Gradient-based Stochastic Variational Inference

Related Reading

- Murphy: Chapter 18

Overview

- Basics of Variational Inference
- Deriving the evidence lower bound (ELBO)
- Properties of the ELBO
- Estimating gradients of the ELBO
    - Simple Monte Carlo
    - Reparameterization trick

Posterior Inference for Latent Variable Models

So far in this course we've worked with a few latent variable models, such as the generative image model and the trueskill model.

These models have a factorization $p(x, z) = p(z)p(x|z)$ where

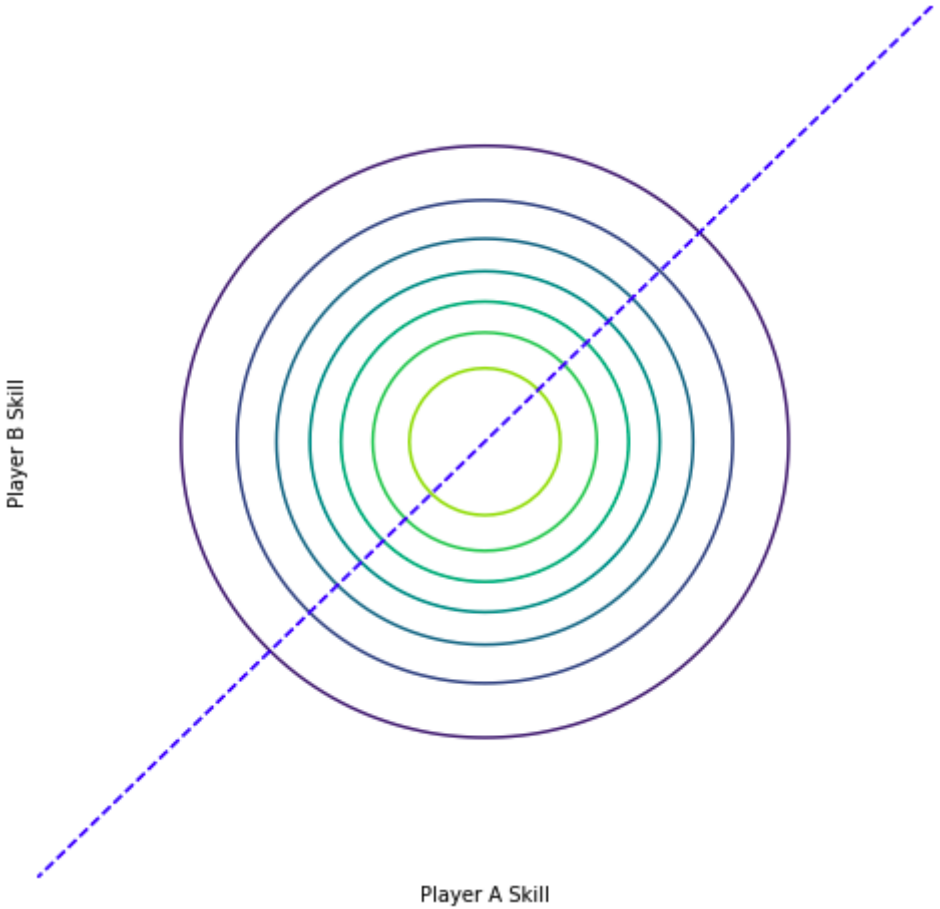
- x are the observations or data,
- z are the unobserved (latent) variables
- $p(z)$ is usually called the *prior*
- $p(x|z)$ is usually called the *likelihood*

The conditional distribution of the unobserved variables given the observed variables (aka the *posterior*) is

$$p(z|x) = \frac{p(x|z)}{p(x)} = \frac{p(x|z)}{\int p(x, z)dz}$$

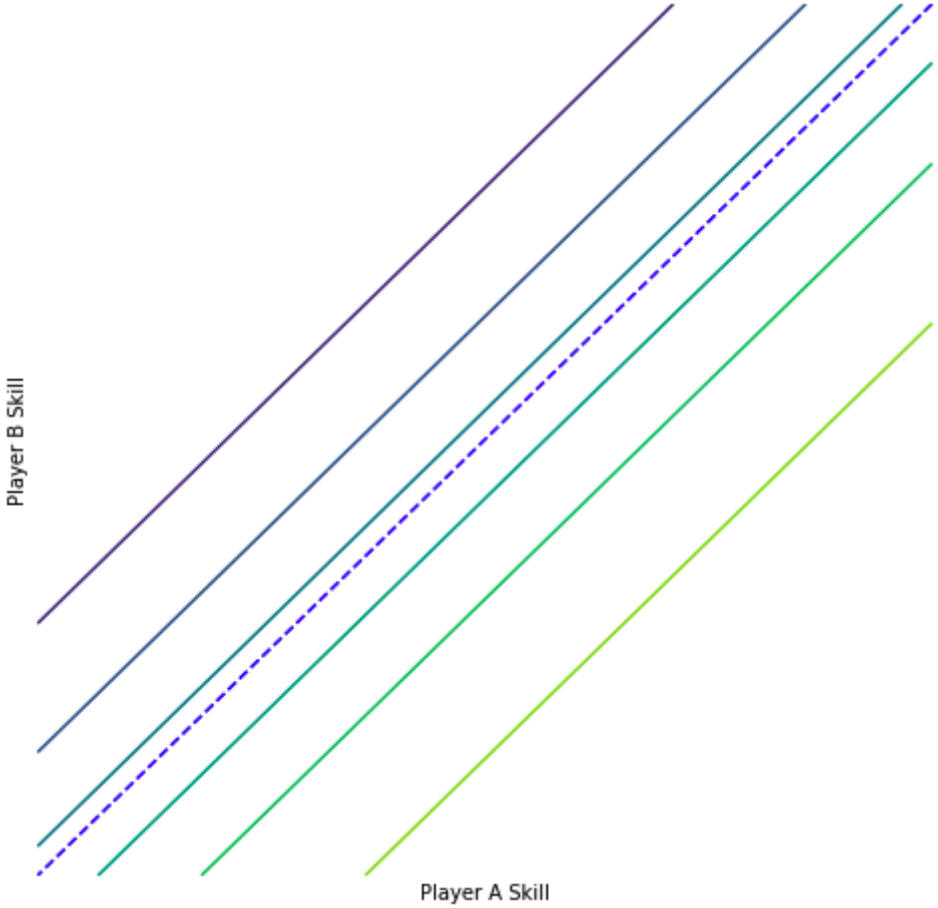
Concrete example: Prior, likelihood and posterior in Trueskill

Prior:



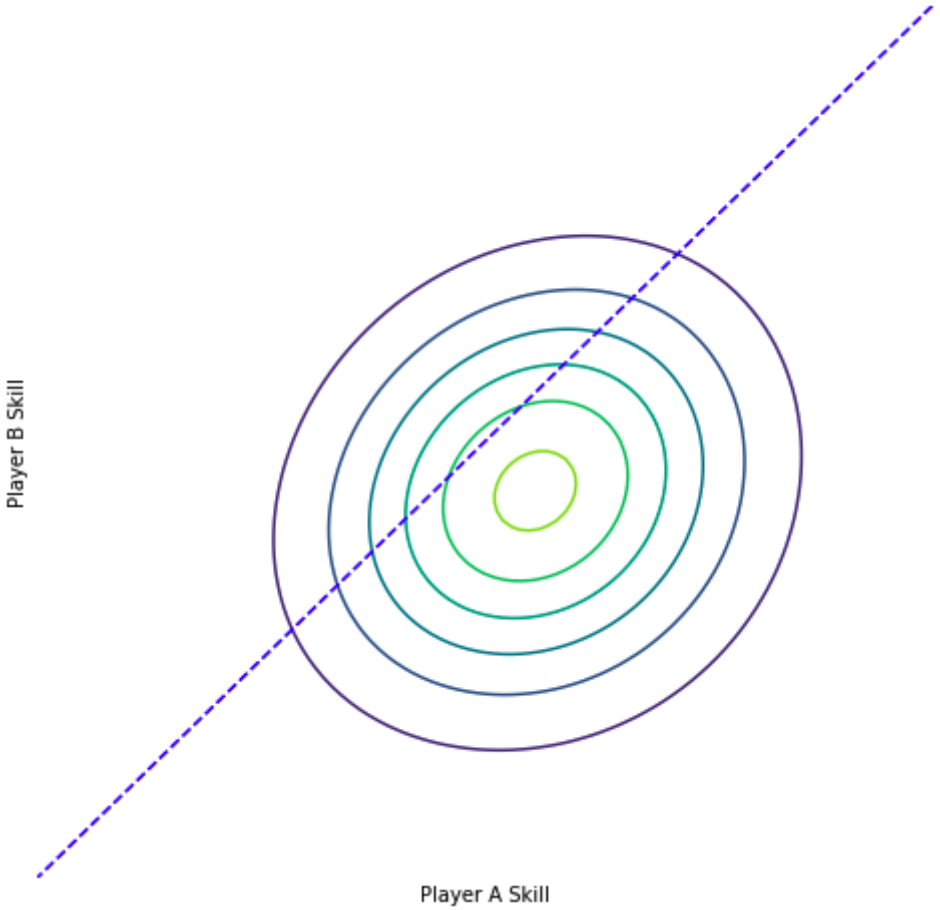
Says we're very uncertain about both player's skill.

Likelihood:



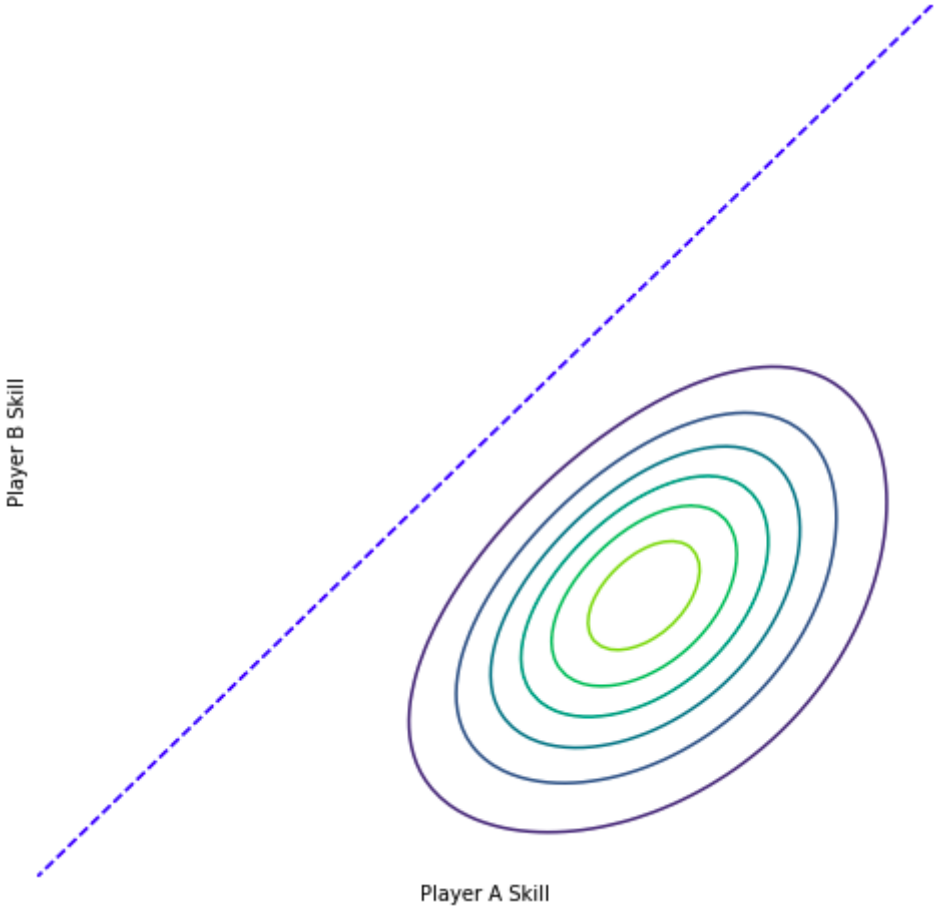
This is the part of the model that gives meaning to the latent variables. For example, if we reversed the sign on the skills inside the likelihood, a lower skill would be better. If we want to include multiple types of skill per player, we'd have to decide what that meant in terms of the probability of winning as a function of both player's sets of skills. We could also learn this function!

Posterior:



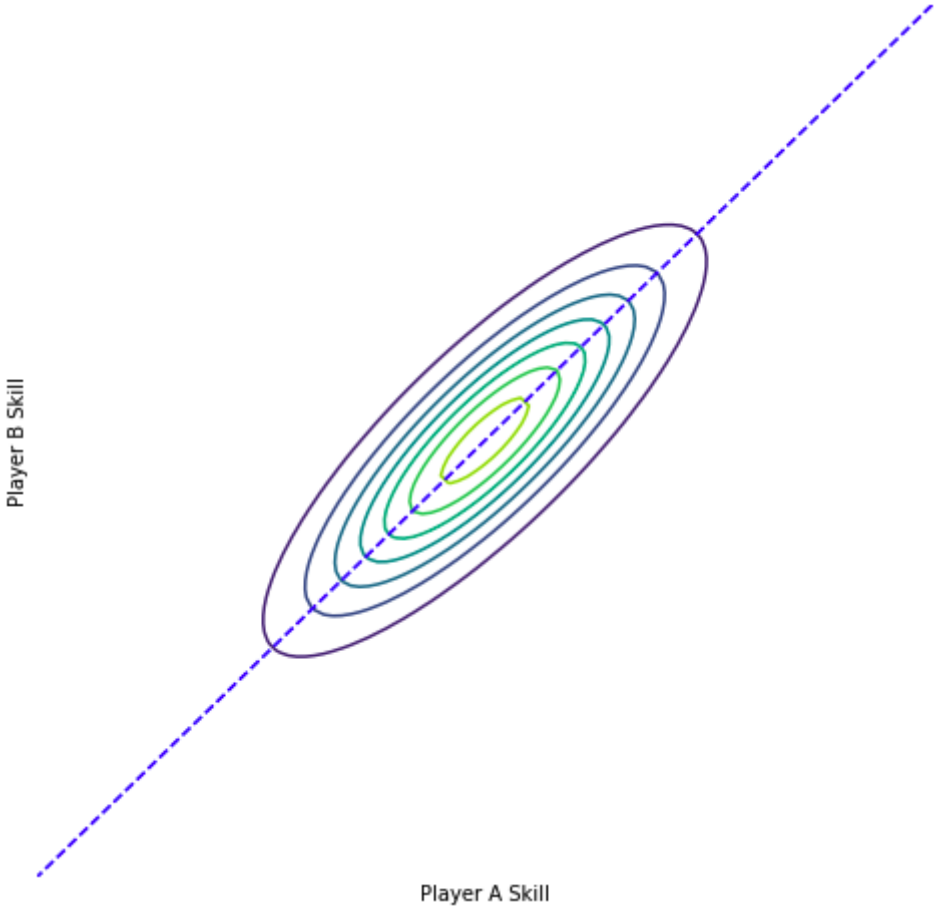
The posterior isn't Gaussian anymore.

Posterior after A beats B 10 times:



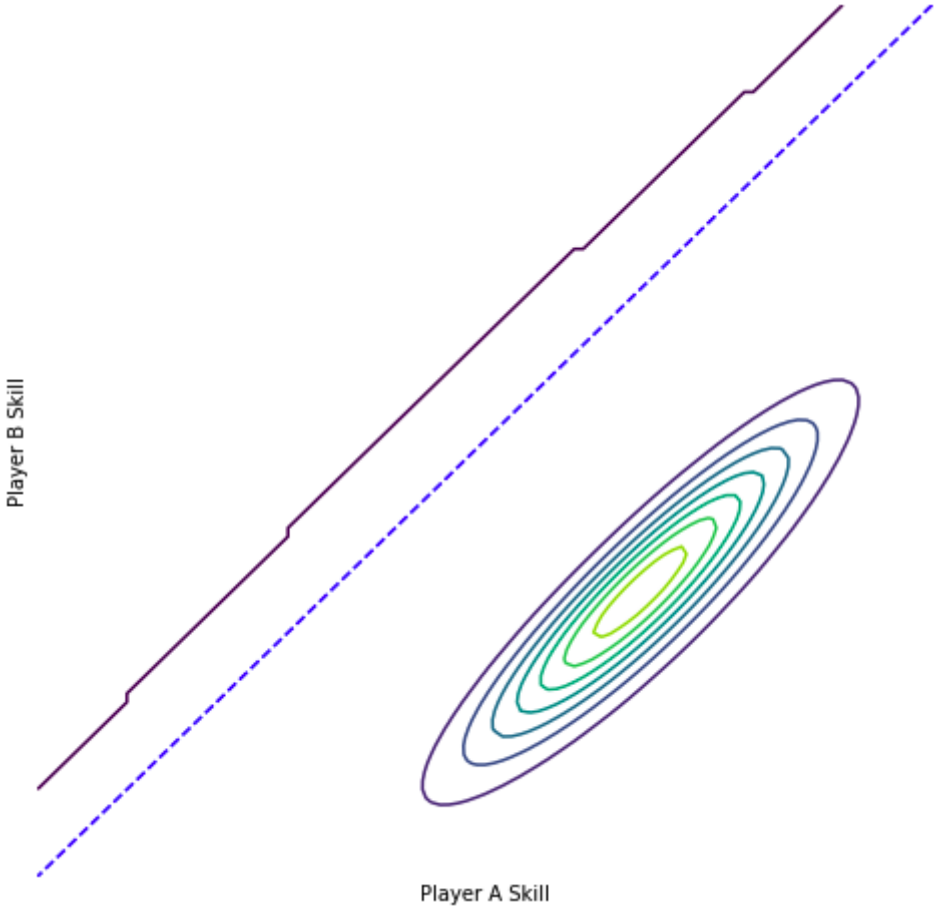
Now the posterior is certain that A is better than B.

Posterior after both beat each other 10 times:



Now the posterior is certain that neither player is much better than the other, but is uncertain how good they both are in an absolute sense.

Posterior after 90 wins vs 10:



In general, the more evidence we have, the more the posterior will shrink.

What is hard to compute about the posterior?

The integral $p(x) = \int p(x, z)dz$ is intractable whenever z is high dimensional. This makes evaluating or sampling from the normalized posterior $p(z|x)$ for a given x and z also intractable. This is a problem, because many of the queries we'd like to make of our model require these abilites, and tuning the parameters of a model requires computing $p(x)$.

Here is a list of operations that are cheap (don't require integrating over all settings of z):

1. Sampling $z, x \sim p(z, x)$: Simply sample $z \sim p(z)$, then $x \sim p(x|z)$, and return z and x.
2. Sampling $x \sim p(x)$: Simply sample $z \sim p(z)$, then $x \sim p(x|z)$, and return x.
    - This is useful for sanity checking your model to see if the data it generates looks vaguely like real data (we did this in HW1 with the sampled handwritten digits)
3. Computing a likelihood ratio: $\frac{p(x|z_1)}{p(x|z_2)}$
    - This is useful if we only have two specific hypotheses in mind that we want to compare.
4. Computing a joint probability ratio: $\frac{p(z_1,x)}{p(z_2,x)} = \frac{p(z_1)p(x|z_1)}{p(z_2)p(x|z_2)}$
5. Computing a posterior ratio: $\frac{p(z_1|x)}{p(z_2|x)} = \frac{\frac{p(z_1)p(x|z_1)}{p(x)}}{\frac{p(z_2)p(x|z_2)}{p(x)}} = \frac{p(z_1)p(x|z_1)}{p(z_2)p(x|z_2)}$
    - Same as a likelihood ratio but takes into account the prior $p(z)$.

Here is a list of operations that are expensive (require integrating over all settings of z, or something equally hard):

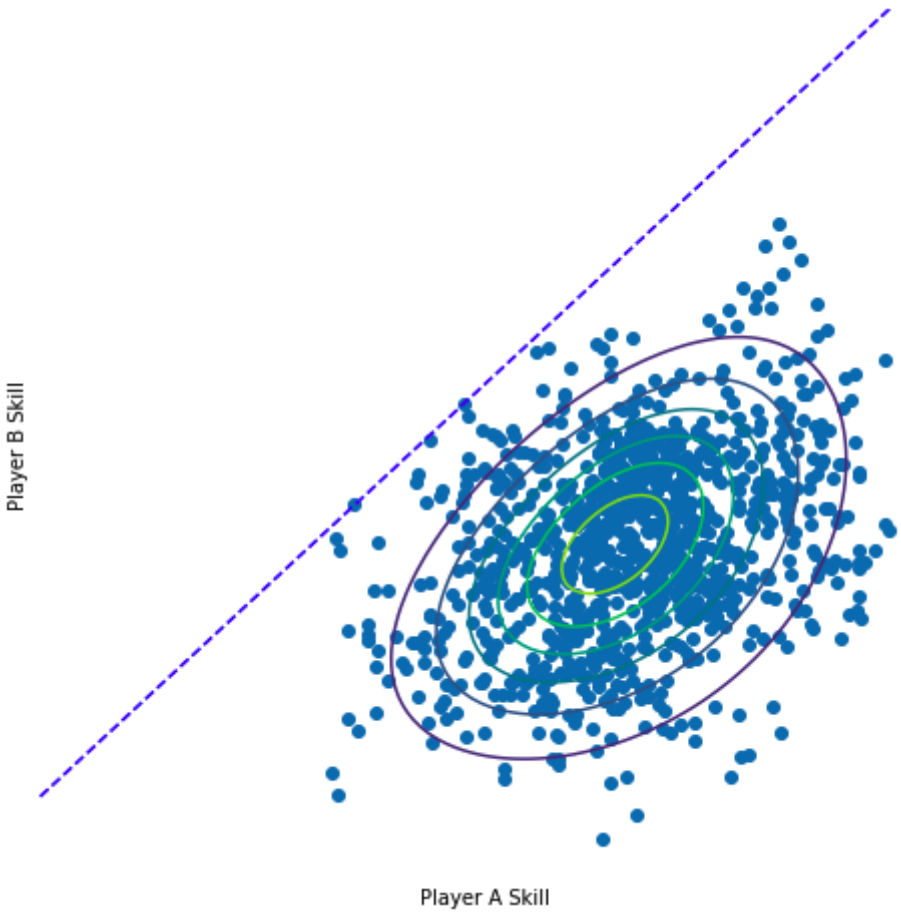
1. Computing a posterior probability: $p(z|x) = \frac{p(z)p(x)}{p(x)}$
2. Computing the evidence / marginal likelihood $p(x) = \int p(z, x) dz$
    - Useful for choosing between models, or fitting model parameters.
3. Computing marginals of $p(z1|x) = \int p(z1, z2, \dots z_D|x) dz_2, dz_3, \dots dz_D$
    - E.g. finding the posterior over a single tennis player's skill given all games.
4. Sampling $z \sim p(z|x)$
    - Useful for summarizing which hypotheses are likely given the data, making predictions, and decisions.
    - E.g. estimating a single player's skill using simple Monte Carlo:

$$\mathbb{E}_{p(z_1|x)}\left[z_1\right] \approxeq \frac{1}{N} \sum_{i=1}^{N} z_i \quad \text{where each} z_i \sim p(z_1|x)$$

Note that as soon as we write down the prior $p(z)$ and the likelihood $p(x|z)$, we've also defined the posterior $p(z|x)$. It exists, but we just don't know what it is until we compute $p(x)$.

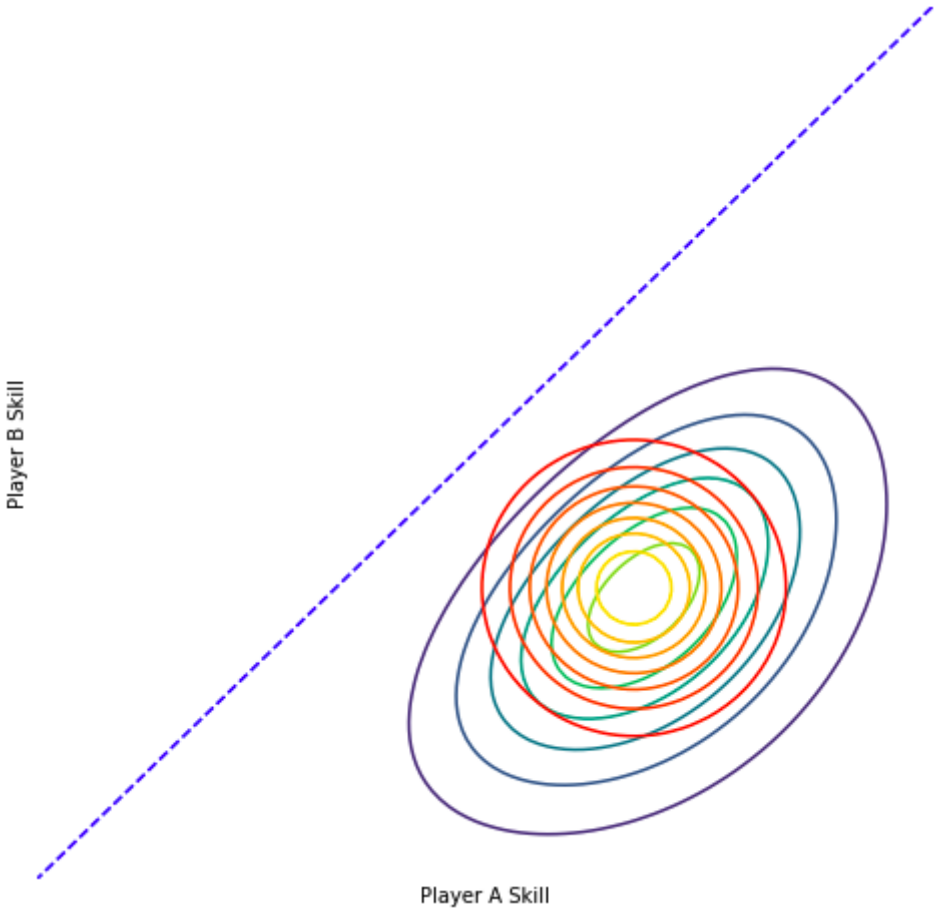
Approximating the Posterior with another, more tractable, distribution.

Last week, we learned how to approximately draw samples from the true posterior using MCMC:



Samples drawn from running HMC on the unnormalized posterior.

This week, we'll learn an alternative family of methods called *variational inference*. This approach finds another distribution that approximates the true posterior, one which we can cheaply sample from exactly:

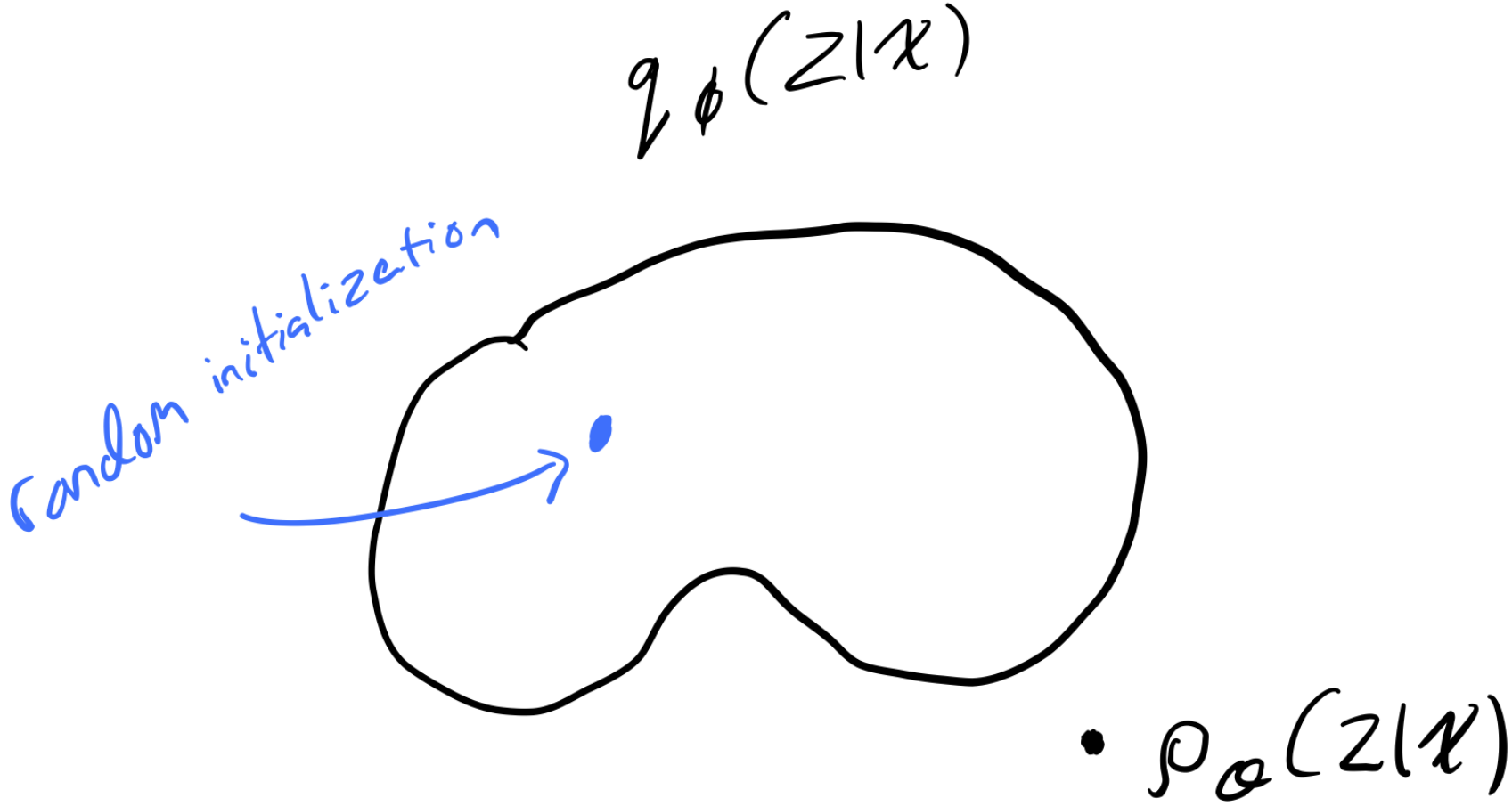


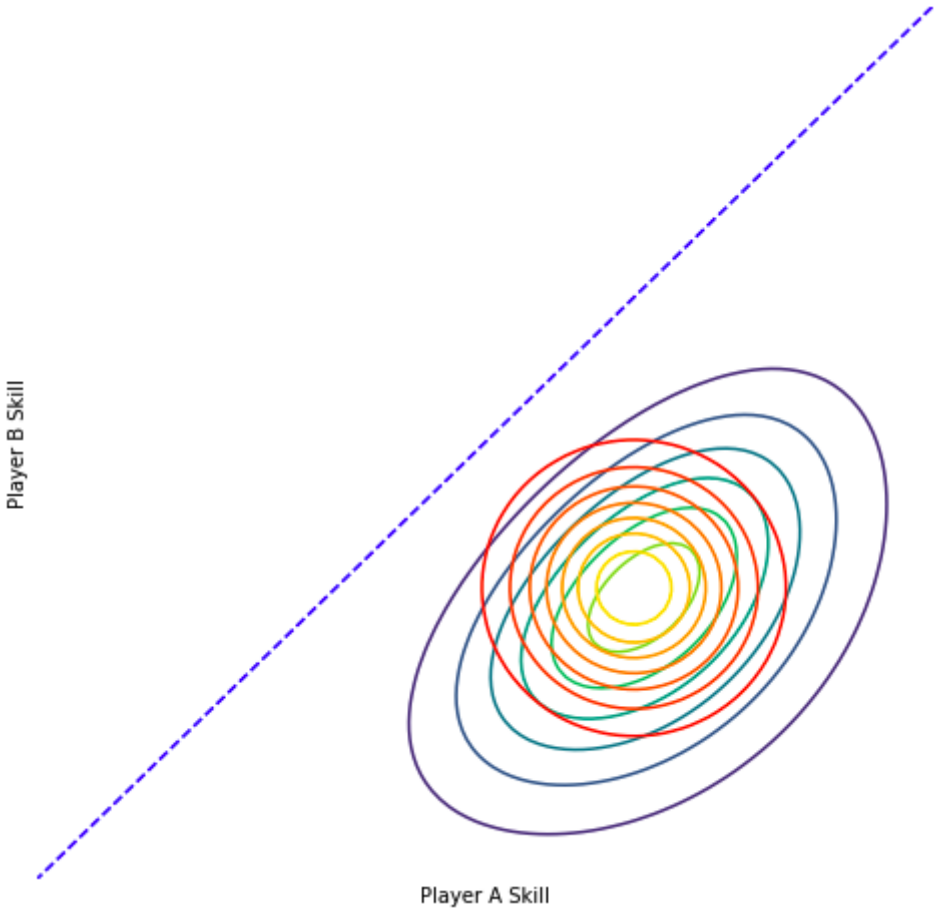
A factorized Gaussian fit to approximate the true posterior.

So: MCMC approximately samples from the true posterior, and variational inference exactly samples from an approximate posterior.

To be more formal, variational inference works as follows:

1. Choose a tractable parametric distribution $q_\phi(z)$ with parameters ϕ. This distribution will be used to approximate $p(z|x)$. For example, $q_\phi(z) = \mathcal{N}(z|\mu, \Sigma)$, where $\phi = (\mu, \Sigma)$. The idea is that we'll try choose a ϕ that makes $q_\phi(z)$ a good approximation of the true posterior $p(z|x)$.
2. Encode some notion of "difference" between $p(z|x)$ and q_ϕ that can be effciently estimated. Usually we will use the KL divergence.
3. Minimize this difference. Usually we will use an iterative optimization method like stochastic gradient descent.


$q_\phi(z|x)$
random initialization
$p_\theta(z|x)$



Whatever parametric distribution we choose for $q(z)_\phi$, it's usually not the case that there is any setting of ϕ that exactly matches the true posterior $p(z|x)$. But this isn't as bad as it sounds:

- Computing the true posterior is intractable, so we have to take a shortcut somewhere.
- Once we find a good ϕ, if we have time left over, we can start again with a more complicated parametric distribution and get a closer fit.

Kullback-Leibler Divergence

We will measure the difference between q_ϕ and p using the **Kullback-Leibler divergence**:

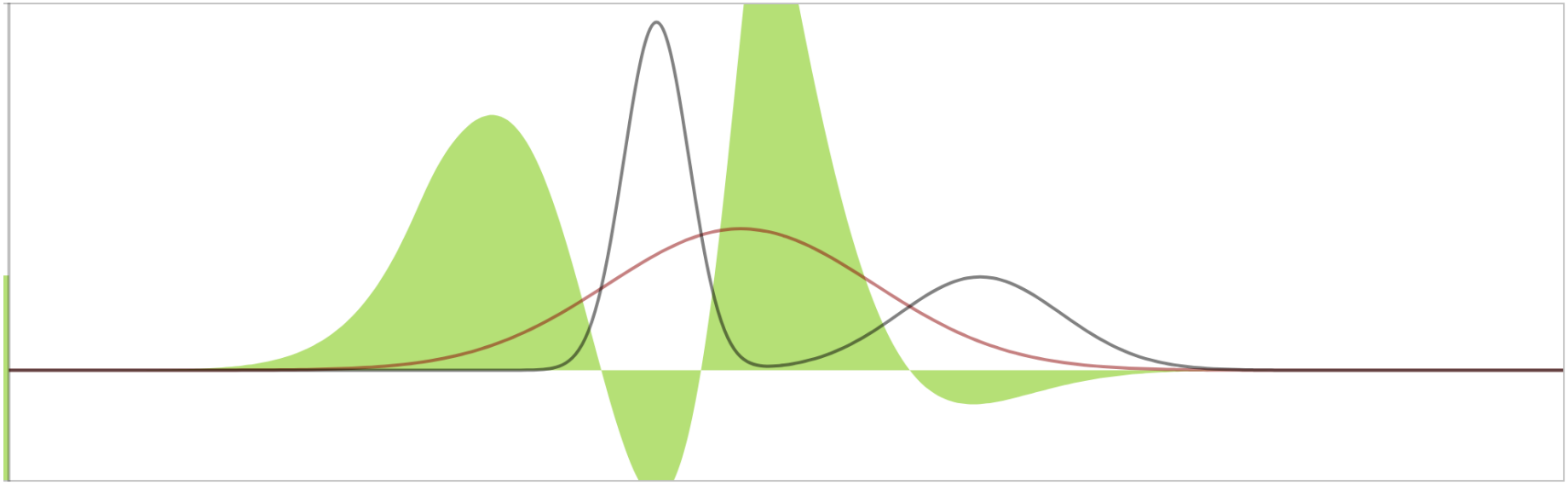
$$\begin{aligned} KL(q_\phi(z)||p(z|x)) &= \int q_\phi(z) \log \frac{q_\phi(z)}{p(z|x)} dz \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z)}{p(z|x)} \end{aligned}$$

Properties of the KL Divergence

1. $KL(q_\phi||p) \geq 0$
2. $KL(q_\phi||p) = 0 \Leftrightarrow q_\phi = p$
3. $KL(q_\phi||p) \neq KL(p||q_\phi)$

The last property means that KL is *not* a distance, since it's not symmetric.

You can play with a widget on this Bayesian Neural Net demo to get a feel for the KL:



The evidence lower bound

We want to approximate $p(z|x)$ by finding a q_ϕ such that $q_\phi(z) \approx p(z|x)$. $Eqactequivalentwillbeacheivedwhen KL(q_\phi||p(z|x)) = 0$.

Exactly evaluating $KL(q_\phi(z)||p(z|x))$ is intractable because of the integral over z. But it's even intractable to approximate with simple Monte Carlo, because it contains the term $p(z|x)$, which we have already established, is intractable to normalize.

It turns out we can still measure optimize this KL without knowing the normalization constant $p(x)$. There is a related quantity called the **evidence lower bound (ELBO)**. Maximizing the ELBO is the same as minimizing $KL(q_\phi||p(z|x))$.

$$
\begin{aligned}
KL(q_\phi(z)||p(z|x)) &= \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z)}{p(z|x)} \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi} \left[\log \left(q_\phi(z) \cdot \frac{p(x)}{p(z,x)} \right) \right] \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z)}{p(z,x)} + \mathop{\mathbb{E}}_{z\sim q_\phi} \log p(x) \\
&= -\mathcal{L}(\phi) + \log p(x)
\end{aligned}
$$

Where $\mathcal{L}(\phi)$ is the **ELBO**.

Rearranging, we get $$ \begin{aligned} KL(q_\phi (z) || p(z | x)) &= -\mathcal L(\phi) + \log p(x) \\ \Rightarrow \mathcal L(\phi) + KL(q_\phi (z) || p(z | x)) &= \log p(x) \\ \end{aligned} $$

Because $KL(q_\phi(z)||p(z|x)) \geq 0$,

$$\mathcal{L}(\phi) \leq \log p(x)$$

$\therefore$ maximizing the ELBO $\Rightarrow$ minimizing $KL(q_\phi(z|x)||p(z|x))$.

Optimizing the ELBO

Now we know that maximizing the ELBO wrt ϕ will mean finding a better approximation to the true posterior, as measured by the KL.

The remaing part of this lecture is about how to compute unbiased estimates of the gradient of the ELBO wrt ϕ, so that we can use stochastic gradient descent.

We have that

$$\begin{aligned}\mathcal{L}(\phi) &= -\mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z|x)}{p(x,z)} \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi}\Big[\log p(x|z) - \log q_\phi(z|x)\Big]\end{aligned}$$

If we want to optimize this with gradient methods, we will need to compute an unbiased estimate of $\nabla_\phi \mathcal{L}(\phi)$.

The reparameterization trick Nowadays, we have [automatic differentiation (AD)](https://en.wikipedia.org/wiki/Automatic_differentiation). However, there is a bit of subtle to differentiating through a stochastic estimator.

We can use autodiff to compute unbiased gradients automatically if:

1. $p(x, z)$ is differentiable wrt z. Can have discrete x such as text, no problem.
2. We need to sample $z \sim q_\phi(z)$ to estimate the ELBO with simple Monte Carlo. We need to break this sampling process into two parts:
    1. Sample a random variable ϵ that has fixed (or no) parameters, such as a uniform distribution or standard normal.
    2. Determinsitically compute z's as a function ϕ and the noise ϵ, such that:
        1. $\epsilon \sim p(\epsilon)$
        2. $z = T(\epsilon, \phi)$
        3. implies
        4. $z \sim q_phi(z)$

A simple example is that you can sample from a $\mathcal{N}(\mu, \sigma)$ by:

1. $\epsilon \sim \mathcal{N}(0, 1)$
2. $z = \mu + \epsilon\sigma$
3. implies
4. $z \sim \mathcal{N}(\mu, \sigma)$

```
### Define variational family by a reparameterized sampler,
### and a matching normalized density.

def diag_gaussian_sample(rng_key, mean, std):
  return mean + std * random.normal(rng_key, mean.shape)

def diag_gaussian_logpdf(z, mean, std):
  return np.sum(norm.logpdf(z, mean, std), axis=-1)
```

Applying this transformation to the ELBO lets us move the gradient inside the expectation. We want an expectation on the outside wrt q, because this will let us use simple Monte Carlo:

$$
\begin{aligned}
\nabla_\phi \mathcal{L}(\phi) &= \nabla_\phi \mathbb{E}_{z \sim q_\phi(z|x)} \Big[\log p(x, z) - \log q_\phi(z) \Big] \\
&= \nabla_\phi \mathbb{E}_{\epsilon \sim p(\epsilon)} \Big[\log p(x, T(\phi, \epsilon)) - \log q_\phi(T(\phi, \epsilon)) \Big] \\
&= \mathbb{E}_{\epsilon \sim p(\epsilon)} \nabla_\phi \Big[\log p(x, T(\phi, \epsilon)) - \log q_\phi(T(\phi, \epsilon)) \Big]
\end{aligned}
$$

Code for gradient-based stochastic variational inference:

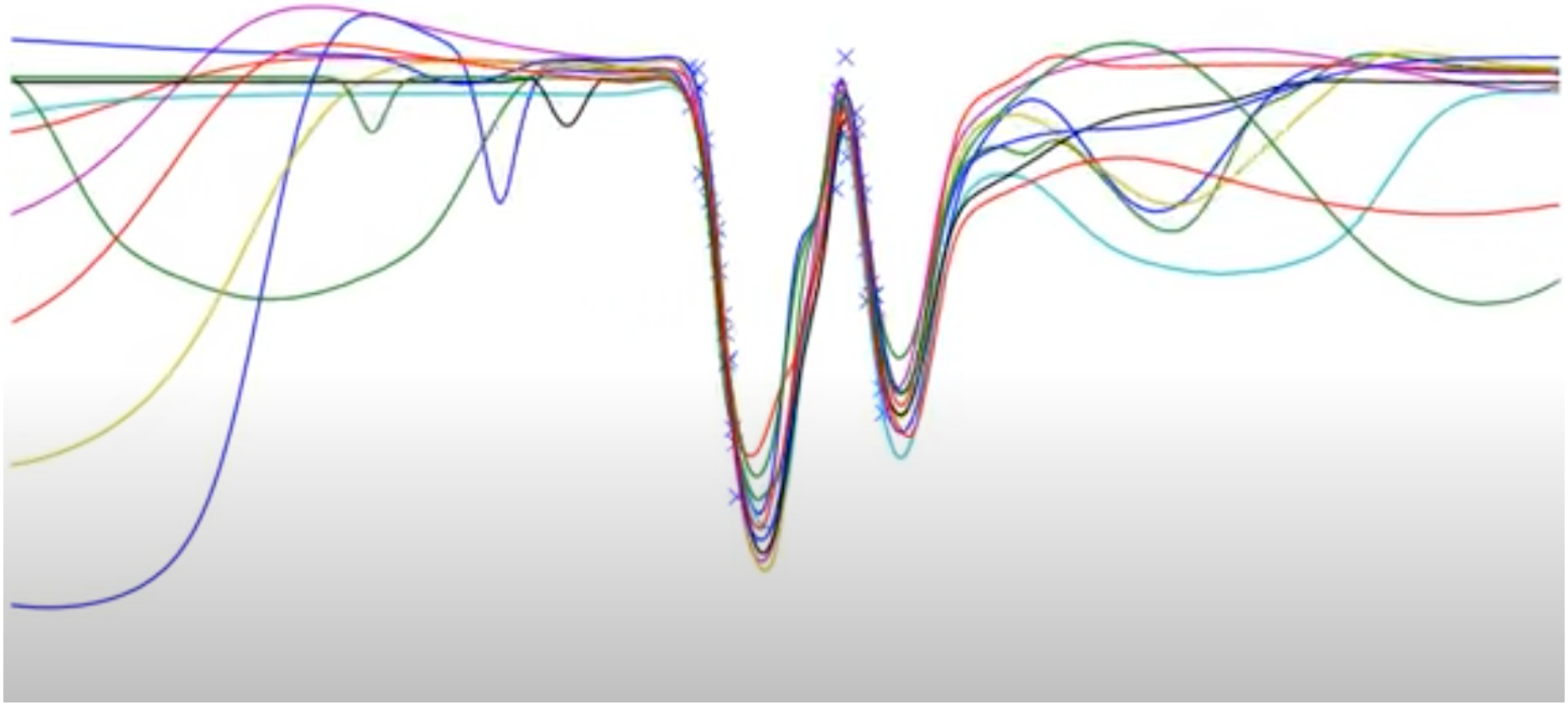
When using, for example, a Gaussian variational approximate posterior, the JAX code is very simple:

```
## Define unbiased estimate of ELBO and its gradient wrt
## parameters of variational distribution.

def elbo(phi, log_p_x_and_z, rng_key):
  mean, std = phi
  z_sample = diag_gaussian_sample(rng_key, mean, std)
  return log_p_x_and_z(sample) - diag_gaussian_logpdf(z_sample, mean, std)

grad_elbo_estimate = grad(elbo)
```

This code is very general: It works for any unnormalized log-posterior that's differentiable wrt z.

Worked example: Bayesian neural network

- z are weights of neural network
- x are all observed input + output pairs
- $p(z)$ prior on weights, usually standard normal (hard to set)
- $p(x|z) = \prod_i p(y_i|x_i, z)$
  - for regression: $p(y_i|x_i, z) = \mathcal{N}(nnet(x_i, z), \sigma^2)$
  - for classification: $p(y_i|x_i, z) = \text{Categorical}(y_i|\text{softargmax}(nnet(x_i, z)))$
- $p(z|x)$ is a collection of plausible sets of parameters that all fit the data (and have some probablity under the prior). Certain where the data is, uncertain where extrapolation is ambiguous.



MCMC vs SVI

Pros of MCMC:

- Asymptotically exact

Cons of MCMC:

- Easy to have a bug and not know it
- Hard to tune hyperparameters
- Hard to tell if you'pre making progress
- Can't use minibatches (easily - there has been limited progress in this area)

Pros of SVI:

- Simple
- Can tell if making progress
- Can use minibatches for fitting to large datasets

Cons of SVI:

- Limited flexibility of variational approximation
    - Can use as many parameters as needed (e.g. mixture of Gaussians)

Future week: Variational Autoencoders

This lecture covered the basics of gradient-based stochastic variational infernece. In future lectures, we'll cover a couple more refinements:

1. If our model $p(z, x|\theta)$ also has some parameters we'd like to make a point estimate of θ, we can also optimize those parameters to match the data at the same time as we optimize our variational posterior. For example, we might want to tune the shape of the likelihood.
2. If our likelihood depends on tons of data, both MCMC and SVI can be slow. However, SVI has a unique advantage, in that it works without modification even if we can only get an unbiased estimate of the likelihood. This means that we'll be able to apply SVI to posteriors conditioned on massive datasets, as long as the likelihood factorizes given the latent variables: $p(x_1, x_2, \ldots, x_N|z) = \prod_i p(x_i|z)$. In this case, we can estimate the total log-likelihood unbiasedly by only evaluating a single randomly-chosen datapoint.
3. If the true posterior factorizes given the data, we can break variational inference into a bunch of smaller problems. But we don't have to solve them all separately. Instead, we can use a neural network to look at the data and guess good variational parameters for each z. Training such a neural network (called a *recognition network*) is called *amortized inference*, because we spread the work out over time and get to re-use the recognition network again later.

Putting these all together will let us train *Variational Autoencoders*, a state-of-the-art deep generative model. We'll get to these after the midterm.

Tutorial

Alternative Derivation of ELBO

Starting with **Jenson's inequality**,

$$f(E[X]) \leq E[f(x)]$$

if X is a random variable and f is a convex function.

Given that $\log$ is a concave function, we have

$$\begin{aligned}
\log p(x) &= \log \int p_\theta(x,z) dz \\
&= \log \int p_\theta(x,z) \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \\
&= \log \mathop{\mathbb{E}}_{z \sim q_\phi} \frac{p_\theta(x,z)}{q_\phi(z|x)} \\
\Rightarrow \log \mathop{\mathbb{E}}_{z \sim q_\phi} \frac{p_\theta(x,z)}{q_\phi(z|x)} &\geq \mathop{\mathbb{E}}_{z \sim q_\phi} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \\
&= - \mathop{\mathbb{E}}_{z \sim q_\phi} \log \frac{q_\phi(z|x)}{p_\theta(x,z)} \\
&= \mathcal{L}(\theta, \phi; x)
\end{aligned}$$

Alternative Forms of ELBO and Intuitions

$$\mathcal{L}(\theta, \phi; x) = \text{ELBO} = -\mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z|x)}{p_\theta(x, z)}$$

1. The most general interpretation of the ELBO is given by

$$\begin{aligned}
\mathcal{L}(\theta, \phi; x) &= -\mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{q_\phi(z|x)}{p_\theta(x, z)} \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{p_\theta(z)p_\theta(x|z)}{q_\phi(z|x)} \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi} \Big[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x) \Big]
\end{aligned}$$

2. We can also re-write 1) using entropy

$$\mathop{\mathbb{E}}_{z\sim q_\phi} \Big[\log p_\theta(x|z) + \log p_\theta(z) \Big] \mathbb{H}\Big[q_\phi(z|x) \Big]$$

3. Another re-write and we arrive at

$$\mathop{\mathbb{E}}_{z\sim q_\phi} \Big[\log p_\theta(x|z) \Big] - KL(q_\phi(z|x)||p_\theta(z))$$

This frames the ELBO as a tradeoff. The first term can be thought of as a "reconstruction likelihood", i.e. how probable is x given z, which encourages the model to choose the distribution which best reconstructs the data. The second term acts as regularization, by enforcing the idea that our parameterization shouldn't move us too far from the true distribution.

Score Function Gradient Estimator

Also called the likelihood ratio, or REINFORCE, was independently developed in 1990, 1992, 2013, and 2014 (twice). It is given by

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z)} f(z) = \nabla_\phi \int f(z) q_\phi(z) dz$$

if we assume that $q_\phi(z)$ is a continous function of ϕ, then

$$\begin{aligned} &= \int \nabla_\phi f(z) q_\phi(z) dz \\ &= \int f(z) \nabla_\phi q_\phi(z) dz \end{aligned}$$

using the log-derivative trick $\left(\nabla_\phi \log q_\phi = \frac{\nabla_\phi q_\phi}{q_\phi}\right)$:

$$\begin{aligned} &= \int f(z) q_\phi(z|x) \nabla_\phi \big[\log q_\phi(z|x)\big] dz \\ &= \mathbb{E}_{z\sim q_\phi(z)} \Big[f(z) \nabla_\phi \big[\log q_\phi(z|x)\big] \Big] \end{aligned}$$

where $q_\phi(z|x)$ is the score function. Finally, we have

$$\nabla_\phi \mathcal{L}(\phi; x) = \mathbb{E}_{z\sim q_\phi(z)} \Big[\big(\log p_\theta(x,z) - \log q_\phi(z|x)\big) \nabla_\phi \big[\log q_\phi(z|x)\big] \Big]$$

which is *unbiased*, but *high variance*.