

# Deep Generative Models: Continuous Latent Variables

Philip Schulz and Wilker Aziz

[https:](https://github.com/philschulz/VITutorial)

[//github.com/philschulz/VITutorial](https://github.com/philschulz/VITutorial)  
branch: yandex2019    module: modules/M3a

# Deep Generative Models

## Variational Autoencoders

# Deep Generative Models

## Variational Autoencoders

# Generative Model with NN Likelihood

## Goal

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

# Example: Language Model

A deterministic language model is **one** distribution over observations:

$$p(x|\theta) = \prod_{i=1}^n p(x_i|x_{<i}, \theta)$$

Every sentence gets mapped from the same conditioning context, namely, the beginning of sequence symbol.

## Example: Language Model (cont.)

With latent variables we can model the data as a draw from a complex marginal, which mixes conditionals from different points in space

$$p(x|\theta) = \int p(\mathbf{z}) \prod_{i=1}^n p(x_i | \mathbf{z}, x_{<i}, \theta) d\mathbf{z}$$

## Example: Language Model (cont.)

With latent variables we can model the data as a draw from a complex marginal, which mixes conditionals from different points in space

$$p(x|\theta) = \int p(\mathbf{z}) \prod_{i=1}^n p(x_i | \mathbf{z}, x_{<i}, \theta) d\mathbf{z}$$

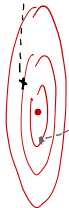
Good training can lead to considerable amount of structure in the posterior

$$p(\mathbf{z} | \mathbf{x}, \theta) = \frac{p(\mathbf{z}) p(\mathbf{x} | \mathbf{z}, \theta)}{p(\mathbf{x} | \theta)}$$

$$z \in \mathbb{R}^2$$

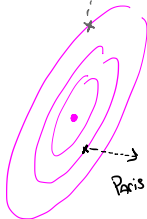


I did not like Paris



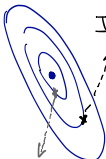
I found the UK too odd

The UK is rainy



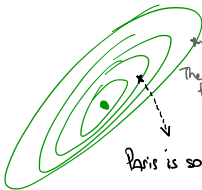
Paris is too busy

I loved Paris



I enjoyed the climate in the UK

The UK is fun!



Paris is so beautiful!



# Example: Language Model (architecture)

Generative model:

$$Z \sim \mathcal{N}(0, I)$$

$$X_i | z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$$

# Example: Language Model (architecture)

Generative model:

$$Z \sim \mathcal{N}(0, I)$$

$$X_i | z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$$

Example architecture:

$$h_0 = \tanh\left(W^{(\text{init})}z + b^{(\text{init})}\right)$$

# Example: Language Model (architecture)

Generative model:

$$Z \sim \mathcal{N}(0, I)$$

$$X_i | z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$$

Example architecture:

$$h_0 = \tanh\left(W^{(\text{init})}z + b^{(\text{init})}\right)$$

$$h_i = \text{rnn}(h_{i-1}, E_{x_{i-1}}; \theta_{\text{rnn}})$$

# Example: Language Model (architecture)

Generative model:

$$Z \sim \mathcal{N}(0, I)$$

$$X_i | z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$$

Example architecture:

$$h_0 = \tanh\left(W^{(\text{init})}z + b^{(\text{init})}\right)$$

$$h_i = \text{rnn}(h_{i-1}, E_{x_{i-1}}; \theta_{\text{rnn}})$$

$$f(z, x_{<i}) = \text{softmax}(W^{(\text{out})}h_i + b^{(\text{out})})$$

# Example: Language Model (architecture)

Generative model:

$$Z \sim \mathcal{N}(0, I)$$

$$X_i | z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$$

Example architecture:

$$h_0 = \tanh(W^{(\text{init})} z + b^{(\text{init})})$$

$$h_i = \text{rnn}(h_{i-1}, E_{x_{i-1}}; \theta_{\text{rnn}})$$

$$f(z, x_{<i}) = \text{softmax}(W^{(\text{out})} h_i + b^{(\text{out})})$$

$$\theta = \theta_{\text{rnn}} \cup \{W^{(\text{init})}, b^{(\text{init})}, W^{(\text{out})}, b^{(\text{out})}\}$$

# Generative Model with NN Likelihood

## Goal

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

## Problem

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

## Deep Generative Models

### Variational Autoencoders

# Solution: Variational Inference

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



# Solution: Variational Inference

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

# Solution: Variational Inference

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

# Solution: Variational Inference

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

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

# Solution: Variational Inference

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

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

- assume  $\text{KL}(q(z|x, \lambda) \parallel p(z))$  analytical  
true for exponential families

# Solution: Variational Inference

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

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

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

# Generator Network Gradient

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

# Generator Network Gradient

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

# Generator Network Gradient

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

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



# Generator Network Gradient

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

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

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

# Inference Network Gradient

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

# Inference Network Gradient

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

# Inference Network Gradient

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

The first term again requires approximation by  
sampling

# Inference Network Gradient

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

# Inference Network Gradient

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

# Inference Network Gradient

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

# Inference Network Gradient

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

Not an expected gradient!



# Score function estimator?

Can we apply the log-derivative trick?

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

# Score function estimator?

Can we apply the log-derivative trick?

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

# Score function estimator?

Can we apply the log-derivative trick?

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

Yes, it's a general result!

# What about variance?

The learning signal can only scale the gradient:

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

# What about variance?

The learning signal can only scale the gradient:

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

Can we do better?

# Inference Network Gradient

## Problem

We need to re-express the gradient, but the measure of integration depends on  $\lambda$

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

# Inference Network Gradient

## Problem

We need to re-express the gradient, but the measure of integration depends on  $\lambda$

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

What if we could re-express  $q(z|x, \lambda)$  in terms of some other distribution that does not depend on  $\lambda$ ?

# Inference Network Gradient

## Reparametrisation trick

Find a transformation  $h : z \mapsto \epsilon$  such that  $\epsilon$  does not depend on  $\lambda$ .

- ▶  $h(z, \lambda)$  needs to be invertible
- ▶  $h(z, \lambda)$  needs to be differentiable



# Inference Network Gradient

## Reparametrisation trick

Find a transformation  $h : z \mapsto \epsilon$  such that  $\epsilon$  does not depend on  $\lambda$ .

- ▶  $h(z, \lambda)$  needs to be invertible
- ▶  $h(z, \lambda)$  needs to be differentiable
- ▶  $h(z, \lambda) = \epsilon$
- ▶  $h^{-1}(\epsilon, \lambda) = z$

# Gaussian Transformation

## Affine property

$$Az + b \sim \mathcal{N}(\mu + b, A\Sigma A^T) \text{ for } z \sim \mathcal{N}(\mu, \Sigma)$$

# Gaussian Transformation

## Affine property

$$Az + b \sim \mathcal{N}(\mu + b, A\Sigma A^T) \text{ for } z \sim \mathcal{N}(\mu, \Sigma)$$

## Special case

$$Az + b \sim \mathcal{N}(b, AA^T) \text{ for } z \sim \mathcal{N}(0, I)$$

# Gaussian Transformation

Let an inference network compute

$$u = \mu(x; \lambda) \quad s = \sigma(x; \lambda)$$

for a posterior  $Z \sim \mathcal{N}(u, s^2)$ , then we have:

# Gaussian Transformation

Let an inference network compute

$$u = \mu(x; \lambda) \quad s = \sigma(x; \lambda)$$

for a posterior  $Z \sim \mathcal{N}(u, s^2)$ , then we have:

$$h(z, \lambda; x) = \frac{z - \mu(x; \lambda)}{\sigma(x; \lambda)} = \epsilon \sim \mathcal{N}(0, 1)$$

# Gaussian Transformation

Let an inference network compute

$$u = \mu(x; \lambda) \quad s = \sigma(x; \lambda)$$

for a posterior  $Z \sim \mathcal{N}(u, s^2)$ , then we have:

$$h(z, \lambda; x) = \frac{z - \mu(x; \lambda)}{\sigma(x; \lambda)} = \epsilon \sim \mathcal{N}(0, I)$$

and conversely, for  $\epsilon \sim \mathcal{N}(0, I)$ , we have:

$$h^{-1}(\epsilon, \lambda; x) = \mu(x; \lambda) + \sigma(x; \lambda) \odot \epsilon = z \sim \mathcal{N}(u, s^2)$$

# Inference Network Gradient

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

# Inference Network Gradient

$$\begin{aligned} &= \frac{\partial}{\partial \lambda} \int q(z|x, \lambda) \log p(x|z, \theta) dz \\ &= \frac{\partial}{\partial \lambda} \int q(\epsilon) \log \left( p(x | \overbrace{h^{-1}(\epsilon, \lambda)}^{=z}, \theta) \right) d\epsilon \end{aligned}$$



# Inference Network Gradient

$$\begin{aligned}
 &= \frac{\partial}{\partial \lambda} \int q(z|x, \lambda) \log p(x|z, \theta) dz \\
 &= \frac{\partial}{\partial \lambda} \int q(\epsilon) \log \left( p(x | \overbrace{h^{-1}(\epsilon, \lambda)}^{=z}, \theta) \right) d\epsilon \\
 &= \int q(\epsilon) \frac{\partial}{\partial \lambda} \left[ \log p(x | \overbrace{h^{-1}(\epsilon, \lambda)}^{=z}, \theta) \right] d\epsilon
 \end{aligned}$$

# Inference Network Gradient

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

# Inference Network Gradient

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

$$\stackrel{\text{MC}}{\approx} \frac{1}{S} \sum_{i=1}^S \frac{\partial}{\partial \lambda} \log p(x | \overbrace{h^{-1}(\epsilon_i, \lambda)}^{=z}, \theta)$$

where  $\epsilon_i \sim q(\epsilon)$

# Inference Network Gradient

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

$$\approx_{\text{MC}} \frac{1}{S} \sum_{i=1}^S \frac{\partial}{\partial \lambda} \log p(x | \overbrace{h^{-1}(\epsilon_i, \lambda)}^{=z}, \theta)$$

where  $\epsilon_i \sim q(\epsilon)$

$$\approx_{\text{MC}} \frac{1}{S} \sum_{i=1}^S \underbrace{\frac{\partial}{\partial z} \log p(x | \overbrace{h^{-1}(\epsilon_i, \lambda)}^{=z}, \theta) \times \frac{\partial}{\partial \lambda} h^{-1}(\epsilon_i, \lambda)}_{\text{chain rule}}$$

# Derivatives of Gaussian transformation

Recall:

$$h^{-1}(\epsilon, \lambda) = \mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon .$$

We get two gradient paths!

# Derivatives of Gaussian transformation

Recall:

$$h^{-1}(\epsilon, \lambda) = \mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon .$$

We get two gradient paths!

► one is **deterministic**

$$\frac{\partial h^{-1}(\epsilon, \lambda)}{\partial \mu(x, \lambda)} = \frac{\partial}{\partial \mu(x, \lambda)} [\mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon] = 1$$

# Derivatives of Gaussian transformation

Recall:

$$h^{-1}(\epsilon, \lambda) = \mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon .$$

We get two gradient paths!

- ▶ one is **deterministic**

$$\frac{\partial h^{-1}(\epsilon, \lambda)}{\partial \mu(x, \lambda)} = \frac{\partial}{\partial \mu(x, \lambda)} [\mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon] = 1$$

- ▶ the other is **stochastic**

$$\frac{\partial h^{-1}(\epsilon, \lambda)}{\partial \sigma(x, \lambda)} = \frac{\partial}{\partial \sigma(x, \lambda)} [\mu(x, \lambda) + \sigma(x, \lambda) \odot \epsilon] = \epsilon$$

# Gaussian KL

## ELBO

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



# Gaussian KL

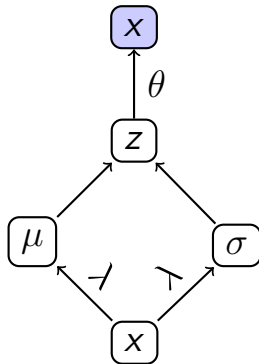
## ELBO

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

Analytical computation of  $-\text{KL} (q(z|x, \lambda) \parallel p(z))$ :

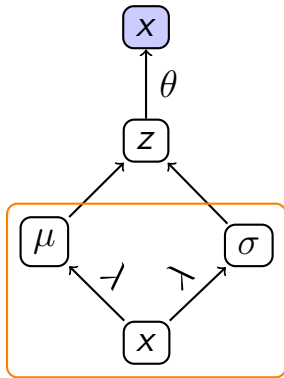
$$\frac{1}{2} \sum_{i=1}^N (1 + \log (\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

# Computation Graph



# Computation Graph

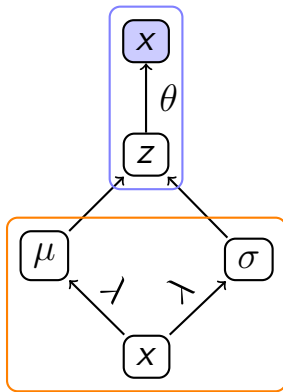
inference model



# Computation Graph

generation model

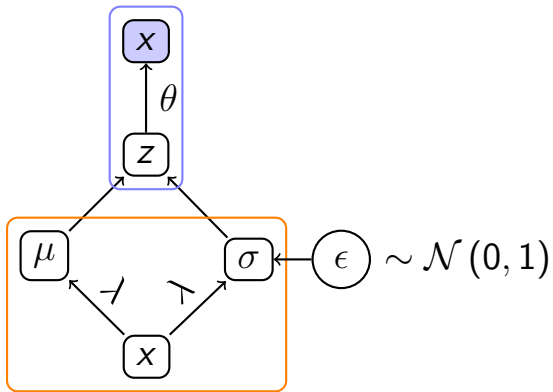
inference model



# Computation Graph

generation model

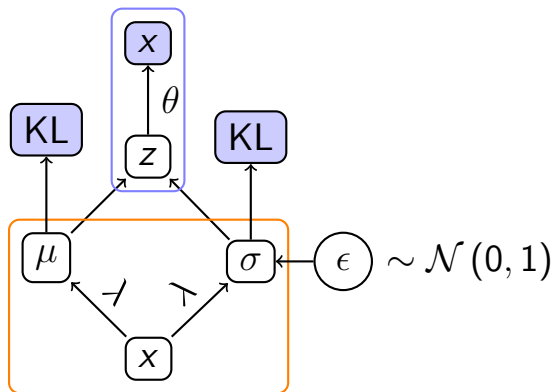
inference model



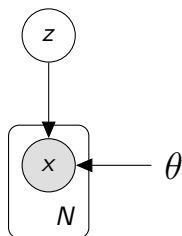
# Computation Graph

generation model

inference model



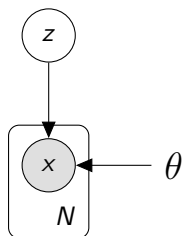
# Example: Unigram Document Model



Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$

# Example: Unigram Document Model



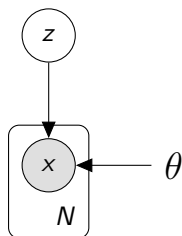
Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$

Designing  $f(z, \theta)$



# Example: Unigram Document Model



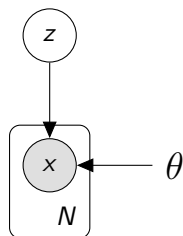
Generative story

- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i|z \sim \text{Cat}(f(z; \theta))$

Designing  $f(z, \theta)$

$$h = \text{relu}(W_1 z + b_1)$$

# Example: Unigram Document Model



Generative story

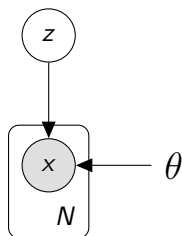
- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i|z \sim \text{Cat}(f(z; \theta))$

Designing  $f(z, \theta)$

$$h = \text{relu}(W_1 z + b_1)$$

$$f(z, \theta) = \text{softmax}(W_2 h + b_2)$$

# Example: Unigram Document Model



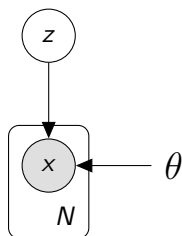
Generative story

- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i | z \sim \text{Cat}(f(z; \theta))$

Designing  $f(z, \theta)$

$$\begin{aligned}
 h &= \text{relu}(W_1 z + b_1) \\
 f(z, \theta) &= \text{softmax}(W_2 h + b_2) \\
 \theta &= \{W_1, b_1, W_2, b_2\}
 \end{aligned}$$

# Example: Unigram Document Model

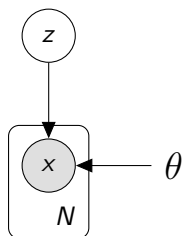


Likelihood

Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$

# Example: Unigram Document Model



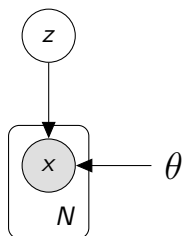
Likelihood

Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$

$$p(x|z, \theta) = \prod_{i=1}^N p(x_i|z, \theta)$$

# Example: Unigram Document Model



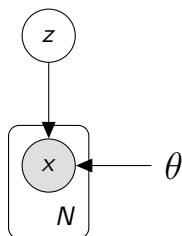
Likelihood

Generative story

- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i|z \sim \text{Cat}(f(z; \theta))$

$$p(x|z, \theta) = \prod_{i=1}^N p(x_i|z, \theta) = \prod_{i=1}^N \text{Cat}(x_i | \underbrace{f(z; \theta)}_{=\psi})$$

# Example: Unigram Document Model



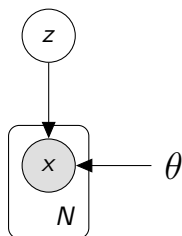
Likelihood

Generative story

- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i | z \sim \text{Cat}(f(z; \theta))$

$$\begin{aligned}
 p(x|z, \theta) &= \prod_{i=1}^N p(x_i|z, \theta) = \prod_{i=1}^N \text{Cat}(x_i | \underbrace{f(z; \theta)}_{=\psi}) \\
 &= \prod_{i=1}^N \psi_{x_i}
 \end{aligned}$$

# Example: Unigram Document Model



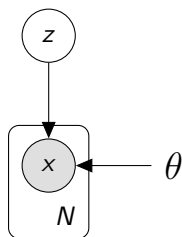
Marginal

Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$



# Example: Unigram Document Model



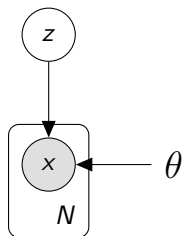
Marginal

Generative story

- ▶ Draw a document embedding  
 $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  
 $X_i|z \sim \text{Cat}(f(z; \theta))$

$$p(x|\theta) = \int p(z) \prod_{i=1}^N p(x_i|z, \theta) \, dz$$

# Example: Unigram Document Model



Marginal

Generative story

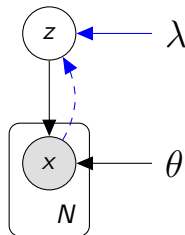
- ▶ Draw a document embedding  $Z \sim \mathcal{N}(0, I)$
- ▶ Draw  $N$  words  $X_i|z \sim \text{Cat}(f(z; \theta))$

$$\begin{aligned}
 p(x|\theta) &= \int p(z) \prod_{i=1}^N p(x_i|z, \theta) \, dz \\
 &= \int \mathcal{N}(z|0, I) \prod_{i=1}^N \text{Cat}(x_i|f(z; \theta)) \, dz
 \end{aligned}$$

# Example: Unigram Document Model

Inference model

►  $Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$

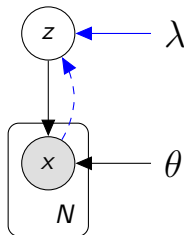


# Example: Unigram Document Model

## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$

Designing the *inference network*



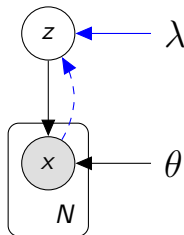
# Example: Unigram Document Model

## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$

Designing the *inference network*

$$s = \sum_{i=1}^N E_{x_i}$$



# Example: Unigram Document Model

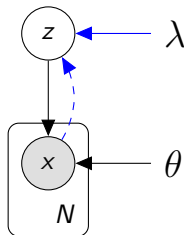
## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$

Designing the *inference network*

$$s = \sum_{i=1}^N E_{x_i}$$

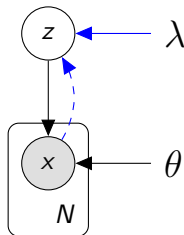
$$h = \text{relu}(M_1 s + c_1)$$



# Example: Unigram Document Model

## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$



Designing the *inference network*

$$s = \sum_{i=1}^N E_{x_i}$$

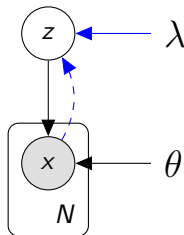
$$\mu(x; \lambda) = M_2 h + c_2$$

$$h = \text{relu}(M_1 s + c_1)$$

# Example: Unigram Document Model

## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$



Designing the *inference network*

$$s = \sum_{i=1}^N E_{x_i}$$

$$h = \text{relu}(M_1 s + c_1)$$

$$\mu(x; \lambda) = M_2 h + c_2$$

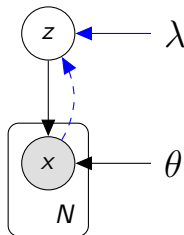
$$\sigma(x; \lambda) = \text{softplus}(M_3 h + c_3)$$



# Example: Unigram Document Model

## Inference model

$$\blacktriangleright Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$$



Designing the *inference network*

$$s = \sum_{i=1}^N E_{x_i}$$

$$h = \text{relu}(M_1 s + c_1)$$

$$\mu(x; \lambda) = M_2 h + c_2$$

$$\sigma(x; \lambda) = \text{softplus}(M_3 h + c_3)$$

$$\lambda = \{E, M_1^3, c_1^3\}$$

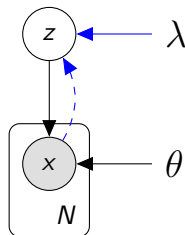
# Example: Unigram Document Model

## Generative Model

- ▶ Prior:  $Z \sim \mathcal{N}(0, I)$
- ▶ Likelihood:  $X_i|z \sim \text{Cat}(f(z; \theta))$

## Inference Model

- ▶  $Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$



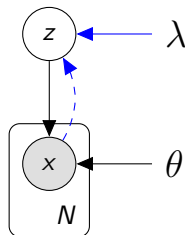
# Example: Unigram Document Model

## Generative Model

- Prior:  $Z \sim \mathcal{N}(0, I)$
- Likelihood:  $X_i|z \sim \text{Cat}(f(z; \theta))$

## Inference Model

- $Z|x \sim \mathcal{N}(\mu(x; \lambda), \sigma(x; \lambda)^2)$



## ELBO

$$\begin{aligned} \log p(x|\theta) \geq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[ \sum_{i=1}^N \log \psi_{x_i} \right] \\ - \text{KL} \left( \mathcal{N}(z|u, s^2) \parallel \mathcal{N}(z|0, I) \right) \end{aligned}$$

where  $u = \mu(x; \lambda)$ ,  $s = \sigma(x; \lambda)$ , and  $\psi = f(z = u + \epsilon \odot s, \theta)$

# Aside

If your likelihood model is able to express dependencies between the output variables (e.g. an RNN), the model may simply ignore the latent code. In that case one often scales the KL term. The scale factor is increased gradually.

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

where  $\beta \rightarrow 1$ .

# Aside

Another strategy is to promote the posterior to deviate a bit from the prior by not penalising for the first few nats of information:

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

where  $R \geq 0$  is known as “free bits”

# Aside

But note that if we scale down the KL term permanently, or allow too many free bits, then the conditional  $p(x|z, \theta)$  will over-specialise to samples from the approximate posterior  $q(z|x, \lambda)$ . This can lead to bad generalisation and/or poor samples when generating from the prior.

# Variational Autoencoder

## Advantages

- ▶ Backprop training
- ▶ Easy to implement
- ▶ Posterior inference possible
- ▶ One objective for both NNs
- ▶ Amortised inference

# Variational Autoencoder

## Advantages

- ▶ Backprop training
- ▶ Easy to implement
- ▶ Posterior inference possible
- ▶ One objective for both NNs
- ▶ Amortised inference

## Drawbacks

- ▶ Discrete latent variables are not possible
- ▶ Optimisation may be difficult with several latent variables



# Summary

- ▶ Wake-Sleep: train inference and generation networks with separate objectives
- ▶ VAE: train both networks with same objective
- ▶ Reparametrisation
  - ▶ Transform parameter-free variable  $\epsilon$  into latent value  $z$
  - ▶ Update parameters with stochastic gradient estimates

# Implementation

Try one of our notebooks, e.g.

- ▶ Original VAE: MNIST

[https://github.com/philschulz/  
VITutorial/blob/master/code/vae\\_  
notebook\\_pytorch.ipynb](https://github.com/philschulz/VITutorial/blob/master/code/vae_notebook_pytorch.ipynb)

- ▶ SentenceVAE

[https://github.com/probabll/dgm4nlp/  
tree/master/notebooks/sentencevae](https://github.com/probabll/dgm4nlp/tree/master/notebooks/sentencevae)

# 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](http://www.gatsby.ucl.ac.uk/~dayan/papers/hdfn95.pdf).

Diederik P. Kingma and Max Welling.  
Auto-Encoding Variational Bayes. 2013. URL  
<http://arxiv.org/abs/1312.6114>.

# Literature II

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017. URL

<http://jmlr.org/papers/v18/16-107.html>.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286, 2014. URL

# Literature III

<http://jmlr.org/proceedings/papers/v32/rezende14.pdf>.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In Tony Jebara and Eric P. Xing, editors, *ICML*, pages 1971–1979, 2014. URL <http://jmlr.org/proceedings/papers/v32/titsias14.pdf>.