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

gourmet

Discrete Latent Variables

- Language is an inherently discrete structure.

Discrete Latent Variables

- Language is an inherently discrete structure.
- Many structures used to describe language are discrete, e.g. trees, graphs, tags, etc.

1. First Attempt: Wake-Sleep
2. Neural Variational Inference and Learning
3. Score Function Estimator

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 neural network 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 neural network 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 neural networks

Goals

We want

- richer probabilistic models
- complex observation models
  parameterised by neural networks

but we can't perform gradient-based MLE

Goals

We want

- richer probabilistic models
- complex observation models
  parameterised by neural networks

but we can't perform gradient-based MLE

We need approximate inference techniques!

ELBO recap

And we've developed the ELBO

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

ELBO recap

And we've developed the ELBO

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

ELBO recap

And we've developed the ELBO

$$\log p(x) = \log \int p(z,x)\mathrm{d}z \;= \log \int q(z|x)\frac{p(z,x)}{q(z|x)}\mathrm{d}z$$
$$\overset{\text{JI}}{\geq} \underbrace{\mathbb{E}_{q(z|x)}\left[\log \frac{p(z,x)}{q(z|x)}\right]}_{\text{ELBO}}$$

ELBO recap

And we've developed the ELBO

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

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

$$\text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log \frac{p(z|x)p(x)}{q(z|x)}\right]$$

ELBO recap

And we've developed the ELBO

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

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

$$\text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log \frac{p(z|x)p(x)}{q(z|x)}\right]$$

$$= -\underbrace{\text{KL}\left(q(z|x) \mid\mid p(z|x)\right)}_{\text{gap}} + \log p(x)$$

Wake-Sleep Algorithm

- Generalise latent variables to neural networks.
- Train generative neural model.
- Use variational inference! (kind of)
- Hinton et al. (1995)

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:

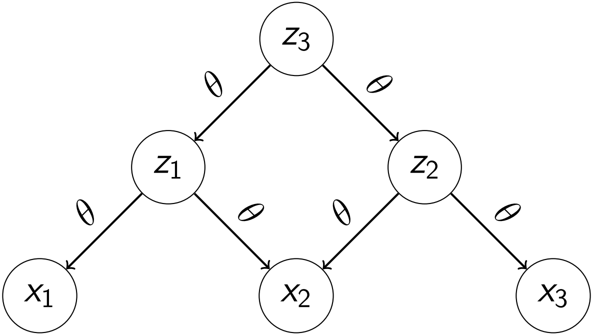
- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units

Wake-Sleep Architecture

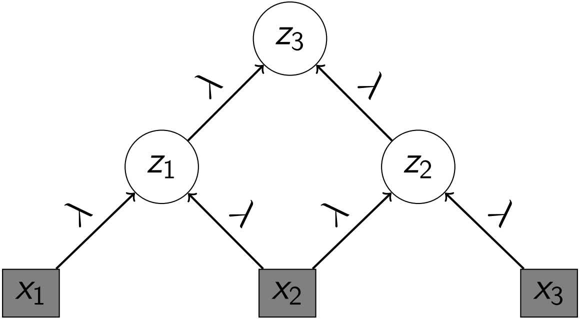
2 neural networks:

- 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

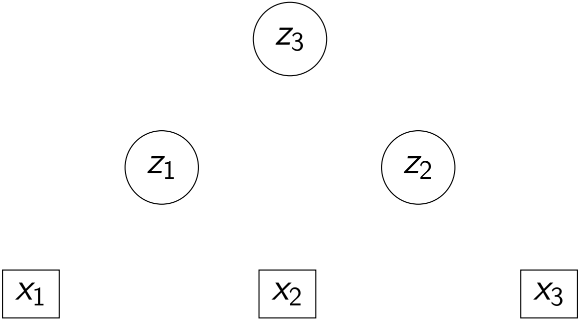
- Use inference network to sample hidden unit setting z from $q(z|x, \lambda)$
- Update generation parameters θ to maximize joint log-likelihood of data and latents $p(x, z|\theta)$

Sleep Phase

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

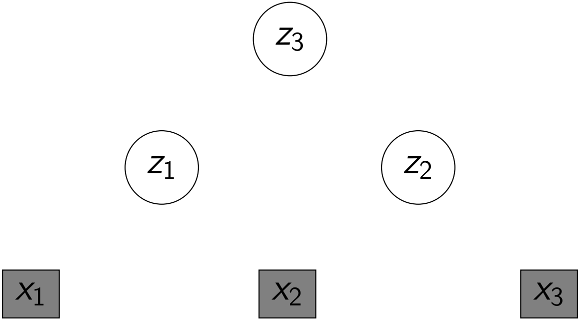
Wake Phase Sampling

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



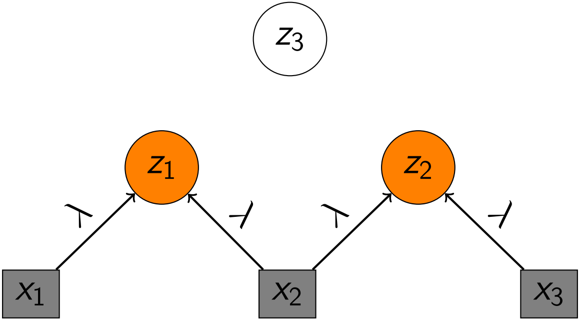
Wake Phase Sampling

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



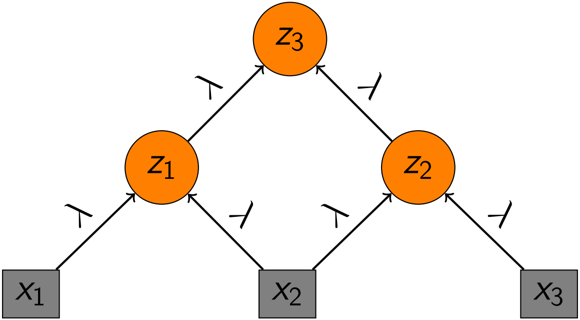
Wake Phase Sampling

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



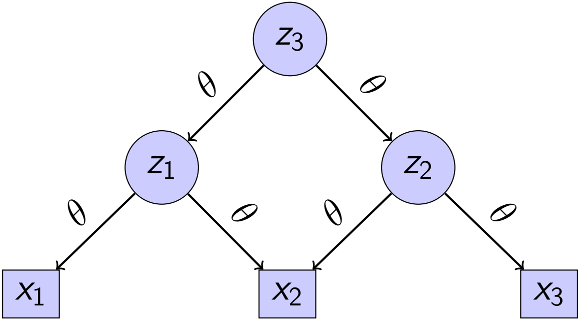
Wake Phase Sampling

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



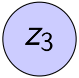
Wake Phase Update

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



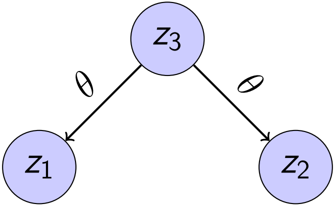
Sleep Phase Sampling

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



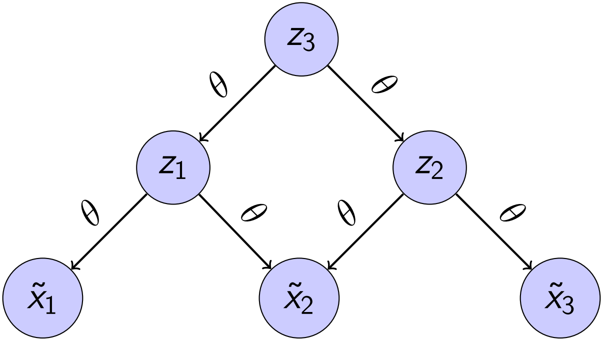
Sleep Phase Sampling

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



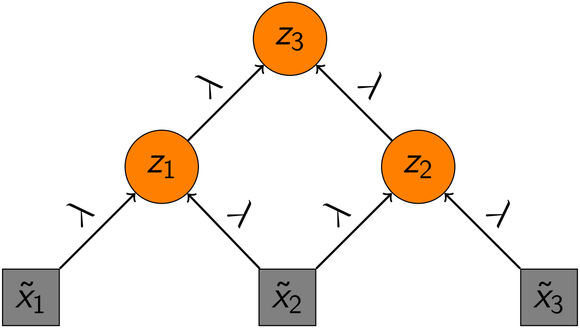
Sleep Phase Sampling

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



Sleep Phase Update

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



Wake Phase Objective

Objective

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

Wake Phase Objective

Objective

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

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

Wake Phase Objective

Objective

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

Approximation: optimize the lower-bound alone.

Wake Phase Objective

Objective

$$
\arg\max_{\theta} \; \mathbb{E}_{p(x)} \left[\text{ELBO}(\theta, \lambda | x) \right]
$$

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

Wake Phase Objective

Objective

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

Gradient wrt θ for $x \sim p(x)$

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

Wake Phase Objective

Objective

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

Gradient wrt θ for $x \sim p(x)$

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

Wake Phase Objective

Objective

$$\arg\max_{\theta} \; \mathbb{E}_{p(x)} \left[\mathrm{ELBO}(\theta, \lambda | x) \right]$$

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

Gradient wrt θ for $x \sim p(x)$

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

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!

Sleep Phase Objective

Objective

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

Sleep Phase Objective

Objective

$$\arg\max_{\lambda} \ \mathbb{E}_{p(x)} \left[\text{ELBO}(\theta, \lambda | x)\right]$$

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

Gradient wrt λ for $x \sim p(x)$

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

Sleep Phase Objective

Objective

$$\underset{\lambda}{\arg\max}\ \mathbb{E}_{p(x)}\left[\mathrm{ELBO}(\theta, \lambda | x)\right]$$
$$= \underset{\lambda}{\arg\max}\ \mathbb{E}_{p(x)}\left[\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z, x|\theta)\right] + \mathbb{H}[q(z|x, \lambda)]\right]$$

Gradient wrt λ for $x \sim p(x)$

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

Let's change the objective!

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Gradient wrt λ

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

Gradient wrt λ

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

Sleep Phase (Convenient) Objective

Flip the direction of the KL

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

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

$$\stackrel{\text{asm}}{=} \arg\max_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda) \right] - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta) \right]}_{\text{constant}}$$

Gradient wrt λ

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

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

$$\stackrel{\text{MC}}{\approx} \nabla_{\lambda} \log q(z|\tilde{x},\lambda) \quad \text{where } z \sim p(z|\theta)$$

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

Sleep Phase (Convenient) Objective

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

$$
\begin{aligned}
z &\sim p(z|\theta) \\
\tilde{x} &\sim p(x|z, \theta)
\end{aligned}
$$

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

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 and Learning
3. Score Function Estimator

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 $x = (x_1, \dots, x_N)$ 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.

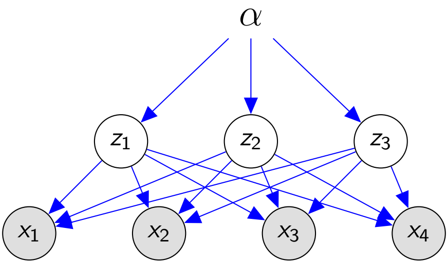
Latent factor model

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

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.

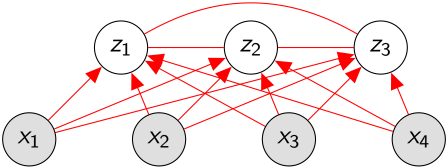
$$
\begin{aligned}
f(z;\theta) &= \text{softmax}(Wz + b) \\
\theta &= \{W, b\}
\end{aligned}
$$

Example Model



Joint distribution: independent latent variables

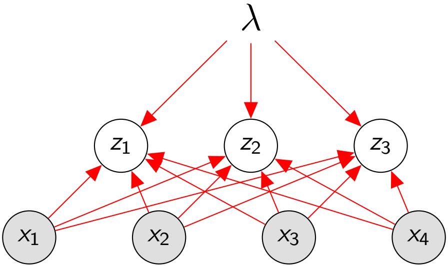
Example Model



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

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

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

Example architecture

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

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

Objective

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

Objective

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

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

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

KL

KL between K independent Bernoulli distributions is tractable

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

$$= \sum_{k=1}^{K} \mathrm{KL}\left(\mathrm{Bernoulli}\left(b_k\right)\right) \mid\mid \mathrm{Bernoulli}\left(\alpha\right))$$

KL

KL between K independent Bernoulli distributions is tractable

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

$$= \sum_{k=1}^{K} \mathrm{KL}\left(\mathrm{Bernoulli}\left(b_k\right)\right)\;||\;\mathrm{Bernoulli}\left(\alpha\right))$$

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

Generative Network Gradient

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

Generative Network Gradient

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

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

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

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

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

Inference Network Gradient

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

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

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

Inference Network Gradient

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

Inference Network Gradient

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

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

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

- MC estimator is non-differentiable

Inference Network Gradient

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

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

$$= \underbrace{\sum_z \frac{\partial}{\partial \lambda} (q(z|x,\lambda)) \log p(x|z,\theta)}_{\text{not an expectation}}$$

- 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

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

Score Function Estimator

We can again use the log identity for derivatives

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

Score Function Estimator

We can now build an MC estimator

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

Score Function Estimator

We can now build an MC estimator

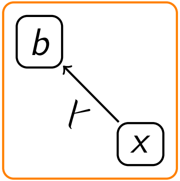
$$\frac{\partial}{\partial \lambda} \mathbb{E}_{q(z|x,\lambda)} [\log p(x|z,\theta)]$$

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

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

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

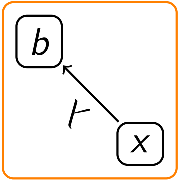
Computation Graph



inference model

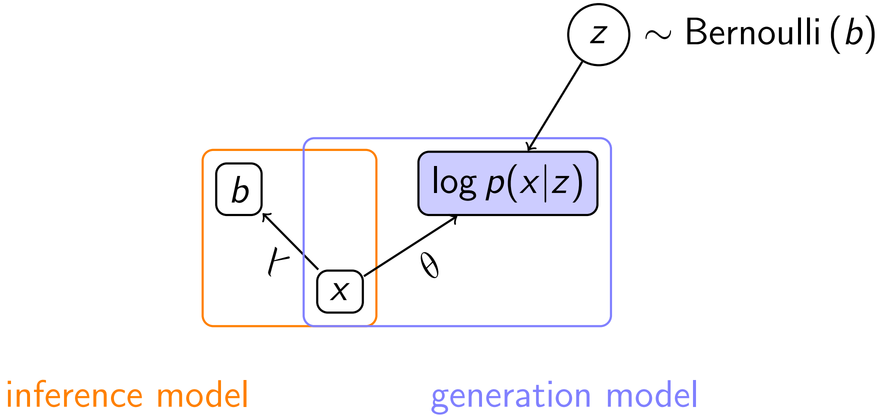
Computation Graph

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

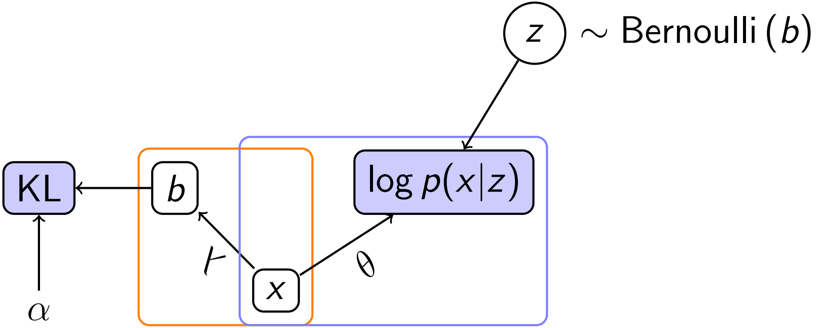


inference model

Computation Graph



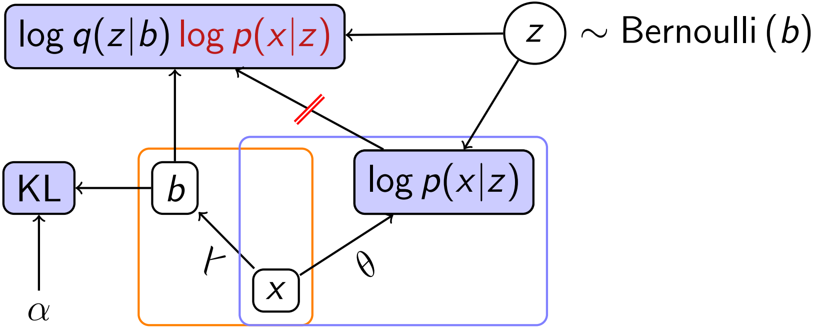
Computation Graph



inference model

generation model

Computation Graph



inference model

generation model

Score Function Estimator: Variance

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

Score Function Estimator: Variance

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

Empirically this estimator often exhibits high variance.

Score Function Estimator: Variance

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

Empirically this estimator often exhibits high variance.

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

Score Function Estimator: Variance

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

Empirically this estimator often exhibits high variance.

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

Score Function Estimator: Variance

We could:

Score Function Estimator: Variance

We could:

- sample more (better MC estimates)

Score Function Estimator: Variance

We could:

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

Score Function Estimator: Variance

Idea: standardise the "reward" $r(z) := \log p(x|z, \theta)$ to have a mean at 0 and a variance of 1

Score Function Estimator: Variance

Idea: standardise the "reward" $r(z) := \log p(x|z, \theta)$ to have a mean at 0 and a variance of 1

- Keep a moving average of the mean and variance $\log p(x|z, \theta)$: $\hat{\mu}$ and $\hat{\sigma^2}$.

Score Function Estimator: Variance

Idea: standardise the “reward” $r(z) := \log p(x|z,\theta)$ to have a mean at 0 and a variance of 1

- Keep a moving average of the mean and variance $\log p(x|z,\theta)$: $\hat{\mu}$ and $\hat{\sigma^2}$.
- $\hat{r}(z) = \frac{\log p(x|z,\theta) - \hat{\mu}}{\hat{\sigma^2}}$

Score Function Estimator: Variance

Idea: standardise the "reward" $r(z) := \log p(x|z, \theta)$ to have a mean at 0 and a variance of 1

- Keep a moving average of the mean and variance $\log p(x|z, \theta)$: $\hat{\mu}$ and $\hat{\sigma^2}$.
- $\hat{r}(z) = \frac{\log p(x|z,\theta) - \hat{\mu}}{\hat{\sigma^2}}$

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

Score Function Estimator: Variance

Idea: standardise the "reward" $r(z) := \log p(x|z, \theta)$ to have a mean at 0 and a variance of 1

- Keep a moving average of the mean and variance $\log p(x|z, \theta)$: $\hat{\mu}$ and $\hat{\sigma^2}$.
- $\hat{r}(z) = \frac{\log p(x|z,\theta) - \hat{\mu}}{\hat{\sigma^2}}$

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

$$\approx E_{q(z|x,\lambda)} \left[\hat{r}(z) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda) \right]$$

Score Function Estimator: Variance

- We can show that using these *baselines* do not bias the estimator.

Score Function Estimator: Variance

- We can show that using these *baselines* do not bias the estimator.
- We can add more advanced *control variates* and other *baselines* to further reduce variance.

Score Function Estimator: Variance

- We can show that using these *baselines* do not bias the estimator.
- We can add more advanced *control variates* and other *baselines* to further reduce variance.
- More about this tomorrow!

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!

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.

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.

Literature II

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.

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.