Deep Generative Models: Continuous Latent Variables

Philip Schulz and Wilker Aziz

https://github.com/philschulz/VITutorial

Deep Generative Models

First Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Deep Generative Models

First Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Generative Models

Joint distribution over observed data x and latent variables Z.

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{likelihood}}$$

The likelihood and prior are often standard distributions (Gaussian, Bernoulli) with simple dependence on conditioning information.

Deep generative models

Joint distribution with **deep observation model**

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{likelihood}}$$

mapping from z to $p(x|z, \theta)$ is a NN with parameters θ

Deep generative models

Joint distribution with **deep observation model**

$$p(x, z|\theta) = \underbrace{p(z)}_{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{likelihood}}$$

mapping from z to $p(x|z, \theta)$ is a NN with parameters θ

Marginal likelihood

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

intractable in general

Goals

We want

- richer probabilistic models

Goals

We want

- richer probabilistic models
- complex observation models parameterised by NNs

Goals

We want

- richer probabilistic models
- complex observation models parameterised by NNs

but we can't perform gradient-based MLE

Goals

We want

- richer probabilistic models
- complex observation models parameterised by NNs

but we can't perform gradient-based MLE

We need approximate inference techniques!

Deep Generative Models

First Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Wake-sleep Algorithm

- Generalise latent variables to Neural Networks
- Train generative neural model
- Use variational inference! (kind of)

Wake-sleep Architecture

2 Neural Networks:

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ

Wake-sleep Architecture

2 Neural Networks:

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

Wake-sleep Architecture

2 Neural Networks:

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

Wake-sleep Architecture

2 Neural Networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units
- Training is performed in a "hard EM" fashion

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 liklelihood of data given latent state $p(x|z, \theta)$

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 liklelihood of data given latent state $p(x|z, \theta)$

Sleep Phase

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

Wake Phase Objective

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

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

Wake Phase Objective

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

$$\max_{\theta} \; \mathbb{E}_{q(z|x,\lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]$$
$$\overset{\text{MC}}{\approx} \max_{\theta} \; \log p(z, x|\theta)$$

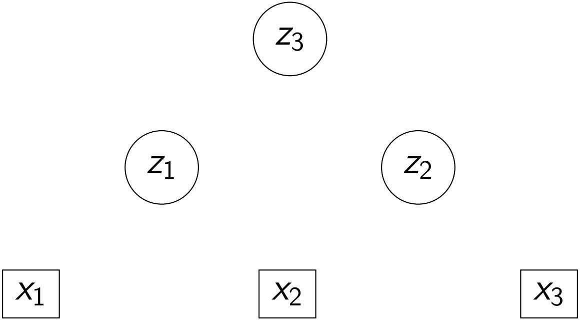
Wake Phase Objective

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

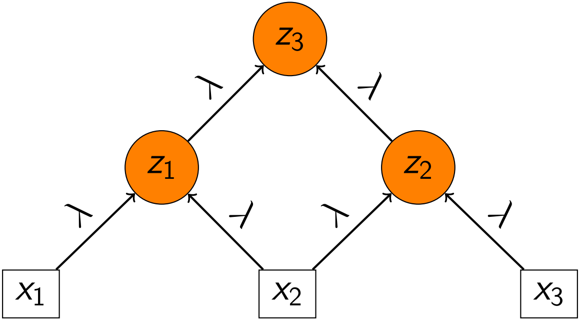
$$\max_{\theta} \; \mathbb{E}_{q(z|x,\lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]$$
$$\overset{\text{MC}}{\approx} \max_{\theta} \; \log p(z, x|\theta)$$

This is simply supervised learning with imputed latent data!

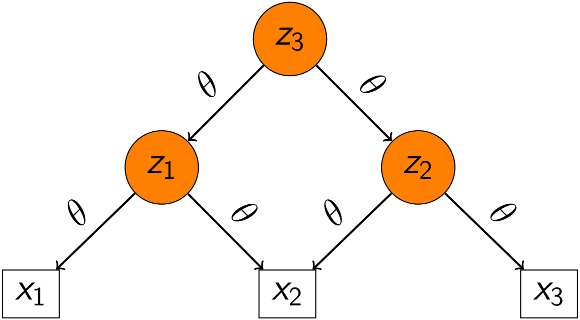
Wake Phase Sampling



Wake Phase Sampling



Wake Phase Update



Sleep Phase Objective

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

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

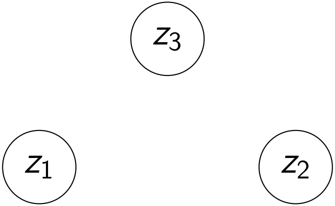
Sleep Phase Objective

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

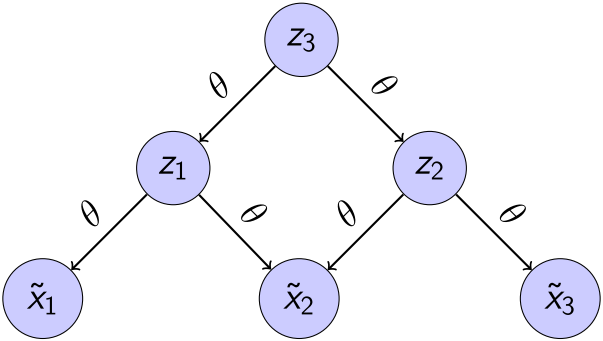
$$\max_{\lambda} \; \mathbb{E}_{p(\tilde{x},z|\theta)} [\log q(z|\tilde{x}, \lambda)] + \mathbb{E}_{p(\tilde{x})} [\mathbb{H} (p(z|\tilde{x}, \theta))]$$

$$\overset{\text{MC}}{\approx} \max_{\lambda} \; \log q(z|\tilde{x}, \lambda)$$

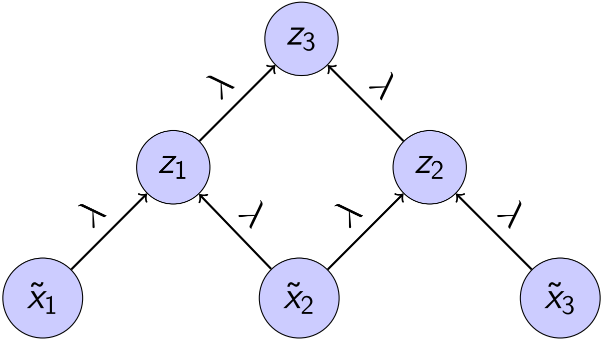
Sleep Phase Sampling



Sleep Phase Sampling



Sleep Phase Update



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

Deep Generative Models

First Attempt: Wake-Sleep

This is how we do: 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.)

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

$p(x) = \int p(x|z, \theta)p(z)\mathrm{d}z$ is hard to compute.

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

$p(x) = \int \underbrace{p(x|z, \theta)}_{\text{highly non-linear!}} p(z)\mathrm{d}z$ is hard to compute.

Solution: Variational Inference

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

Solution: Variational Inference

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

Solution: Variational Inference

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

Solution: Variational Inference

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

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

Solution: Variational Inference

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

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

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

Solution: Variational Inference

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

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

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

Generator Network Gradient

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

Generator Network Gradient

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

Generator Network Gradient

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

Generator Network Gradient

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

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

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

The first term again requires approximation by sampling

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

Not an expected gradient!

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
- $h(z, \lambda) = \epsilon$
- $h^{-1}(\epsilon, \lambda) = z$

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

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

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

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

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

Inference Network Gradient

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

Inference Network Gradient

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

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

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

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

Inference Network Gradient

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

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

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

Derivatives of Gaussian transformation

Recall:

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

Derivatives of Gaussian transformation

Recall:

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

This gives us 2 gradient paths.

Derivatives of Gaussian transformation

Recall:

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

This gives us 2 gradient paths.

$$\frac{dh^{-1}(\epsilon, \lambda)}{d\mu(\phi, \lambda)} = \frac{d}{d\mu(\phi, \lambda)} \left[\mu(\phi, \lambda) + \sigma(\phi, \lambda) \odot \epsilon\right] = 1$$

$$\frac{dh^{-1}(\epsilon, \lambda)}{d\sigma(\phi, \lambda)} = \frac{d}{d\sigma(\phi, \lambda)} \left[\mu(\phi, \lambda) + \sigma(\phi, \lambda) \odot \epsilon\right] = \epsilon$$

Gaussian KL

ELBO

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

Gaussian KL

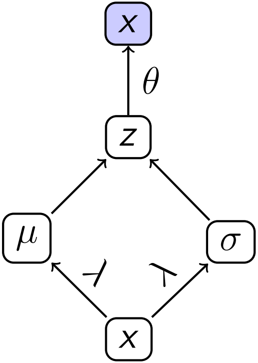
ELBO

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

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

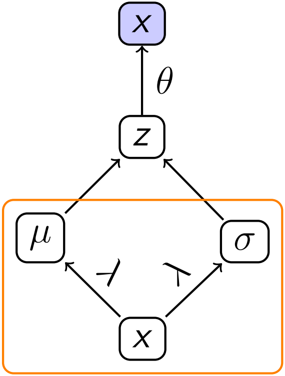
$$\frac{1}{2}\sum_{i=1}^{N}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

Computation Graph



Computation Graph

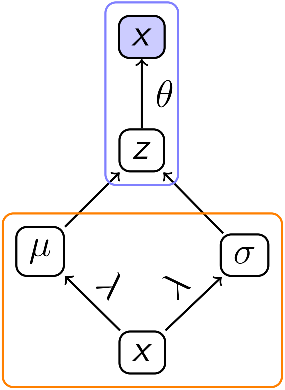
inference model



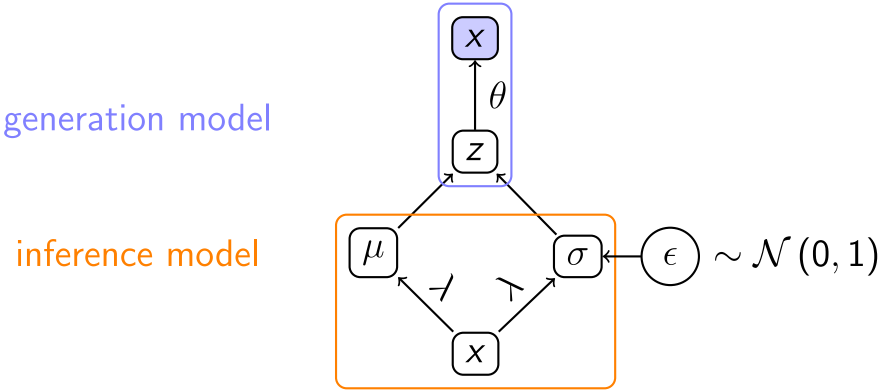
Computation Graph

generation model

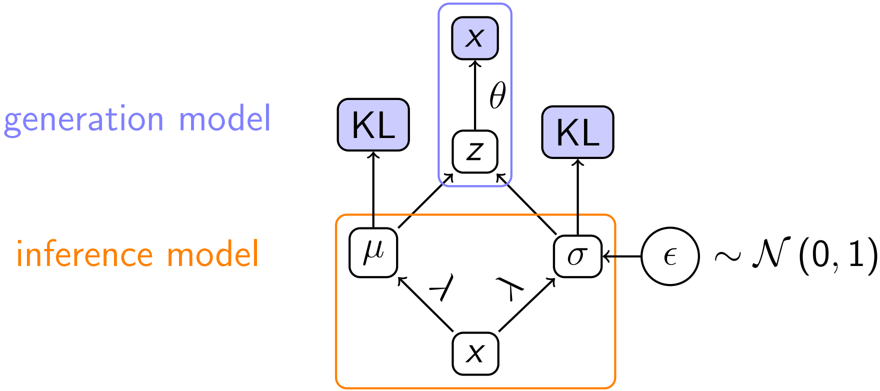
inference model



Computation Graph



Computation Graph



Example

- Data: binary mnist
- Likelihood: product of Bernoullis
  - Let $\phi = \sigma(\mathrm{NN}(z))$
  - $\prod_{i=1}^{N} p(x_i|\phi) = \prod_{i=1}^{N} \phi^{x_i} \times (1-\phi)^{1-x_i}$
- Prior over z: $\mathcal{N}(0,1)$
- $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$
- $\mu(x,\lambda) = \mathrm{NN}_{\mu}(x;\lambda)$
- $\sigma(x,\lambda) = \mathrm{NN}_{\sigma}(x;\lambda)$

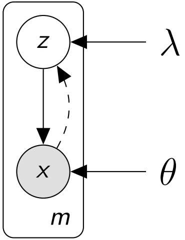
Example

- Data: binary mnist
- Likelihood: product of Bernoullis
  - Let $\phi = \sigma(\text{NN}(z))$
  - $\prod_{i=1}^{N} p(x_i|\phi) = \prod_{i=1}^{N} \phi^{x_i} \times (1-\phi)^{1-x_i}$
- Prior over z: $\mathcal{N}(0, 1)$
- $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- $\mu(x, \lambda) = \text{NN}_{\mu}(x; \lambda)$
- $\sigma(x, \lambda) = \text{NN}_{\sigma}(x; \lambda)$

Mean Field assumption

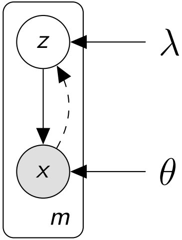
Variational approximation factorises over latent dimensions.

Graphical Model



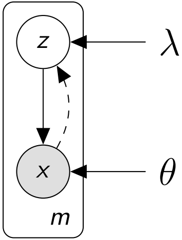
- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$

Graphical Model



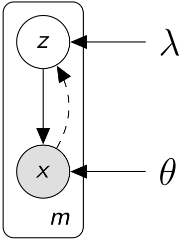
- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- where
  - $\mu(x, \lambda) = \mathrm{NN}_\mu(x; \lambda)$
    e.g. $\mu(x, \lambda) = W^{(u)}x + b^{(u)}$

Graphical Model



- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- where
  - $\mu(x, \lambda) = \mathrm{NN}_\mu(x; \lambda)$
    e.g. $\mu(x, \lambda) = W^{(u)}x + b^{(u)}$
  - $\sigma(x, \lambda) = \exp(\mathrm{NN}_\sigma(x; \lambda))$
    e.g. $\sigma(x, \lambda) = \log\left(1 + \exp\left(W^{(v)}x + b^{(v)}\right)\right)$

Graphical Model



- approximate posterior
  $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$
- where
  - $\mu(x,\lambda) = \mathrm{NN}_\mu(x;\lambda)$
    e.g. $\mu(x,\lambda) = W^{(u)}x + b^{(u)}$
  - $\sigma(x,\lambda) = \exp(\mathrm{NN}_\sigma(x;\lambda))$
    e.g. $\sigma(x,\lambda) =$
    $\log\left(1 + \exp\left(W^{(v)}x + b^{(v)}\right)\right)$
  - $\lambda = \left(W^{(u)}, W^{(v)}, b^{(u)}, b^{(v)}\right)$

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 $\beta \rightarrow 1$.

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

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.