Deep Generative Models: Continuous Latent Variables

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

Score Function Estimator

Integration by substitution

Variational Autoencoders

Semisupervised Learning

Score Function Estimator

Integration by substitution

Variational Autoencoders

Semisupervised Learning

Score Function Estimator

- Stochastic Gradient Estimator
- $\frac{\partial}{\partial \lambda} \mathbb{E}_{q(z|\lambda)} [\log p(x|z,\theta)] = \mathbb{E}_{q(z|\lambda)} \left[\frac{\partial}{\partial \lambda} \log q(z|\lambda) \log p(x|z,\theta)\right]$
- Very general
- High variance

Score Function Estimator

- Stochastic Gradient Estimator
- $\frac{\partial}{\partial \lambda} \mathbb{E}_{q(z|\lambda)} [\log p(x|z, \theta)] = \mathbb{E}_{q(z|\lambda)} \left[\frac{\partial}{\partial \lambda} \log q(z|\lambda) \log p(x|z, \theta) \right]$
- Very general
- High variance
- Can we do better?

Score Function Estimator

Integration by substitution

Variational Autoencoders

Semisupervised Learning

Basic Proof

We can express an integral over x in terms of an integral over y.
Let f be continuous and $h(x) = y$ be invertible s.t $h^{-1}(y) = x$.

$$\int_a^b f(x)\mathrm{d}x$$

Basic Proof

We can express an integral over x in terms of an integral over y.
Let f be continuous and $h(x) = y$ be invertible s.t $h^{-1}(y) = x$.

$$\int_a^b f(x)\mathrm{d}x = F(x)\Big|_a^b$$

Basic Proof

We can express an integral over x in terms of an integral over y.
Let f be continuous and $h(x) = y$ be invertible s.t $h^{-1}(y) = x$.

$$\int_a^b f(x)\mathrm{d}x = F(x)\Big|_a^b$$
$$= F\left(\underbrace{h^{-1}(y)}_{x}\right)\Bigg|_{h(a)}^{h(b)}$$

Basic Proof

We can express an integral over x in terms of an integral over y.
Let f be continuous and $h(x) = y$ be invertible s.t $h^{-1}(y) = x$.

$$\int_a^b f(x)\mathrm{d}x = F(x)\Big|_a^b$$

$$= F\left(\underbrace{h^{-1}(y)}_{x}\right)\Big|_{h(a)}^{h(b)} = \int_{h(a)}^{h(b)} f\left(h^{-1}(y)\right) \underbrace{\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}}_{\frac{\mathrm{d}x}{\mathrm{d}y}} \mathrm{d}y$$

Observations

$$\int_a^b f(x)\mathrm{d}x = \int_{h(a)}^{h(b)} f\left(h^{-1}(y)\right) \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\mathrm{d}y$$

- h^{-1} increasing $\implies \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} > 0$

Observations

$$\int_a^b f(x)\mathrm{d}x = \int_{h(a)}^{h(b)} f\left(h^{-1}(y)\right) \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\mathrm{d}y$$

- h^{-1} increasing $\implies \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} > 0$
- h^{-1} decreasing $\implies \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} < 0$ and $h(a) \geq h(b)$

Observations

$$\int_a^b f(x)\mathrm{d}x = \int_{h(a)}^{h(b)} f\left(h^{-1}(y)\right) \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\mathrm{d}y$$

- h^{-1} increasing $\implies \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} > 0$
- h^{-1} decreasing $\implies \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} < 0$ and $h(a) \geq h(b)$
  - $\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}$ corrects for the bounds of integration

$$\int f(x)\mathrm{d}x = \int f\left(h^{-1}(y)\right) \left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right| \mathrm{d}y$$

Observations

$$\int f\left(h^{-1}(y)\right)\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right|\mathrm{d}y$$

Observations

$$\int f\left(h^{-1}(y)\right)\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right|\mathrm{d}y$$

- $\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right| > 1 \implies h^{-1}$ locally expands area around y

Observations

$$\int f\left(h^{-1}(y)\right)\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right| \mathrm{d}y$$

- $\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right| > 1 \implies h^{-1}$ locally expands area around y
- $\left|\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y}\right| < 1 \implies h^{-1}$ locally shrinks area around y

Probability Densities

Fact

As before $h(x) = y$ and $h^{-1}(y) = x$. Let X and Y be random variables with densities f_x and f_y. Then

$$F(x) = F(y = h(x)) \text{ or } 1 - F(y = h(x))$$

where

$$F(x) = \int^x p_x(x)\mathrm{d}x \ .$$

Probability Densities

$$F(x) = \int^{x} p_x(t)\mathrm{d}t =$$

Probability Densities

$$F(x) = \int^{x} p_x(t)\mathrm{d}t =$$

$$F(y = h(x)) = \int^{x} p_y(h(t)) \left| \frac{\mathrm{d}h(t)}{\mathrm{d}t} \right| \mathrm{d}x$$

Probability Densities

$$F(x) = \int^{x} p_x(t)\mathrm{d}t =$$

$$F(y = h(x)) = \int^{x} p_y(h(t)) \left| \frac{\mathrm{d}h(t)}{\mathrm{d}t} \right| \mathrm{d}x$$

$$\implies p_x(x) = p_y(y = h(x)) \left| \frac{\mathrm{d}h(x)}{\mathrm{d}x} \right|$$

Probability Densities

$$F(x) = \int^{x} p_x(t)\mathrm{d}t =$$

$$F(y = h(x)) = \int^{x} p_y(h(t)) \left| \frac{\mathrm{d}h(t)}{\mathrm{d}t} \right| \mathrm{d}x$$

$$\implies p_x(x) = p_y(y = h(x)) \left| \frac{\mathrm{d}h(x)}{\mathrm{d}x} \right|$$

Fact

$$\frac{\mathrm{d}h(x)}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}h^{-1}(y)} = \left(\frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} \right)^{-1}$$

Score Function Estimator

Integration by substitution

Variational Autoencoders

Semisupervised Learning

The Original Problem

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

The Original Problem

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

The Original Problem

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

The first term again requires approximation by sampling

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

Not an expectation!

Possible Solutions

- We could just use the score function gradient estimator
- Or we could use what we have learned about variable substitution . . .

Inference Network Gradient

Reparametrisation trick

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

- $h(z, \lambda)$ 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 ϵ does not depend on λ.

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

Inference Network Gradient

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

Inference Network Gradient

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

$$= \frac{\partial}{\partial \lambda} \int q(\epsilon) \left| \frac{\mathrm{d}h(z,\lambda)}{\mathrm{d}z} \right|$$

Inference Network Gradient

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

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

Inference Network Gradient

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

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

Inference Network Gradient

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

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

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

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

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

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

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

Inference Network Gradient

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

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

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

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

Gaussian Transformation

Gaussian Transformation

Affine property

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

Gaussian Transformation

Affine property

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

Special case

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

Gaussian Transformation

Affine property

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

Special case

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

Gaussian transformation

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

$$\underbrace{h^{-1}(\epsilon)}_{=z} = \mu + \sigma \odot \epsilon \quad \epsilon \sim \mathcal{N}(0, \mathrm{I})$$

Gaussian Transformation

$$
\begin{aligned}
&\mathbb{E}_{q(\epsilon)}\left[\frac{\partial}{\partial\lambda}\log p(x|\overbrace{h^{-1}(\epsilon,\lambda)}^{=z},\theta)\right]\\
=&\mathbb{E}_{q(\epsilon)}\left[\frac{\partial}{\partial\lambda}\log p(x|\overbrace{\mu(x,\lambda)+\epsilon\odot\sigma(x,\lambda)}^{=z},\theta)\right]
\end{aligned}
$$

Gaussian Transformation

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

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

where

$$q(\epsilon)=\mathcal{N}(\epsilon;0,\mathrm{I})$$

Derivatives of Gaussian transformation

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

We get two gradient paths!

Derivatives of Gaussian transformation

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

We get two gradient paths!

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

Derivatives of Gaussian transformation

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

We get two gradient paths!

- one is deterministic
  $\frac{\partial h^{-1}(\epsilon,\lambda)}{\partial \mu(\phi,\lambda)} = \frac{\partial}{\partial \mu(\phi,\lambda)}[\mu(\phi, \lambda) + \sigma(\phi, \lambda) \odot \epsilon] = 1$
- the other is stochastic
  $\frac{\partial h^{-1}(\epsilon,\lambda)}{\partial \sigma(\phi,\lambda)} = \frac{\partial}{\partial \sigma(\phi,\lambda)}[\mu(\phi, \lambda) + \sigma(\phi, \lambda) \odot \epsilon] = \epsilon$

Gaussian KL

ELBO

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

Gaussian KL

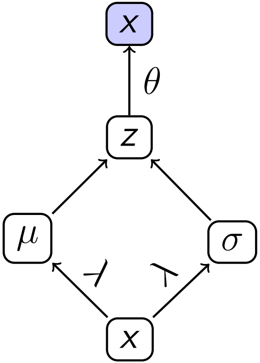
ELBO

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

Analytical computation of $-\mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z)\right)$:

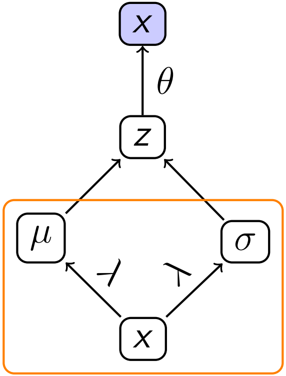
$$\frac{1}{2}\sum_{i=1}^{N}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

Computation Graph



Computation Graph

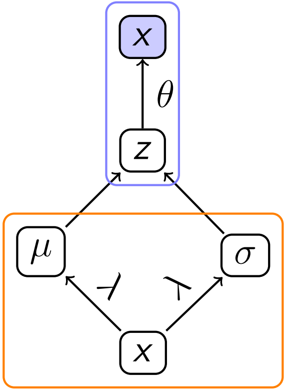
inference model



Computation Graph

generation model

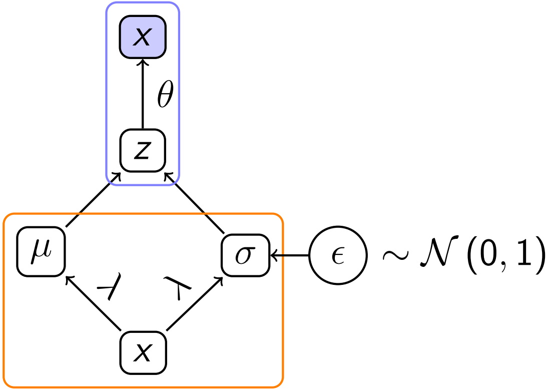
inference model



Computation Graph

generation model

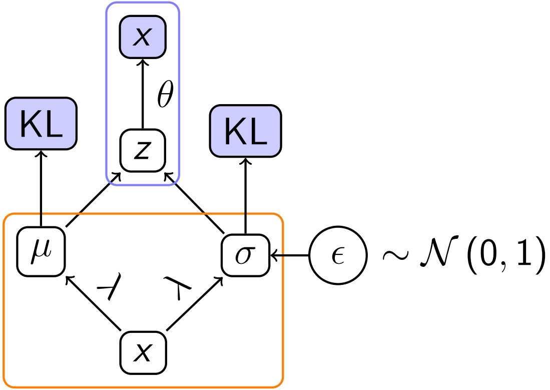
inference model



Computation Graph

generation model

inference model



Comparison Between Estimators

- Score function gradient

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

- Reparametrisation gradient

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

Discrete Variables

Why not use reparametrization for discrete variables?

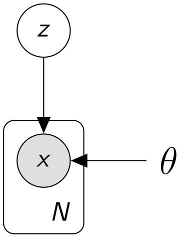
Discrete Variables

Why not use reparametrization for discrete variables?

Discrete CDF are step functions.

- not continuous
- derivative 0
- no invertible mappings

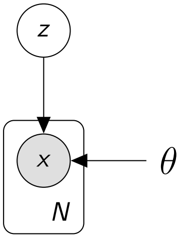
Example: Binarized MNIST



Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Example: Binarized MNIST

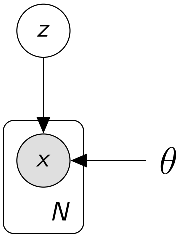


Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Designing $f(z, \theta)$

Example: Binarized MNIST



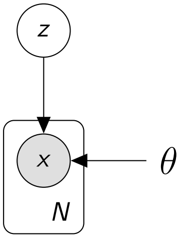
Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Designing $f(z, \theta)$

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

Example: Binarized MNIST



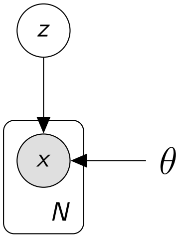
Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Designing $f(z, \theta)$

$$h = \text{relu}(W_1 z + b_1)$$
$$f(z, \theta) = \sigma(W_2 h + b_2)$$

Example: Binarized MNIST



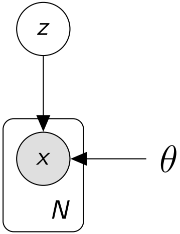
Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Designing $f(z, \theta)$

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

Example: Binarized MNIST

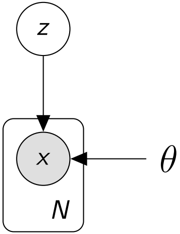


Marginal

Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

Example: Binarized MNIST



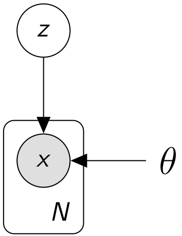
Marginal

Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

$$p(x_1^N | \theta) = \int p(z) \prod_{i=1}^{N} p(x_i | z, \theta) \, \mathrm{d}z$$

Example: Binarized MNIST



Marginal

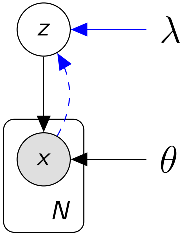
Generative story

- Draw an image embedding $Z \sim \mathcal{N}(0, I)$
- Draw N pixels
  $X_i | z \sim \text{Bernoulli}(f(z, \theta))$

$$p(x_1^N | \theta) = \int p(z) \prod_{i=1}^{N} p(x_i | z, \theta) \, \mathrm{d}z$$

$$= \int \mathcal{N}(z | 0, I) \prod_{i=1}^{N} \text{Bernoulli}(x_i | f(z, \theta)) \, \mathrm{d}z$$

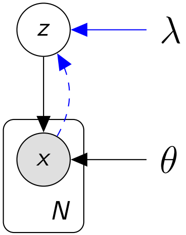
Example: Binarized MNIST



Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Example: Binarized MNIST

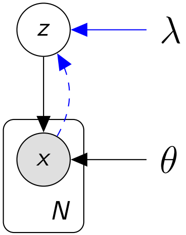


Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Designing the *inference network*

Example: Binarized MNIST



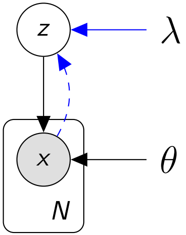
Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Designing the *inference network*

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

Example: Binarized MNIST



Inference model

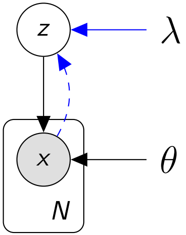
- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Designing the *inference network*

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

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

Example: Binarized MNIST



Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

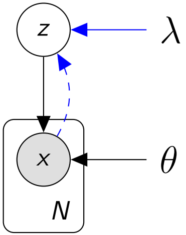
Designing the *inference network*

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

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

$$\mu(x_1^N, \lambda) = M_2 h + c_2$$

Example: Binarized MNIST



Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Designing the *inference network*

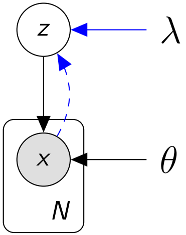
$$s = \sum_{i=1}^{N} E_{x_i}$$

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

$$\mu(x_1^N, \lambda) = M_2 h + c_2$$

$$\sigma(x_1^N, \lambda) = \text{softplus}(M_3 h + c_3)$$

Example: Binarized MNIST



Inference model

- $Z|x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$

Designing the *inference network*

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

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

$$\mu(x_1^N, \lambda) = M_2 h + c_2$$

$$\sigma(x_1^N, \lambda) = \text{softplus}(M_3 h + c_3)$$

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

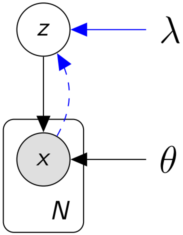
Example: Binarized MNIST

Generative Model

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

Inference Model

- $Z | x_1^N \sim \mathcal{N}(\mu(x_1^N, \lambda), \sigma(x_1^n, \lambda)^2)$



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.

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

where $\beta \to 1$.

Variational Autoencoder

Advantages

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

Variational Autoencoder

Advantages

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

Drawbacks

- Discrete latent variables are difficult
- 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 ϵ into latent value z
  - Update parameters with stochastic gradient estimates

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.

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

Literature III

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.

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *ACL*, pages 310–320, 2017. doi: 10.18653/v1/P17-1029. URL http://www.aclweb.org/anthology/P17-1029.

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

- plays $\rightarrow$ played

Putting it all together

We now know how to handle continuous and discrete latent variables. Let us combine these two treat partially observed data.

Morphological Reinflection

Transform an inflected form of a verb into another.

- plays $\rightarrow$ played
- walking $\rightarrow$ walks

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

- lemma

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

- lemma (real vector)

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

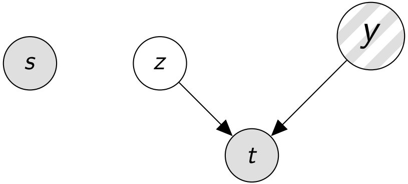
- lemma (real vector)
- morphological information

A Simple Model (Zhou and Neubig, 2017)

What do we need to correctly inflect a word?

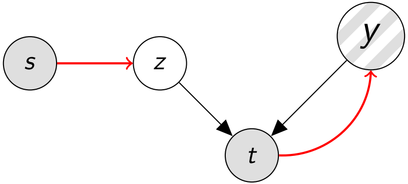
- lemma (real vector)
- morphological information (discrete vector)

A Simple Model (Zhou and Neubig, 2017)



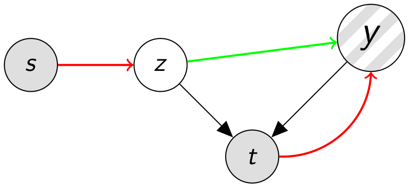
- z = lemma (continuous)
- y = morphological features (discrete)
- s = source form (inflected)
- t = target form (inflected)

A Simple Model (Zhou and Neubig, 2017)



- z = lemma (continuous)
- y = morphological features (discrete)
- s = source form (inflected)
- t = target form (inflected)

A Simple Model (Zhou and Neubig, 2017)



- z = lemma (continuous)
- y = morphological features (discrete)
- s = source form (inflected)
- t = target form (inflected)