

Deep Generative Models: Discrete Latent Variables

Bryan Eikema and Wilker Aziz

<https://vitutorial.github.io/tour/ua2020>



UNIVERSITY OF AMSTERDAM
Institute for Logic, Language and Computation



- 1 First Attempt: Wake-Sleep
- 2 Neural Variational Inference
- 3 Score function estimator
- 4 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

$$p(x|\theta) = \int p(x, z|\theta) \, dz = \int p(z)p(x|z, \theta) \, dz$$

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!

ELBO recap

And we've developed the ELBO

$$\log p(x) = \log \int p(z, x) dz$$

ELBO recap

And we've developed the ELBO

$$\log p(x) = \log \int p(z, x) dz = \log \int q(z|x) \frac{p(z, x)}{q(z|x)} dz$$

ELBO recap

And we've developed the ELBO

$$\begin{aligned}\log p(x) &= \log \int p(z, x) dz = \log \int q(z|x) \frac{p(z, x)}{q(z|x)} dz \\ &\stackrel{\text{JL}}{\geq} \underbrace{\mathbb{E}_{q(z|x)} \left[\log \frac{p(z, x)}{q(z|x)} \right]}_{\text{ELBO}}\end{aligned}$$

ELBO recap

And we've developed the ELBO

$$\log p(x) = \log \int p(z, x) dz = \log \int q(z|x) \frac{p(z, x)}{q(z|x)} dz$$

$$\stackrel{\text{JL}}{\geq} \underbrace{\mathbb{E}_{q(z|x)} \left[\log \frac{p(z, x)}{q(z|x)} \right]}_{\text{ELBO}}$$

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_{q(z|x)} \left[\log \frac{p(z|x)p(x)}{q(z|x)} \right] \\ &= - \underbrace{\text{KL} (q(z|x) \parallel p(z|x))}_{\text{gap}} + \log p(x) \end{aligned}$$

- 1 First Attempt: Wake-Sleep
- 2 Neural Variational Inference
- 3 Score function estimator
- 4 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:

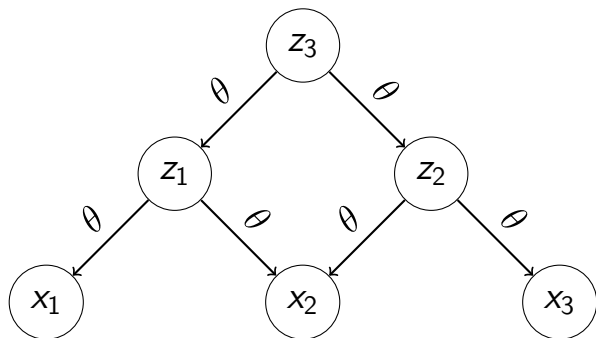
- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units

Wake-sleep Architecture

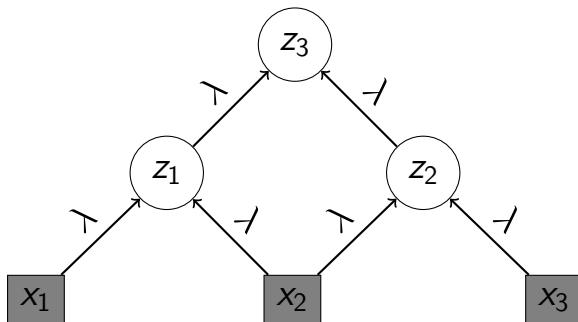
2 Neural Networks:

- 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 joint log-likelihood 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 joint log-likelihood 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

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

Wake Phase Objective

Objective

$$\begin{aligned} \arg \max_{\theta} \mathbb{E}_{p(x)} [\text{ELBO}(\theta, \lambda|x)] \\ = \arg \max_{\theta} \mathbb{E}_{p(x)} [\mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]] \end{aligned}$$

Gradient wrt θ for $x \sim p(x)$

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

Wake Phase Objective

Objective

$$\begin{aligned} \arg \max_{\theta} \mathbb{E}_{p(x)} [\text{ELBO}(\theta, \lambda|x)] \\ = \arg \max_{\theta} \mathbb{E}_{p(x)} [\mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]] \end{aligned}$$

Gradient wrt θ for $x \sim p(x)$

$$\begin{aligned} \nabla_{\theta} \mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \nabla_{\theta} \mathbb{H}[q(z|x, \lambda)] \\ = \mathbb{E}_{q(z|x, \lambda)} [\nabla_{\theta} \log p(z, x|\theta)] \end{aligned}$$

Wake Phase Objective

Objective

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

Gradient wrt θ for $x \sim p(x)$

$$\begin{aligned} & \nabla_{\theta} \mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \nabla_{\theta} \mathbb{H}[q(z|x, \lambda)] \\ &= \mathbb{E}_{q(z|x, \lambda)} [\nabla_{\theta} \log p(z, x|\theta)] \\ &\stackrel{\text{MC}}{\approx} \nabla_{\theta} \log p(z, x|\theta) \quad \text{where } z \sim q(z|x, \lambda) \end{aligned}$$

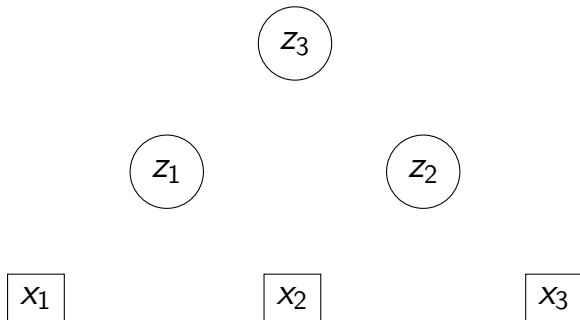
Wake Phase Objective

Assumes z to be fixed random draw from $q(z|x, \lambda)$
and maximises $\log p(z, x|\theta)$.

This is simply supervised learning with imputed latent data!

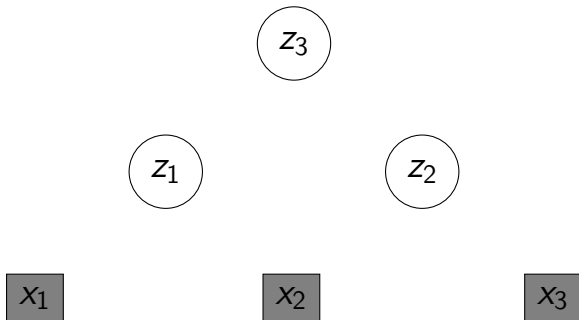
Wake Phase Sampling

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



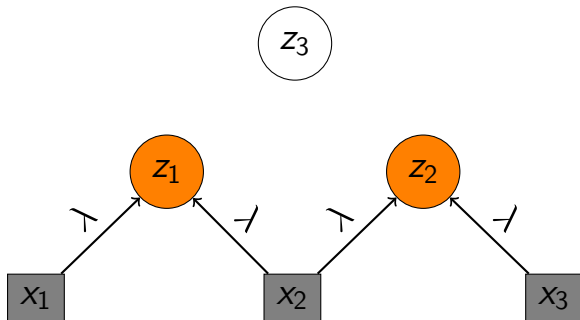
Wake Phase Sampling

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



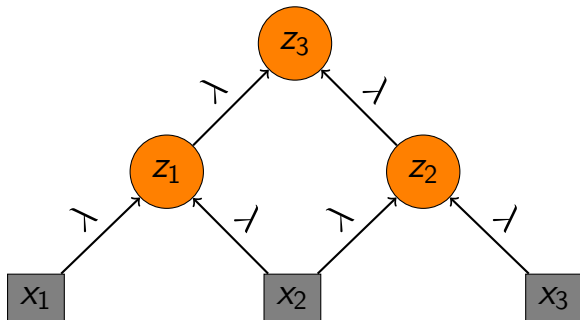
Wake Phase Sampling

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



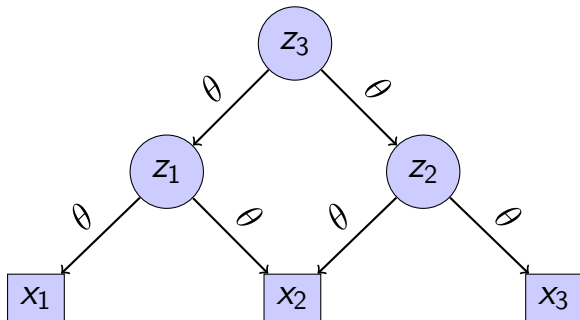
Wake Phase Sampling

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



Wake Phase Update

Compute $\log p(x, z|\theta)$ and update θ



Sleep Phase Objective

Objective

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

Sleep Phase Objective

Objective

$$\begin{aligned} \arg \max_{\lambda} \mathbb{E}_{p(x)} [\text{ELBO}(\theta, \lambda|x)] \\ = \arg \max_{\lambda} \mathbb{E}_{p(x)} [\mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]] \end{aligned}$$

Gradient wrt λ for $x \sim p(x)$

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

Sleep Phase Objective

Objective

$$\begin{aligned} \arg \max_{\lambda} \mathbb{E}_{p(x)} [\text{ELBO}(\theta, \lambda|x)] \\ = \arg \max_{\lambda} \mathbb{E}_{p(x)} [\mathbb{E}_{q(z|x, \lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]] \end{aligned}$$

Gradient wrt λ for $x \sim p(x)$

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

Let's change the objective!

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\begin{aligned} & \arg \min_{\lambda} \mathbb{E}_{p(x)} [\text{KL} (p(z|x, \theta) || q(z|x, \lambda))] \\ &= \arg \min_{\lambda} \mathbb{E}_{p(x)} \mathbb{E}_{p(z|x, \theta)} [\log p(z|x, \theta) - \log q(z|x, \lambda)] \end{aligned}$$

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Gradient wrt λ

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Gradient wrt λ

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Gradient wrt λ

$$\begin{aligned}
 & \nabla_{\lambda} \mathbb{E}_{p(x, z|\theta)} [\log q(z|x, \lambda)] \\
 &= \mathbb{E}_{p(x, z|\theta)} [\nabla_{\lambda} \log q(z|x, \lambda)] \\
 &\stackrel{\text{MC}}{\approx} \nabla_{\lambda} \log q(z|\tilde{x}, \lambda) \quad \text{where } z \sim p(z|\theta)
 \end{aligned}$$

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

Sleep Phase (Convenient) Objective

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

$$z \sim p(z|\theta)$$

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

and maximises $\log q(z|\tilde{x}, \lambda)$.

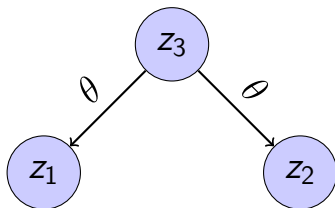
Sleep Phase Sampling

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



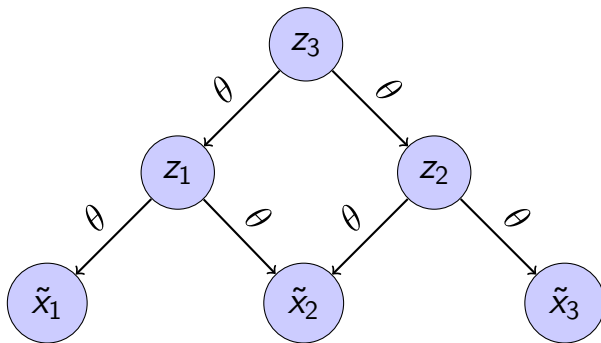
Sleep Phase Sampling

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



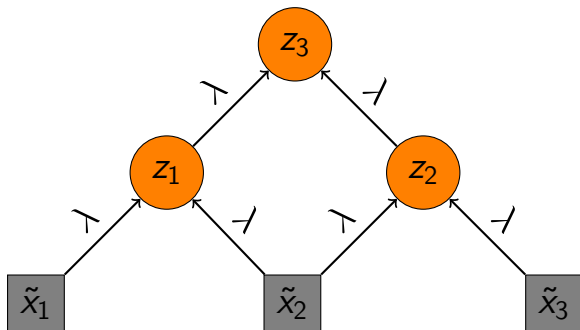
Sleep Phase Sampling

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



Sleep Phase Update

Compute $\log q(z|\tilde{x}, \lambda)$ and 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

- 1 First Attempt: Wake-Sleep
- 2 Neural Variational Inference**
- 3 Score function estimator
- 4 Variance reduction

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

Latent factor model

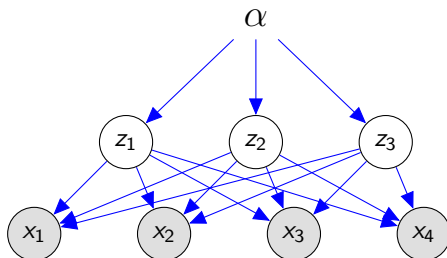
$$Z_j \sim \text{Bernoulli}(\alpha) \quad (1 \leq k \leq K)$$

$$X_i|z \sim \text{Categorical}(f(z; \theta)) \quad (1 \leq i \leq N)$$

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

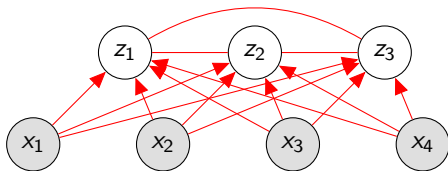
$$f(z; \theta) = \text{softmax}(Wz + b)$$
$$\theta = \{W, b\}$$

Example Model



Joint distribution: independent latent variables

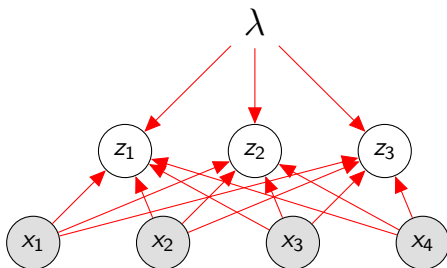
Example Model



Posterior: 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

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

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

Example architecture

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

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

Objective

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

Objective

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

Objective

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

Parameter estimation

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

KL

KL between K independent Bernoulli distributions is tractable

$$\text{KL} (q(z|x, \lambda) \parallel p(z|\alpha)) = \sum_{k=1}^K \text{KL} (q(z_k|x, \lambda) \parallel p(z_k|\alpha))$$

KL

KL between K independent Bernoulli distributions is tractable

$$\begin{aligned}\text{KL}(q(z|x, \lambda) \parallel p(z|\alpha)) &= \sum_{k=1}^K \text{KL}(q(z_k|x, \lambda) \parallel p(z_k|\alpha)) \\ &= \sum_{k=1}^K \text{KL}(\text{Bernoulli}(b_k) \parallel \text{Bernoulli}(\alpha))\end{aligned}$$

KL

KL between K independent Bernoulli distributions is tractable

$$\begin{aligned}\text{KL}(q(z|x, \lambda) \parallel p(z|\alpha)) &= \sum_{k=1}^K \text{KL}(q(z_k|x, \lambda) \parallel p(z_k|\alpha)) \\ &= \sum_{k=1}^K \text{KL}(\text{Bernoulli}(b_k) \parallel \text{Bernoulli}(\alpha)) \\ &= \sum_{k=1}^K b_k \log \frac{b_k}{\alpha} + (1 - b_k) \log \frac{1 - b_k}{1 - \alpha}\end{aligned}$$

Generative Network Gradient

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

Generative Network Gradient

$$\begin{aligned}
 & \frac{\partial}{\partial \theta} \left(\mathbb{E}_{q(z|x, \lambda)} [\log p(x|z, \theta)] - \overbrace{\text{KL}(q(z|x, \lambda) || p(z))}^{\text{constant wrt } \theta} \right) \\
 &= \underbrace{\mathbb{E}_{q(z|x, \lambda)} \left[\frac{\partial}{\partial \theta} \log p(x|z, \theta) \right]}_{\text{expected gradient :)}}
 \end{aligned}$$

Generative Network Gradient

$$\begin{aligned}
 & \frac{\partial}{\partial \theta} \left(\mathbb{E}_{q(z|x, \lambda)} [\log p(x|z, \theta)] - \overbrace{\text{KL}(q(z|x, \lambda) || p(z))}^{\text{constant wrt } \theta} \right) \\
 &= \underbrace{\mathbb{E}_{q(z|x, \lambda)} \left[\frac{\partial}{\partial \theta} \log p(x|z, \theta) \right]}_{\text{expected gradient :)}} \\
 &\stackrel{\text{MC}}{\approx} \frac{1}{S} \sum_{s=1}^S \frac{\partial}{\partial \theta} \log p(x|z^{(s)}, \theta) \quad \text{where } z^{(s)} \sim q(z|x, \lambda)
 \end{aligned}$$

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

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

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

Inference Network Gradient

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

Inference Network Gradient

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

- MC estimator is non-differentiable

Inference Network Gradient

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

- 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

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

Score function estimator

We can again use the log identity for derivatives

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

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

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

Score function estimator: remarks

We can now build an MC estimator

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

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

Score function estimator: remarks

We can now build an MC estimator

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

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

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

Score function estimator: remarks

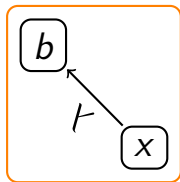
We can now build an MC estimator

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

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

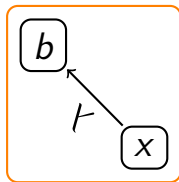
- magnitude of $\log p(x|z, \theta)$ varies widely
- model likelihood does not contribute to direction of gradient

Computation Graph



inference model

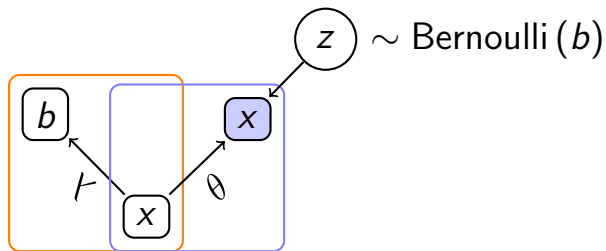
Computation Graph



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

inference model

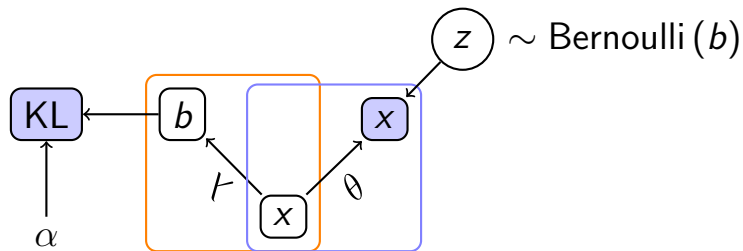
Computation Graph



inference model

generation model

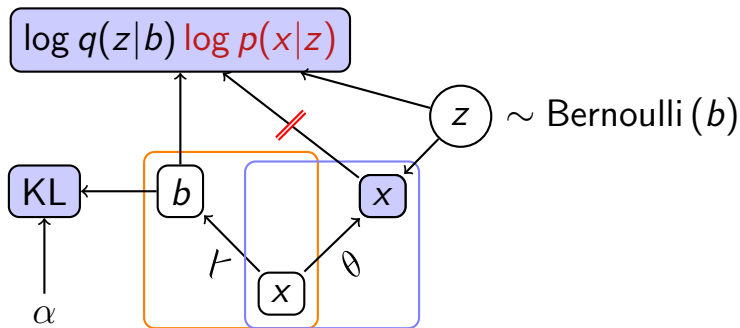
Computation Graph



inference model

generation model

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!

- 1 First Attempt: Wake-Sleep
- 2 Neural Variational Inference
- 3 Score function estimator
- 4 **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

Intuition

To estimate $\mathbb{E}[f(z)]$ via Monte Carlo we compute the empirical average of $\hat{f}(z)$ where $\hat{f}(z)$ is chosen so that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$ and $\text{Var}(f) > \text{Var}(\hat{f})$.

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
it holds that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
it holds that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$
- and $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

Choosing the control variate

- 1 $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
- 2 $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

How do we choose b and $c(z)$?

Choosing the control variate

- 1 $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
- 2 $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance

Choosing the control variate

- 1 $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
- 2 $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b} \text{Var}(\hat{f}) = 0$

Choosing the control variate

- 1 $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
- 2 $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b} \text{Var}(\hat{f}) = 0$ yields $b^* = \text{Cov}(f, c) / \text{Var}(c)$

Choosing the control variate

- 1 $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
- 2 $\text{Var}(\hat{f}) = \text{Var}(f) - 2b \text{Cov}(f, c) + b^2 \text{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b} \text{Var}(\hat{f}) = 0$ yields $b^* = \text{Cov}(f, c) / \text{Var}(c)$

Of course, $\mathbb{E}[c(z)]$ must be known!

MC

We then use the estimate

$$\bar{f} \stackrel{\text{MC}}{\approx} \frac{1}{S} \left(\sum_{s=1}^S f(z^{(s)}) - bc(z^{(s)}) \right) + b\bar{c}$$

MC

We then use the estimate

$$\bar{f} \stackrel{\text{MC}}{\approx} \frac{1}{S} \left(\sum_{s=1}^S f(z^{(s)}) - bc(z^{(s)}) \right) + b\bar{c}$$

And recall that for us

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

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

Expected score

The Expectation of the score function is 0.

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

Expected score

The Expectation of the score function is 0.

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

Expected score

The Expectation of the score function is 0.

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

Expected score

The Expectation of the score function is 0.

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

Expected score

The Expectation of the score function is 0.

$$\begin{aligned} & \mathbb{E}_{q(z|x, \lambda)} \left[\frac{\partial}{\partial \lambda} \log q(z|x, \lambda) \right] \\ &= \int q(z|x, \lambda) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda) dz \\ &= \int \frac{\partial}{\partial \lambda} q(z|x, \lambda) dz \\ &= \frac{\partial}{\partial \lambda} \int q(z|x, \lambda) dz \\ &= \frac{\partial}{\partial \lambda} 1 = 0 \end{aligned}$$

Baselines

With

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

and

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

we have

$$\hat{f}(z) =$$

Baselines

With

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

and

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

we have

$$\hat{f}(z) = (\log p(x|z, \theta) - b) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

Baselines

With

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

and

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

we have

$$\hat{f}(z) = (\log p(x|z, \theta) - b) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

b is known as *baseline* in RL literature.

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x; \omega)$

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x; \omega)$
- The likelihood assessed at a deterministic point, e.g. $b(x) = \log p(x|z^*, \theta)$ where $z^* = \arg \max_z q(z|x, \lambda)$

Trainable baselines

Baselines are predicted by a regression model (e.g. a neural net).

The model is trained using an L_2 -loss.

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

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!

Implementation

Check one of our notebooks, e.g.

- inducing rationales for sentiment classification
github.com/vitutorial/exercises/tree/master/SST

Literature I

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

Literature II

John W. Paisley, David M. Blei, and Michael I. Jordan.
Variational bayesian inference with stochastic search.
In *ICML*, 2012. URL

<http://icml.cc/2012/papers/687.pdf>.

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](http://proceedings.mlr.press/v33/ranganath14.pdf).

Literature III

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>.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter.

Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih.

Muprop: Unbiased backpropagation for stochastic neural networks. In *ICLR*, 2016.

Literature IV

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636, 2017.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.