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

Gradient

Exact gradient is intractable

$$\nabla_\theta \log p(x|\theta)$$

Gradient

Exact gradient is intractable

$$\nabla_\theta \log p(x|\theta) = \nabla_\theta \log \underbrace{\int p(x, z|\theta)\,\mathrm{d}z}_{\text{marginal}}$$

Gradient

Exact gradient is intractable

$$\begin{aligned}
\nabla_\theta \log p(x|\theta) &= \nabla_\theta \log \underbrace{\int p(x, z|\theta)\, \mathrm{d}z}_{\text{marginal}} \\
&= \underbrace{\frac{1}{\int p(x, z|\theta)\, \mathrm{d}z} \int \nabla_\theta p(x, z|\theta)\, \mathrm{d}z}_{\text{chain rule}}
\end{aligned}$$

Gradient

Exact gradient is intractable

$$\begin{aligned}\nabla_\theta \log p(x|\theta) &= \nabla_\theta \log \underbrace{\int p(x,z|\theta)\,\mathrm{d}z}_{\text{marginal}} \\ &= \underbrace{\frac{1}{\int p(x,z|\theta)\,\mathrm{d}z} \int \nabla_\theta p(x,z|\theta)\,\mathrm{d}z}_{\text{chain rule}} \\ &= \frac{1}{p(x|\theta)} \int \underbrace{p(x,z|\theta)\nabla_\theta \log p(x,z|\theta)}_{\text{log-identity for derivatives}}\,\mathrm{d}z\end{aligned}$$

Gradient

Exact gradient is intractable

$$
\begin{aligned}
\nabla_\theta \log p(x|\theta) &= \nabla_\theta \log \underbrace{\int p(x,z|\theta)\,\mathrm{d}z}_{\text{marginal}} \\
&= \underbrace{\frac{1}{\int p(x,z|\theta)\,\mathrm{d}z} \int \nabla_\theta p(x,z|\theta)\,\mathrm{d}z}_{\text{chain rule}} \\
&= \frac{1}{p(x|\theta)} \int \underbrace{p(x,z|\theta)\nabla_\theta \log p(x,z|\theta)}_{\text{log-identity for derivatives}}\,\mathrm{d}z \\
&= \int p(z|x,\theta)\nabla_\theta \log p(x,z|\theta)\,\mathrm{d}z
\end{aligned}
$$

Gradient

Exact gradient is intractable

$$
\begin{aligned}
\boldsymbol{\nabla}_\theta \log p(x|\theta) &= \boldsymbol{\nabla}_\theta \log \underbrace{\int p(x, z|\theta)\,\mathrm{d}z}_{\text{marginal}} \\
&= \underbrace{\frac{1}{\int p(x, z|\theta)\,\mathrm{d}z} \int \boldsymbol{\nabla}_\theta p(x, z|\theta)\,\mathrm{d}z}_{\text{chain rule}} \\
&= \frac{1}{p(x|\theta)} \int \underbrace{p(x, z|\theta) \boldsymbol{\nabla}_\theta \log p(x, z|\theta)}_{\text{log-identity for derivatives}}\,\mathrm{d}z \\
&= \int p(z|x, \theta) \boldsymbol{\nabla}_\theta \log p(x, z|\theta)\,\mathrm{d}z \\
&= \underbrace{\mathbb{E}_{p(z|x,\theta)} \left[\boldsymbol{\nabla}_\theta \log p(x, Z|\theta)\right]}_{\text{expected gradient :)}}
\end{aligned}
$$

Can we get an estimate?

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

Can we get an estimate?

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

$$\overset{\text{MC}}{\approx} \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p(x, z^{(k)}|\theta)$$

$$Z^{(k)} \sim p(z|x,\theta)$$

Can we get an estimate?

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

$$\overset{\mathrm{MC}}{\approx} \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p(x, z^{(k)}|\theta)$$

$$Z^{(k)} \sim p(z|x,\theta)$$

MC estimate requires sampling from posterior

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

unavailable due to the intractability of the marginal

Summary

We want

- richer probabilistic models

Summary

We want

- richer probabilistic models
- complex observation models parameterised by NNs

Summary

We want

- richer probabilistic models
- complex observation models parameterised by NNs

but we can't perform gradient-based MLE

Summary

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 $q(z|x,\lambda)$.

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

$$\overset{\text{MC}}{\approx} \max_{\theta} \log p(z,x|\theta)$$

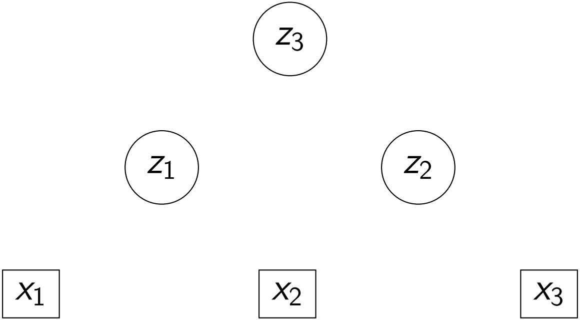
Wake Phase Objective

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

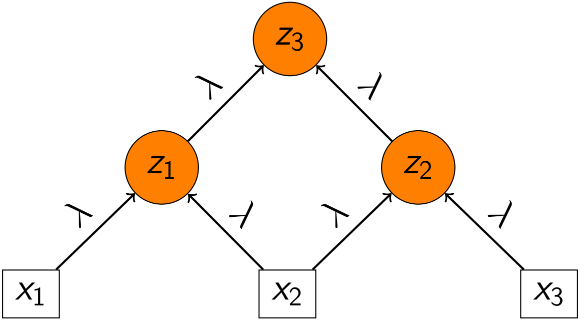
$$\max_{\theta} \mathbb{E}_{q(z|x,\lambda)} [\log p(z, x|\theta)] + \mathbb{H}[q(z|x, \lambda)]$$
$$\overset{\text{MC}}{\approx} \max_{\theta} \log p(z, x|\theta)$$

This is simply supervised learning with imputed latent data!

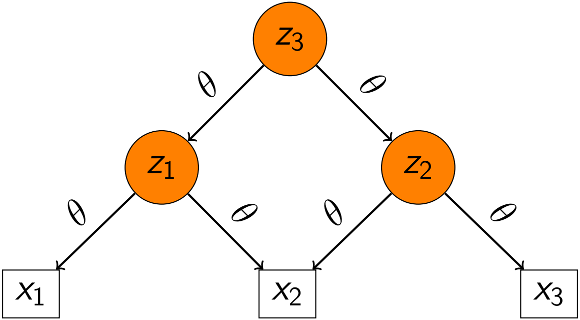
Wake Phase Sampling



Wake Phase Sampling



Wake Phase Update



Sleep Phase Objective

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

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

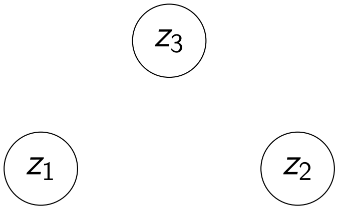
Sleep Phase Objective

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

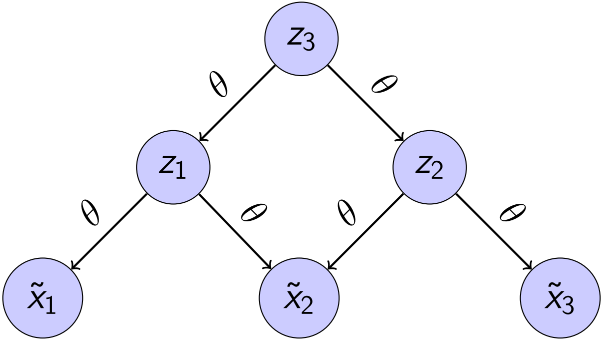
$$\max_{\lambda} \ \mathbb{E}_{p(\tilde{x}, z|\theta)} \left[\log q(z|\tilde{x}, \lambda)\right] + \mathbb{E}_{p(\tilde{x})} \left[\mathbb{H}\left(p(z|\tilde{x}, \theta)\right)\right]$$

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

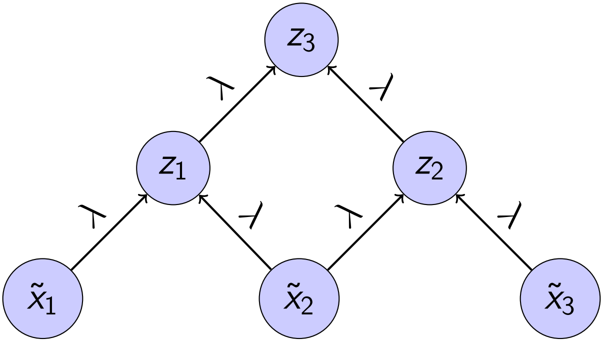
Sleep Phase Sampling



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.

Generative Model with NN Likelihood

Solution: VI

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

Generative Model with NN Likelihood

Solution: VI

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

Generative Model with NN Likelihood

Solution: VI

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

Generative Model with NN Likelihood

Solution: VI

$$
\begin{aligned}
\log p(x) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta) + \log p(z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right] - \underbrace{\mathrm{KL}\left(q(z|x,\lambda) \;||\; p(z)\right)}_{\substack{\text{assume analytical} \\ \text{(true for exponential families)}}}
\end{aligned}
$$

Generative Model with NN Likelihood

Solution: VI

$$
\begin{aligned}
\log p(x) &\geq \overbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x, z|\theta)\right] + \mathbb{H}\left(q(z|x,\lambda)\right)}^{\text{ELBO}} \\
&= \mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta) + \log p(z)\right] + \mathbb{H}\left(q(z|x,\lambda)\right) \\
&= \underbrace{\mathbb{E}_{q(z|x,\lambda)}\left[\log p(x|z,\theta)\right]}_{\text{approximate by sampling}} - \underbrace{\mathrm{KL}\left(q(z|x,\lambda) \,||\, p(z)\right)}_{\substack{\text{assume analytical} \\ \text{(true for exponential families)}}}
\end{aligned}
$$

Generator Network Gradient

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

Generator Network Gradient

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

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

Generator Network Gradient

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

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

$$\overset{\mathrm{MC}}{\approx} \frac{1}{S}\sum_{i=1}^{S}\frac{d}{d\theta}\log p(x|z_i,\theta)$$

Generator Network Gradient

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

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

$$\overset{\text{MC}}{\approx} \frac{1}{S}\sum_{i=1}^{S}\frac{d}{d\theta}\log p(x|z_i,\theta)$$

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

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

Inference Network Gradient

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

The first term again requires approximation by sampling

Inference Network Gradient

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

Inference Network Gradient

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

MC estimator non-differentiable

Inference Network Gradient

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

MC estimator non-differentiable

- Sampling z neglects $\frac{d}{d\lambda}q(z|x,\lambda)$

Inference Network Gradient

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

MC estimator non-differentiable

- Sampling z neglects $\frac{d}{d\lambda}q(z|x,\lambda)$
- Differentiating $q(z|x,\lambda)$ breaks the expectation

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

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

Gaussian Transformation

Affine property

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

Special case

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

Gaussian Transformation

Affine property

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

Special case

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

Gaussian transformation

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

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

Inference Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

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

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

Inference Network Gradient

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

Inference Network Gradient

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

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

Inference Network Gradient

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

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

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

Derivatives of Gaussian transformation

Recall:

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

Derivatives of Gaussian transformation

Recall:

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

This gives us 2 gradient paths.

Derivatives of Gaussian transformation

Recall:

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

This gives us 2 gradient paths.

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

Gaussian KL

ELBO

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

Gaussian KL

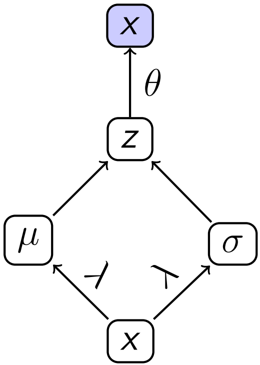
ELBO

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

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

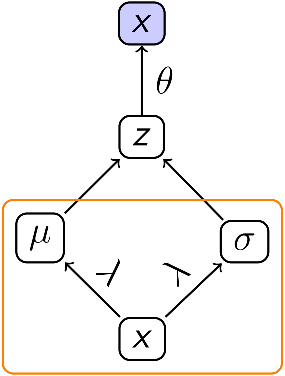
$$\frac{1}{2}\sum_{i=1}^{N}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

Computation Graph



Computation Graph

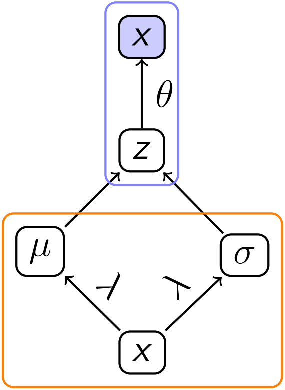
inference model



Computation Graph

generation model

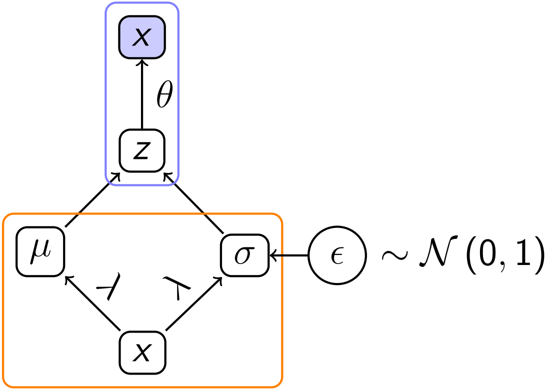
inference model



Computation Graph

generation model

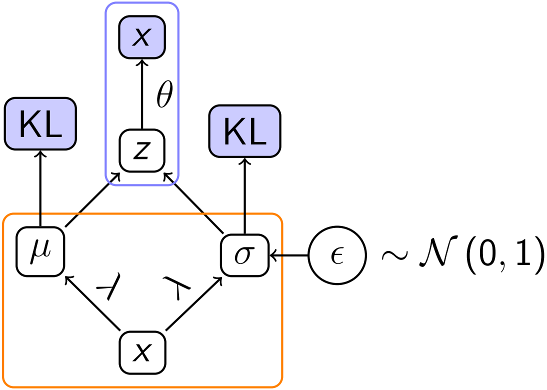
inference model



Computation Graph

generation model

inference model



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: $\mathcal{N}(0, 1)$
- $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- $\mu(x, \lambda) = \text{NN}_\mu(x; \lambda)$
- $\sigma(x, \lambda) = \text{NN}_\sigma(x; \lambda)$

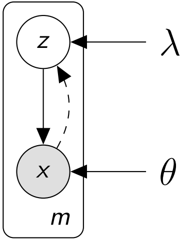
Example

- 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: $\mathcal{N}(0, 1)$
- $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- $\mu(x, \lambda) = \mathrm{NN}_\mu(x; \lambda)$
- $\sigma(x, \lambda) = \mathrm{NN}_\sigma(x; \lambda)$

Mean Field assumption

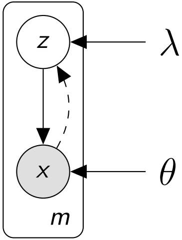
Variational approximation factorises over latent dimensions.

Graphical Model



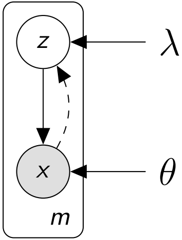
- approximate posterior
  $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$

Graphical Model



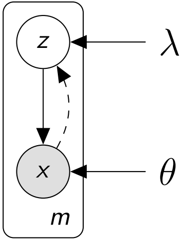
- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- where
  - $\mu(x, \lambda) = \mathrm{NN}_\mu(x; \lambda)$
    e.g. $\mu(x, \lambda) = W^{(u)}x + b^{(u)}$

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.

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

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