Deep Generative Models: Discrete Latent Variables

Philip Schulz
https://vitutorial.github.io
https://github.com/vitutorial/VITutorial

Deep Generative Models

First Attempt: Wake-Sleep

Revisiting the Inference Gradient

Control Variates and Baselines

Generative Models

Joint distribution over observed data x and latent variables Z.

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{likelihood}}$$

The likelihood and prior are often standard distributions (Gaussian, Bernoulli) with simple dependence on conditioning information.

Deep generative models

Joint distribution with **deep observation model**

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{likelihood}}$$

mapping from z to $p(x|z, \theta)$ is a NN with parameters θ

Deep generative models

Joint distribution with **deep observation model**

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z,\theta)}_{\text{likelihood}}$$

mapping from z to $p(x|z,\theta)$ is a NN with parameters θ

Marginal likelihood

$$p(x|\theta) = \int p(x, z|\theta)\,\mathrm{d}z = \int p(z) \underbrace{p(x|z,\theta)}_{\text{highly nonlinear!}} \,\mathrm{d}z$$

intractable in general

Goals

We want

- richer probabilistic models

Goals

We want

- richer probabilistic models
- complex observation models parameterised by NNs

Goals

We want

- richer probabilistic models
- complex observation models parameterised by NNs

but we can't perform gradient-based MLE

Goals

We want

- richer probabilistic models
- complex observation models
  parameterised by NNs

but we can't perform gradient-based MLE

We need approximate inference techniques!

Wake-sleep Algorithm

- Generalise latent variables to Neural Networks
- Train generative neural model
- Use variational inference! (kind of)

Wake-sleep Architecture

2 Neural Networks:

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units
- Training is performed in a “hard EM” fashion

Wake-sleep Training

Wake Phase

- Use inference network to sample hidden unit setting z from $q(z|x, \lambda)$
- Update generation parameters θ to maximize liklelihood of data given latent state $p(x|z, \theta)$

Wake-sleep Training

Wake Phase

- Use inference network to sample hidden unit setting z from $q(z|x, \lambda)$
- Update generation parameters θ to maximize liklelihood of data given latent state $p(x|z, \theta)$

Sleep Phase

- Produce dream sample $\tilde{x}$ from random hidden unit z
- Update inference parameters λ to maximize probability of latent state $q(z|\tilde{x}, \lambda)$

Wake Phase Objective

Assumes latent state z to be fixed random draws from $q(z|x,\lambda)$.

$$\max_{\theta} \; \mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]$$

Wake Phase Objective

Assumes latent state z to be fixed random draws from $q(z|x, \lambda)$.

$$\max_{\theta} \; \mathbb{E}_{q(z|x,\lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]$$

$$\stackrel{\text{MC}}{\approx} \max_{\theta} \; \frac{1}{S} \sum_{s=1}^{S} \log p(z^{(s)}, x|\theta), \quad z^{(s)} \sim q(z|x, \lambda)$$

Wake Phase Objective

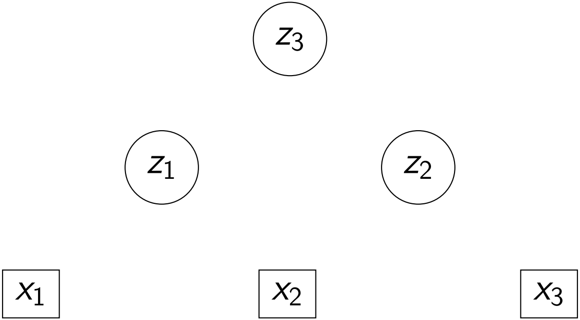
Assumes latent state z to be fixed random draws from $q(z|x, \lambda)$.

$$\max_{\theta} \ \mathbb{E}_{q(z|x,\lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]$$

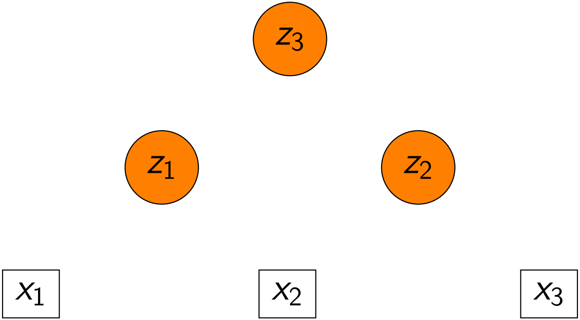
$$\overset{\text{MC}}{\approx} \max_{\theta} \ \frac{1}{S} \sum_{s=1}^{S} \log p(z^{(s)}, x|\theta), \quad z^{(s)} \sim q(z|x, \lambda)$$

This is simply supervised learning with imputed latent data!

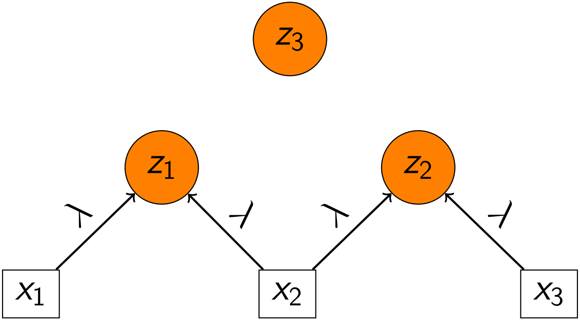
Wake Phase Sampling



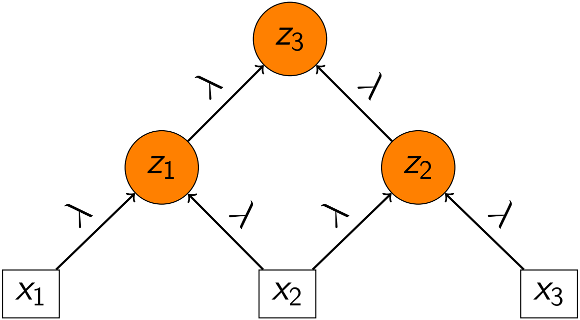
Wake Phase Sampling



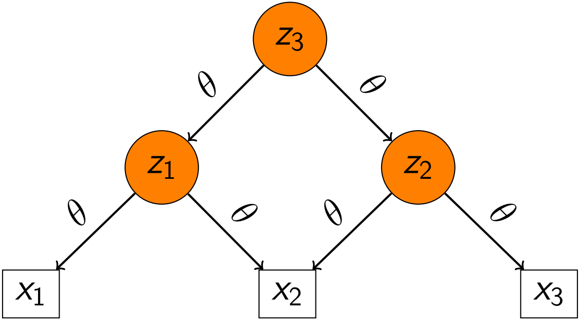
Wake Phase Sampling



Wake Phase Sampling



Wake Phase Update



Sleep Phase Objective

Needed to find alternative objective because

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

is not an expectation!

Sleep Phase Objective

Needed to find alternative objective because

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

is not an expectation!

Idea

Optimize q towards a sample from p.

Sleep Phase Objective

Assumes fake data $\tilde{x}$ and latent variables z to be fixed random draw from $p(x, z|\theta)$.

$$\max_{\lambda} \; \mathbb{E}_{p(\tilde{x}, z|\theta)} \left[\log q(z|\tilde{x}, \lambda)\right] + \mathbb{E}_{p(\tilde{x})} \left[\mathbb{H}\left(p(z|\tilde{x}, \theta)\right)\right]$$

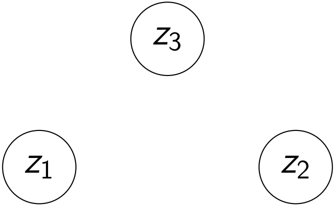
Sleep Phase Objective

Assumes fake data $\tilde{x}$ and latent variables z to be fixed random draw from $p(x, z|\theta)$.

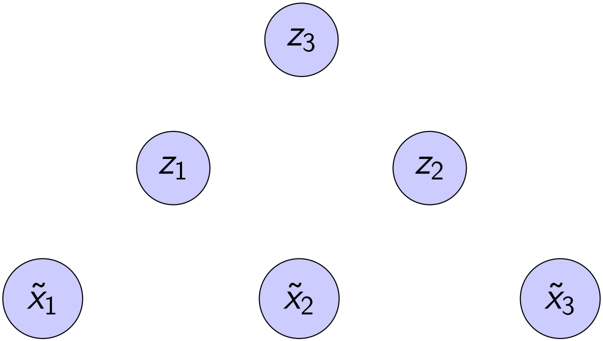
$$\max_{\lambda} \; \mathbb{E}_{p(\tilde{x},z|\theta)}\left[\log q(z|\tilde{x},\lambda)\right] + \mathbb{E}_{p(\tilde{x})}\left[\mathbb{H}\left(p(z|\tilde{x},\theta)\right)\right]$$

$$\overset{\text{MC}}{\approx} \max_{\lambda} \; \frac{1}{S}\sum_{s=1}^{S} \log q(z^{(s)}|\tilde{x}^{(s)},\lambda), \quad (\tilde{x}, z)^{(s)} \sim p(x, z|\theta)$$

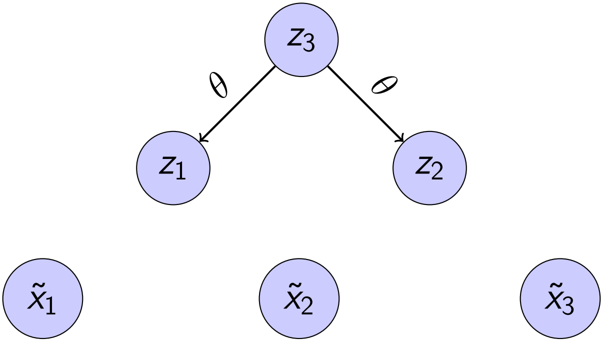
Sleep Phase Sampling



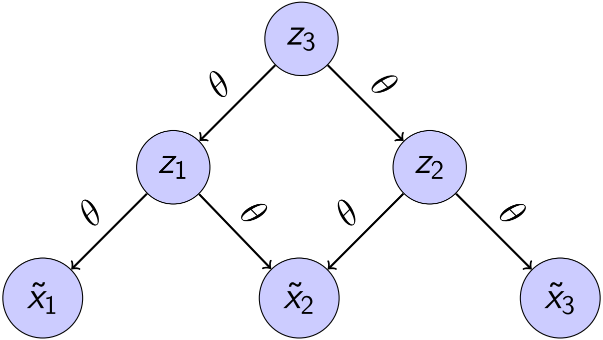
Sleep Phase Sampling



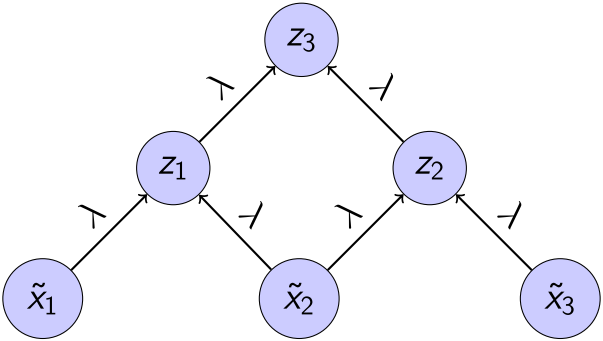
Sleep Phase Sampling



Sleep Phase Sampling



Sleep Phase Update



Wake-sleep Algorithm

Advantages

- Simple layer-wise updates
- Amortised inference: all latent variables are inferred from the same weights λ

Wake-sleep Algorithm

Advantages

- Simple layer-wise updates
- Amortised inference: all latent variables are inferred from the same weights λ

Drawbacks

- Inference and generative networks are trained on different objectives
- Inference weights λ are updated on fake data $\tilde{x}$
- Generative weights are bad initially, giving wrong signal to the updates of λ

Generative Model with NN Likelihood

Goal

Define model $p(x, z|\theta) = p(x|z, \theta)p(z)$ where the likelihood $p(x|z, \theta)$ is given by a neural network. (We fix $p(z)$ for simplicity.)

Generative Model with NN Likelihood

Goal

Define model $p(x, z|\theta) = p(x|z, \theta)p(z)$ where the likelihood $p(x|z, \theta)$ is given by a neural network. (We fix $p(z)$ for simplicity.)

Problem

$p(x) = \int p(x|z, \theta)p(z)\mathrm{d}z$ is hard to compute.

Generative Model with NN Likelihood

Goal

Define model $p(x, z|\theta) = p(x|z, \theta)p(z)$ where the likelihood $p(x|z, \theta)$ is given by a neural network. (We fix $p(z)$ for simplicity.)

Problem

$p(x) = \int \underbrace{p(x|z, \theta)}_{\text{highly non-linear!}} p(z)\mathrm{d}z$ is hard to compute.

Solution: Variational Inference

$$\log p(x|\theta) \geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}}$$

Solution: Variational Inference

$$
\begin{aligned}
\log p(x|\theta) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta) + \log p(Z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)
\end{aligned}
$$

Solution: Variational Inference

$$\begin{aligned}
\log p(x|\theta) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta) + \log p(Z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)
\end{aligned}$$

Solution: Variational Inference

$$\begin{aligned}
\log p(x|\theta) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta) + \log p(Z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)
\end{aligned}$$

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

Solution: Variational Inference

$$
\begin{aligned}
\log p(x|\theta) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta) + \log p(Z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)
\end{aligned}
$$

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

- assume $\mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)$ analytical
  true for exponential families

Solution: Variational Inference

$$
\begin{aligned}
\log p(x|\theta) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, Z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta) + \log p(Z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)
\end{aligned}
$$

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

- assume $\mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)$ analytical
  true for exponential families
- approximate $\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right]$ by sampling
  feasible because $q(z|x,\lambda)$ is simple

Generator Network Gradient

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

Generator Network Gradient

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

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

Generator Network Gradient

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

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

$$\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{i=1}^{S}\frac{\partial}{\partial\theta}\log p(x|z_i,\theta)$$

where $z_i \sim q(z|x,\lambda)$

Generator Network Gradient

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

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

$$\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{i=1}^{S}\frac{\partial}{\partial\theta}\log p(x|z_i,\theta)$$

where $z_i \sim q(z|x,\lambda)$

Note: $q(z|x,\lambda)$ does not depend on θ.

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda}\left[\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right] - \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)\right] \\
=&\frac{\partial}{\partial \lambda}\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right] - \underbrace{\frac{\partial}{\partial \lambda}\mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)}_{\text{analytical computation}}
\end{aligned}
$$

The first term again requires approximation by sampling

Inference Network Gradient

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

Inference Network Gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda} \mathbb{E}_{q(z|x,\lambda)} [\log p(x|z,\theta)] \\
&= \frac{\partial}{\partial \lambda} \sum_{z} q(z|x,\lambda) \log p(x|z,\theta)
\end{aligned}
$$

Inference Network Gradient

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

Inference Network Gradient

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

Not an expectation!

Back to Basic Calculus

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

Back to Basic Calculus

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

Back to Basic Calculus

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

Consequence

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

Score Function Estimator

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

Score Function Estimator

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

Apply this to the ELBO derivative.

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

Score Function Estimator

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

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

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

Apply this to the ELBO derivative.

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

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

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

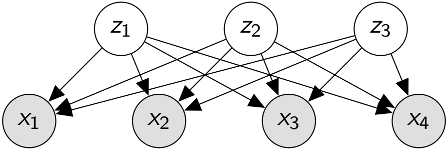
Example Model

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.

$$\begin{aligned} Z_j &\sim \text{Bernoulli}(\phi) & (1 \le j \le k) \\ X_i | z &\sim \text{Categorical}(g(z)) & (1 \le i \le n) \end{aligned}$$

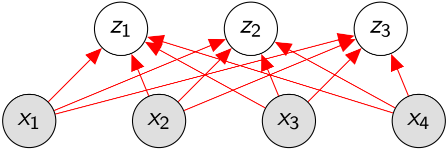
Here $g(\cdot)$ is a function computed by neural network with softmax output.

Example Model



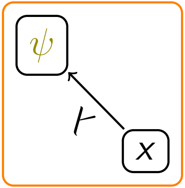
At inference time the latent variables are marginally dependent. For our variational distribution we are going to assume that they are not (recall: mean field assumption).

Inference Network



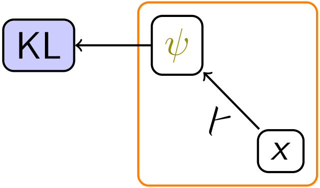
The inference network needs to predict k Bernoulli parameters ψ. Any neural network with sigmoid output will do that job.

Computation Graph



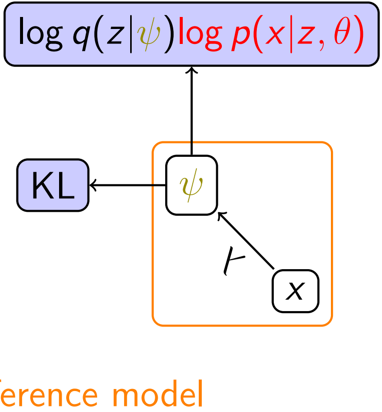
inference model

Computation Graph

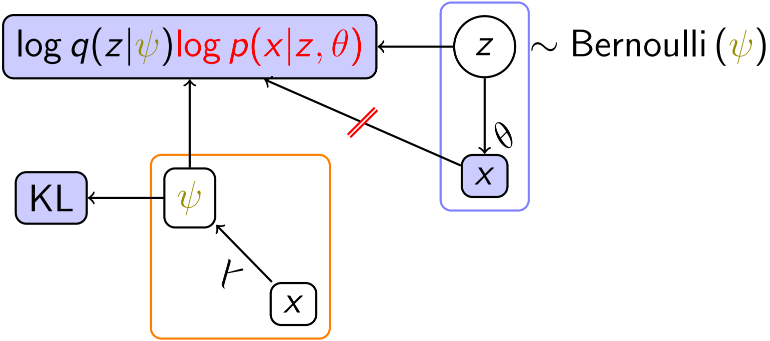


inference model

Computation Graph



Computation Graph



inference model

generation model

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

Fact

The Expectation of the score function is 0.

Baselines

Fact

The Expectation of the score function is 0.

$$\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] = 0$$

Baselines

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

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

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

Baselines

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

Baselines

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

$$\underbrace{\mathbb{E}_{q(z|\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|\lambda)\log p(x|z,\theta)\right]}_{\text{score function gradient}} -$$

Baselines

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

$$\underbrace{\mathbb{E}_{q(z|\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|\lambda)\log p(x|z,\theta)\right]}_{\text{score function gradient}} -$$

$$\underbrace{\mathbb{E}_{q(z|\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|\lambda)\right]}_{0} C$$

Baselines

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

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

Baselines

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

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

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.

$$\min_{\omega} \left(C(x; \omega) - \log p(x|z, \theta) \right)^2$$

Summary

Summary

- Differentiating ELBO wrt λ does not yield an expectation.

Summary

- Differentiating ELBO wrt λ does not yield an expectation.
- Use score function estimator.

Summary

- Differentiating ELBO wrt λ does not yield an expectation.
- Use score function estimator.
- High variance.

Summary

- Differentiating ELBO wrt λ does not yield an expectation.
- Use score function estimator.
- High variance.
- Always use baselines for variance reduction!

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.