Variational inference (part 2)

Brooks Paige

COMP0171 Week 5

Variational inference recap

Last week we looked at a fairly "general" variational inference formulation:

1. Define the ELBO, a lower bound on the marginal log likelihood, as

$$\mathcal{L}(\lambda, \mathcal{D}) = \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{p(\mathbf{z}, \mathcal{D})}{q_\lambda(\mathbf{z})}\right] \leq \log p(\mathcal{D})$$

2. Use the "reparameterization trick" to re-write $q_\lambda(\mathbf{z})$ in terms of an independent noise distribution $p(\boldsymbol{\epsilon})$ and a transformation function $\mathbf{z} = r(\lambda, \boldsymbol{\epsilon})$
3. Using samples from $p(\boldsymbol{\epsilon})$, compute a stochastic gradient estimate of

$$\nabla_\lambda \mathcal{L}(\lambda, \mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\lambda \log \frac{p(r(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(r(\lambda, \boldsymbol{\epsilon}))}\right].$$

This lecture

- Simple demo and practical notes on reparameterization gradients in pytorch
- What is "approximate" about this sort of approximate inference?
- Using the ELBO for maximum likelihood estimation in latent variable models
- How to handle discrete variables (or not)

Demo #1: fit a Gaussian

One-dimensional Gaussian

Suppose our data $\mathcal{D} = \{x_1, \ldots, x_N\}$ is drawn from a 1-d Gaussian distribution with unknown mean μ and unknown precision τ.

We will define a model

$$\begin{aligned} \mu &\sim \mathcal{N}(0, 1) \\ \tau &\sim \text{Gamma}(2, 2) \\ x_i|\mu, \tau &\sim \mathcal{N}(\mu, \tau^{-1}) \qquad \text{for } i = 1, \ldots, N. \end{aligned}$$

One-dimensional Gaussian

Suppose our data $\mathcal{D} = \{x_1, \ldots, x_N\}$ is drawn from a 1-d Gaussian distribution with unknown mean μ and unknown precision τ.

We will define a model

$$\begin{aligned} \mu &\sim \mathcal{N}(0, 1) \\ \tau &\sim \text{Gamma}(2, 2) \\ x_i|\mu, \tau &\sim \mathcal{N}(\mu, \tau^{-1}) \qquad \text{for } i = 1, \ldots, N. \end{aligned}$$

Following the example in Bishop, Ch. 10, we will learn an approximate posterior $q(\mu, \tau) = q(\mu)q(\tau)$ where

$$q(\mu) = \mathcal{N}(\mu|m, s^2) \qquad q(\tau) = \text{Gamma}(\tau|a, b).$$

We then optimize the ELBO for $\lambda = \{m, s, a, b\}$.

Before we look at the code (1/2)

We will be optimizing the ELBO

$$\mathcal{L}(\mathcal{D};\lambda) = \mathbb{E}_{q(\mu|m,s)q(\tau|a,b)}\left[\log \frac{p(\mu)p(\tau)\prod_{i=1}^{N} p(x_i|\mu,\tau)}{q(\mu|m,s)q(\tau|a,b)}\right]$$

with respect to m, s, a, b. Three of these parameters have constraints:

$$s > 0, \qquad a > 0, \qquad b > 0.$$

To avoid "issues" we will optimize with respect to unconstrained transformed parameters $s' = \log s$, $a' = \log a$, and $b' = \log b$.

Before we look at the code (2/2)

There are two ways to sample from a pytorch distribution:

```
>> a = torch.tensor(1.0, requires_grad=True)

>> dist.Gamma(a, 1).sample()
tensor(1.0297)

>> dist.Gamma(a, 1).rsample()
tensor(0.7706, grad_fn=<DivBackward0>)
```

If you call `.rsample`, then internally sampling will be done such that gradients can be computed, i.e. internally it samples a value z by performing

$$\epsilon \sim p(\epsilon) \qquad\qquad z = r(\epsilon)$$

with appropriate transform r (may depend on parameters of the distribution!)

(Switch to notebook for demo now!)

What are we "missing"
if q_λ is too simple?

Variational inference goal

If all went well, after maximizing the ELBO, we have estimated parameters λ such that

$$q_\lambda(\mathbf{z}) \approx p(\mathbf{z}|\mathcal{D}).$$

- How can we tell if it worked?
- What if we picked a q_λ family that was too "simple"?

KL divergence is asymmetric

Maximizing the ELBO as defined minimizes

$$D_{KL}(q_\lambda(\mathbf{z}) \| p(\mathbf{z}|\mathcal{D})) = \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{q_\lambda(\mathbf{z})}{p(\mathbf{z}|\mathcal{D})}\right].$$

This divergence is not symmetric: in general it differs from

$$D_{KL}(p(\mathbf{z}|\mathcal{D}) \| q_\lambda(\mathbf{z})) = \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}\left[\log \frac{p(\mathbf{z}|\mathcal{D})}{q_\lambda(\mathbf{z})}\right]$$

(though $D_{KL}(p\|q_\lambda) = D_{KL}(q_\lambda\|p) = 0$ if $q_\lambda = p$).

KL divergence is asymmetric

Maximizing the ELBO as defined minimizes

$$D_{KL}(q_\lambda(\mathbf{z}) \| p(\mathbf{z}|\mathcal{D})) = \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{q_\lambda(\mathbf{z})}{p(\mathbf{z}|\mathcal{D})}\right].$$

This divergence is not symmetric: in general it differs from

$$D_{KL}(p(\mathbf{z}|\mathcal{D}) \| q_\lambda(\mathbf{z})) = \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}\left[\log \frac{p(\mathbf{z}|\mathcal{D})}{q_\lambda(\mathbf{z})}\right]$$

(though $D_{KL}(p\|q_\lambda) = D_{KL}(q_\lambda\|p) = 0$ if $q_\lambda = p$).

Practical reason for $D_{KL}(q_\lambda\|p)$: computing expectations under $p(\mathbf{z}|\mathcal{D})$ is hard.

If q_λ is too simple (part 1)

- The target distribution (green): multivariate gaussian $p(z_1, z_2)$ with positive correlation (non-diagonal covariance matrix)
- The approximating distribution (red): $q(z_1, z_2) = q(z_1)q(z_2)$, i.e. the product of two univariate Gaussians, with zero correlation between z_1, z_2 (diagonal covariance matrix)

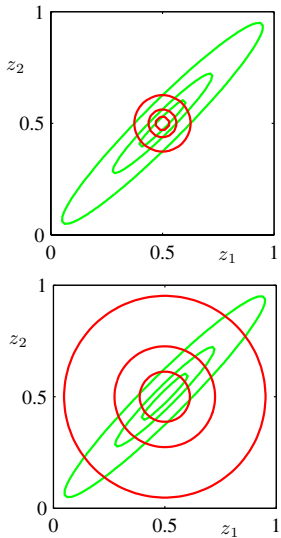


Figure: Bishop PRML, Ch. 10

If q_λ is too simple (part 1)

- The target distribution (green): multivariate gaussian $p(z_1, z_2)$ with positive correlation (non-diagonal covariance matrix)
- The approximating distribution (red): $q(z_1, z_2) = q(z_1)q(z_2)$, i.e. the product of two univariate Gaussians, with zero correlation between z_1, z_2 (diagonal covariance matrix)
- (top) q_λ minimizes $D_{KL}(q_\lambda \| p)$
- (bottom) q_λ minimizes $D_{KL}(p \| q_\lambda)$

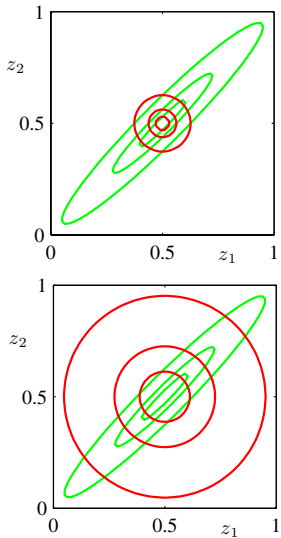


Figure: Bishop PRML, Ch. 10

If q_λ is too simple (part 1)

- The target distribution (green): multivariate gaussian $p(z_1, z_2)$ with positive correlation (non-diagonal covariance matrix)
- The approximating distribution (red): $q(z_1, z_2) = q(z_1)q(z_2)$, i.e. the product of two univariate Gaussians, with zero correlation between z_1, z_2 (diagonal covariance matrix)
- (top) q_λ minimizes $D_{KL}(q_\lambda \| p)$
- (bottom) q_λ minimizes $D_{KL}(p \| q_\lambda)$

VB **systematically underestimates** marginal variances!

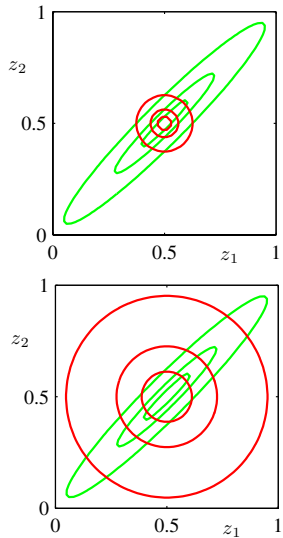
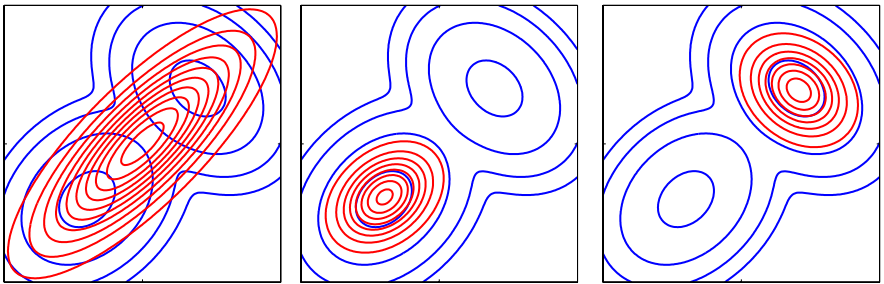


Figure: Bishop PRML, Ch. 10

If q_λ is too simple (part 2)

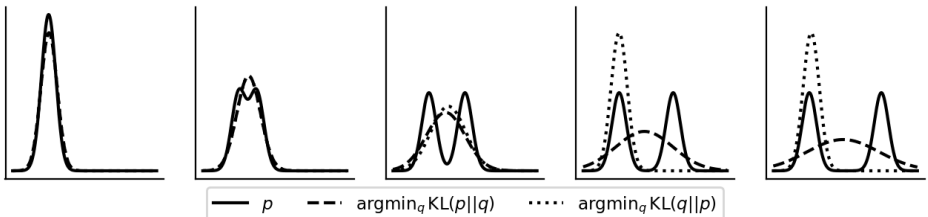


We also have a problem with **multimodal targets**.

- (left) q_λ minimizes $D_{KL}(p\|q_\lambda)$
- (middle, right) q_λ minimizes $D_{KL}(q_\lambda\|p)$ and finds a single local optimum

Figure: Bishop PRML, Ch. 10

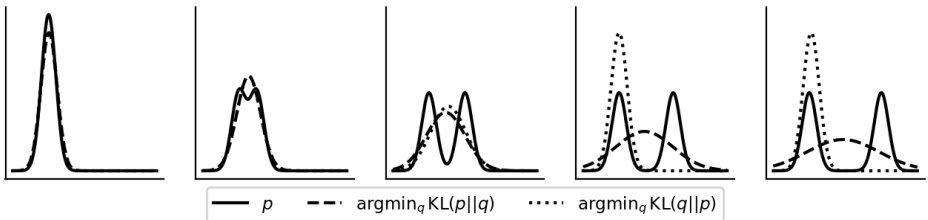
Understanding KL divergences



$D_{KL}(q_\lambda \| p)$ has **mode-seeking** or **zero-forcing** behavior.

Figure: Tuan-Anh Le

Understanding KL divergences



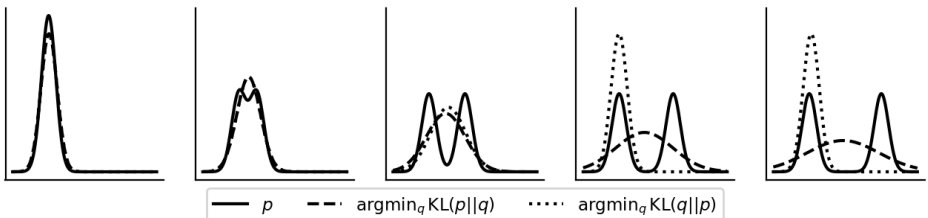
$D_{KL}(q_\lambda \| p)$ has **mode-seeking** or **zero-forcing** behavior. If you look at

$$\mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{q_\lambda(\mathbf{z})}{p(\mathbf{z}|\mathcal{D})}\right] = \mathbb{E}_{q_\lambda(\mathbf{z})}[\log q_\lambda(\mathbf{z})] - \mathbb{E}_{q_\lambda(\mathbf{z})}[\log p(\mathbf{z}|\mathcal{D})]$$

you will see a very large loss is assigned if there are any values $\mathbf{z}$ where $q_\lambda(\mathbf{z}) > 0$ and $p(\mathbf{z}|\mathcal{D}) \approx 0$.

Figure: Tuan-Anh Le

Understanding KL divergences

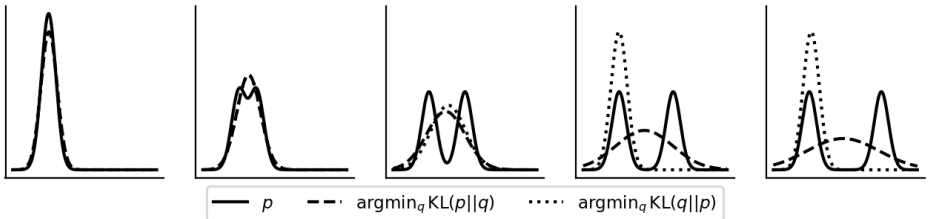


$D_{KL}(p\|q_\lambda)$ has **mass-covering** or **mean-seeking** behavior, for the opposite reason:

$$\mathbb{E}_{p(\mathbf{z}|\mathcal{D})}\left[\log \frac{p(\mathbf{z}|\mathcal{D})}{q_\lambda(\mathbf{z})}\right] = \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}[\log p(\mathbf{z}|\mathcal{D})] - \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}[\log q_\lambda(\mathbf{z})]$$

Figure: Tuan-Anh Le

Understanding KL divergences



$D_{KL}(p\|q_\lambda)$ has **mass-covering** or **mean-seeking** behavior, for the opposite reason:

$$\mathbb{E}_{p(\mathbf{z}|\mathcal{D})}\left[\log \frac{p(\mathbf{z}|\mathcal{D})}{q_\lambda(\mathbf{z})}\right] = \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}[\log p(\mathbf{z}|\mathcal{D})] - \mathbb{E}_{p(\mathbf{z}|\mathcal{D})}[\log q_\lambda(\mathbf{z})]$$

Exercise: Which is better as an approximate posterior? When? What about as an importance sampling proposal?

Figure: Tuan-Anh Le

Parameter estimation

Maximum likelihood in latent variable models

When we talked about maximum likelihood estimation before, it was in the context of models of the form $p(\mathcal{D}|\theta)$. (For example, in linear regression, the parameters were the regression weights.) We maximize

$$\theta_{MLE} = \arg\max_{\theta} \log p(\mathcal{D}|\theta).$$

What happens if you have a mix of latent variables, and unknown parameters? In that case,

$$\log p(\mathcal{D}|\theta) = \log \int p(\mathcal{D}, \mathbf{z}|\theta) \mathrm{d}\mathbf{z}.$$

How could we estimate θ?

Concrete example

In our previous Bayesian linear regression discussion, we assumed a few parameters were fixed, and only did inference over the weights.

The full model is

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$
$$p(y_i|\mathbf{x}_i, \mathbf{w}, \beta) = \mathcal{N}(y_i|\mathbf{w}^\top \phi(\mathbf{x}_i), \beta^{-1}) \qquad \text{for } i = 1, \ldots, N.$$

In this model we have latent variables $\mathbf{w}$, for which we will compute $p(\mathbf{w}|\mathcal{D})$, as well as parameters $\theta = \{\alpha, \beta\}$.

Concrete example

In our previous Bayesian linear regression discussion, we assumed a few parameters were fixed, and only did inference over the weights.

The full model is

$$
\begin{aligned}
p(\mathbf{w}|\alpha) &= \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \\
p(y_i|\mathbf{x}_i, \mathbf{w}, \beta) &= \mathcal{N}(y_i|\mathbf{w}^\top \phi(\mathbf{x}_i), \beta^{-1}) \qquad \text{for } i = 1, \ldots, N.
\end{aligned}
$$

In this model we have latent variables $\mathbf{w}$, for which we will compute $p(\mathbf{w}|\mathcal{D})$, as well as parameters $\theta = \{\alpha, \beta\}$.

Note: obviously, we could place priors on α, β and also find their posterior! But for now, suppose we are content simply optimizing them.

Standard answer: Expectation-Maximization

The **EM algorithm** is the "classic" answer for how to perform maximum likelihood estimation in models with latent variables. It is done by alternately

1. Computing the posterior $p(\mathbf{z}|\mathcal{D}, \theta^{\text{old}})$ for a current value θ^{old}
2. Maximizing $Q(\theta, \theta^{\text{old}}) = \mathbb{E}_{p(\mathbf{z}|\mathcal{D},\theta^{\text{old}})}[\log p(\mathcal{D}, \mathbf{z}|\theta)]$ with respect to θ

Standard answer: Expectation-Maximization

The **EM algorithm** is the "classic" answer for how to perform maximum likelihood estimation in models with latent variables. It is done by alternately

1. Computing the posterior $p(\mathbf{z}|\mathcal{D}, \theta^{\text{old}})$ for a current value θ^{old}
2. Maximizing $Q(\theta, \theta^{\text{old}}) = \mathbb{E}_{p(\mathbf{z}|\mathcal{D},\theta^{\text{old}})}[\log p(\mathcal{D}, \mathbf{z}|\theta)]$ with respect to θ

This works incredibly well when either (or both!) of those steps are easy to do analytically.

(Examples: Gaussian mixture models; hidden Markov models)

Handling intractable posteriors

If the posterior $p(\mathbf{z}|\mathcal{D}, \theta)$ doesn't have a closed form, but we have an approximation $q_\lambda(\mathbf{z})$, then we can use the ELBO. Recall

$$\mathcal{L}(\lambda, \mathcal{D}, \theta) = \mathbb{E}_{q_\lambda(\mathbf{z})} \left[\log p(\mathbf{z}, \mathcal{D}|\theta) - \log q_\lambda(\mathbf{z})\right] \leq \log p(\mathcal{D}|\theta).$$

Since $\mathcal{L}(\lambda, \mathcal{D}, \theta)$ is a lower bound on $\log p(\mathcal{D}|\theta)$, we can use it as a surrogate objective for (approximate) maximum likelihood estimation.

Handling intractable posteriors

If the posterior $p(\mathbf{z}|\mathcal{D}, \theta)$ doesn't have a closed form, but we have an approximation $q_\lambda(\mathbf{z})$, then we can use the ELBO. Recall

$$\mathcal{L}(\lambda, \mathcal{D}, \theta) = \mathbb{E}_{q_\lambda(\mathbf{z})} \left[\log p(\mathbf{z}, \mathcal{D}|\theta) - \log q_\lambda(\mathbf{z})\right] \leq \log p(\mathcal{D}|\theta).$$

Since $\mathcal{L}(\lambda, \mathcal{D}, \theta)$ is a lower bound on $\log p(\mathcal{D}|\theta)$, we can use it as a surrogate objective for (approximate) maximum likelihood estimation.

Using gradient-based optimization, we maximize it following

$$\nabla_\theta \mathcal{L}(\lambda, \mathcal{D}, \theta) = \mathbb{E}_{q_\lambda(\mathbf{z})} \left[\nabla_\theta \log p(\mathbf{z}, \mathcal{D}|\theta)\right].$$

(Note there are no issues here as $q_\lambda(\mathbf{z})$ does not depend on θ!)

Optimizing variational and model parameters

In this sense the ELBO is a unified objective: by gradient-based optimization on

$$\nabla_{\lambda,\theta}\mathcal{L}(\lambda,\mathcal{D},\theta) = \nabla_{\lambda,\theta}\mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{p(\mathbf{z},\mathcal{D}|\theta)}{q_\lambda(\mathbf{z})}\right].$$

we can simultaneously:

- maximize w.r.t. λ to fit an approximate posterior;
- maximize w.r.t. θ to find maximum likelihood estimates.

Optimizing variational and model parameters

In this sense the ELBO is a unified objective: by gradient-based optimization on

$$\nabla_{\lambda,\theta}\mathcal{L}(\lambda,\mathcal{D},\theta) = \nabla_{\lambda,\theta}\mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{p(\mathbf{z},\mathcal{D}|\theta)}{q_\lambda(\mathbf{z})}\right].$$

we can simultaneously:

- maximize w.r.t. λ to fit an approximate posterior;
- maximize w.r.t. θ to find maximum likelihood estimates.

It might not be as efficient as EM (nor is it exact), but it somehow has the same flavor.

Back to concrete example

For the Bayesian linear regression example, we had

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$
$$p(y_i|\mathbf{x}_i, \mathbf{w}, \beta) = \mathcal{N}(y_i|\mathbf{w}^\top \phi(\mathbf{x}_i), \beta^{-1}) \qquad \text{for } i = 1, \ldots, N.$$

If we define an approximate posterior $q_\lambda(\mathbf{w})$, then we just need to perform stochastic gradient optimization on

$$\mathcal{L}(\mathcal{D}; \alpha, \beta, \lambda) = \mathbb{E}_{q_\lambda(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}, \beta) + \log p(\mathbf{w}|\alpha) - \log q_\lambda(\mathbf{w})\right]$$

by computing $\nabla_{\alpha,\beta,\lambda}\mathcal{L}(\mathcal{D}; \alpha, \beta, \lambda)$ on (reparameterized) samples from $q_\lambda(\mathbf{w})$.

This simultaneously fits $q_\lambda(\mathbf{w}) \approx p(\mathbf{w}|\mathcal{D})$ and estimates $\hat{\alpha}_{MLE}, \hat{\beta}_{MLE}$.

(Switch to notebook for demo now!)

Sequential learning

If you are adding data points sequentially, your old posterior is your new prior.

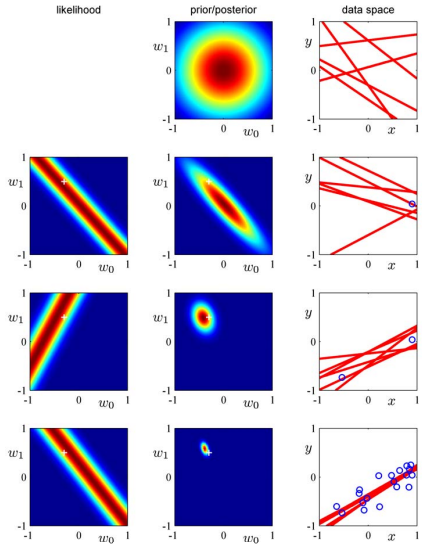


Figure: Bishop PRML, Ch. 3

What about discrete latent variables?

What if our latent variables are discrete?

Suppose the latent variable $\mathbf{z}$ has as its domain a subset of the integers.

Then we have a problem: $\nabla_{\mathbf{z}} p(\mathbf{z}, \mathcal{D})$ does not exist: we cannot use the "reparameterization trick".

What if our latent variables are discrete?

Suppose the latent variable $\mathbf{z}$ has as its domain a subset of the integers.

Then we have a problem: $\nabla_{\mathbf{z}} p(\mathbf{z}, \mathcal{D})$ does not exist: we cannot use the "reparameterization trick".

Example: Suppose $z \sim \text{Bernoulli}(\theta)$, and suppose we would like to compute $\frac{\partial}{\partial \theta} \mathbb{E}_{p(z|\theta)}[f(z, \theta)]$. For a reparameterization we need to define an independent random variable $p(\epsilon)$ and a transformation $z = r(\theta, \epsilon)$:

$$p(\epsilon) = \text{Uniform}([0, 1]) \qquad r(\theta, \epsilon) = \mathbb{I}[\epsilon \leq \theta]$$

What if our latent variables are discrete?

Suppose the latent variable $\mathbf{z}$ has as its domain a subset of the integers.

Then we have a problem: $\nabla_{\mathbf{z}} p(\mathbf{z}, \mathcal{D})$ does not exist: we cannot use the "reparameterization trick".

Example: Suppose $z \sim \text{Bernoulli}(\theta)$, and suppose we would like to compute $\frac{\partial}{\partial \theta} \mathbb{E}_{p(z|\theta)}[f(z, \theta)]$. For a reparameterization we need to define an independent random variable $p(\epsilon)$ and a transformation $z = r(\theta, \epsilon)$:

$$p(\epsilon) = \text{Uniform}([0, 1]) \qquad\qquad r(\theta, \epsilon) = \mathbb{I}[\epsilon \leq \theta]$$

The problem is not that $r(\theta, \epsilon)$ doesn't exist, but that it is non-differentiable; $\frac{\partial r}{\partial \theta} = 0$ almost everywhere (and is undefined at θ).

(. . . confirming it doesn't work)

Let

$$p(\epsilon) = \text{Uniform}([0, 1]) \qquad\qquad r(\theta, \epsilon) = \mathbb{I}[\epsilon \leq \theta].$$

With $z = r(\theta, \epsilon)$ we would have

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}[f(z,\theta)] &= \mathbb{E}_{p(\epsilon)}\left[\frac{\partial}{\partial\theta}f(r(\theta,\epsilon),\theta)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\left(\frac{\partial}{\partial z}f(z,\theta)\right)\left(\frac{\partial}{\partial\theta}r(\theta,\epsilon)\right)\right]\end{aligned}$$

Can't evaluate it at θ; zero everywhere else. . .

Approach #1: Score-function estimators

We talked about this briefly before, but the following can work:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}[f(z,\theta)] = \mathbb{E}_{p(z|\theta)}\left[\frac{\partial}{\partial\theta}f(z,\theta) + f(z,\theta)\frac{\partial}{\partial\theta}\log p(z|\theta)\right]$$
$$\approx \frac{1}{L}\sum_{\ell=1}^{L}\frac{\partial}{\partial\theta}\left(f(z^{(\ell)},\theta)\log p(z^{(\ell)}|\theta)\right),$$

for $z^{(\ell)} \sim p(z|\theta)$.

Good news: we can apply it for any discrete z, even if the support is infinite (e.g. $z \in \mathbb{Z}$ or $z \in \mathbb{N}$), as long as f is differentiable w.r.t. θ

Bad news: it might require a much larger sample size L than we would like

Approach #2: Explicit marginalization

If the support for z is finite, then maybe we can just marginalize it out!

In the Bernoulli example, $z \in \{0, 1\}$. So

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}[f(z,\theta)] &= \frac{\partial}{\partial\theta}\sum_{z\in\{0,1\}} p(z|\theta)f(z,\theta) \\ &= \sum_{z\in\{0,1\}} p(z|\theta)\frac{\partial}{\partial\theta}f(z,\theta) + f(z,\theta)\frac{\partial}{\partial\theta}p(z|\theta)\end{aligned}$$

It's easy to forget this is an option, somehow!

Bad news: If the number of possible values z can take is very large, then this will be infeasible

Approach #2: Explicit marginalization

If the support for z is finite, then maybe we can just marginalize it out!

In the Bernoulli example, $z \in \{0, 1\}$. So

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}[f(z,\theta)] &= \frac{\partial}{\partial\theta}\sum_{z\in\{0,1\}} p(z|\theta)f(z,\theta) \\ &= \sum_{z\in\{0,1\}} p(z|\theta)\frac{\partial}{\partial\theta}f(z,\theta) + f(z,\theta)\frac{\partial}{\partial\theta}p(z|\theta)\end{aligned}$$

It's easy to forget this is an option, somehow!

Bad news: If the number of possible values z can take is very large, then this will be infeasible

Exercise: Show that this is identical to exact marginalization on the expectation in the previous slide

Explicit marginalization and the ELBO

Suppose we have both continuous latents $\mathbf{z}_c$ and discrete latents $\mathbf{z}_d$ in a model $p(\mathcal{D}, \mathbf{z}_c, \mathbf{z}_d)$.

If we marginalize out the discrete latent variable, we have

$$\log p(\mathcal{D}, \mathbf{z}_c) = \log \sum_{\mathbf{z}_d} p(\mathcal{D}, \mathbf{z}_c, \mathbf{z}_d).$$

We can then define the ELBO just on $\mathbf{z}_c$ along, i.e. after marginalizing out the discrete variable, and fit $q_\lambda(\mathbf{z}_c)$.

Explicit marginalization and the ELBO

Suppose we have both continuous latents $\mathbf{z}_c$ and discrete latents $\mathbf{z}_d$ in a model $p(\mathcal{D}, \mathbf{z}_c, \mathbf{z}_d)$.

If we marginalize out the discrete latent variable, we have

$$\log p(\mathcal{D}, \mathbf{z}_c) = \log \sum_{\mathbf{z}_d} p(\mathcal{D}, \mathbf{z}_c, \mathbf{z}_d).$$

We can then define the ELBO just on $\mathbf{z}_c$ along, i.e. after marginalizing out the discrete variable, and fit $q_\lambda(\mathbf{z}_c)$.

If we are later interested in the posterior over the discrete latent variables, we can consider

$$p(\mathbf{z}_d = k|\mathcal{D}) = \int p(\mathbf{z}_d = k|\mathbf{z}_c, \mathcal{D}) p(\mathbf{z}_c|\mathcal{D}) \mathrm{d}\mathbf{z}_c \approx \frac{1}{C} \int p(\mathbf{z}_d = k, \mathbf{z}_c, \mathcal{D}) q_\lambda(\mathbf{z}_c) \mathrm{d}\mathbf{z}_c.$$

Real-world usage

This is how discrete latent variables are handled in **STAN**.

Stan is a "probabilistic programming" system: it includes

1. a **modeling language**: a specialized programming language for specifying a probabilistic model
2. an **inference backend**: software which compiles the model into code which performs inference automatically

Stan is built around an efficient automatic differentiation implementation in C++. It is very easy to use for continuous latent variables, and performs MCMC and VB automatically.

But: it doesn't handle discrete latent variables natively, but instead requires them to be marginalized out "by hand".

(See: `https://mc-stan.org`)