COMPSCI 688: Probabilistic Graphical Models

Lecture 19: Black-Box Stochastic Variational Inference

Dan Sheldon

Manning College of Information and Computer Sciences
University of Massachusetts Amherst

Partially based on materials by Benjamin M. Marlin (marlin@cs.umass.edu) and Justin Domke (domke@cs.umass.edu)

Review

Variational Inference

1. **Input**: $p(z, x)$ and fixed x
2. Choose some approximating family $q_\phi(z)$
3. Maximize ELBO(ϕ) wrt ϕ — equivalent to minimizing KL$(q_\phi(z) \,\|\, p(z|x))$
4. Use $q_\phi(z)$ as a proxy for $p(z|x)$

$$\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(Z)}\left[\log \frac{p(Z,x)}{q_\phi(Z)}\right] = \mathbb{E}_{q_\phi(Z)}\left[\log p(Z,x)\right] - \mathbb{E}_{q_\phi}\left[\log q_\phi(Z)\right]$$

Variational Inference

Something we skipped: $p(z, x)$ discrete graphical model, $q(z) = \prod_j q_j(z_j)$ ("mean field")

Today: z continuous, $p(z, x)$ black box, $q(z)$ TBD

Motivation: Continuous Latent Variable Models

Factor Analysis

Factor analysis is a classical statistical model. It posits an observed vector $\mathbf{x} \in \mathbb{R}^d$ is generated as a linear combination of basis vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ with weights $z_1, \ldots, z_m$ plus noise:

Probabilistic factor analysis assumes the weights are drawn from a standard normal. The generative process is:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{Wz}, \Psi)$$

(Typically Ψ is diagonal and the data is pre-processed so $\mathbf{x}$ has zero mean.)

Visualization: PCA Demo

Factor Analysis: Learning

Consider learning the parameters $\theta = (\mathbf{W}, \Psi)$ given data $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ assumed to be independently drawn from this model.

Since $\mathbf{z}$ is latent, the log-likelihood of a single datum $\mathbf{x}$ is $\log p(\mathbf{x})$, the "log-marginal likelihood".

In this model, the marginal likelihood is available in *closed form*:

$$p(\mathbf{x}) = \int \mathcal{N}(\mathbf{z}; 0, I)\mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z}, \Psi) d\mathbf{z} = \mathcal{N}(\mathbf{x}; 0, \mathbf{W}\mathbf{W}^\top + \Psi)$$

Therefore, we can learn by maximizing the log-marginal likelihood:

$$\mathcal{L}(\theta) = -\frac{N}{2}\log(|2\pi\Sigma|) - \frac{1}{2}\sum_{n=1}^{N} \mathbf{x}^{(n)\top}\Sigma^{-1}\mathbf{x}^{(n)}, \qquad \Sigma = \mathbf{W}\mathbf{W}^\top + \Psi$$

Alternately, there is an EM algorithm for this model where both E and M steps have simple forms.

Factor Analysis: Generalizations

This model is "easy", but factor analysis has many generalizations that make exact learning and inference intractable.

A variational autoencoder (VAE) uses a nonlinear-function f_θ instead of a linear transformation $\mathbf{W}$ to map from $\mathbf{z}$ to the mean of $\mathbf{x}$:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta(\mathbf{z}), \Psi)$$

A typical structure for f_θ is a multi-layer neural network, e,g.

$$f_\theta(\mathbf{z}) = h_2(\mathbf{b}_2 + \mathbf{W}_2 \cdot h_1(\mathbf{b}_1 + \mathbf{W}_1\mathbf{z}))$$

where h_2, h_1 are element-wise nonlinear functions.

Another generalization changes the likelihood, e.g., to a Bernoulli distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$$
$$p(x_j|\mathbf{z}) = \text{Bernoulli}(x_j; (f_\theta(\mathbf{z}))_j), \qquad j = 1, \ldots, d$$

Inference and Learning in Generalized Models

Almost any change from the basic factor analysis model makes it so we can't compute the marginal likelihood $p(\mathbf{x})$ exactly, so inference and learning become hard.

The model is *only* tractable with linear transformations and a Gaussian likelihood.

We need additional inference tools for the generalizations.

Black-Box Stochastic Variational Inference

Black-Box Stochastic Variational Inference

A general inference approach that works well for models with continuous latent variables, including factor analysis, is *black-box stochastic variational inference*:

- **Black box**: only requires computing $\log p(z, x)$ and its gradients for different z
- **Stochastic**: optimizes the ELBO using Monte Carlo estimates

Stochastic Variational Inference

- **Input**: $p(z, x)$ and fixed x
- Start with some ϕ
- For $t = 1, 2, \dots, T$
  - $g \leftarrow$ **unbiased estimate** of ∇_ϕELBO
  - Take a small step: $\phi \leftarrow \phi + \epsilon g$ (or Adam or other optimizer)
- **Return** ϕ

Main issue: how to get g?

Warmup: Estimating the ELBO

It's easy to estimate the ELBO via Monte Carlo samples:

$$\text{ELBO}(\phi) \approx \frac{1}{K} \sum_{i=1}^{K} \log \frac{p(z^{(i)}, x)}{q_\phi(z^{(i)})} \qquad z^{(i)} \sim q_\phi$$

This is unbiased: expected value of RHS = ELBO(ϕ) for any value of K.

Estimating the Gradient? (False Start)

What happens if we try to estimate the gradient the same way? We want:

$$\nabla_\phi \text{ELBO}(\phi) = \nabla_\phi \mathbb{E}_{q_\phi}\left[\log \frac{p(Z, x)}{q_\phi(Z)}\right]$$

Consider the estimate:

$$\nabla_\phi \log \frac{p(z, x)}{q_\phi(z)}, \quad z \sim q_\phi.$$

This is not unbiased! It neglects the fact that the *distribution* of z depends on ϕ:

$$\nabla_\phi \text{ELBO}(\phi) = \nabla_\phi \int q_\phi(z) \log \frac{p(z, x)}{q_\phi(z)} dz$$

The false start incorrectly interchanges the gradient and the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi}\left[\log \frac{p(Z, x)}{q_\phi(Z)}\right] \neq \mathbb{E}_{q_\phi}\left[\nabla_\phi \log \frac{p(Z, x)}{q_\phi(Z)}\right]$$

Reparameterization Trick

The reparameterization trick is a way to convert the ELBO into an expectation with respect to a *fixed* distribution (independent of ϕ) so we can interchange the gradient and expectation. The idea is to draw samples of z by transforming a random variable from a fixed base distribution.

Example: $z = \mu + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0,1) \implies z \sim \mathcal{N}(\mu, \sigma^2)$

General case: $z = \mathcal{T}_\phi(\epsilon),\ \epsilon \sim q(\epsilon) \implies z \sim q_\phi(z)$

We call $\mathcal{T}_\phi$ and $q(\epsilon)$ a *reparameterization* of q_ϕ

ELBO Gradient with Reparameterization

With reparameterization, we can write the ELBO as an expectation over $q(\epsilon)$:

$$\text{ELBO}(\phi) = \mathbb{E}_{q(\epsilon)}\left[\log \frac{p\left(\mathcal{T}_\phi(\epsilon), x\right)}{q_\phi\left(\mathcal{T}_\phi(\epsilon)\right)}\right]$$

Now we can interchange the gradient and expectation

$$\nabla_\phi \text{ELBO}(\phi) = \nabla_\phi \mathbb{E}_{q(\epsilon)}\left[\log \frac{p\left(\mathcal{T}_\phi(\epsilon), x\right)}{q_\phi\left(\mathcal{T}_\phi(\epsilon)\right)}\right] = \mathbb{E}_{q(\epsilon)}\left[\nabla_\phi \log \frac{p\left(\mathcal{T}_\phi(\epsilon), x\right)}{q_\phi\left(\mathcal{T}_\phi(\epsilon)\right)}\right]$$

Reparameterization Gradient Estimate

This gives a simple unbiased Monte Carlo estimate of the gradient:

$$g = \nabla_\phi \left(\frac{1}{K}\sum_{i=1}^{K} \log \frac{p\left(\mathcal{T}_\phi(\epsilon^{(i)}), x\right)}{q_\phi\left(\mathcal{T}_\phi(\epsilon^{(i)})\right)}\right), \qquad \epsilon^{(1)}, \ldots, \epsilon^{(K)} \sim q(\epsilon)$$

We can compute it as follows:

1. Draw $\epsilon^{(1)}, \ldots, \epsilon^{(K)} \sim q(\epsilon)$
2. Compute $\widehat{\text{ELBO}}(\phi, \epsilon^{(1)}, \ldots, \epsilon^{(K)})$ = term in parentheses above
3. Use autodiff to get $g = \nabla_\phi \widehat{\text{ELBO}}(\phi, \epsilon^{(1:K)})$

Reparameterization with Gaussians

Multivariate Gaussians are common variational distributions, and easy to reparameterize:

$$\epsilon \sim \mathcal{N}(0, I) \Longrightarrow \mu + L\epsilon \sim \mathcal{N}(\mu, LL^\top).$$

This is a reparameterization with

$$q(\epsilon) = \mathcal{N}(\epsilon|0, I), \qquad \mathcal{T}_\phi(\epsilon) = \mu + L\epsilon \qquad \phi = (L, \mu)$$

It covers any multivariate Gaussian, since an arbitrary covariance matrix Σ can be written as $\Sigma = LL^\top$ for some L (e.g., a Cholesky factor)

Reparameterization with Diagonal Gaussians

Another common variational distribution is a diagonal Gaussians:

$$q_\phi(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$$

This can be reparameterized as:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

($\odot$ = elementwise multiplication)

Example: Factor Analysis

Let's return to our Bernoulli VAE model

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, I) \\ p(x_j|\mathbf{z}) &= \text{Bernoulli}(x_j; (f_\theta(\mathbf{z}))_j), \qquad j = 1, \dots, d \end{aligned}$$

Suppose we choose a diagonal Gaussian variational family

$$q_\phi(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mu, \text{diag}(\boldsymbol{\sigma}^2))$$

We now know how to use BBSVI with the reparameterization trick to optimize the ELBO.

With the optimized parameters $\phi = (\mu, \boldsymbol{\sigma})$ we can

- approximate $p(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z})$
- lower bound the log-marginal likelihood $\log p(\mathbf{x}) \geq \text{ELBO}(\phi)$

Next time: learning the model f_θ. Thoughts?

Bonus: Handling Some Terms in Closed Form

The ELBO can be decomposed into several terms with different computational properties:

$$\begin{aligned} \text{ELBO}(\phi) &= \mathbb{E}_{q_\phi}\left[\log \frac{p(Z, x)}{q_\phi(Z)}\right] \\ &= \underbrace{\mathbb{E}_{q_\phi}[\log p(Z)]}_{\text{"cross entropy"}} + \underbrace{\mathbb{E}_{q_\phi}[\log p(x|Z)]}_{\text{"energy"}} - \underbrace{\mathbb{E}_{q_\phi}[\log q_\phi(Z)]}_{\text{"entropy"}} \end{aligned}$$

With simple distributions (esp. Gaussians) the cross entropy and entropy terms can often be computed in closed form.

Example: cross entropy standard normal and diagonal Gaussian

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, I) \\ q_\phi(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mu, \mathsf{diag}(\boldsymbol{\sigma}^2)) \end{aligned} \quad \Longrightarrow \quad \int q_\phi(\mathbf{z}) \log p(\mathbf{z}) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{d}(\mu_j^2 + \sigma_j^2)$$

When possible, it's usually (but not always) best to compute these terms and their gradients analytically, and only use Monte Carlo estimation for the energy term.

This is because lower variance gradient estimates will make the stochastic optimization converge faster.