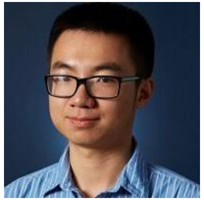
CS294-158 Deep Unsupervised Learning

Lecture 3b: Likelihood Models III: Latent variable models

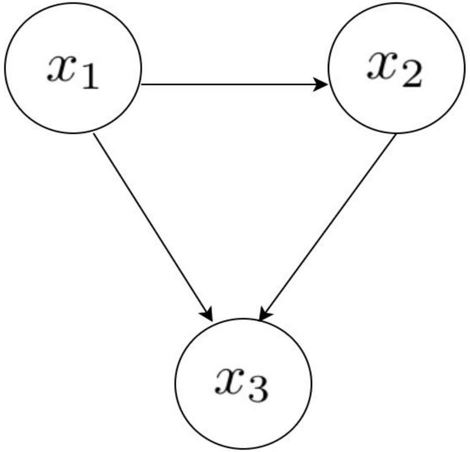
   

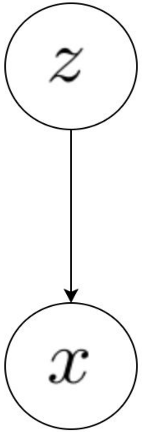
Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas

UC Berkeley

Latent Variable Models

- Autoregressive models + Flows
  - All random variables are observed
- Latent Variable Models (LVMs):
  - Some random variables are hidden - we do not get to observe





Why Latent Variable Models?

- Simpler, lower-dimensional representations of data often possible
  - Latent variable models hold the promise of automatically identifying those hidden representations

Why Latent Variable Models?

- AR models are slow to sample because all pixels (observation dims) are assumed to be dependent on each other
- We can make part of observation space independent *conditioned on some latent variables*
  - Latent variable models *can* have faster sampling by exploiting statistical patterns

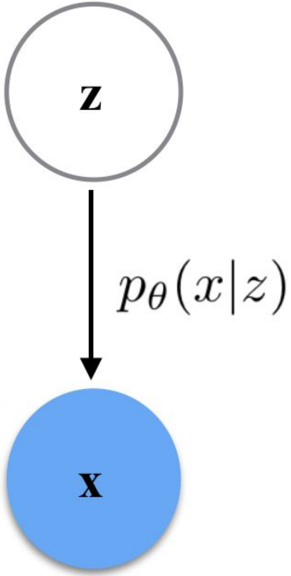
Latent Variable Models

- Sometimes, it’s possible to design a latent variable model with an understanding of the causal process that generates data
- In general, we don’t know what are the latent variables and how they interact with observations
  - Most popular models make little assumption about what are the latent variables
  - Best way to specify latent variables is still an active area of research

A simple latent variable model

$$z = (z_1, z_2, \cdots, z_K) \sim p(z; \beta) = \prod_{k=1}^{K} \beta_k^{z_k} (1 - \beta_k)^{1 - z_k}$$

$$x = (x_1, x_2, \cdots, x_L) \sim p_\theta(x|z) \Leftrightarrow \text{Bernoulli}(x_i; \text{DNN}(z))$$



Training LVM

- Now we have a model with parameters, how to train it?
- Maximum Likelihood again!
- Even for small K = 32, it's a summation over 4B terms

$$\log p(x) = \log \left(\sum_z p(x|z)p(z) \right)$$

$$z = (z_1, z_2, \cdots, z_K)$$

$$p(z;\phi) = \prod_{k=1}^{K} \phi_k^{z_k} (1-\phi_k)^{1-z_k}$$

$$\theta \leftarrow \text{argmax}_\theta \left[\log p_\theta(x) = \log \left(\sum_z p_\theta(x|z)p(z;\phi) \right) \right]$$

Variational Inference (VI)

- Recap: we have O(1) access to p(z) and p(x|z) and want to efficiently compute p(x)
- Assume we have an oracle: p(z|x)
  - intuition: what was the z that generates this x?

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

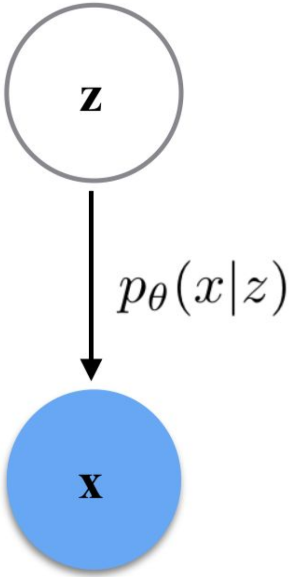
- But how do we obtain p(z|x)?

Variational Inference (VI)

Given this simple graphical model, we are interested in the posterior distribution of the latent variables.

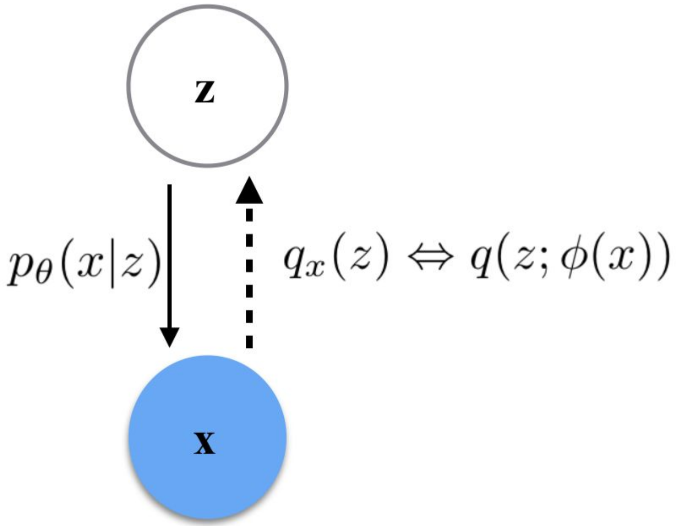
$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x,z)}{\sum_z p(x|z)p(z)}$$

Intractable



Variational Inference (VI)

Key idea in variational inference is to approximate the true posterior with a variational distribution. This results in an optimization problem that minimizes the distance between these two distributions, ex: KL divergence.



Variational Inference (VI)

- Let $q_x(z)$ be our approximation of true posterior $p(z|x)$
  - Then we pick a divergence to minimize between these two distributions

$$
\begin{aligned}
D_{\mathrm{KL}}\left[q_x(z) \,\|\, p(z|x)\right] &= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z|x)\right] \\
&= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log \frac{p(z,x)}{p(x)}\right] \\
&= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z) + \log p(x)\right] \\
&= \underbrace{\mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right]}_{\text{Only this part depends on } z} + \log p(x)
\end{aligned}
$$

- note: the expectation can be approximated by stochastic samples, and every term in expectation can be computed in O(1) now

Variational Lower Bound (VLB)

- We now have an objective amenable to stochastic optimization

$$D_{\mathrm{KL}}\left[q_x(z) \,\|\, p(z|x)\right] = \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + \log p(x)$$

- Turns out we can get more out of this exercise

$$\begin{aligned}\log p(x) &= -\mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log p(z) - \log p(x|z)\right] + D_{\mathrm{KL}}\left[q_x(z) \,\|\, p(z|x)\right] \\ &= \underbrace{\mathbb{E}_{z\sim q_x(z)}\left[\log p(z) + \log p(x|z) - \log q_x(z)\right]}_{\text{Variational Lower Bound}} + \underbrace{D_{\mathrm{KL}}\left[q_x(z) \,\|\, p(z|x)\right]}_{\geq 0}\end{aligned}$$

  - note: the optimal $q_x(z)$ of VLB is p(z|x), at which point VLB is tight (= log p(x))

VLB Maximization

- Given a data distribution x ~ p_{data}, we can know train the generative model by maximizing the VLB under data distribution

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\mathbb{E}_{\mathbf{z}\sim q_x(\mathbf{z})}\left[\log p(\mathbf{z}) + \log p(\mathbf{x}|\mathbf{z}) - \log q_x(\mathbf{z})\right]\right]$$
$$\leq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p(\mathbf{x})\right]$$

Variational Inference

- Core idea: learn an approximation of the intractable posterior
- Widely applicable:
  - learning latent variable generative models
  - estimation / maximization of Mutual Information
  - learnable dequantization
  - ….

Stochastic optimization of VLB

$$z = (z_1, z_2, \cdots, z_K) \sim p(z;\beta) = \prod_{k=1}^{K} \beta_k^{z_k}(1-\beta_k)^{1-z_k}$$

$$x = (x_1, x_2, \cdots, x_L) \sim p_\theta(x|z) \Leftrightarrow \mathrm{Bernoulli}(x_i; \mathrm{DNN}(z))$$

$$\mathrm{VLB} = \mathbb{E}_{z\sim q(z;\phi(x))}\left[\log p(x|z) - \log q(z;\phi(x)) + \log p(z)\right]$$

In the Bernoulli setting, this becomes:

$$\mathbb{E}_{z\sim q(z;\phi(x))}\left[\log p(x|z;\theta) - \log q(z;\phi(x)) + \log p(z;\beta)\right]$$

Core problem: how to optimize the expectation from which z is drawn

Wake-Sleep algorithm

- Note: VLB was derived from min KL[$q_x(z)||p(z|x)$], hard to optimize because z drawn from $q_x(z)$
- What if we instead minimize KL[$p(z|x)||q_x(z)$] for any given x
  - still hard because we don't know $p(z|x)$
- Trick: we know z if we generate them! $\mathbf{z} \sim p_\beta(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$
  - caveat $x \sim p_{model}$ instead of $x \sim p_{data}$
- Problem?
  - VLB is maximized with x drawn from p_{data}
  - minimizing kl(p || q) with x drawn from p_{model} doesn't guarantee the bound is tight

Wake-Sleep algorithm

Wake Phase

- Sample $x \sim p_{\text{data}}$, $z \sim q(z; \phi(x))$.
- Maximize VLB with respect to θ, β.

Sleep Phase (model *dreaming* samples)

- Sample $z \sim p(z; \beta)$, $x \sim p(x|z; \theta)$.
- Minimize $KL(p(z|x)||q(z; \phi(x)))$ with respect to ϕ now that we have samples from p_{model}.

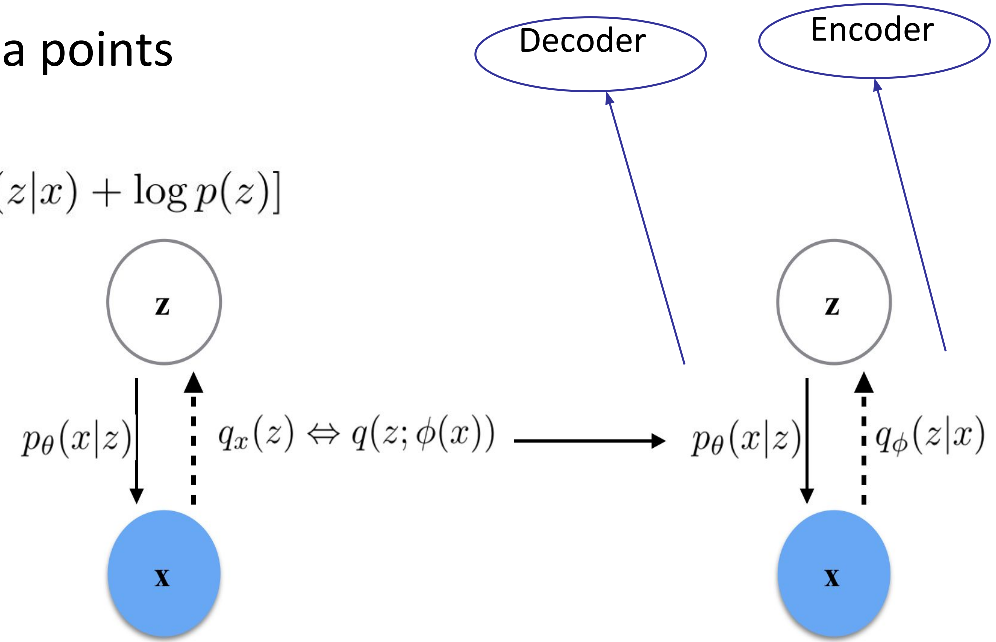
Reverse KL

$$KL(p(z|x)||q(z; \phi(x)) = E_{z \sim p(z|x)} \left[\underbrace{\log p(z|x)}_{\text{indepndent of } \phi} - \log q(z; \phi(x)) \right]$$

Amortized Inference

- $q(z;\phi(x))$
  - Not scalable to large dataset
  - Expensive to evaluate new data points
- Amortized Inference

$$VLB = \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[\log p_\theta(x|z) - \log q_\phi(z|x) + \log p(z)\right]$$



Helmholtz Machine

- Helmholtz Machine [Dayan, P., Hinton, G. E.,... 1995]
  - Bernoulli latent code + observation space
  - Learned with Wake-sleep algorithm
- Did not scale to solve more complex problems due to limitations of wake-sleep

Directly optimizing VLB

- Wake-Sleep not effective, especially when p_{model} is far away from p_{data}
- Can we directly optimize VLB?

Recall, we want

$$\phi, \theta \leftarrow \text{argmax}_{\theta,\phi} \left(\mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[\log p_\theta(x|z) - \log q_\phi(z|x) + \log p(z) \right] \right)$$

Optimization with respect to ϕ is of the form

$$\text{argmax}_\phi \mathbb{E}_{z \sim q_\phi} \left[f(z) \right]$$

Well studied problem in reinforcement learning where no assumption on f is made.

Likelihood Ratio Estimator

We are interested in $\text{argmax}_\phi \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[f(z)\right]$

How do we compute $\nabla_\phi \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[f(z)\right]$?

$$\nabla_\phi \sum_z q_\phi(z|x) f(z) = \sum_z \nabla_\phi q_\phi(z|x) f(z) = \sum_z \underbrace{\frac{\nabla_\phi q_\phi(z|x)}{q_\phi(z|x)}} f(z) q_\phi(z|x)$$

$$\Rightarrow \nabla_\phi \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[f(z)\right] = \sum_z \left(\nabla_\phi \log q_\phi(z|x) f(z)\right) q_\phi(z|x) = \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[\nabla_\phi \log q_\phi(z|x) f(z)\right]$$

$$\phi \leftarrow \phi + \alpha \nabla_\phi \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[\nabla_\phi \log q_\phi(z|x) f(z)\right]$$

Issue: High variance gradients, needs many samples of z to form a good estimate [Demo]

Pathwise Derivative (PD)

One other way to optimize this objective when z is continuous is to cast z as a function of a simple fixed noise such as standard gaussian.

$$z = g(\epsilon, \phi), \epsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[f(g(\epsilon, \phi))\right]$$

When f is differentiable,

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(\cdot|x)}\left[f(z)\right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[\nabla_\phi f(g(\epsilon, \phi))\right]$$

Pathwise Derivative (PD)

- Stochastic gradient possible if z is continuous now (more technical condition?)
  - Common choice: \eps ~ Normal, f(\eps) = \mu + \sigma \eps
  - Any flow that you just learned!
- Also known as reparameterization trick
- Can work with only 1~2 samples
- Demo

PD applied to VI

Variational AutoEncoder

$q_\phi(z|x)$ is modeled as a Gaussian with parameters μ and σ a DNN encoder (parameters ϕ) of x. The DNN decoder $p_\theta(x|z)$ is differentiable.

$$\text{Let } z = \Sigma^{1/2}(x;\phi)\epsilon + \mu(x;\phi)$$

$$\begin{aligned}
\text{VLB} &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(x|z) - \log q_\phi(z|x) + \log p(z)\right] \\
&= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log p_\theta(x|z)\right] - KL(q_\phi(z|x)||p(z))
\end{aligned}$$

$\nabla_\theta\,[\text{VLB}]$ and $\nabla_\phi\,[\text{VLB}]$ can now be efficiently computed with SGD.

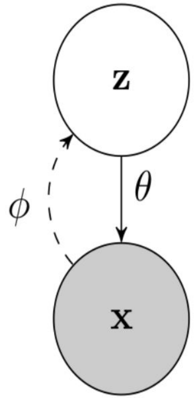
Why is it called an autoencoder?

- We have seen that a variational autoencoder is a latent variable model with Gaussian prior p(z) and approximate posterior q(z|x).
  - Why is it called an "autoencoder"?

$$\log p_\theta(x) \geq \underbrace{\underbrace{\left(E_{z\sim q_x(z)} \log p_\theta(x|z)\right)}_{\text{Reconstruction loss}} \quad - \quad \underbrace{KL(q_\phi(z|x)||p(z))}_{\text{Regularization}}}_{L(\theta,\phi)\text{ - VAE objective}}$$

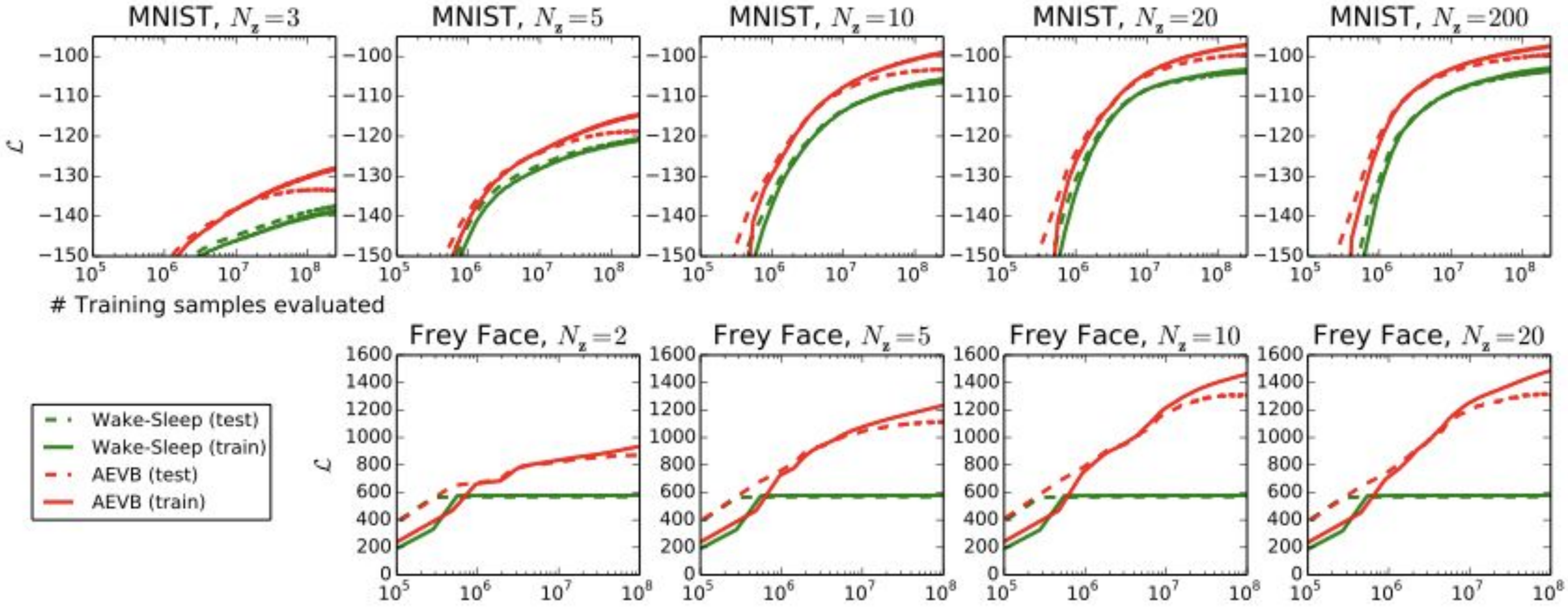
VAE

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$



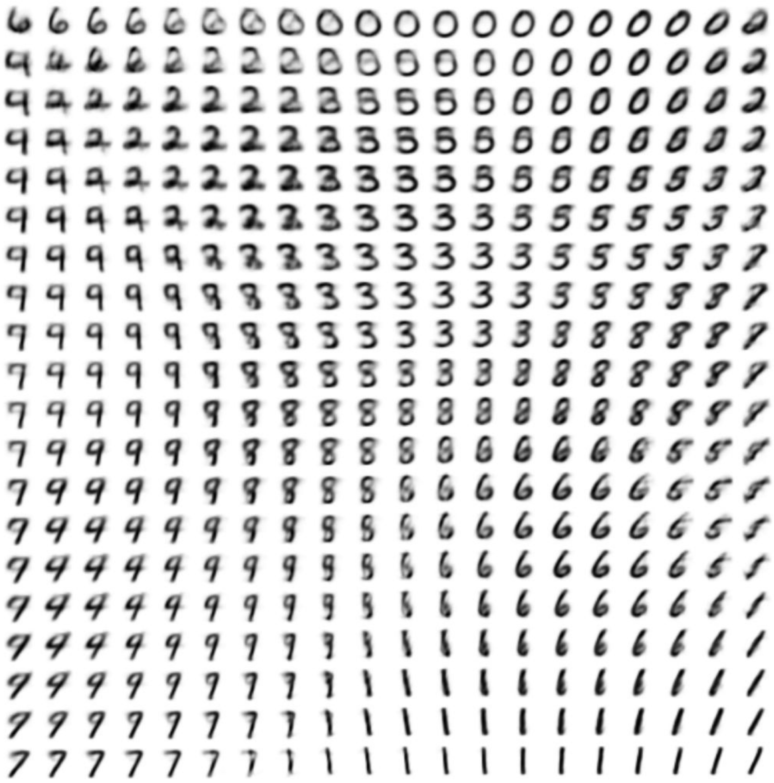
$$p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$$

VAE



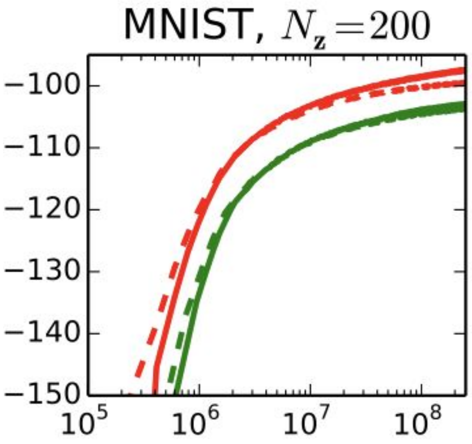
MNIST, $N_z = 3$
MNIST, $N_z = 5$
MNIST, $N_z = 10$
MNIST, $N_z = 20$
MNIST, $N_z = 200$
$\mathcal{L}$
−100
−110
−120
−130
−140
−150
10^5
10^6
10^7
10^8
Training samples evaluated
Frey Face, $N_z = 2$
Frey Face, $N_z = 5$
Frey Face, $N_z = 10$
Frey Face, $N_z = 20$
1600
1400
1200
1000
800
600
400
200
0
Wake-Sleep (test)
Wake-Sleep (train)
AEVB (test)
AEVB (train)

VAE



Compared to AR

- We now have a family of trainable latent variable models!
- But performance is lacking



Model	NLL Test
DBM 2hl [1]:	≈ 84.62
DBN 2hl [2]:	≈ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	≈ 86.60
DLGM 8 leapfrog steps [6]:	≈ 85.51
DARN 1hl [7]:	≈ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	≤ 80.97
PixelCNN:	81.30

Bridging the gap between AR and VAE

- Improve Variational Inference
- Flexible decoder
- More expressive model architectures