Discrete Variables in DGMs

Philip Schulz and Wilker Aziz
https://github.com/philschulz/VITutorial

What we know so far

- Deep Generative Models are probabilistic models where the parameters of the conditional distributions are computed by neural networks
- Because the ELBO cannot be computed exactly, we need to sample latent values
- Main problem: the MC estimator is not differentiable
- Solution: reparametrisation gradient

Reparametrisation Gradient

Model Gradient

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(z|\lambda)}\left[\log p(x|z,\theta)\right] - \frac{\partial}{\partial\theta}\mathrm{KL}\left(q(z|\lambda)\ ||\ p(z|\theta)\right)$$

Inference Network Gradient

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{q(z|\lambda)}\left[\log p(x|z,\theta)\right] - \frac{\partial}{\partial\lambda}\mathrm{KL}\left(q(z|\lambda)\ ||\ p(z|\theta)\right)$$

Reparametrisation Gradient

$$
\begin{aligned}
\frac{\partial}{\partial \lambda}\mathbb{E}_{q(z|\lambda)}\left[\log p(x|z,\theta)\right] &= \\
\frac{\partial}{\partial \lambda}\mathbb{E}_{\phi(\epsilon)}\left[\log p(x|\overbrace{h^{-1}(\epsilon,\lambda)}^{z},\theta)\right] &= \\
\mathbb{E}_{\phi(\epsilon)}\left[\frac{\partial}{\partial z}\log p(x|\overbrace{h^{-1}(\epsilon,\lambda)}^{z},\theta) \times \frac{\partial}{\partial \lambda}\overbrace{h^{-1}(\epsilon,\lambda)}^{z}\right]
\end{aligned}
$$

Reparametrisation for Discrete Variables?

Revisiting the Inference Gradient

Reparametrisation for Discrete Variables?

Revisiting the Inference Gradient

Reparametrisation

In order to tranform variables, we need to compute the Jacobian (matrix of derivatives).

$$p(z) = \phi(h(z)) \left| \frac{d}{dz} h(z) \right|$$

The Jacobian is generally not available for discrete variables.

Cumulative Distribution Function

Insert picture here

Continuity

The outcome space of discrete variables is non-continuous. Thus, we cannot take derivatives with respect to real variables.

Reparametrisation for Discrete Variables?

Revisiting the Inference Gradient

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{q(z|\lambda)}\left[\log p(x|z,\theta)\right] =$$

$$\frac{\partial}{\partial\lambda}\sum_z q(z|\lambda)\log p(x|z,\theta) =$$

$$\sum_z \frac{\partial}{\partial\lambda} q(z|\lambda)\log p(x|z,\theta)$$

Back to Basic Calculus

$$\frac{d}{d\lambda} \log f(\lambda) = \frac{\frac{d}{d\lambda} f(\lambda)}{f(\lambda)}$$

Consequence

$$\frac{d}{d\lambda} f(\lambda) = \frac{d}{d\lambda} \log f(\lambda) \times f(\lambda)$$

Score Function Estimator

$$\frac{d}{d\lambda} f(\lambda) = \frac{d}{d\lambda} \log f(\lambda) \times f(\lambda)$$

Apply this to the red derivative.

$$\sum_z \frac{\partial}{\partial \lambda} q(z|\lambda) \log p(x|z,\theta) =$$

$$\sum_z q(z|\lambda) \frac{\partial}{\partial \lambda} \log q(z|\lambda) \times \log p(x|z,\theta) =$$

$$\mathbb{E}_{q(z|\lambda)} \left[\frac{\partial}{\partial \lambda} \log q(z|\lambda) \times \log p(x|z,\theta) \right]$$

Comparison Between Estimators

- Score function gradient

$$\mathbb{E}_{q(z|\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|\lambda)\times\log p(x|z,\theta)\right]$$

- Reparametrisation gradient

$$\mathbb{E}_{\phi(\epsilon)}\left[\frac{\partial}{\partial\lambda}\log p(x|h^{-1}(\epsilon,\lambda),\theta)\right]$$

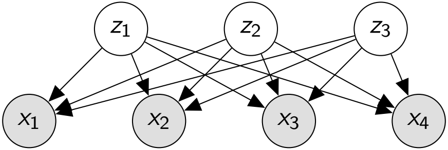
Example Model

Let us consider a latent factor model for topic modelling. Each document x consists of n i.i.d. categorical draws from that model. The categorical distribution in turn depends on the binary latent factors $z = (z_1, \ldots, z_k)$ which are also i.i.d.

$$\begin{aligned} z_j &\sim \text{Bernoulli}(\phi) & (1 \leq j \leq k) \\ x_i &\sim \text{Categorical}(g(z)) & (1 \leq i \leq n) \end{aligned}$$

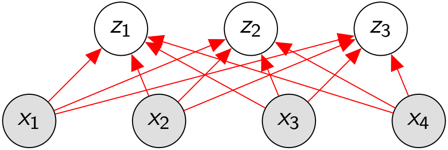
Here $g(\cdot)$ is a function computed by neural network with softmax output.

Example Model



At inference time the latent variables are marginally dependent. For our variational distribution we are going to assume that they are not (recall: mean field assumption).

Inference Network



The inference network needs to predict k Bernoulli parameters ψ. Any neural network with sigmoid output will do that job.

Computation Graph