Deep Generative Models: Discrete Latente Variables

Philip Schulz and Wilker Aziz

https://github.com/philschulz/VITutorial

First Attempt: Wake-Sleep

Neural Variational Inference

Score function estimator

Variance reduction

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

$$p(x|\theta) = \int p(x, z|\theta) \, \mathrm{d}z = \int p(z)p(x|z, \theta) \, \mathrm{d}z$$

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!

First Attempt: Wake-Sleep

Neural Variational Inference

Score function estimator

Variance reduction

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:

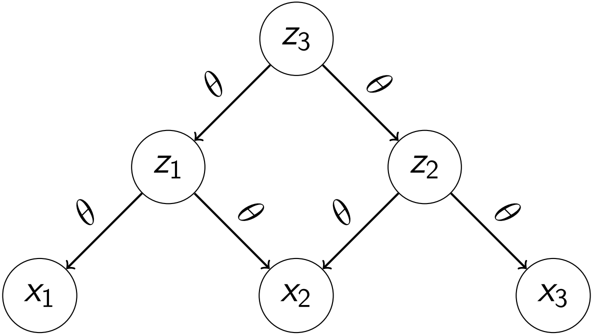
- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units

Wake-sleep Architecture

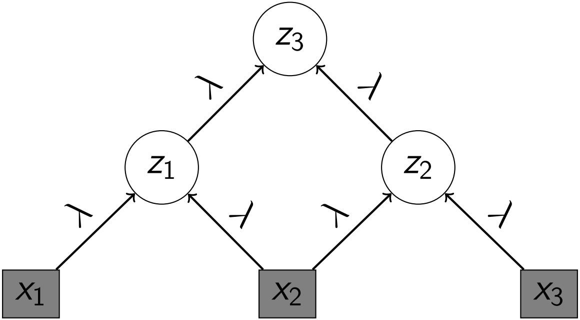
2 Neural Networks:

- 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

Generator



Inference Network



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 join log-liklelihood of data and latents $p(x, z|\theta)$

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 join log-liklelihood of data and latents $p(x, z|\theta)$

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

Objective

$$\arg\min_{\theta} \mathrm{KL}\left(q(z|x,\lambda) \;||\; p(z|x,\theta)\right)$$

Wake Phase Objective

Objective

$$\begin{aligned}
&\arg\min_{\theta} \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z|x,\theta)\right) \\
&= \arg\max_{\theta} \ \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{G}(\theta)}
\end{aligned}$$

Wake Phase Objective

Objective

$$
\begin{aligned}
&\arg\min_{\theta} \mathrm{KL}\left(q(z|x,\lambda) \,||\, p(z|x,\theta)\right) \\
&= \arg\max_{\theta} \; \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{G}(\theta)}
\end{aligned}
$$

Gradient estimate

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

Wake Phase Objective

Objective

$$\arg\min_{\theta} \mathrm{KL}\left(q(z|x,\lambda) \;||\; p(z|x,\theta)\right)$$
$$= \arg\max_{\theta} \; \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{G}(\theta)}$$

Gradient estimate

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

Wake Phase Objective

Objective

$$\arg\min_\theta \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z|x,\theta)\right)$$
$$= \arg\max_\theta \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{G}(\theta)}$$

Gradient estimate

$$\nabla_\theta \mathcal{G}(\theta) = \nabla_\theta \mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \nabla_\theta \mathbb{H}[q(z|x,\lambda)]$$
$$= \mathbb{E}_{q(z|x,\lambda)}\left[\nabla_\theta \log p(z,x|\theta)\right]$$
$$\overset{\text{MC}}{\approx} \nabla_\theta \log p(z,x|\theta) \quad \text{where } z \sim q(z|x,\lambda)$$

Wake Phase Objective

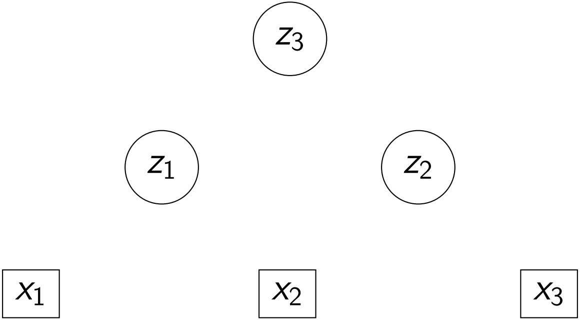
Assumes latent state z to be fixed random draws from $q(z|x, \lambda)$.

$$\arg\min_{\theta} \mathrm{KL}\left(q(z|x, \lambda) \,||\, p(z|x, \theta)\right)$$
$$\overset{\mathrm{MC}}{\approx} \arg\max_{\theta} \log p(z, x|\theta)$$

This is simply supervised learning with imputed latent data!

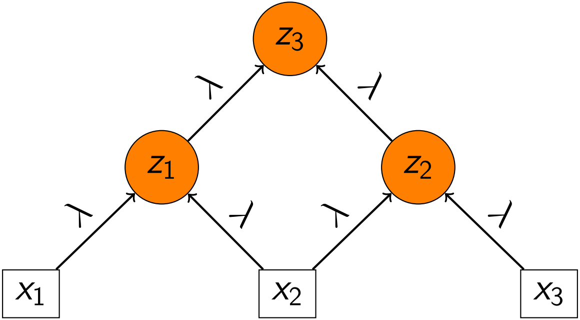
Wake Phase Sampling

Sampling $z \sim q(z|x, \lambda)$



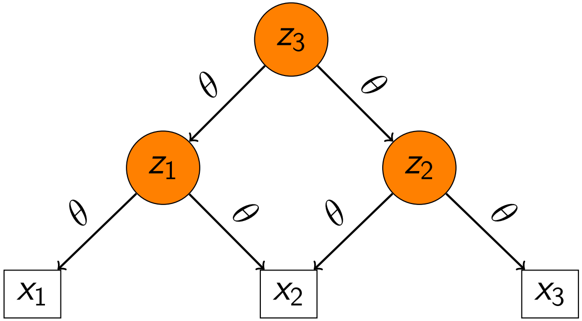
Wake Phase Sampling

Sampling $z \sim q(z|x, \lambda)$



Wake Phase Update

Update θ



Sleep Phase Objective

Objective

$$\arg\min_{\lambda} \mathrm{KL}\left(q(z|x,\lambda) \,||\, p(z|x,\theta)\right)$$

Sleep Phase Objective

Objective

$$\arg\min_{\lambda} \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z|x,\theta)\right)$$
$$= \arg\max_{\lambda} \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{R}(\lambda)}$$

Sleep Phase Objective

Objective

$$
\begin{aligned}
&\arg\min_{\lambda} \mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z|x,\theta)\right) \\
&= \arg\max_{\lambda} \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}\left[q(z|x,\lambda)\right]}_{\mathcal{R}(\lambda)}
\end{aligned}
$$

Gradient estimate

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

Sleep Phase Objective

Objective

$$
\begin{aligned}
&\arg\min_{\lambda} \mathrm{KL}\left(q(z|x,\lambda)\;||\;p(z|x,\theta)\right)\\
&= \arg\max_{\lambda}\;\underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]}_{\mathcal{R}(\lambda)}
\end{aligned}
$$

Gradient estimate

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

Let's change the objective!

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$
\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{p(x)} \left[\mathrm{KL}\left(p(z|x,\theta) \;||\; q(z|x,\lambda)\right)\right] \\
&= \arg\min_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right] \\
&= \arg\max_{\lambda} \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right]}_{\mathcal{R}(\lambda)} - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta)\right]}_{\text{constant}}
\end{aligned}
$$

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$
\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{p(x)} \left[\mathrm{KL}\left(p(z|x,\theta) \,||\, q(z|x,\lambda) \right) \right] \\
&= \arg\min_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda) \right] \\
&= \arg\max_{\lambda} \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda) \right]}_{\mathcal{R}(\lambda)} - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) \right]}_{\text{constant}}
\end{aligned}
$$

Gradient

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{p(x)} \left[\mathrm{KL}\left(p(z|x,\theta) \,||\, q(z|x,\lambda) \right) \right] \\
&= \arg\min_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda) \right] \\
&= \arg\max_{\lambda} \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda) \right]}_{\mathcal{R}(\lambda)} - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) \right]}_{\text{constant}}
\end{aligned}$$

Gradient

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

Sleep Phase (Convenient) Objective

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

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

Sleep Phase (Convenient) Objective

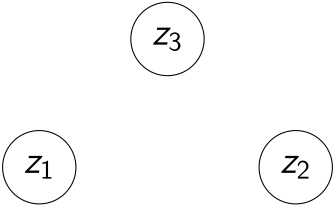
Assumes fake data $\tilde{x}$ and latent variables z to be fixed random draws from $p(x, z|\theta)$.

$$\begin{aligned} &\arg\max_{\lambda} \; \mathbb{E}_{p(x,z|\theta)} [\log q(z|x, \lambda)] \\ \overset{\text{MC}}{\approx} \; &\arg\max_{\lambda} \; \log q(z|\tilde{x}, \lambda) \end{aligned}$$

where $z \sim p(z)$ and $\tilde{x} \sim p(x|z)$ (fake data!)

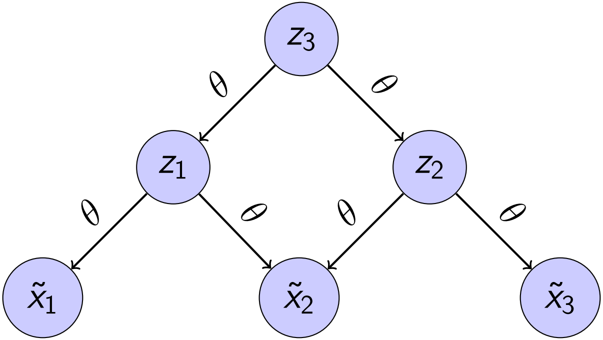
Sleep Phase Sampling

Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$



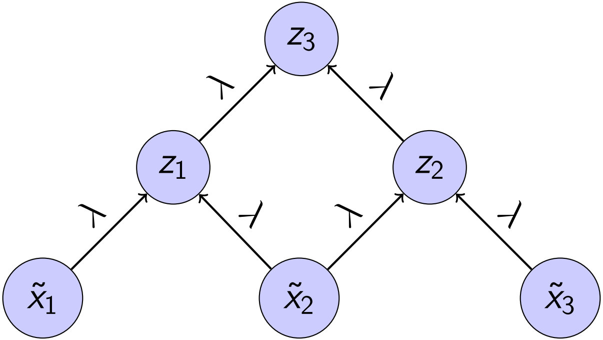
Sleep Phase Sampling

Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$



Sleep Phase Update

Update λ



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

Variational Inference Learning (NVIL)

Generative model with NN likelihood

Variational Inference Learning (NVIL)

Generative model with NN likelihood

Let us consider a latent factor model for topic modelling:

Variational Inference Learning (NVIL)

Generative model with NN likelihood

Let us consider a latent factor model for topic modelling:

- a document $x = (x_1, \ldots, x_N)$ consists of n i.i.d. categorical draws from that model

Variational Inference Learning (NVIL)

Generative model with NN likelihood

Let us consider a latent factor model for topic modelling:

- a document $x = (x_1, \ldots, x_N)$ consists of n i.i.d. categorical draws from that model
- the categorical distribution in turn depends on binary latent factors $z = (z_1, \ldots, z_K)$ which are also i.i.d.

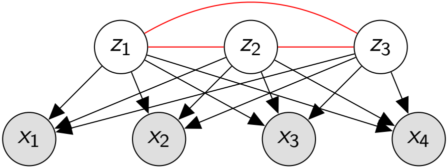
Latent factor model

$$
\begin{aligned}
Z_j &\sim \text{Bernoulli}\,(\phi) && (1 \leq k \leq K) \\
X_i|z &\sim \text{Categorical}\,(f(z;\theta)) && (1 \leq i \leq N)
\end{aligned}
$$

Here $0 < \phi < 1$ specifies a Bernoulli prior and $f(\cdot;\theta)$ is a function computed by a neural network with softmax output, e.g.

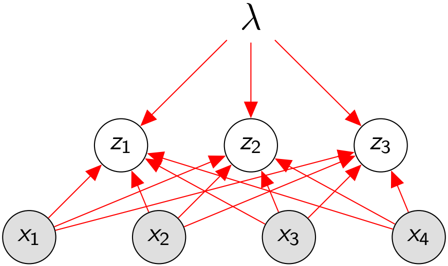
$$
\begin{aligned}
f(z;\theta) &= \text{softmax}(Wz + b) \\
\theta &= \{W, b\}
\end{aligned}
$$

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).

Mean Field Inference



The inference network needs to predict K Bernoulli parameters b_1^K. Any neural network with sigmoid output will do that job.

Inference Network

$$q(z|x,\lambda) = \prod_{k=1}^{K} \text{Bern}(z_k|b_k)$$

$$\text{where } b_1^K = g(x;\lambda)$$

Example architecture

$$h = \frac{1}{N}\sum_{i=1}^{N} E_{x_i} \qquad b_1^K = \text{sigmoid}(Mh + c)$$

$$\lambda = \{E, M, c\}$$

Objective

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

Parameter estimation

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

KL

KL between K independent Bernoulli distributions is tractable

$$\mathrm{KL}\left(q(z|x,\lambda) \,||\, p(z|\phi)\right) = \sum_{k=1}^{K} \mathrm{KL}\left(q(z_k|x,\lambda) \,||\, p(z_k|\phi)\right)$$

$$= \sum_{k=1}^{K} b_k \log \frac{b_k}{\phi} + (1-b_k) \log \frac{1-b_k}{1-\phi}$$

Generative Network Gradient

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

Generative Network Gradient

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

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

Generative Network Gradient

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

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

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

Inference Network Gradient

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

Inference Network Gradient

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

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

Inference Network Gradient

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

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

The first term again requires approximation by sampling, but there is a problem

Inference Network Gradient

$$\frac{\partial}{\partial \lambda} \mathbb{E}_{q_\lambda(z|x)} [\log p_\theta(x|z)]$$

Inference Network Gradient

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

Inference Network Gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda} \mathbb{E}_{q_\lambda(z|x)} \left[\log p_\theta(x|z)\right] \\
&= \frac{\partial}{\partial \lambda} \sum_z q(z|x,\lambda) \log p(x|z,\theta) \\
&= \underbrace{\sum_z \frac{\partial}{\partial \lambda}(q(z|x,\lambda)) \log p(x|z,\theta)}_{\text{not an expectation}}
\end{aligned}
$$

Inference Network Gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda} \mathbb{E}_{q_\lambda(z|x)} \left[\log p_\theta(x|z)\right] \\
&= \frac{\partial}{\partial \lambda} \sum_z q(z|x,\lambda) \log p(x|z,\theta) \\
&= \underbrace{\sum_z \frac{\partial}{\partial \lambda}(q(z|x,\lambda)) \log p(x|z,\theta)}_{\text{not an expectation}}
\end{aligned}
$$

- MC estimator is non-differentiable

Inference Network Gradient

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda} \mathbb{E}_{q_\lambda(z|x)} \left[\log p_\theta(x|z)\right] \\
&= \frac{\partial}{\partial \lambda} \sum_z q(z|x,\lambda) \log p(x|z,\theta) \\
&= \underbrace{\sum_z \frac{\partial}{\partial \lambda}(q(z|x,\lambda)) \log p(x|z,\theta)}_{\text{not an expectation}}
\end{aligned}
$$

- MC estimator is non-differentiable
- Differentiating the expression does not yield an expectation: cannot approximate via MC

Score function estimator

We can again use the log identity for derivatives

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{q_\lambda(z|x)}\left[\log p_\theta(x|z)\right]$$

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

Score function estimator

We can again use the log identity for derivatives

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{q_\lambda(z|x)}\left[\log p_\theta(x|z)\right]$$

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

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

Score function estimator

We can again use the log identity for derivatives

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

Score function estimator: remarks

We can now build an MC estimator

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

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

Score function estimator: remarks

We can now build an MC estimator

$$\begin{aligned}
&\frac{\partial}{\partial \lambda}\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right] \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\frac{\partial}{\partial \lambda}\log q(z|x,\lambda)\right] \\
&\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{s=1}^{S} \log p(x|z^{(s)},\theta)\frac{\partial}{\partial \lambda}\log q(z^{(s)}|x,\lambda)
\end{aligned}$$

where $z^{(s)} \sim q(z|x,\lambda)$

Score function estimator: remarks

We can now build an MC estimator

$$\frac{\partial}{\partial \lambda}\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right]$$
$$= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\frac{\partial}{\partial \lambda}\log q(z|x,\lambda)\right]$$
$$\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{s=1}^{S}\log p(x|z^{(s)},\theta)\frac{\partial}{\partial \lambda}\log q(z^{(s)}|x,\lambda)$$

where $z^{(s)} \sim q(z|x,\lambda)$

- magnitude of $\log p(x|z,\theta)$ varies widely

Score function estimator: remarks

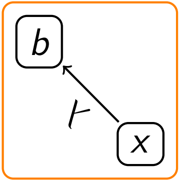
We can now build an MC estimator

$$\frac{\partial}{\partial \lambda}\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right]$$
$$= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\frac{\partial}{\partial \lambda}\log q(z|x,\lambda)\right]$$
$$\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{s=1}^{S} \log p(x|z^{(s)},\theta)\frac{\partial}{\partial \lambda}\log q(z^{(s)}|x,\lambda)$$

where $z^{(s)} \sim q(z|x,\lambda)$

- magnitude of $\log p(x|z,\theta)$ varies widely
- model likelihood does not contribute to direction of gradient

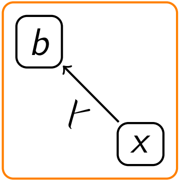
Computation Graph



inference model

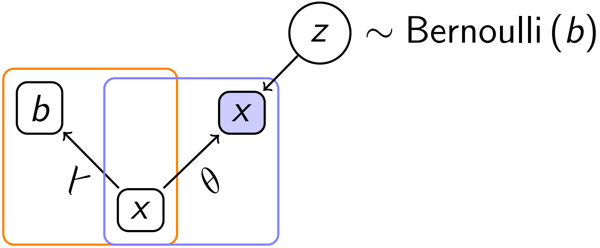
Computation Graph

$z \sim \text{Bernoulli}\,(b)$



inference model

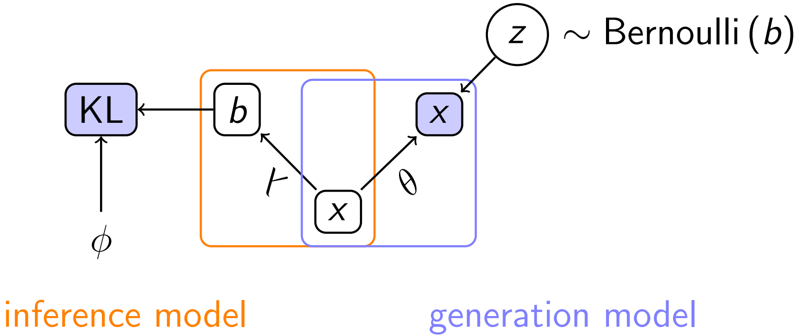
Computation Graph



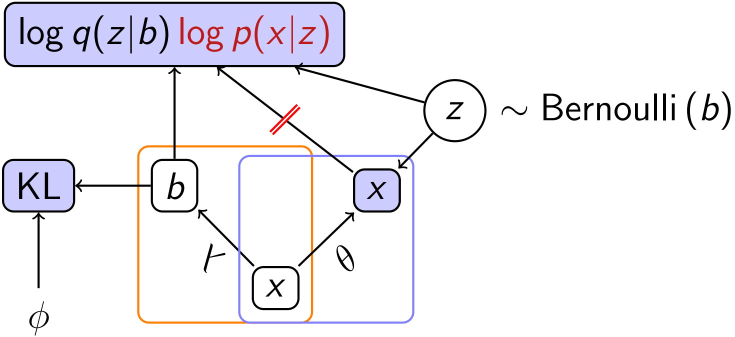
inference model

generation 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!

First Attempt: Wake-Sleep

Neural Variational Inference

Score function estimator

Variance reduction

When variance is high we can

When variance is high we can

- sample more

When variance is high we can

- sample more
- use variance reduction techniques (e.g. baselines and control variates)

Control variates

Suppose we want to estimate $\mathbb{E}[f(Z)]$

$$\hat{f} \overset{\text{MC}}{\approx} \frac{1}{S} \sum_{s=1}^{S} f(z^{(s)})$$

and we know the expected value of another function $\psi(z)$ on the same support.

Control variates

Suppose we want to estimate $\mathbb{E}[f(Z)]$

$$\hat{f} \overset{\text{MC}}{\approx} \frac{1}{S} \sum_{s=1}^{S} f(z^{(s)})$$

and we know the expected value of another function $\psi(z)$ on the same support.

It holds that

$$\mathbb{E}[f(Z)] = \mathbb{E}[f(Z) - \psi(Z)] + \mathbb{E}[\psi(Z)]$$

Variance reduction

$$\hat{d} = \frac{1}{S} \left(\sum_{s=1}^{S} f(z^{(s)}) - \psi(z^{(s)}) \right) + \underbrace{\mathbb{E}[\psi(Z)]}_{\mu_\psi}$$

In general

$$\mathrm{Var}(\hat{d}) = \mathrm{Var}(\hat{f}) - 2\,\mathrm{Cov}(\hat{f}, \hat{\psi}) + \underbrace{\mathrm{Var}(\mu_\psi)}_{0}$$

If f and ψ are strongly correlated, then we improve on the original estimation problem.

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

- Wake-Sleep: train inference and generation networks with separate objectives

Summary

- Wake-Sleep: train inference and generation networks with separate objectives
- NVIL: a single objective (ELBO) for both models

Summary

- Wake-Sleep: train inference and generation networks with separate objectives
- NVIL: a single objective (ELBO) for both models
- Use score function estimator

Summary

- Wake-Sleep: train inference and generation networks with separate objectives
- NVIL: a single objective (ELBO) for both models
- Use score function estimator
- Always use baselines for variance reduction!

Literature I

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.

Literature II

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995. URL http://www.gatsby.ucl.ac.uk/~dayan/papers/hdfn95.pdf.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1791–II–1799.

Literature III

JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3045092.

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.

Literature IV

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.