Deep Generative Models: Discrete Latent Variables

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

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

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

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

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

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

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

Reparametrisation Gradient

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

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

Reparametrisation Gradient

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

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

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

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.

Continuity

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

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

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

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

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

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

Back to Basic Calculus

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

Back to Basic Calculus

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

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

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

Score Function Estimator

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

Apply this to the ELBO derivative.

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

Score Function Estimator

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

Apply this to the ELBO derivative.

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

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

Score Function Estimator

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

Apply this to the ELBO derivative.

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

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

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

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

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

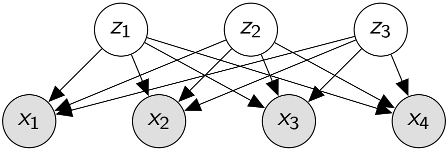
Example Model

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.

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

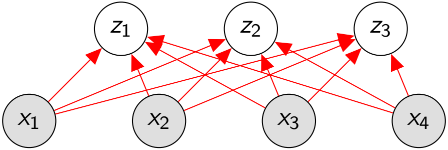
Here $g(\cdot)$ is a function computed by neural network with softmax output.

Example Model



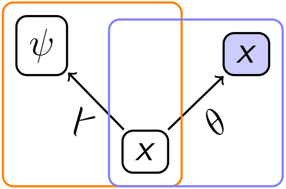
At inference time the latent variables are marginally dependent. For our variational distribution we are going to assume that they are not (recall: mean field assumption).

Inference Network



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

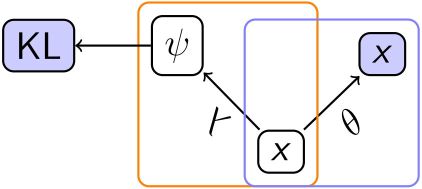
Computation Graph



inference model

generation model

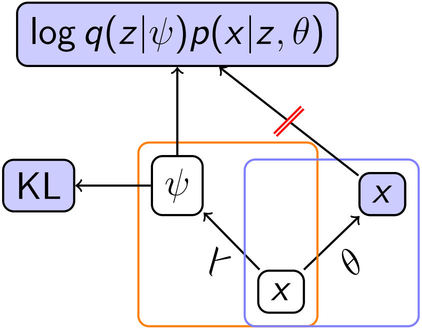
Computation Graph



inference model

generation model

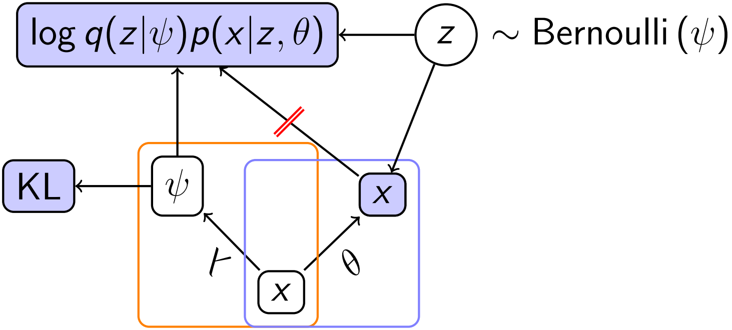
Computation Graph



inference model

generation model

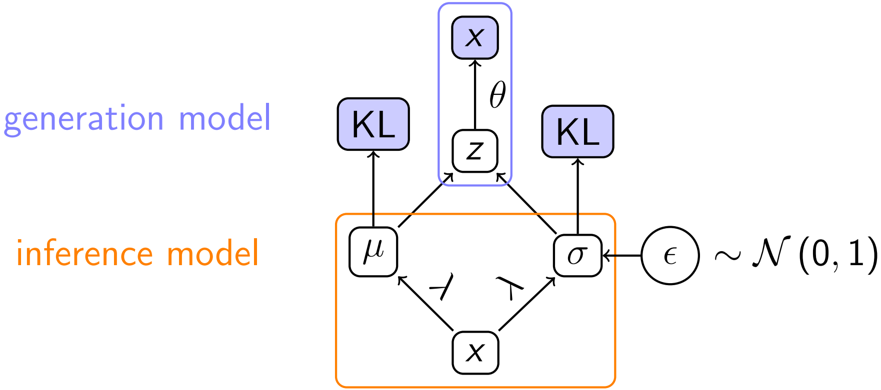
Computation Graph



inference model

generation model

Reparametrisation Gradient



Pros and Cons

- Pros
  - Applicable to all distributions
  - Many libraries come with samplers for common distributions

Pros and Cons

- Pros
  - Applicable to all distributions
  - Many libraries come with samplers for common distributions
- Cons
  - High Variance!

Baselines

We attempt to centre the gradient estimate. To do this we learn a quantity C that we subtract from the reconstruction loss.

$$\log q(z|\lambda)\left(\log p(x|z,\theta) - C\right)$$

We call C a baseline. It does not change the expected gradient (Williams, 1992).

Baselines

We can make baselines input-depdendent to make them more flexible.

$$\log q(z|\lambda)\left(\log p(x|z,\theta) - C(x)\right)$$

However, baselines may not depend on the random value z! Quantities that may depend on the random value ($C(z)$) are called **control variates**. See Blei et al. (2012); Ranganath et al. (2014); Gregor et al. (2014).

Baselines

Baselines are predicted by a regression model (e.g. a neural net). The model is trained using an L_2-loss.

$$\min \left(C(x) - \log p(x|z,\theta) \right)^2$$

Summary

Summary

- Reparametrisation not available for discrete variables.

Summary

- Reparametrisation not available for discrete variables.
- Use score function estimator.

Summary

- Reparametrisation not available for discrete variables.
- Use score function estimator.
- High variance.

Summary

- Reparametrisation not available for discrete variables.
- Use score function estimator.
- High variance.
- Always use baselines for variance reduction!

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

- plays $\rightarrow$ played

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

- plays $\rightarrow$ played
- walking $\rightarrow$ walks

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

- lemma

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

- lemma (real vector)

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

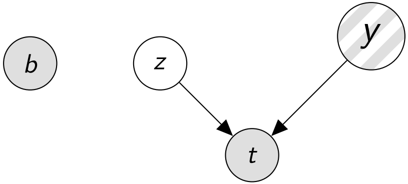
- lemma (real vector)
- morphological information

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

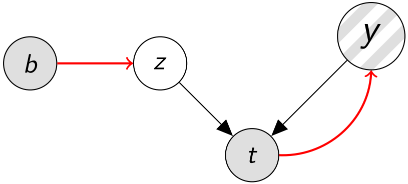
- lemma (real vector)
- morphological information (discrete vector)

A Simple Model (Zhou and Neubig, 2017)



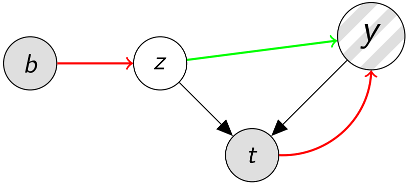
- z = lemma (continuous)
- y = morphological features (discrete)
- b = base form (inflected)
- t = target form (inflected)

A Simple Model (Zhou and Neubig, 2017)



- z = lemma (continuous)
- y = morphological features (discrete)
- b = base form (inflected)
- t = target form (inflected)

A Simple Model (Zhou and Neubig, 2017)



- z = lemma (continuous)
- y = morphological features (discrete)
- b = base form (inflected)
- t = target form (inflected)

David M. Blei, Michael I. Jordan, and John W. Paisley. Variational bayesian inference with stochastic search. In *ICML*, 2012. URL http://icml.cc/2012/papers/687.pdf.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In Eric P. Xing and Tony Jebara, editors, *ICML*, pages 1242–1250, 2014. URL http://proceedings.mlr.press/v32/gregor14.html.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *AISTATS*, pages 814–822, 2014. URL http://proceedings.mlr.press/v33/ranganath14.pdf.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4): 229–256, 1992. URL https://doi.org/10.1007/BF00992696.

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *ACL*, pages 310–320, 2017. doi: 10.18653/v1/P17-1029. URL http://www.aclweb.org/anthology/P17-1029.

Goodbye and thank you!

What we covered today

- Derivation of VI
- Continuous VAE
- Discrete VAE
- Semisupervised Models

Goodbye and thank you!

Next steps

- Do the coding tutorial (https://github.com/philschulz/VITutorial/blob/master/code/vae_notebook.ipynb)
- Come to our talk on Tuesday (10:30am, ROOM 203/204)
- Stay in touch
    - w.aziz@uva.nl
    - phschulz@amazon.com