Deep Generative Models

Philip Schulz and Wilker Aziz

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 little dependence on side information.

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.

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

Log-linear Model

Let us treat z as observed.

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

Weight Gradient

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

Log-linear Model

Let us treat z as observed.

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

Weight Gradient

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

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

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

Log-linear model with latent variables

Posterior

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

Log-linear model with latent variables

Weight Gradient

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

Log-linear model with latent variables

Weight Gradient

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

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

$$\sum_z p(z|x,w)\underbrace{\frac{d}{dw}\log p(x,z|w)}_{\text{We've already solved this!}}$$

Log-linear model with latent variables

Weight Gradient

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

Log-linear model with latent variables

Weight Gradient

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

Procedurally

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

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!

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!

Sleep Phase Objective

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

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

Wake-sleep Algorithm

Advantages

- Backprop can be used without modification
- Amortised inference: all latent variables are inferred from the same weights λ

Wake-sleep Algorithm

Advantages

- Backprop can be used without modification
- 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

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

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

Generative Model with NN Likelihood

Solution: VI

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

Generative Model with NN Likelihood

Solution: VI

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

Generation Network Gradient

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

Inference Network Gradient

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

Inference Network Gradient

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

Problems for MC

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

Inference Network Gradient

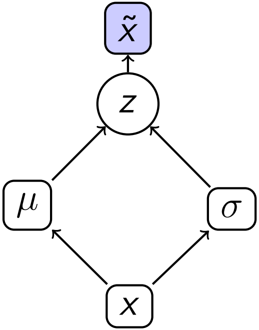
$$= \frac{d}{d\lambda} \int q(z|x,\lambda) \log p(x|z,\theta) \mathrm{dz}$$

$$= \frac{d}{d\lambda} \int q(\epsilon) \log \left(p(x|\overbrace{h(\epsilon,\lambda)}^{=z},\theta) \times \underbrace{\left| \frac{d}{d\epsilon} h(\epsilon,\lambda) \right|}_{\text{constant if h linear}} \right) \mathrm{d}\epsilon$$

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

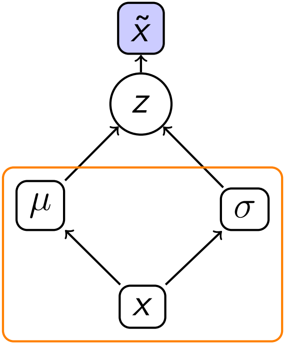
$$= \mathbb{E}_{p(\epsilon)} \left[\frac{d}{d\lambda} \log p(x|h(\epsilon,\lambda),\theta) \right] \overset{\text{MC}}{\approx} \sum_{i=1}^{S} \frac{d}{d\lambda} \log p(x|h(\epsilon,\lambda),\theta)$$

Computation Graph



Computation Graph

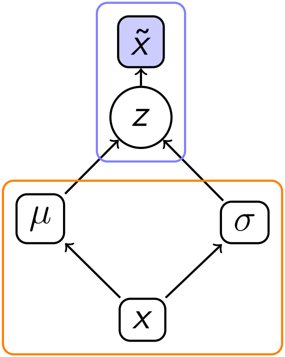
inference model



Computation Graph

generation model

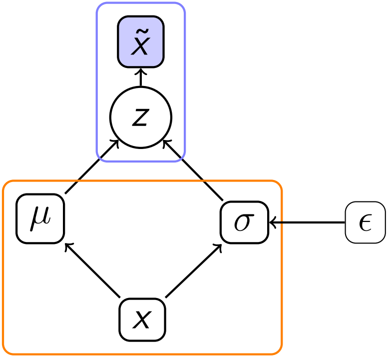
inference model



Computation Graph

generation model

inference model



Example

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

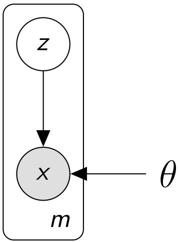
Example

- Data: binary mnist
- Likelihood: product of Bernoullis
- 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

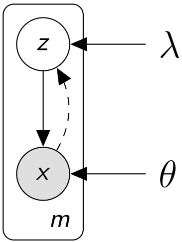
Variational approximation factorises over latent dimensions.

Graphical Model



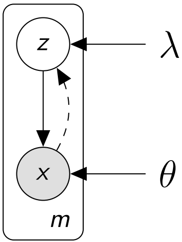
- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- where
  - $\mu(x, \lambda) = \text{NN}_\mu(x; \lambda)$
    e.g.
    $\mu(x, \lambda) = \tanh(W^{(u)} r(x) + b^{(u)})$
  - $\sigma(x, \lambda) = \exp(\text{NN}_\sigma(x; \lambda))$
    e.g. $\sigma(x,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

Graphical Model



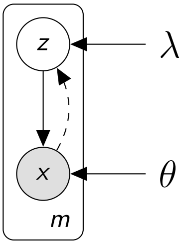
- approximate posterior
  $q(z|x,\lambda) = \mathcal{N}\left(\mu(x,\lambda), \sigma(x,\lambda)^2\right)$

Graphical Model



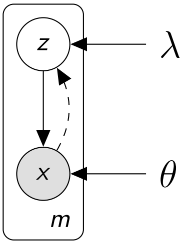
- approximate posterior
  $q(z|x, \lambda) = \mathcal{N}\left(\mu(x, \lambda), \sigma(x, \lambda)^2\right)$
- where
  - $\mu(x, \lambda) = \text{NN}_\mu(x; \lambda)$
    e.g.
    $\mu(x, \lambda) = \tanh(W^{(u)} r(x) + b^{(u)})$

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.
    $\mu(x, \lambda) = \tanh(W^{(u)} r(x) + b^{(u)})$
  - $\sigma(x, \lambda) = \exp(\mathrm{NN}_\sigma(x; \lambda))$
    e.g. $\sigma(x, \lambda) =$
    $\exp\left(\tanh(W^{(v)} r(x) + b^{(v)})\right)$

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.
    $\mu(x, \lambda) = \tanh(W^{(u)} r(x) + b^{(u)})$
  - $\sigma(x, \lambda) = \exp(\mathrm{NN}_\sigma(x; \lambda))$
    e.g. $\sigma(x, \lambda) =$
    $\exp\left(\tanh(W^{(v)} r(x) + b^{(v)})\right)$
  - $\lambda = \left(W^{(u)}, W^{(v)}, b^{(u)}, b^{(v)}\right)$