}^{N} P(x_i)\, P(\theta) \tag{24}$$

$$Z = \int e^{-\sum_{i=1}^{N} L(y_i, g_\theta(x_i))} \prod_{i=1}^{N} P(x_i)\, P(\theta)\; d\theta \tag{25}$$

usually cannot be simplified, and doesn't fit any well-known distribution. We cannot easily calculate the mean or the variance, and we can't straightforwardly draw samples the way we can for a normal distribution or a uniform distribution.

The one thing we *can* do is calculate the log posterior probability of a given set of weights θ:

$$\log P(\theta \mid D) = -\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \log P(\theta) + \sum_{i=1}^{N} \log P(x_i) - \log Z \tag{26}$$

Let's see how we bootstrap from there!

4.1 Maximum Likelihood Estimation

The important quantities we want to calculate are all statistics of the posterior predictive distribution $P(y' \mid x', D)$. Remember that the posterior predictive distribution is given by an expectation over θ:

$$P(y' \mid x', D) = \mathbb{E}_{P(\theta \mid D)}[P(y' \mid x', \theta)]\,. \tag{27}$$

We can approximate this numerically using sampling as in (6):

$$P(y' \mid x', D) \approx \frac{1}{M} \sum_{j=1}^{M} P(y' \mid x', \theta_j)\,, \quad \theta_j \overset{\text{idd}}{\sim} P(\theta \mid D)\,. \tag{28}$$

We can approximate it really badly using a single sample

$$P(y' \mid x', D) \approx P(y' \mid x', \theta)\,, \quad \theta \sim P(\theta \mid D) \tag{29}$$

and if we're only going to use one sample, it may as well be the *mode* $\theta^\star$, where

$$\theta^\star = \arg\max_\theta P(\theta \mid D) \tag{30}$$

Finding the mode is fairly straightforward. Since log is monotonic,

$$\theta^\star = \arg\max_\theta P(\theta \mid D) \tag{31}$$

$$= \arg\max_\theta \log P(\theta \mid D) \tag{32}$$

$$= \arg\max_\theta \left[-\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \log P(\theta) + \sum_{i=1}^{N} \log P(x_i) - \log Z \right]. \tag{33}$$

Because addition and subtraction of constants is monotonic, we can drop the terms that don't vary with θ, leaving us with

$$\theta^\star = \arg\max_\theta \left[-\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \log P(\theta) \right] \tag{34}$$

$$= \arg\min_\theta \left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) - \log P(\theta) \right] \tag{35}$$

Notice how similar this is to the Frequentist optimization objective (14). The only difference is we've added a term $-\log P(\theta)$ saying that the weights shouldn't stray too far from the prior. In practice we often set the prior to $\theta \sim N(0, 1)$, in which case $\log P(\theta) \propto -\|\theta\|_2^2$:

$$\theta^\star = \arg\min_\theta \left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \|\theta\|_2^2 \right] \tag{36}$$

i.e. Bayesian maximum likelihood estimation with a Gaussian prior on the weights is equivalent to Frequentist learning with L2 regularization.

We might also assume an *improper* uninformative prior on the weights, $P(\theta) \propto 1$. In that case the prior term also becomes constant in θ and Bayesian maximum likelihood estimation becomes exactly equivalent to Frequentist learning:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{N} L(y_i, g_\theta(x_i)) \tag{37}$$

Once we have our point estimate of the posterior $\theta^\star$, we can write

$$P(y' \mid x', D) \approx P(y' \mid x', \theta^\star) \tag{38}$$

and read off statistics like the mean, variance and confidence from the model distribution.

4.2 Variational Inference

Okay, that's a neat trick. But if all we wanted to do was get a point estimate for the mode of the posterior, we may as well have stayed in the Frequentist paradigm and not learned all these statistics. We want more!

Bayesian machine learning would be easy if the posterior distribution had a convenient closed form like $N(\mu, \sigma^2)$. What if we just... pretended that it did?

In variational inference, we decide on a *variational distribution* $Q_\varphi(\theta)$ with a convenient closed form, and the goal becomes to find parameters φ for this distribution such that $Q_\varphi(\theta)$ is as close as possible to $P(\theta \mid D)$. Usually we do this by minimizing the Kullback-Liebler divergence (the KL divergence):

$$\mathrm{KL}(Q \,\|\, P) = \int Q(x) \log \frac{Q(x)}{P(x)}\, dx \tag{39}$$

Why the KL divergence in particular? There is a lot of probability theory that backs the KL as the sensible measure of divergence to minimize, but really we pick it because it generates a convenient optimization problem.

Recall from (5) that any integral over a probability distribution is an expected value:

$$\mathrm{KL}(Q \,\|\, P) = \mathbb{E}_{Q(x)}[\log Q(x) - \log P(x)] \tag{40}$$

Specializing that to our setting we get

$$\mathrm{KL}(Q_\varphi(\theta) \,\|\, P(\theta \mid D)) = \mathbb{E}_{Q_\varphi(\theta)}[\log Q_\varphi(\theta) - \log P(\theta \mid D)]\,. \tag{41}$$

$$= \mathbb{E}_{Q_\varphi(\theta)}\left[\log Q_\varphi(\theta) + \sum_{i=1}^{N} L(y_i, g_\theta(x_i)) - \log P(\theta) - \sum_{i=1}^{N} \log P(x_i) + \log Z\right] \tag{42}$$

We want to find $\varphi^\star$ that minimizes the KL divergence:

$$\varphi^\star = \arg\min_\varphi \mathrm{KL}(Q_\varphi(\theta) \,\|\, P(\theta \mid D)) \tag{43}$$

A neat property of expectation $\mathbb{E}_{P(\cdot)}[\cdot]$ is that it distributes over addition, so we can write

$$\begin{aligned}
\mathrm{KL}(Q_\varphi(\theta) \,\|\, P(\theta \mid D)) = {} & \mathbb{E}_{Q_\varphi(\theta)}[\log Q_\varphi(\theta)] \\
& + \mathbb{E}_{Q_\varphi(\theta)}\left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i))\right] \\
& - \mathbb{E}_{Q_\varphi(\theta)}[\log P(\theta)] \\
& - \mathbb{E}_{Q_\varphi(\theta)}\left[\sum_{i=1}^{N} \log P(x_i)\right] \\
& + \mathbb{E}_{Q_\varphi(\theta)}[\log Z]
\end{aligned} \tag{44}$$

For optimization purposes we can ignore the terms that are constant in φ, i.e. the expectations that do not contain a θ term. After rearranging, that gives us

$$\varphi^\star = \arg\min_\varphi \left[\mathbb{E}_{Q_\varphi(\theta)}\left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) - \log P(\theta) + \log Q_\varphi(\theta)\right]\right] \tag{45}$$

We get to choose the form of the variational distribution $Q_\varphi(\theta)$ such that it's easy to sample from (indeed, that was the whole point). We can therefore easily approximate the expectation numerically (6):

$$\varphi^\star = \arg\min_\varphi \left[\frac{1}{M} \sum_{j=1}^{M} \left[\sum_{i=1}^{N} L(y_i, g_{\theta_j}(x_i)) - \log P(\theta_j) + \log Q_\varphi(\theta_j) \right] \right], \quad (46)$$

with $\theta_j \overset{\text{iid}}{\sim} Q_\varphi(\theta)$. How big should the number of samples M be? In theory M should be large, so we can get the best possible estimate of the KL divergence. But for each sample θ_i we have to do a forward pass of the model for every example in our training data to calculate the sum loss $\sum_{i=1}^{N} L(y_i, g_{\theta_j}(x_i))$. That takes a lot of time, and time is money. So in practice we set $M = 1$, giving us

$$\varphi^\star = \arg\min_\varphi \left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) - \log P(\theta) + \log Q_\varphi(\theta) \right] \quad (47)$$

with $\theta \sim Q_\varphi(\theta)$. Now that's an objective we can optimize by gradient descent!

Notice how similar (47) is to the Frequentist optimization objective (14) and maximum likelihood estimation (35). Maximum likelihood estimation selects $\theta^\star$ to minimize the loss, while adding a term $-\log P(\theta)$ representing how far the parameters have diverged from the prior. If we rearrange (47) slightly we get

$$\varphi^\star = \arg\min_\varphi \left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \mathbb{E}_{Q_\varphi(\theta)}[\log Q_\varphi(\theta) - \log P(\theta)] \right] \quad (48)$$

$$\varphi^\star = \arg\min_\varphi \left[\sum_{i=1}^{N} L(y_i, g_\theta(x_i)) + \mathrm{KL}(Q_\varphi(\theta) \parallel P(\theta)) \right]. \quad (49)$$

i.e. variational inference selects $\varphi^\star$ to minimize the expected loss while also minimizing the KL-divergence of the variational distribution $Q_\varphi(\theta)$ from the prior $P(\theta)$.

Variational inference is very powerful! Once we have $Q_{\varphi^\star}(\theta) \approx P(\theta \mid D)$, we can use it to generate samples from the posterior:

$$y' \sim P(y' \mid x', D) = y' \sim P(y' \mid x', \theta), \quad \theta \sim P(\theta \mid D) \quad (50)$$

$$\approx y'_j \sim P(y' \mid x', \theta), \quad \theta \sim Q_{\varphi^\star}(\theta) \quad (51)$$

We can use these samples to easily calculate the mean and variance of the prediction. We can also easily calculate the confidence:

$$P(y' \mid x', D) \approx \mathbb{E}_{Q_{\varphi^\star}(\theta)}[P(y' \mid x', \theta)]. \quad (52)$$

This is much better than maxmimum likelihood estimation, which only gives us gives us access to the posterior mode.

Variational inference is a great technique. It's high-performance, and a straight upgrade on Frequentist learning. But it requires making strong assumptions about the form of the posterior.

For instance, we often assume a Gaussian prior on the weights, because it's computationally convenient. Many machine learning libraries already set the default weights using a Gaussian distribution. We might expect that the posterior has the same form as the prior, and set our variational distribution $Q_\varphi(\theta) = N(\varphi_0, \varphi_1)$. This is very tempting, but hamstrings our analysis. Gaussian distributions are *unimodal* - they only have one peak. Remember ages ago we said there there might be many different functions compatible with your training data? A Gaussian distribution can only concentrate in one place; it can't concentrate at two meaningfully different functions. Much of the work in modern variational inference is in defining expressive variational families that can capture all the nuances of the true posterior.

4.3 MCMC / Posterior Sampling

What should you do if variational inference isn't accurate enough for you?

If you really want direct samples from the posterior, there is a simple algorithm that can get you them. Notice that

$$P(\theta \mid D) = \frac{P(\theta \mid D)}{P(\theta)} P(\theta)\,. \tag{53}$$

i.e. we can get from one distribution to the other by relative reweighting. That gives rise to an algorithm called *importance sampling*:

1. Sample $\{\theta_j\}_{j=1}^N$ from the prior $P(\theta)$
2. For each sample θ_j:
   (a) Calculate $q_j = P(\theta)$
   (b) Calculate $p_j = P(\theta = \theta_j \mid D)$
3. Sample θ from $\{\theta_j\}_{j=1}^N$ according to the relative weights $\frac{p_j}{q_j}$.

This algorithm *sucks*. You usually have to sample from the prior many, many times before you find a part of weight space where the posterior concentrates, and calculating $P(\theta \mid D)$ is difficult and expensive.

Fortunately we can improve this algorithm by turning our random samples into a directed search, seeking areas of low loss / high posterior concentration. This gives rise to a family of methods called Markov Chain Monte Carlo or MCMC. The simplest such algorithm is Metropolis-Hastings.

In Metropolis-Hastings, we pick a proposal distribution $P(\theta' \mid \theta)$ which can be anything we like, and so is usually a Gaussian $N(\theta, \sigma^2)$. Then we:

1. Pick an initial parameter θ_0 from the prior $P(\theta)$. Set $j = 0$.
2. Pick a new parameter $\theta_?$ from the proposal distribution $P(\theta_? \mid \theta_j)$.
3. Calculate the acceptance ratio

$$\alpha = \frac{P(\theta = \theta_? \mid D)}{P(\theta = \theta_j \mid D)} \tag{54}$$

4. With probability α, let $\theta_{j+1} = \theta_?$, jumping to the proposed parameter. Otherwise, let $\theta_{j+1} = \theta_j$, remaining where you are.

5. Set $j \leftarrow j + 1$, go to 2.

It can be shown (and we might, if we have time, show it) that the process $(\theta_0, \theta_1, \dots)$ is a Markov chain with limit distribution $P(\theta \mid D)$. In computer science, a Monte Carlo algorithm has fixed running time but gives random answers (compared to a Las Vegas algorithm which gives correct answers but whose running time is random). Hence the name "Markov Chain Monte Carlo".

Metropolis-Hastings makes two big improvements on importance sampling: it converges much faster, and it only requires us to compute the *ratio* of probability densities of parameters:

$$\log \frac{P(\theta = \theta_? \mid D)}{P(\theta = \theta_j \mid D)} = \log P(\theta = \theta_? \mid D) - \log P(\theta = \theta_j \mid D) \tag{55}$$

$$= -\sum_{i=1}^{N} L(y_i, g_{\theta_?}(x_i)) + \log P(\theta_?) + \cancel{\sum_{i=1}^{N} \log P(x_i)} - \cancel{\log Z}$$

$$+ \sum_{i=1}^{N} L(y_i, g_{\theta_j}(x_i)) - \log P(\theta_i) - \cancel{\sum_{i=1}^{N} \log P(x_i)} + \cancel{\log Z} \tag{56}$$

$$= \log \frac{P(\theta_?)}{P(\theta_i)} - \sum_{i=1}^{N} \big[L(y_i, g_{\theta_?}(x_i)) - L(y_i, g_{\theta_j}(x_i)) \big] \tag{57}$$

$$= \log \frac{P(\theta_?)}{P(\theta_i)} - N \times \mathbb{E}_{P(x,y)} \big[L(y, g_{\theta_?}(x)) - L(y, g_{\theta_j}(x)) \big] \tag{58}$$

Notice that this lets us avoid computing Z, an intractable integral, and $P(x_i)$, the unobservable true data generating distribution. All that remains is the prior log odds ratio and the expected difference in loss, both of which are easy and cheap.

In exchange for being feasible in practice, Metropolis-Hastings gives us one drawback: correlation. We sample by taking a random walk through parameter space, meaning that adjacent samples from the chain are highly correlated. We get around this using *burn in* and *subsampling*. To get representative samples from the posterior:

- Sample θ_0 from the prior $P(\theta)$.
- *Burn in* the chain $(\theta_0, \theta_1, \dots)$ by generating and throwing away thousands of samples. This allows the search process to find a region of high posterior concentration.
- Let, say, θ_{10000} be the first posterior sample.
- Generate and throw away another thousand samples from the chain. This diminishes the correlation between posterior samples.

- Let θ_{11000} by the next posterior sample.
- Lather, rinse, repeat as needed.

Research in MCMC sampling mostly revolves around better choices for the proposal distribution. We want the search process to seek areas of high posterior concentration, and the posterior concentrates near parameters with low loss. Hence, we can use the gradient of the loss to tell us in which direction we should step. Using Hamiltonian dynamics gives us Hamiltonian Monte Carlo or HMC. This is the **gold standard** in posterior sampling.

5 Concluding thoughts

There are two reasons you might do Bayesian machine learning.

1. Bayesian machine learning is a family of techniques aimed at *making richer predictions*. Point estimates aren't enough for you; you have some business reason for getting well-calibrated uncertainty estimates. Perhaps you're doing online learning, and you want to target data collection to areas where the model is uncertain. Bayesian machine learning is more expensive, but the benefits are worth the costs.
2. Bayes is good, you should do a Bayes.

The second reason is actually more sensible than you'd think. There is a sense in which Bayesian machine learning is "the real machine learning". Nobody knew why L2 regularization worked until Bayesians rephrased it as placing a Gaussian prior on the weights, and that rephrasing opened a rich vein of theory. The central question in ML research is why neural networks generalize well, and the most compelling answers to that question, like SLT, have their roots in Bayesian statistics.

A lot of learning is knowing that there are things to be known. I hope having read this document, the next time you encounter a phrase like *variational family* or *MCMC with NUTS sampling*, you think back to this session, remember the shape of the ideas, and know that you could learn the details.