Deep Generative Models: Continous Latent Variables

Philip Schulz and Wilker Aziz

https://github.com/philschulz/VITutorial

Generative Models

First Attempt: Log-linear Models

Second Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Generative Models

First Attempt: Log-linear Models

Second Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Recap: Generative Models

Joint distribution over observed data x and latent variables Z.

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

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

Recap: Variational Inference

Objective

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

- The ELBO is a lower bound on $\log p(x)$
- Mean field assumption: $q(z) = \prod_{i=1}^{N} q(z_i)$

Generative Models

First Attempt: Log-linear Models

Second Attempt: Wake-Sleep

This is how we do: Variational Autoencoders

Feature-rich Generative Models

Let us assume that z has internal structure (features). How can we exploit that?

First Idea

Make $p(x|z, \alpha)$ a log-linear model.

- Only discrete data
- Trainable with EM if we can efficiently enumerate $\mathcal{X}$ and $\mathcal{Z}$.

Log-linear Model

Let us treat z as observed.

$$p(x|z, \alpha = w) = \frac{\exp\left(w^\top f(x, z)\right)}{\sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, z)\right)}$$

Log-linear Model

Let us treat z as observed.

$$p(x|z, \alpha = w) = \frac{\exp\left(w^\top f(x,z)\right)}{\sum_{x \in \mathcal{X}} \exp\left(w^\top f(x,z)\right)}$$

Weight Gradient

$$\frac{d}{dw} \log p(x|z,w) = f(x,z) - \mathbb{E}\left[f(X,z)|z,w\right]$$

Log-linear Model

Let us treat z as observed.

$$p(x|z, \alpha = w) = \frac{\exp\left(w^\top f(x,z)\right)}{\sum_{x \in \mathcal{X}} \exp\left(w^\top f(x,z)\right)}$$

Weight Gradient

$$\frac{d}{dw} \log p(x|z,w) = f(x,z) - \mathbb{E}\left[f(X,z)|z,w\right]$$

Updates need to be performed iteratively.

Log-linear model with latent variables

Now let us treat z as latent.

Log-linear model with latent variables

Now let us treat z as latent.

Model

$$p(x, z|w) = \underbrace{\frac{\exp\left(w^\top f(x, z)\right)}{\sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, z)\right)}}_{p(x|z,w)} \times \underbrace{p(z)}_{arbitrary}$$

Log-linear model with latent variables

Posterior

$$p(z|x, w)$$

Log-linear model with latent variables

Posterior

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

Log-linear model with latent variables

Posterior

$$p(z|x, w) = \frac{p(x, z|w)}{p(x|w)} = \frac{p(x, z|w)}{\sum_z p(x, z|w)} =$$

Log-linear model with latent variables

Posterior

$$p(z|x,w) = \frac{p(x,z|w)}{p(x|w)} = \frac{p(x,z|w)}{\sum_z p(x,z|w)} = \frac{\frac{\exp\left(w^\top f(x,z)\right)}{\sum_{x\in\mathcal{X}} \exp(w^\top f(x,z))} \times p(z)}{\sum_z \frac{\exp(w^\top f(x,z))}{\sum_{x\in\mathcal{X}} \exp(w^\top f(x,z))} \times p(z)}$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

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

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

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

$$\frac{d}{dw}\sum_z p(z|x, w)\log p(x, z|w) =$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x,w)$ as fixed)

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

$$\frac{d}{dw}\sum_{z} p(z|x,w)\log p(x,z|w) =$$

$$\sum_{z} p(z|x,w)\frac{d}{dw}\log p(x,z|w) =$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

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

$$\frac{d}{dw}\sum_{z} p(z|x, w) \log p(x, z|w) =$$

$$\sum_{z} p(z|x, w)\frac{d}{dw} \log p(x, z|w) =$$

$$\sum_{z} p(z|x, w)\left(\frac{d}{dw} \log p(x|z, w) + \frac{d}{dw} p(z)\right)$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

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

$$\frac{d}{dw}\sum_z p(z|x, w)\log p(x, z|w) =$$

$$\sum_z p(z|x, w)\frac{d}{dw}\log p(x, z|w) =$$

$$\sum_z p(z|x, w)\left(\overbrace{\frac{d}{dw}\log p(x|z, w)}^{\text{We've already solved this!}} + \overbrace{\frac{d}{dw}p(z)}^{0}\right)$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

$$\frac{d}{dw}\mathbb{E}_{p(z|x,w)}\left[\log p(x, z|w)\right] =$$
$$\mathbb{E}_{p(z|x,w)}\left[f(x, Z)|x, w\right] - \mathbb{E}_{p(z|x,w)}\left[\mathbb{E}\left[(f(X, Z)|Z, w\right]\right]$$

Log-linear model with latent variables

Weight Gradient (treat $p(z|x, w)$ as fixed)

$$\frac{d}{dw}\mathbb{E}_{p(z|x,w)}\left[\log p(x, z|w)\right] =$$
$$\mathbb{E}_{p(z|x,w)}\left[f(x, Z)|x, w\right] - \mathbb{E}_{p(z|x,w)}\left[\mathbb{E}\left[(f(X, Z)|Z, w\right]\right]$$

Procedurally

E_count$(f(x,z))$ - { E_count$(f(x,z)) \times \mathbb{E}\left[f(X,z)|z,w\right]$ }

EM

E-step $p(z|x, w) = \frac{p(x,z|w)}{\sum_z p(x,z|w)}$ in $\mathcal{O}(|\mathcal{X}| \times |\mathcal{Z}|)$

M-step Iteratively optimise w to match E_count(x, z)
with E_count(x, z) $\times \mathbb{E}[X|z, w]$

Restrictions

- Only log-linear models
- Scales badly

Generative Models

First Attempt: Log-linear Models

Second 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 $q(z|\tilde{x},\lambda)$

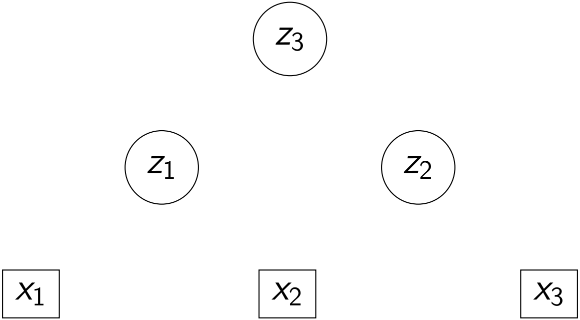
Wake Phase Objective

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

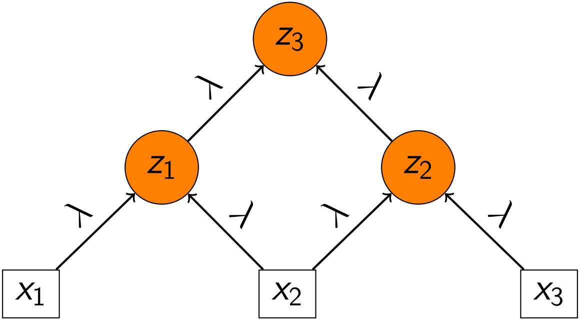
$$\max_{\theta} \log p(x|z, \theta)$$

This is simply supervised learning with imputed latent data!

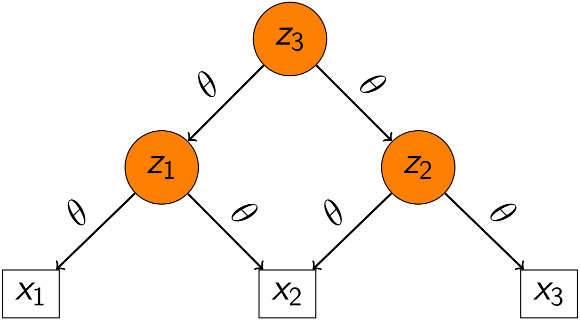
Wake Phase Sampling



Wake Phase Sampling



Wake Phase Update

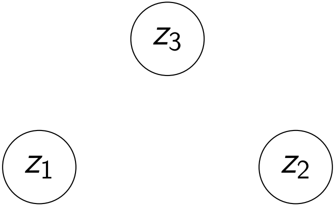


Sleep Phase Objective

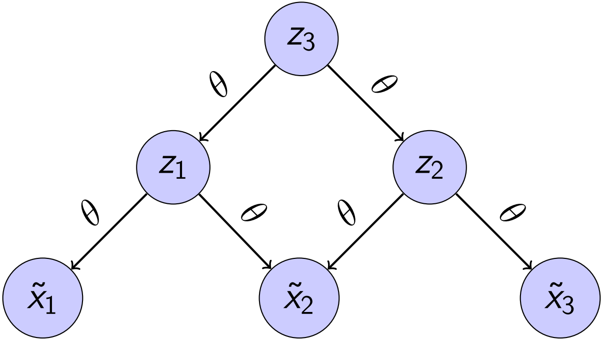
Assumes fake data $\tilde{x}$ and latent variables z to be fixed random draw from $p(x, z|\theta)$.

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

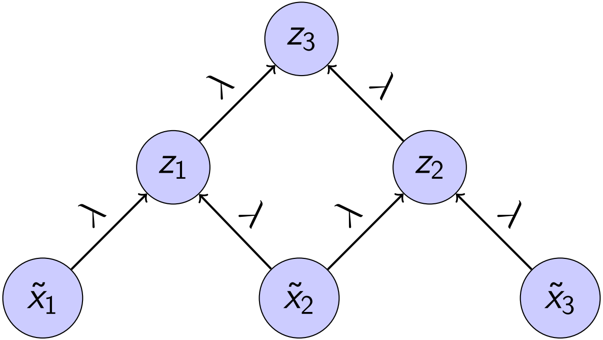
Sleep Phase Sampling



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

Generative Models

First Attempt: Log-linear Models

Second 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

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

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

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

Generation Network Gradient

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

Generation Network Gradient

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

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

Generation Network Gradient

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

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

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

Generation Network Gradient

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

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

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

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

Inference Network Gradient

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

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

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

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

Inference Network Gradient

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

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

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

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.

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

Gaussian KL

ELBO

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

Gaussian KL

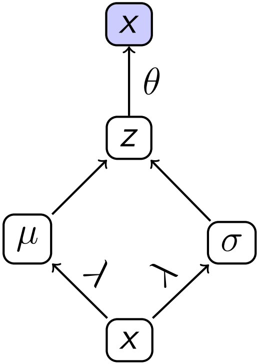
ELBO

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

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

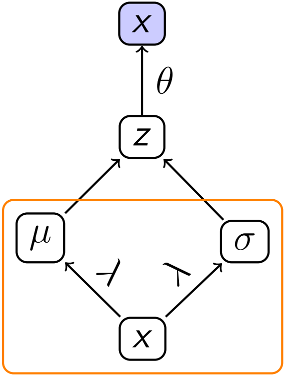
$$\frac{1}{2}\sum_{i=1}^{N}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

Computation Graph



Computation Graph

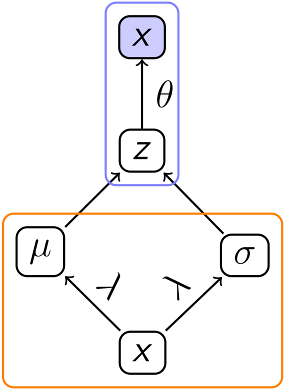
inference model



Computation Graph

generation model

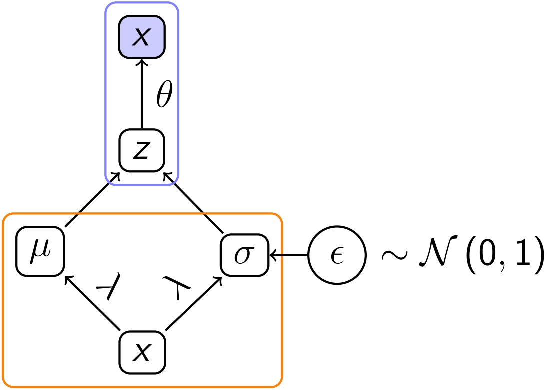
inference model



Computation Graph

generation model

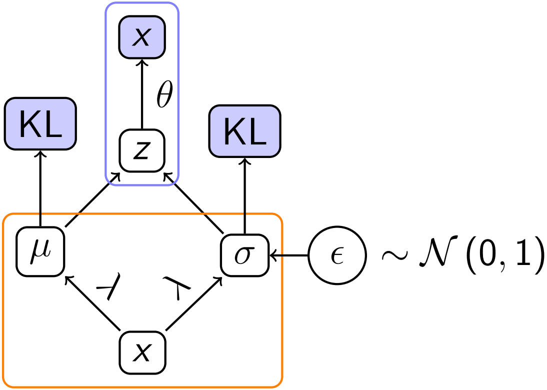
inference model



Computation Graph

generation model

inference model



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

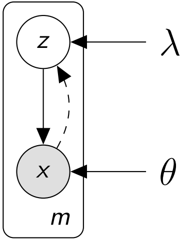
Example

- Data: binary mnist
- Likelihood: product of Bernoullis
  - Let $\phi = \sigma(\mathrm{NN}(z))$
  - $\prod_{i=1}^{N} p(x_i|\phi) = \prod_{i=1}^{N} \phi^{x_i} \times (1-\phi)^{1-x_i}$
- 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

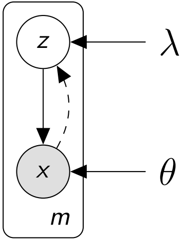
Variational approximation factorises over latent dimensions.

Graphical Model



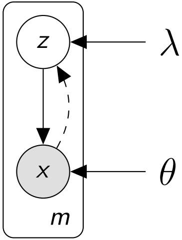
- approximate posterior
  $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$

Graphical Model



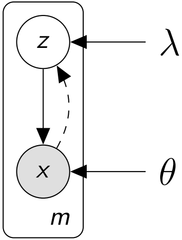
- approximate posterior
  $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$
- where
  - $\mu(x,\lambda) = \mathrm{NN}_\mu(x;\lambda)$
    e.g. $\mu(x,\lambda) = W^{(u)}x + b^{(u)}$

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. $\sigma(x,\lambda) =$
    $\log\left(1 + \exp\left(W^{(v)}x + b^{(v)}\right)\right)$
  - $\lambda = \left(W^{(u)}, W^{(v)}, b^{(u)}, b^{(v)}\right)$

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

- When $|\mathcal{X}|$ and $|\mathcal{Z}|$ are not too large, we can do EM with features
- Otherwise use VI with simple approximation
- 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

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *NAACL*, pages 582–590, 2010. URL http://www.aclweb.org/anthology/N10-1083.

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.

Literature II

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

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.

Literature III

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.

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

Literature IV

http://jmlr.org/proceedings/papers/v32/titsias14.pdf.