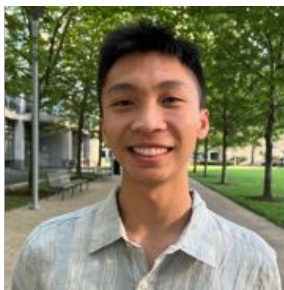


CS294-158 Deep Unsupervised Learning

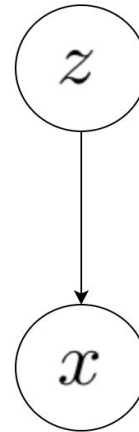
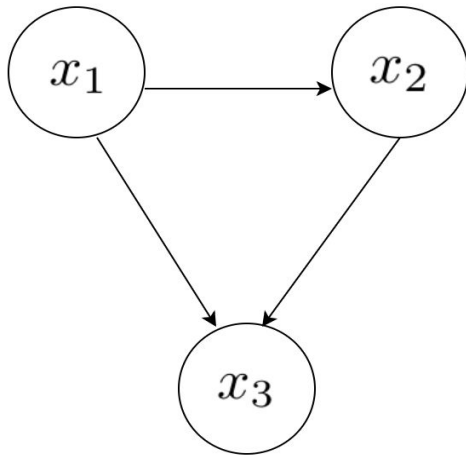
Lecture 4 Latent Variable Models -- Variational AutoEncoder (VAE)



Pieter Abbeel, Wilson Yan, Kevin Frans, Philipp Wu

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

Obj1 @ (x,y)
= Corgi, red
& white

Obj2 @ (x,y)
= Corgi, red
& white,
floppy left
ear



Background
= Wood
bench in a
park

Obj3 @ (x,y)
= Corgi,
Tri-color

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

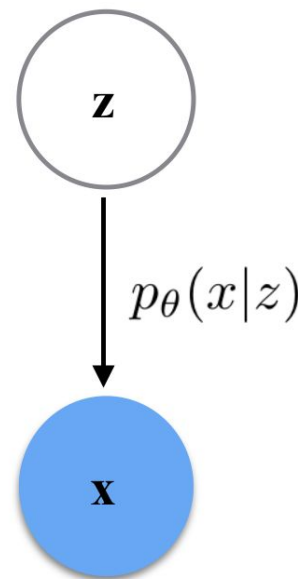
Latent Variable Models

- 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

Example latent variable model

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

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



Latent Variable Model

- Sample

$$z \sim p_Z(z)$$

$$x \sim p_\theta(x | z)$$

- Evaluate likelihood

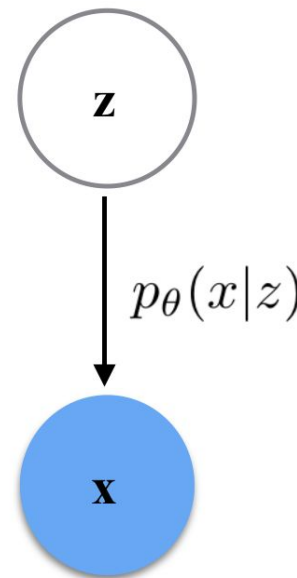
$$p_\theta(x) = \sum_z p_Z(z) p_\theta(x | z)$$

- Train

$$\max_{\theta} \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} | z)$$

- Representation

$$x \rightarrow z$$



Outline

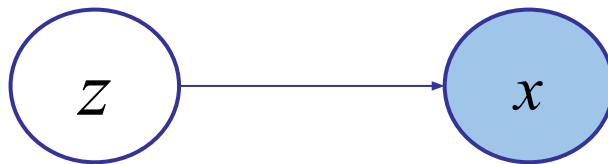
- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - Mutual Information Estimation

Outline

- Motivation
- **Training Latent Variable Models (including VAE and IWAE)**
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - Mutual Information Estimation

Training Latent Variable Models

- Objective:



$$\max_{\theta} \sum_i \log p_{\theta}(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_{\theta}(x^{(i)} | z)$$

- Scenario 1: z can only take on a small number of values $\rightarrow$ exact objective tractable
- Scenario 2: z can take on an impractical number of values to enumerate $\rightarrow$ approximate

How about optimizing $p_Z(z)$? = “learning the prior” and sometimes done [more later]

Outline

- Motivation
- **Training Latent Variable Models (including VAE and IWAE)**
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - Mutual Information Estimation

Exact Likelihood Objective

Example: mixture of 3 Gaussians, with uniform prior over components

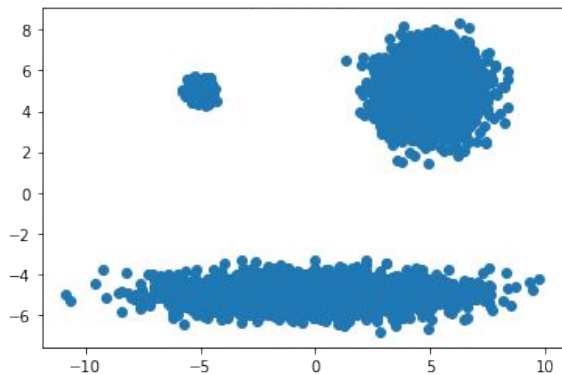
$$p_{\theta}(x) = \sum_z p_Z(z) p_{\theta}(x \mid z) \qquad p_Z(z = A) = p_Z(z = B) = p_Z(z = C) = \frac{1}{3}$$
$$p_{\theta}(x \mid z = k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu_k)^{\top} \Sigma_k^{-1} (x - \mu_k) \right)$$

Training objective:

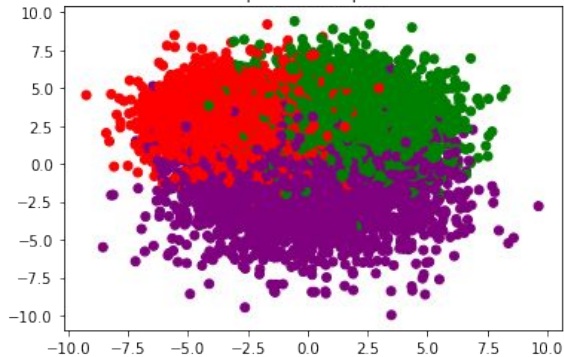
$$\begin{aligned} & \max_{\theta} \sum_i \log p_{\theta}(x^{(i)}) \\ &= \max_{\mu, \Sigma} \sum_i \log \left[\frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_A|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x^{(i)} - \mu_A)^{\top} \Sigma_A^{-1} (x^{(i)} - \mu_A) \right) \right. \\ & \quad + \frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_B|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x^{(i)} - \mu_B)^{\top} \Sigma_B^{-1} (x^{(i)} - \mu_B) \right) \\ & \quad \left. + \frac{1}{3} \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_C|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x^{(i)} - \mu_C)^{\top} \Sigma_C^{-1} (x^{(i)} - \mu_C) \right) \right] \end{aligned}$$

2-D Mixture of Gaussians

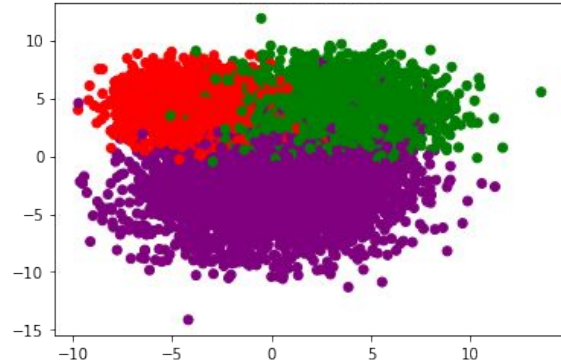
Train Data



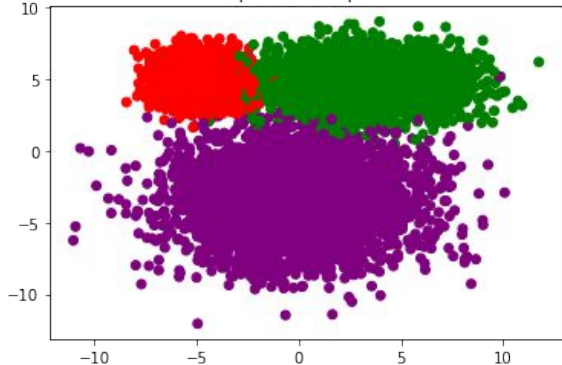
Epoch 0 Samples



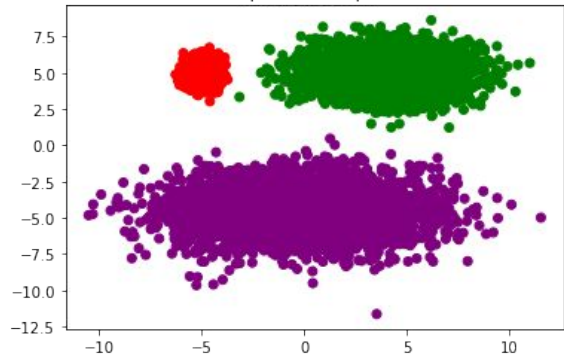
Epoch 2 Samples



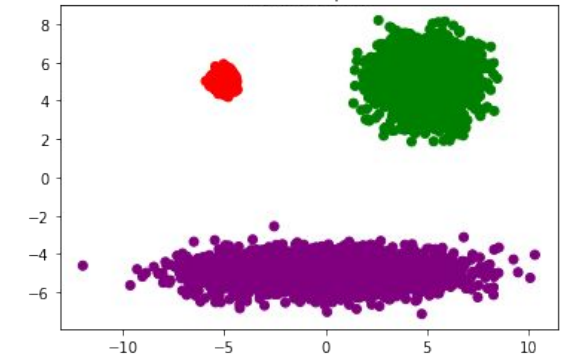
Epoch 4 Samples



Epoch 6 Samples



Final Samples



Prior Sampling

Main idea: if z can take on many values $\rightarrow$ sample z

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} | z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^K p_\theta(x^{(i)} | z_k^{(i)}) \quad z_k^{(i)} \sim p_Z(z)$$

$\rightarrow$ run Stochastic Gradient Descent (SGD) on the approximate objective

Prior Sampling -- Example + Challenge

Consider data in N clusters:

Sampling z uniformly results in only $1/N$ terms being useful.

Recall: z might correspond to many high level properties, e.g., 100 high level properties, probability of correct z for given x : 0.5^{100}

Issue: When going to higher dimensional data, it becomes near impossible to be lucky enough that a sampled z is a good match for a data point $x^{(i)}$

Importance Sampling -- Motivation / Background

Problem setting:

Want to compute $\mathbb{E}_{z \sim p_Z(z)} [f(z)]$

But: (1) hard to sample from $p_Z(z)$
and/or (2) samples from $p_Z(z)$ are not very informative

Example of (2):

Note: our Latent Variable Model objective is also example of (2)

Importance Sampling -- Algorithm

Formulation:

$$\begin{aligned}\mathbb{E}_{z \sim p_Z(z)} [f(z)] &= \sum_z p_Z(z) f(z) \\ &= \sum_z \frac{q(z)}{q(z)} p_Z(z) f(z) \\ &= \mathbb{E}_{z \sim q(z)} \left[\frac{p_Z(z)}{q(z)} f(z) \right] \\ &\approx \frac{1}{K} \sum_{k=1}^K \frac{p_Z(z^{(k)})}{q(z^{(k)})} f(z^{(k)}) \quad \text{with } z^{(k)} \sim q(z)\end{aligned}$$

→ Can sample from q to compute expectation w.r.t. p

Importance Sampling for Latent Variable Model

Training Objective:

$$\sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} | z) \approx \sum_i \log \frac{1}{K} \sum_{k=1}^K \frac{p_Z(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)} | z_k^{(i)}) \quad \text{with } z_k^{(i)} \sim q(z_k^{(i)})$$

Good proposal distribution $q(z)$?

We want samples compatible with $x^{(i)}$

→ How about $q(z) = p_\theta(z | x^{(i)}) = \frac{p_\theta(x^{(i)} | z) p_Z(z)}{p_\theta(x^{(i)})}$

Issue: not clear how to sample from this distribution...

Importance Sampling Proposal Distribution

General Principle of Variational Approach:

We can't directly use p we want

So, instead, we propose a parameterized distribution q we know we can work with easily (in this case, sample from easily), and try to find a parameter setting that makes it as good as possible.

E.g. find $q(z) = \mathcal{N}(z; \mu, \sigma^2)$ as close as possible to $p_\theta(z | x^{(i)}) = \frac{p_\theta(x^{(i)} | z)p_Z(z)}{p_\theta(x^{(i)})}$

Importance Sampling Proposal Distribution

→ optimize to find q

$$\begin{aligned} & \min_{q(z)} \text{KL}(q(z) \| p_{\theta}(z | x^{(i)})) \\ = & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_{\theta}(z | x^{(i)})} \right) \\ = & \min_{q(z)} \mathbb{E}_{z \sim q(z)} \log \left(\frac{q(z)}{p_{\theta}(x^{(i)} | z) p_Z(z) / p_{\theta}(x^{(i)})} \right) \\ = & \min_{q(z)} \mathbb{E}_{z \sim q(z)} [\log q(z) - \log p_Z(z) - \log p_{\theta}(x^{(i)} | z)] + \log p_{\theta}(x^{(i)}) \\ = & \min_{q(z)} \mathbb{E}_{z \sim q(z)} [\log q(z) - \log p_Z(z) - \log p_{\theta}(x^{(i)} | z)] + \text{constant independent of } z \end{aligned}$$

Note: all needed quantities in the objective readily computable

Amortized Inference

General Idea of Amortization: if same inference problem needs to be solved many times, can we parameterize a neural network to solve it?

Our case: for all $x^{(i)}$ we want to solve:

$$\min_{q(z)} \text{KL}(q(z) \| p_{\theta}(z \mid x^{(i)}))$$

Amortized formulation:

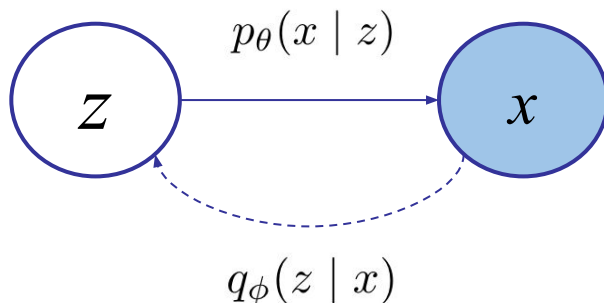
$$\min_{\phi} \sum_i \text{KL}(q_{\phi}(z \mid x^{(i)}) \| p_{\theta}(z \mid x^{(i)}))$$

Trade-off: + : faster, regularization; - : not as precise

Amortized Inference

Amortized formulation:

$$\min_{\phi} \sum_i \text{KL}(q_{\phi}(z \mid x^{(i)}) \parallel p_{\theta}(z \mid x^{(i)}))$$



E.g:

$$q_{\phi}(z \mid x) = \mathcal{N}(\mu_{\phi}(x), \sigma_{\phi}^2(x))$$

$$\text{Equivalently: } z = \mu_{\phi}(x) + \varepsilon \sigma_{\phi}(x) \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, I)$$

Importance Weighted AutoEncoder (IWAE)

Objective:
$$\sum_i \log \frac{1}{K} \sum_{k=1}^K \frac{p_Z(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)} | z_k^{(i)}) \quad \text{with } z_k^{(i)} \sim q(z_k^{(i)})$$

And:
$$\min_{\phi} \sum_i \text{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))$$

→ maximize term1 - term2

Importance Weighted AutoEncoder (IWAE)

Theorem 1. *For all k , the lower bounds satisfy*

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k.$$

Moreover, if $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then $\mathcal{L}_k$ approaches $\log p(\mathbf{x})$ as k goes to infinity.

[Burda et al., 2015]

VI as Importance Sampling [skip]

- We draw multiple z samples from $q(z|x)$ and name them z_i
- Define w_i and L_k :

$$w_i = \frac{p(z_i, x)}{q(z_i|x)}$$

$$\mathcal{L}_k = \mathbb{E} \left[\log \frac{1}{k} \sum_{i=1}^k w_i \right] \leq \log \mathbb{E} \left[\frac{1}{k} \sum_{i=1}^k w_i \right] = \log p(\mathbf{x}),$$

[Burda et al., 2015]

Importance Weighted AutoEncoder (IWAE) [skip]

$$\begin{aligned}\mathcal{L}_k &= \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_k} \left[\log \frac{1}{k} \sum_{i=1}^k \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i | \mathbf{x})} \right] \\ &= \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_k} \left[\log \mathbb{E}_{I=\{i_1, \dots, i_m\}} \left[\frac{1}{m} \sum_{j=1}^m \frac{p(\mathbf{x}, \mathbf{h}_{i_j})}{q(\mathbf{h}_{i_j} | \mathbf{x})} \right] \right] \\ &\geq \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_k} \left[\mathbb{E}_{I=\{i_1, \dots, i_m\}} \left[\log \frac{1}{m} \sum_{j=1}^m \frac{p(\mathbf{x}, \mathbf{h}_{i_j})}{q(\mathbf{h}_{i_j} | \mathbf{x})} \right] \right] \\ &= \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_m} \left[\log \frac{1}{m} \sum_{i=1}^m \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i | \mathbf{x})} \right] = \mathcal{L}_m\end{aligned}$$

[Burda et al., 2015]

Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - **Variational Lower Bound (VLB) / Evidence Lower Bound (ELBO)**
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - Mutual Information Estimation
-

VLB: Derivation 1 (Jensen)

[live derivation]

VLB: Derivation 1 (Jensen)

[live derivation]

Variational Lower Bound:

Derivation:

$$\max_{\theta} \sum_i \log p_{\theta}(x^{(i)})$$

$$= \max_{\theta} \sum_i \log \left(\sum_z p(z) \cdot p_{\theta}(x^{(i)}|z) \right)$$

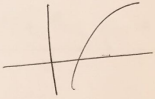
$$= \max_{\theta} \sum_i \log \left(\sum_z \frac{q(z)}{q(z)} p(z) p_{\theta}(x^{(i)}|z) \right)$$

$$\geq \max_{\theta} \sum_i \sum_z \mathbb{E}_{q(z)} \left[\log p(z) - \log q(z|x^{(i)}) + \log p_{\theta}(x^{(i)}|z) \right]$$

we set distribution we like to sample from

$$= \max_{\theta} \max_{\phi} \sum_i \mathbb{E}_{q_{\phi}(z)} \left[\log p(z) - \log q_{\phi}(z|x^{(i)}) + \log p_{\theta}(x^{(i)}|z) \right]$$

where $q(z|x^{(i)}) \propto p(z) p_{\theta}(x^{(i)}|z)$
 $\propto p(z|x^{(i)})$

$$\max_{\phi} () = \min_{\phi} \text{KL} ()$$


VLB: Derivation 2 (KL)

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

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

Same as with Jensen's, but now we know the gap = KL

Variational Lower Bound (VLB)

- We now have an objective amenable to stochastic optimization

$$D_{\text{KL}} [q_x(z) \parallel p(z|x)] = \mathbb{E}_{z \sim q_x(z)} [\log q_x(z) - \log p(z) - \log p(x|z)] + \log p(x)$$

- Turns out we can get more out of this exercise

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

- 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 $\mathbf{x} \sim p_{\text{data}}$, we can train the generative model by maximizing the VLB under data distribution

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

Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - **Optimization**
 - **Likelihood Ratio Gradients vs. Reparameterization Trick Gradients**
 - **Optimizing the VLB/ELBO**
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - Mutual Information Estimation

Likelihood Ratio Gradient

[live derivation -- for general case; for Gaussian on next slide]

Likelihood Ratio Gradient

$$\begin{aligned}
 & \max_{\phi} E_{z \sim q_{\phi}(z)} [f(z)] \\
 & \max_{\phi} \sum_z q_{\phi}(z) f(z) \\
 & \nabla_{\phi} (\quad) = \sum_z \nabla_{\phi} q_{\phi}(z) \cdot f(z) \\
 & = \sum_z \frac{q_{\phi}(z)}{q_{\phi}(z)} \nabla_{\phi} q_{\phi}(z) f(z) \\
 & = \mathbb{E}_{z \sim q_{\phi}(z)} \frac{\nabla_{\phi} q_{\phi}(z)}{q_{\phi}(z)} \cdot f(z) \\
 & = \mathbb{E}_{z \sim q_{\phi}(z)} [\nabla_{\phi} \log q_{\phi}(z) \cdot f(z)]
 \end{aligned}$$

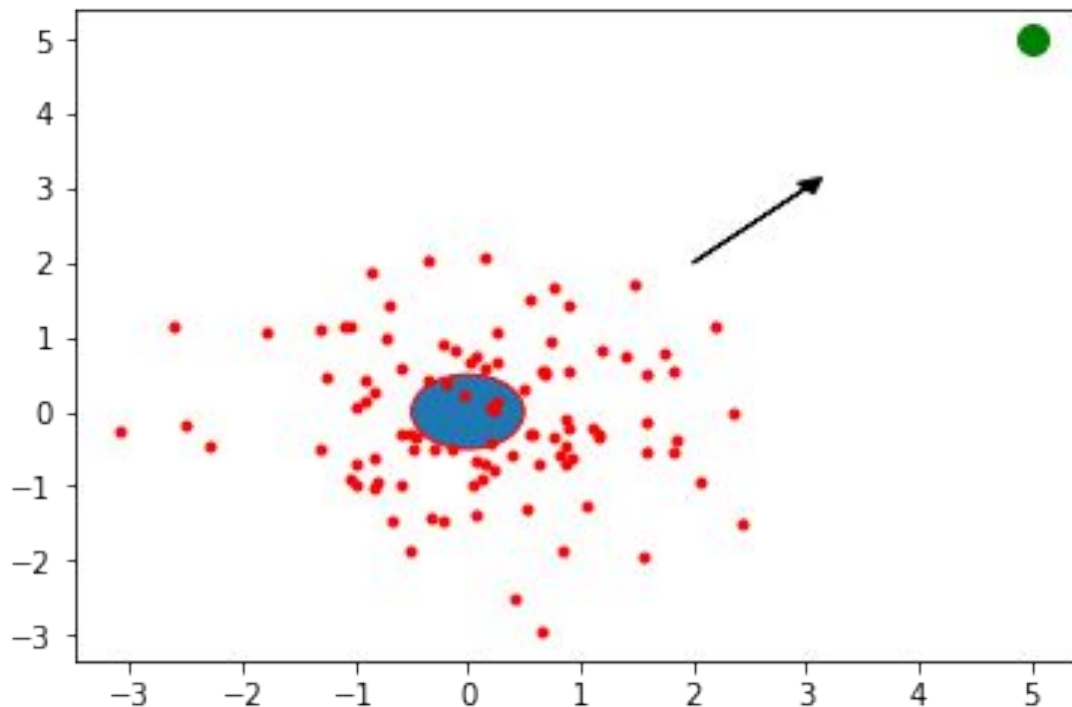
$\mathbb{E}_{z \sim q_{\phi}(z)} \left[\frac{\nabla_{\phi} q_{\phi}(z)}{q_{\phi}(z)} \cdot f(z) \right] = 0$

$\approx \frac{1}{K} \sum_{i=1}^K \nabla_{\phi} \log q_{\phi}(z^{(i)}) \cdot f(z^{(i)})$

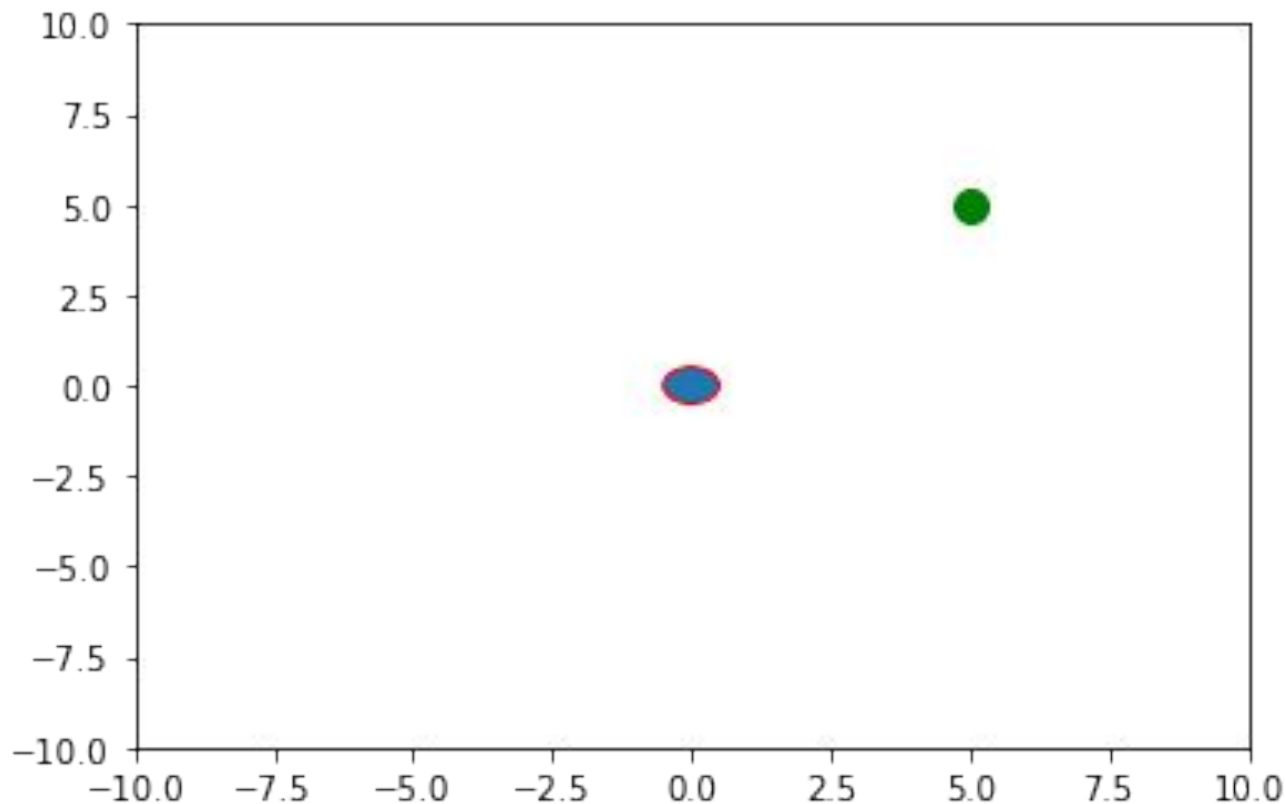
Likelihood Ratio Gradient - Toy Problem

Learn $\mu \in \mathbb{R}^2$ to
minimize the objective
below to reach the green
point

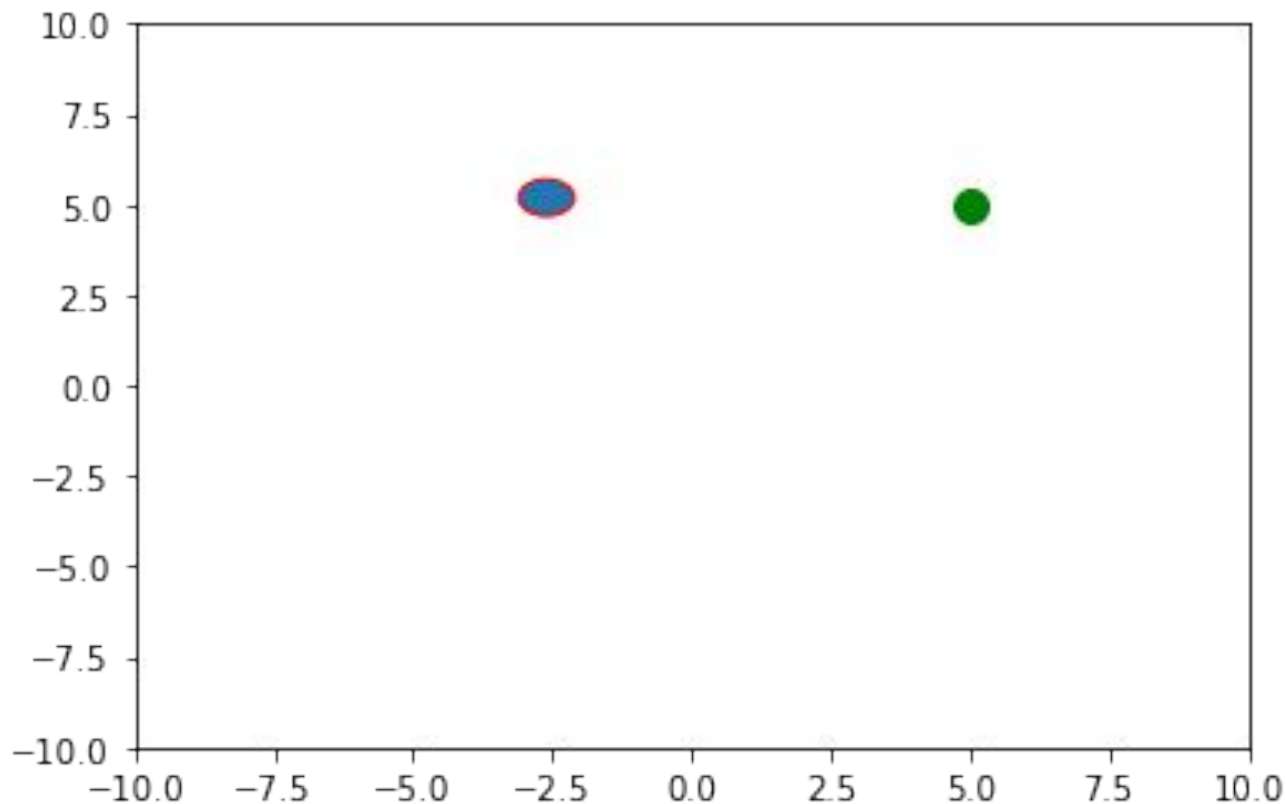
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



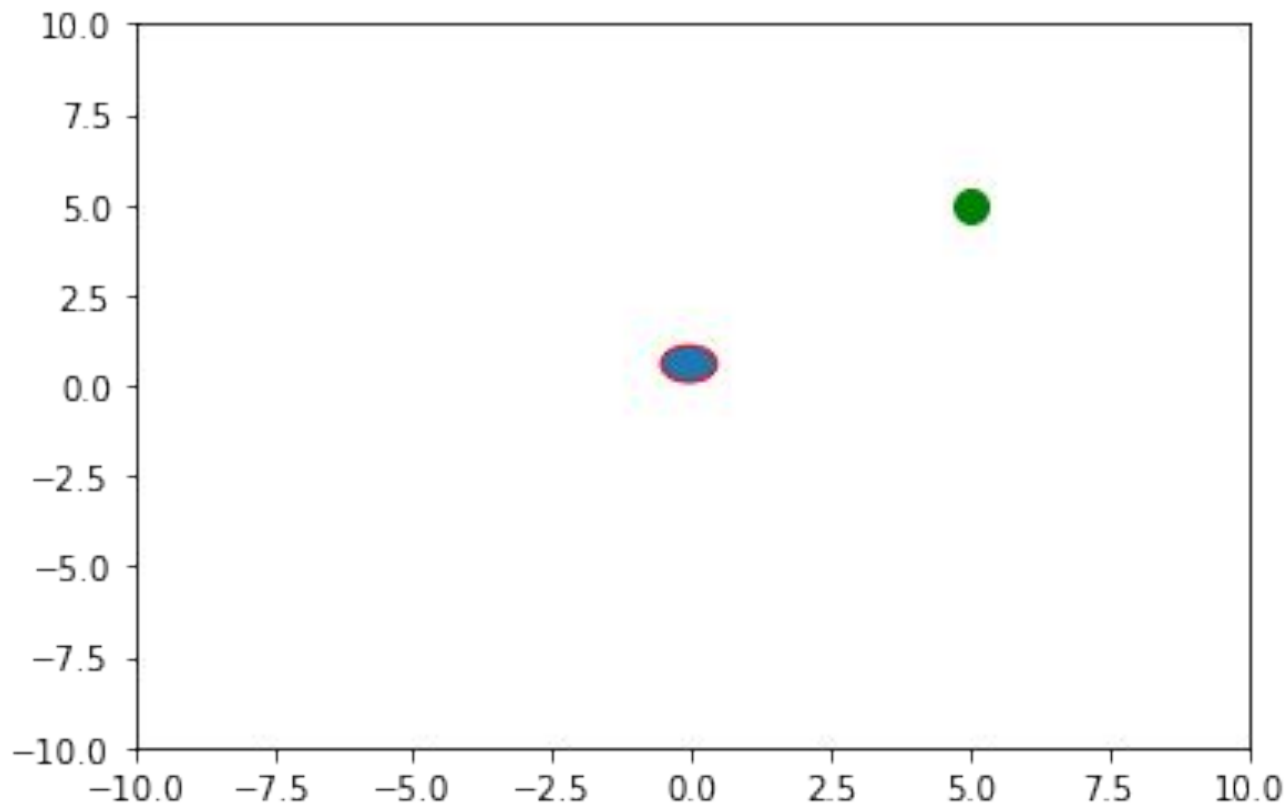
Likelihood Ratio Gradient - Toy Problem



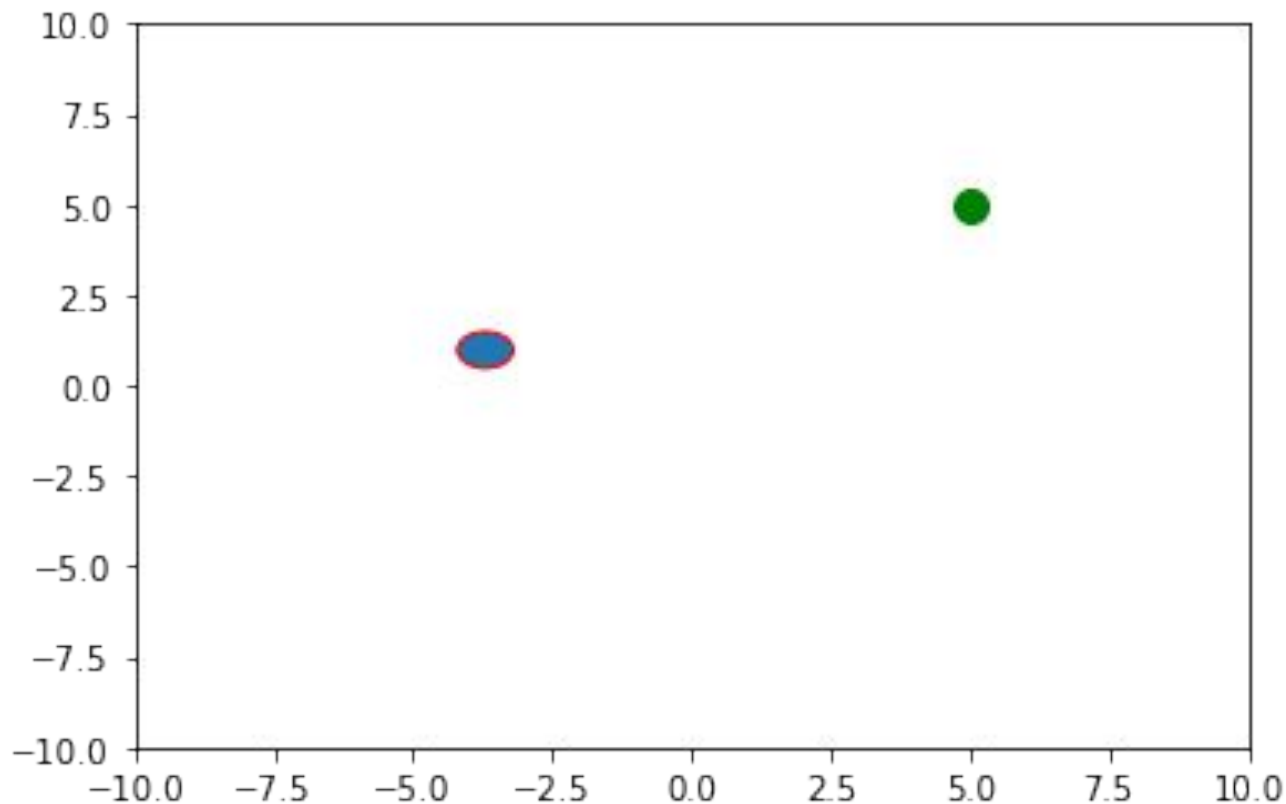
Likelihood Ratio Gradient - Toy Problem



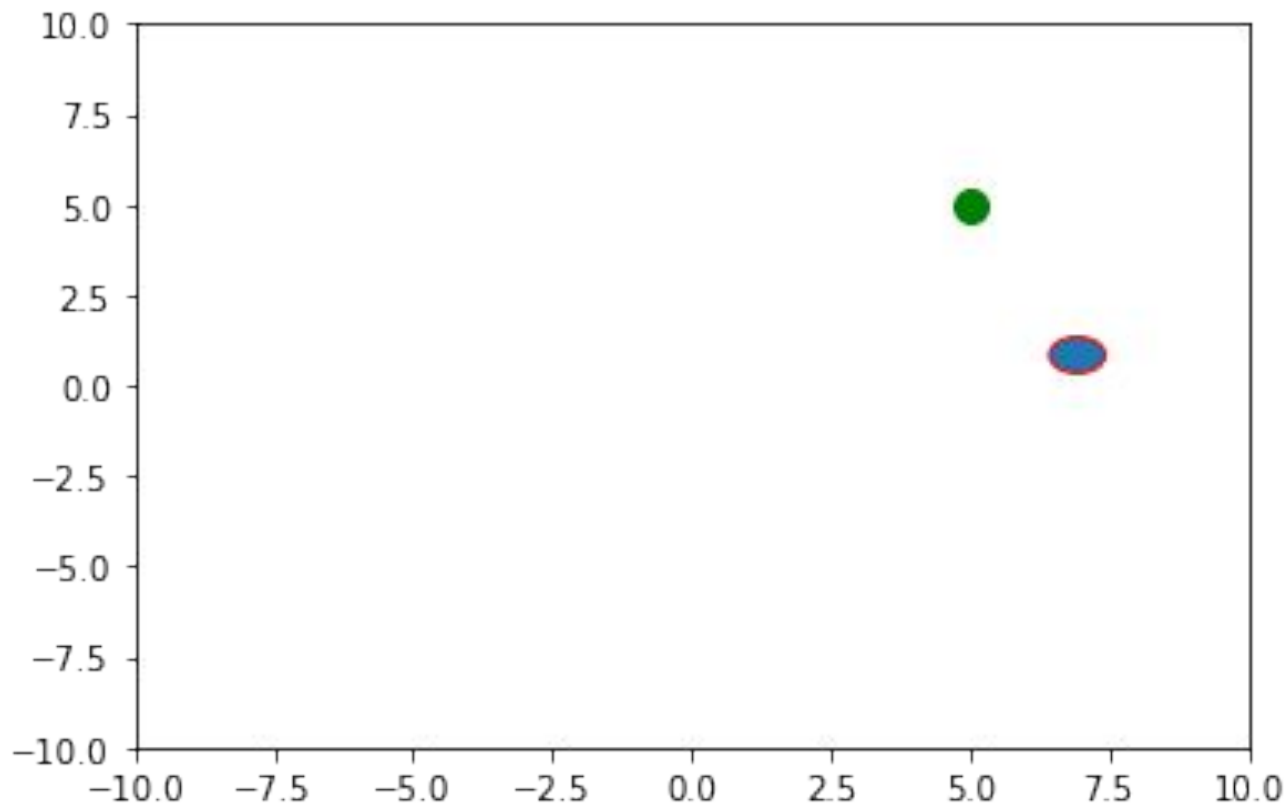
Likelihood Ratio Gradient - Toy Problem



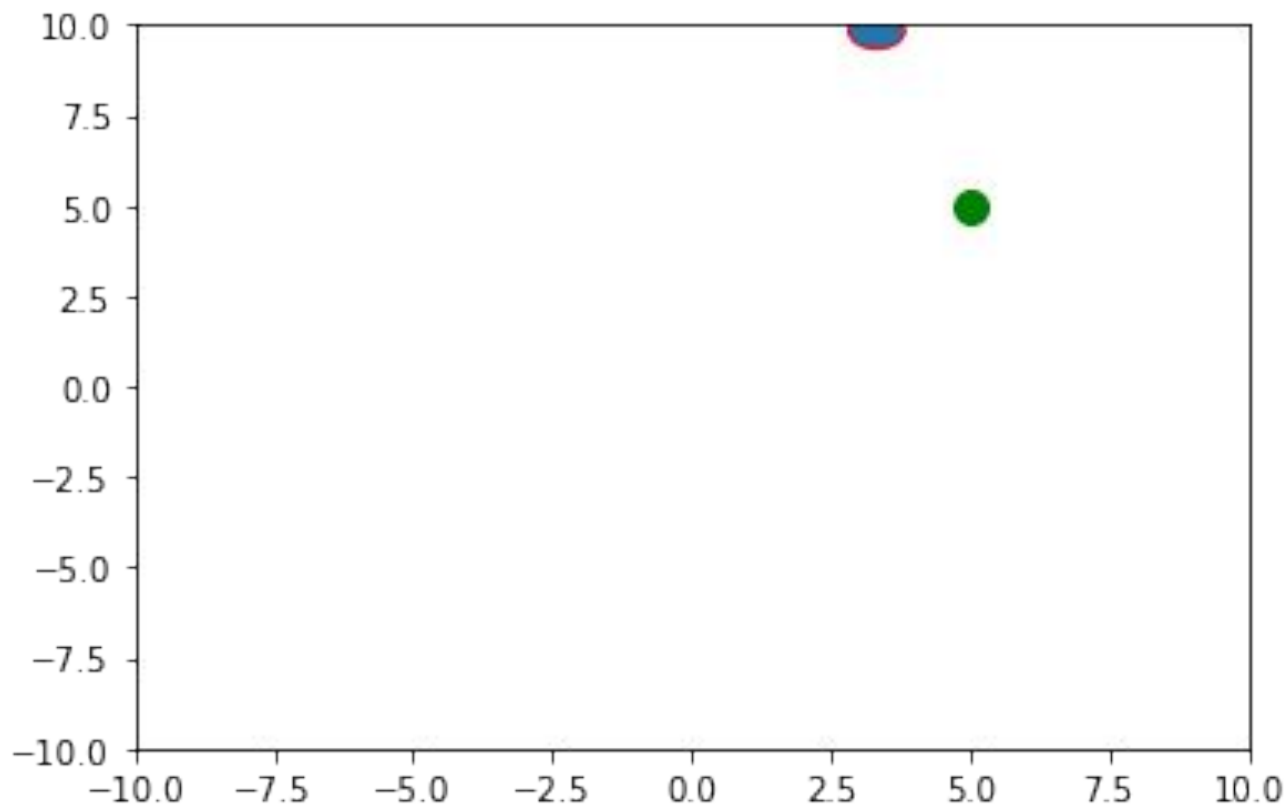
Likelihood Ratio Gradient - Toy Problem



Likelihood Ratio Gradient - Toy Problem



Likelihood Ratio Gradient - Toy Problem



Pathwise Derivative aka Reparameterization Trick

[live derivation -- for general case; for Gaussian of next slide]

Pathwise Derivative aka Reparameterization Trick

$$\begin{aligned} & \mathbb{E}_{z \sim q_{\phi}(z)} [f(z)] \\ &= \mathbb{E}_{\epsilon \sim N(0,1)} [f(\mu + \epsilon \sigma)] \\ & \quad \overline{\text{w.r.t. } \phi} \end{aligned}$$

$$\begin{aligned} & \approx \frac{1}{K} \sum_{i=1}^K f(\mu + \epsilon^{(i)} \sigma) \\ & \quad \nabla_{\mu, \sigma} \left(\right) \end{aligned}$$

$$q_{\phi}(z) = N(\mu, \sigma^2)$$

$$z = \underline{\mu + \epsilon \cdot \sigma} \quad \epsilon \sim N(0,1)$$

$$\begin{aligned} & \nabla_{\mu} f(\mu + \epsilon^{(i)} \cdot \sigma) \\ & \quad \left\| \mu + \epsilon^{(i)} \cdot \sigma - \left(\frac{5}{5} \right) \right\|^2 \end{aligned}$$

Pathwise Derivative (PD)

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

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.

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

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

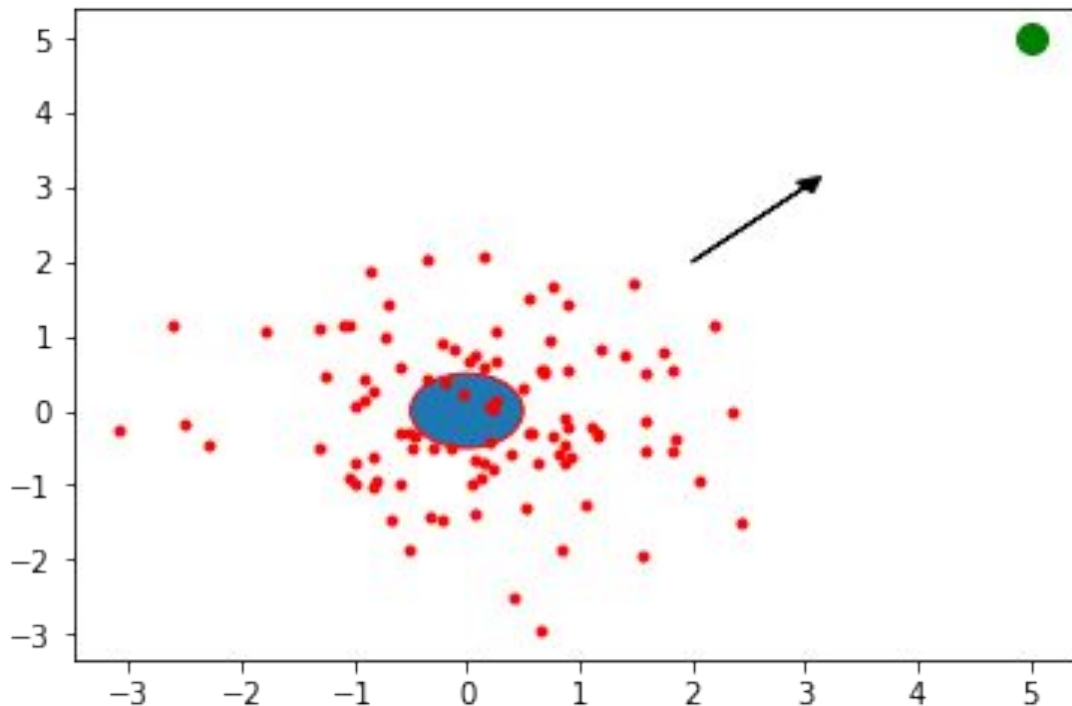
When f is differentiable,

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

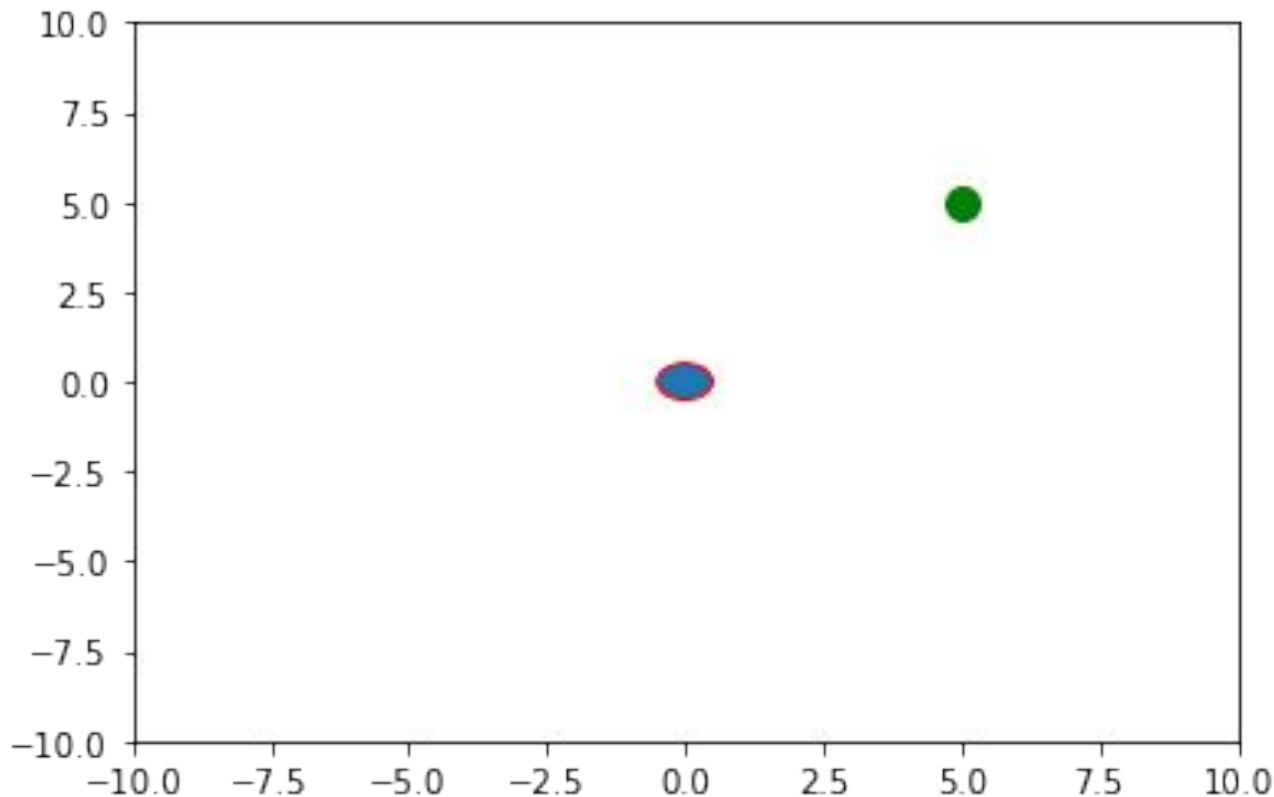
Pathwise Derivative - Toy Problem

Learn $\mu \in \mathbb{R}^2$ to
minimize the objective
below to reach the green
point

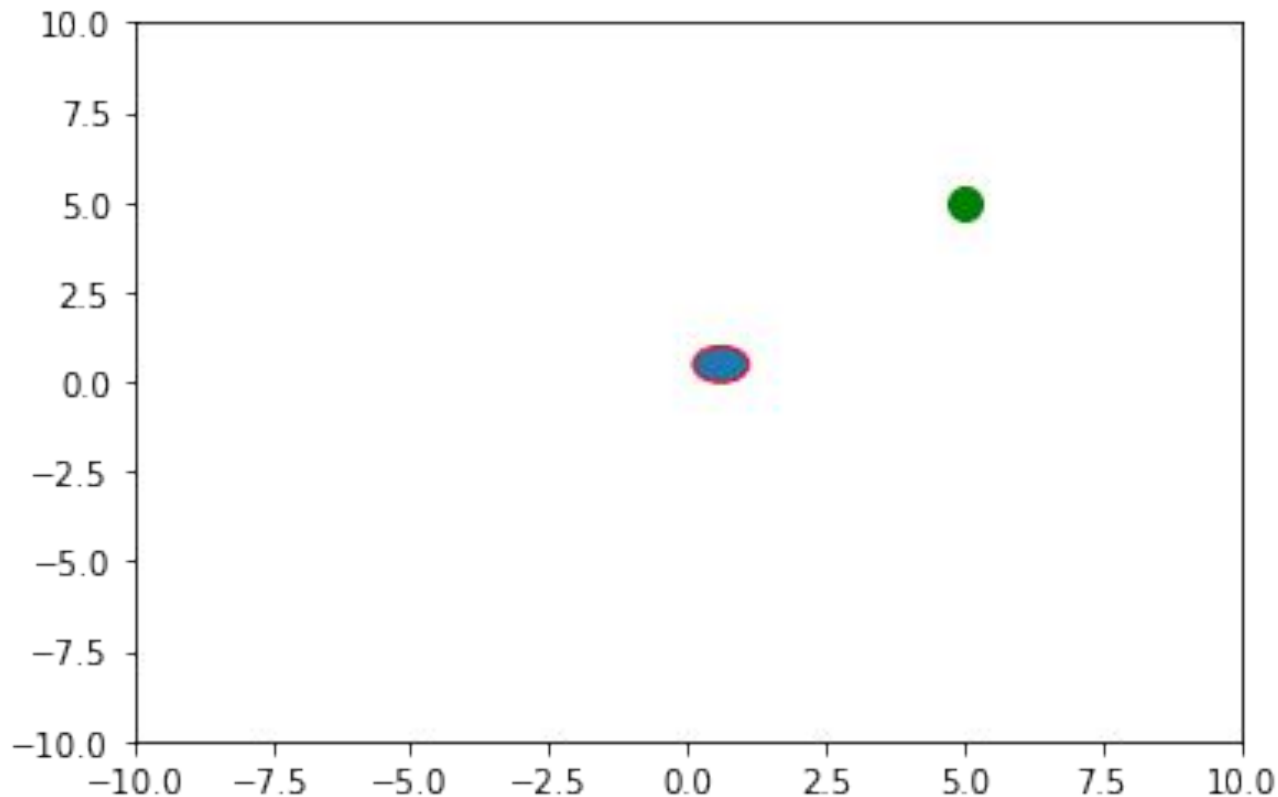
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



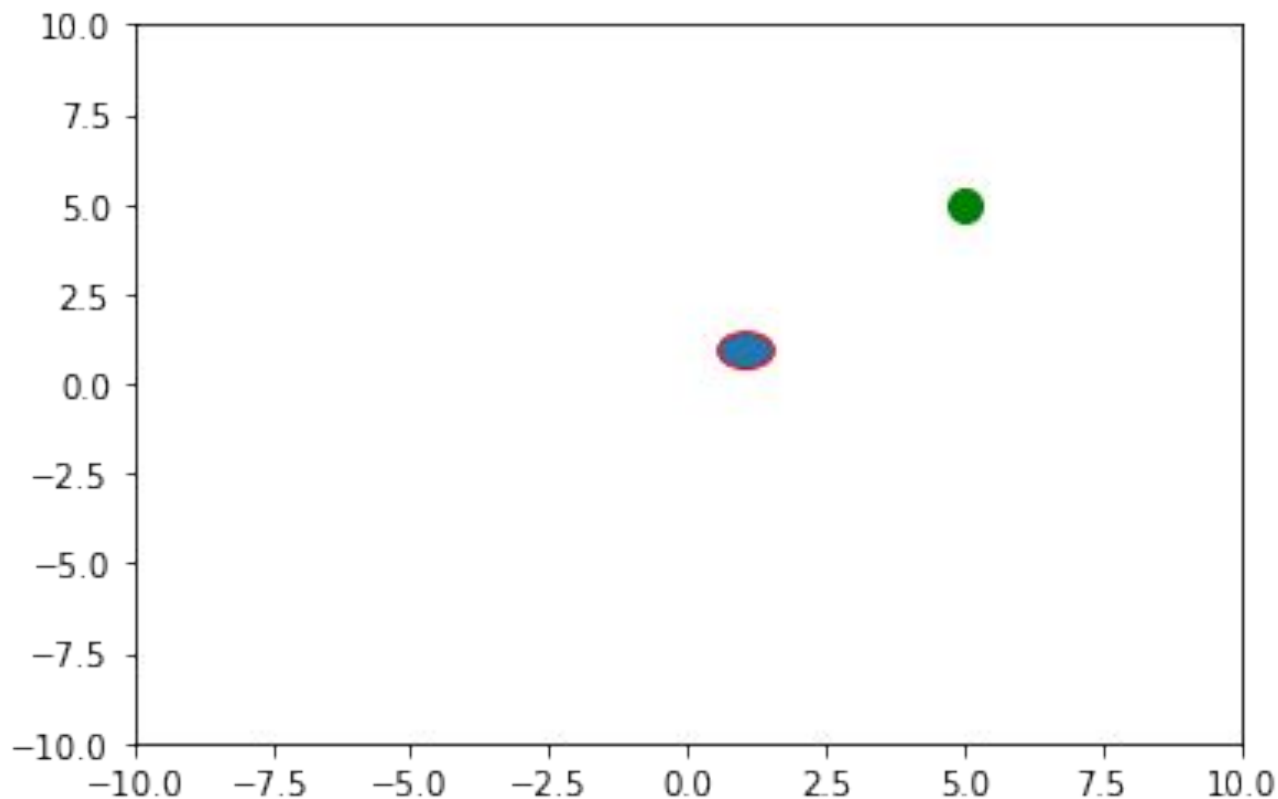
Pathwise Derivative - Toy Problem



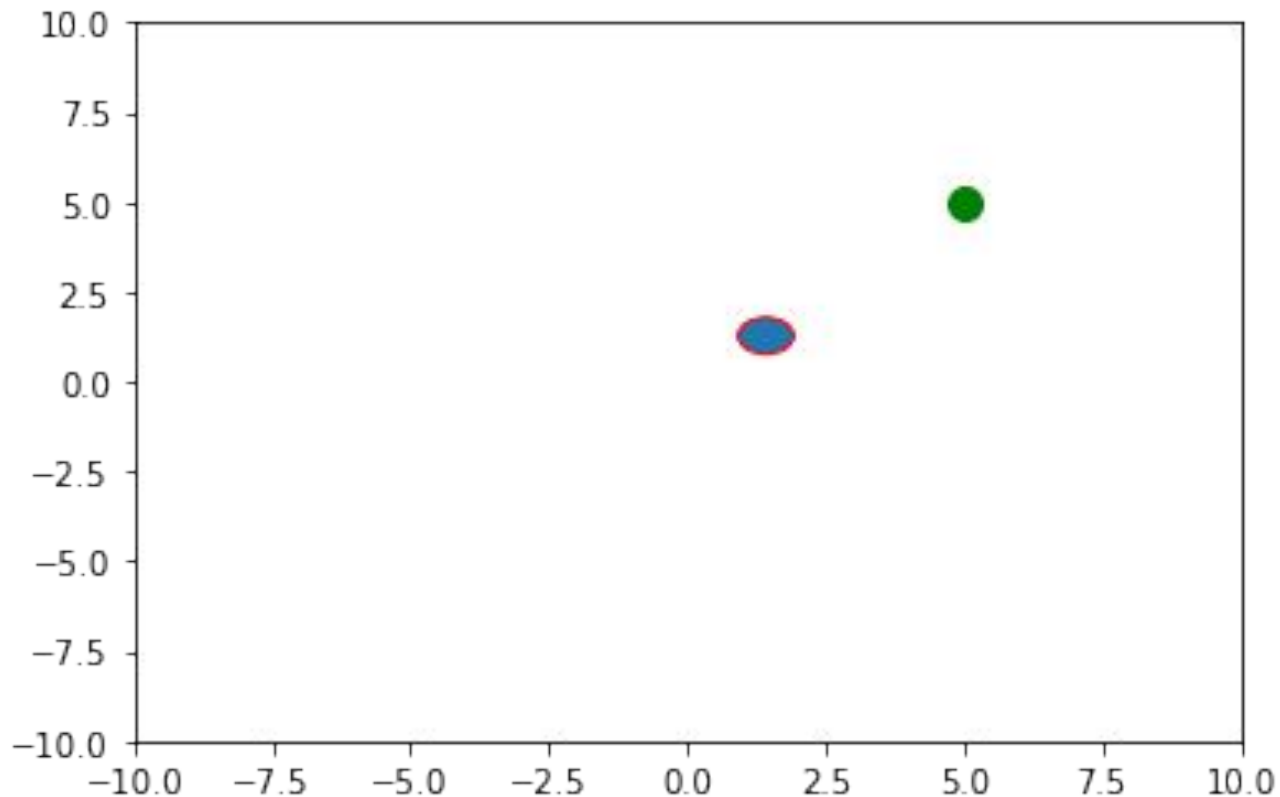
Pathwise Derivative - Toy Problem



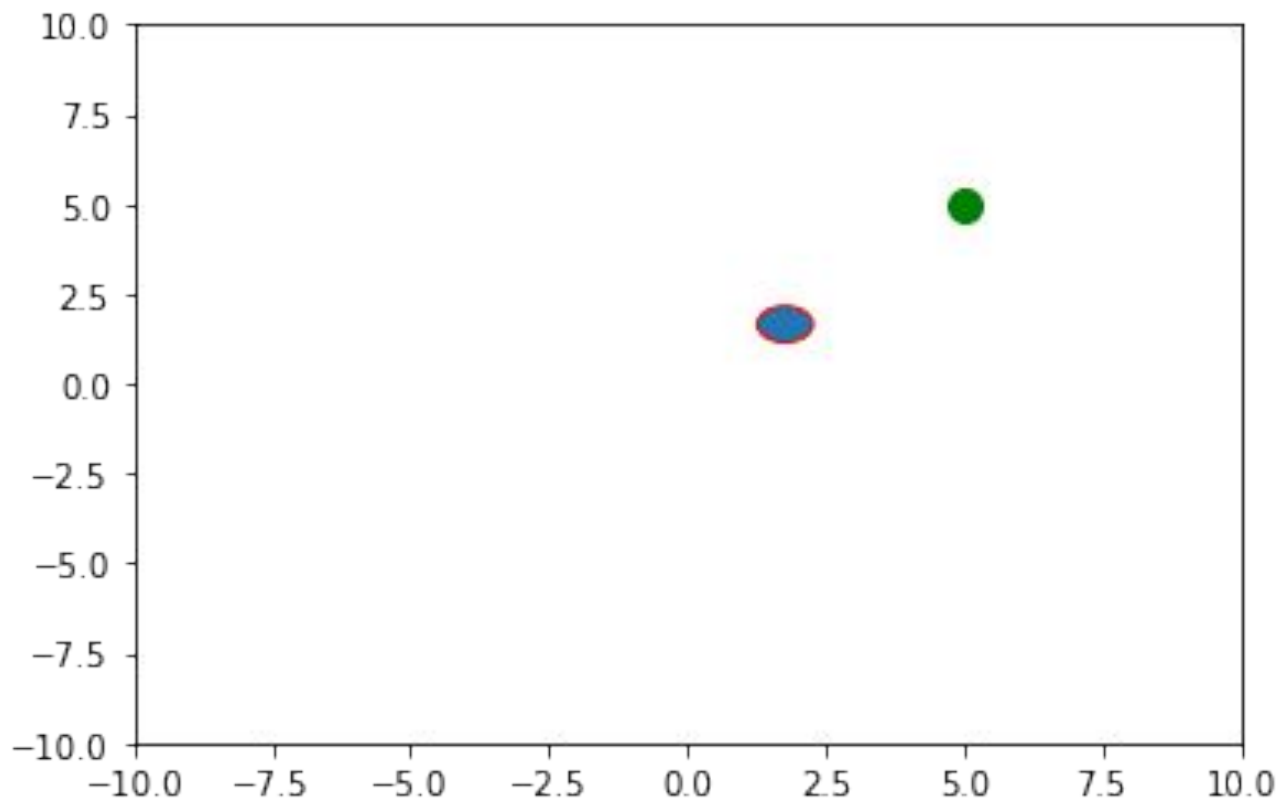
Pathwise Derivative - Toy Problem



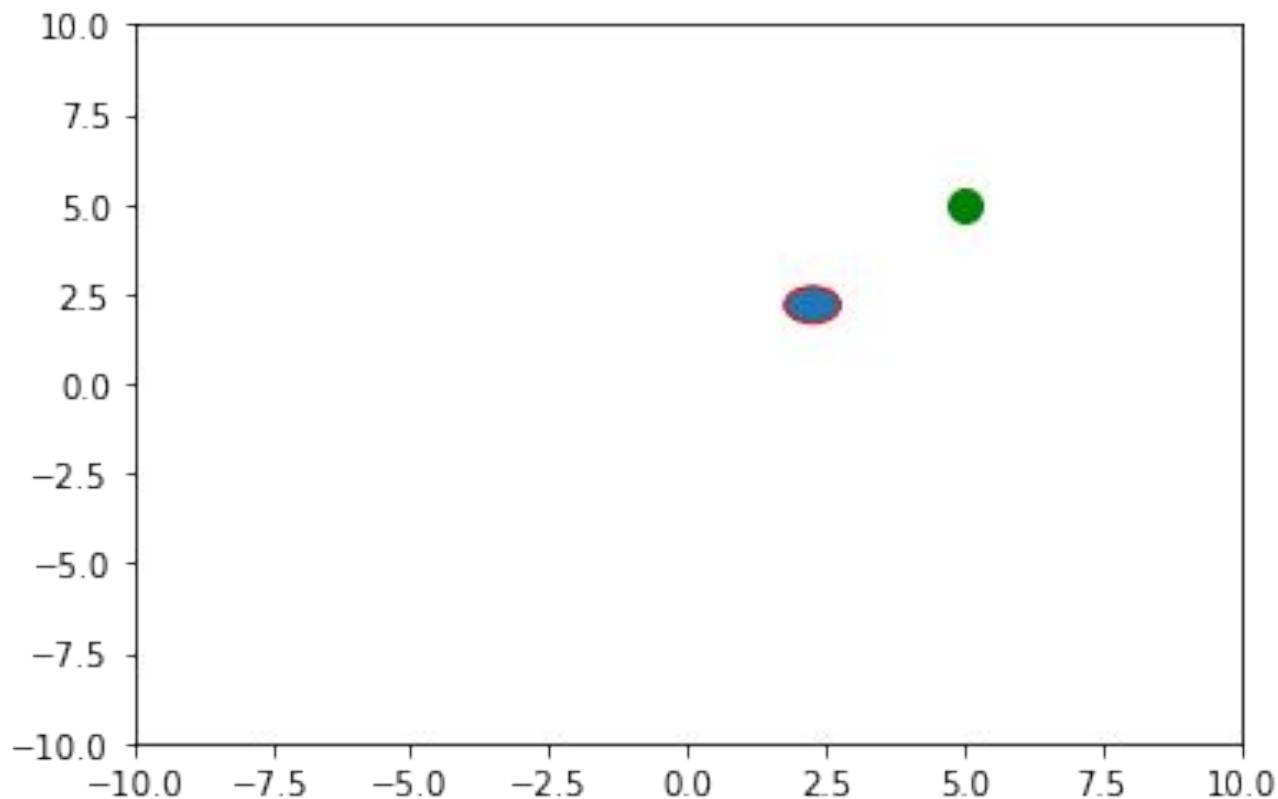
Pathwise Derivative - Toy Problem



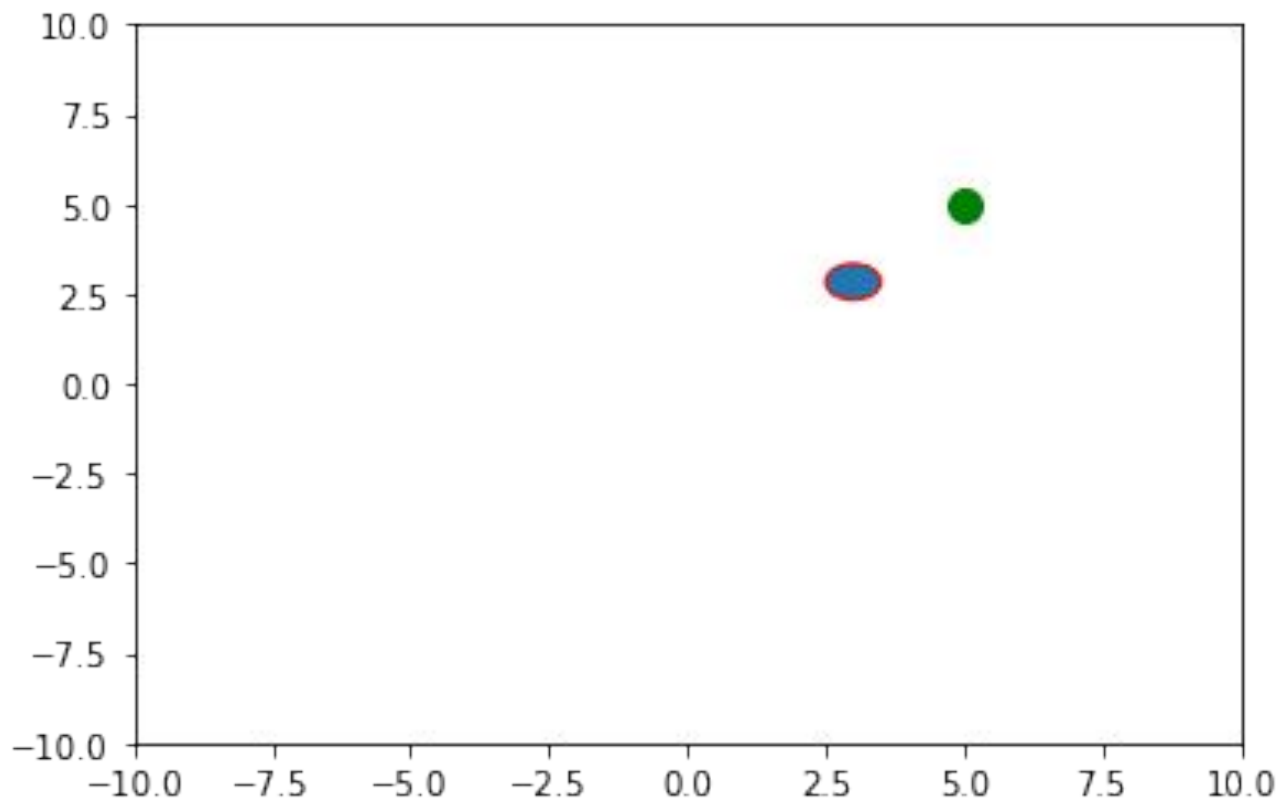
Pathwise Derivative - Toy Problem



Pathwise Derivative - Toy Problem



Pathwise Derivative - Toy Problem



VAE and Likelihood Ratio Gradient

$$\begin{aligned}
 \max_{\theta, \phi} \mathbb{E}_{z \sim q_{\phi}(z|x^{(i)})} & \left[\log p_z(z) + \log p_{\theta}(x^{(i)}|z) - \log q_{\phi}^{\theta}(z|x^{(i)}) \right] \quad \textcircled{2} \rightarrow \textcircled{3} \\
 \nabla_{\theta} &= \nabla_{\theta} \mathbb{E}_{z \sim q_{\phi}} [\log p_{\theta}(x^{(i)}|z)] = \nabla_{\theta} \sum_{k=1}^K \log p_{\theta}(x^{(k)}|z^{(k)}) \\
 & \quad z^{(k)} \sim q_{\phi}(z|x^{(k)}) \\
 \nabla_{\phi} &= \nabla_{\phi} q_{\phi}(z^{(i)}) [\log p_z(z^{(i)}) + \log p_{\theta}(x^{(i)}|z^{(i)}) - \log q_{\phi}(z^{(i)}|x^{(i)})] \\
 & \quad + \mathbb{E}_{z \sim q_{\phi}(z|x^{(i)})} [-\nabla_{\phi} \log q_{\phi}(z|x^{(i)})] \\
 & \quad \frac{\mathbb{E}_{z \sim q_{\phi}} \nabla_{\phi} q_{\phi}(z|x^{(i)})}{\sum_z q_{\phi}(z|x^{(i)})} = \frac{\nabla_{\phi} q_{\phi}(z|x^{(i)})}{q_{\phi}(z|x^{(i)})} \quad \text{if } \nabla_{\phi} \left(\sum_z q_{\phi}(z|x^{(i)}) \right) = 0
 \end{aligned}$$

Likelihood Ratio Estimator

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

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

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

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

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

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

VAE and Pathwise Derivative [skip]

[empty slide for live derivation what's on next slide]

Stochastic optimization of VLB [skip]

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

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

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

In the Bernoulli setting, this becomes:

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

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

Directly optimizing VLB [skip]

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

Recall, we want

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

Optimization with respect to ϕ is of the form

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

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

PD applied to VLB

Variational AutoEncoder

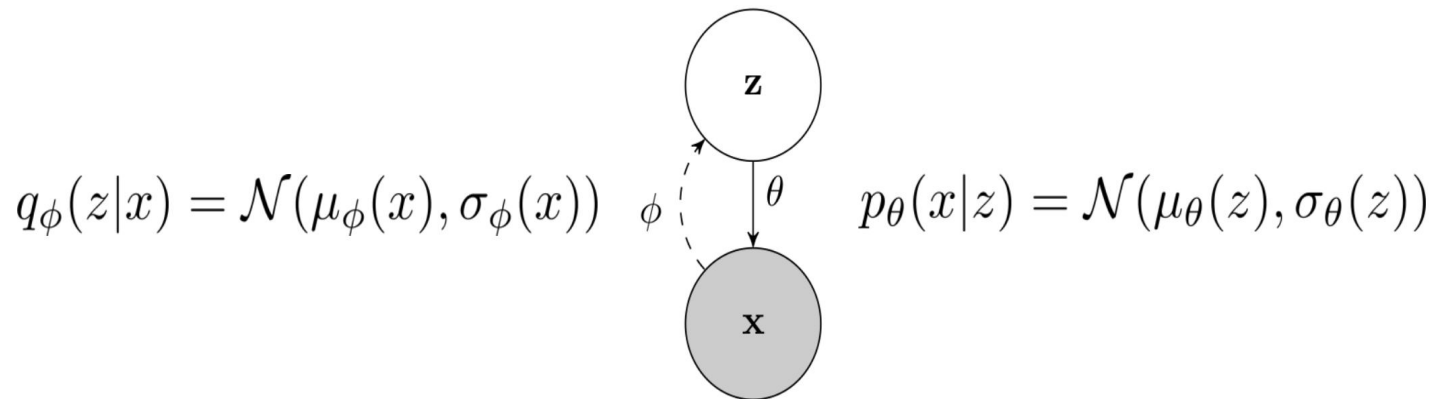
$q_\phi(z|x)$ is modeled as a Gaussian with parameters μ and σ a DNN encoder (parameters ϕ) of x . The DNN decoder $p_\theta(x|z)$ is differentiable.

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

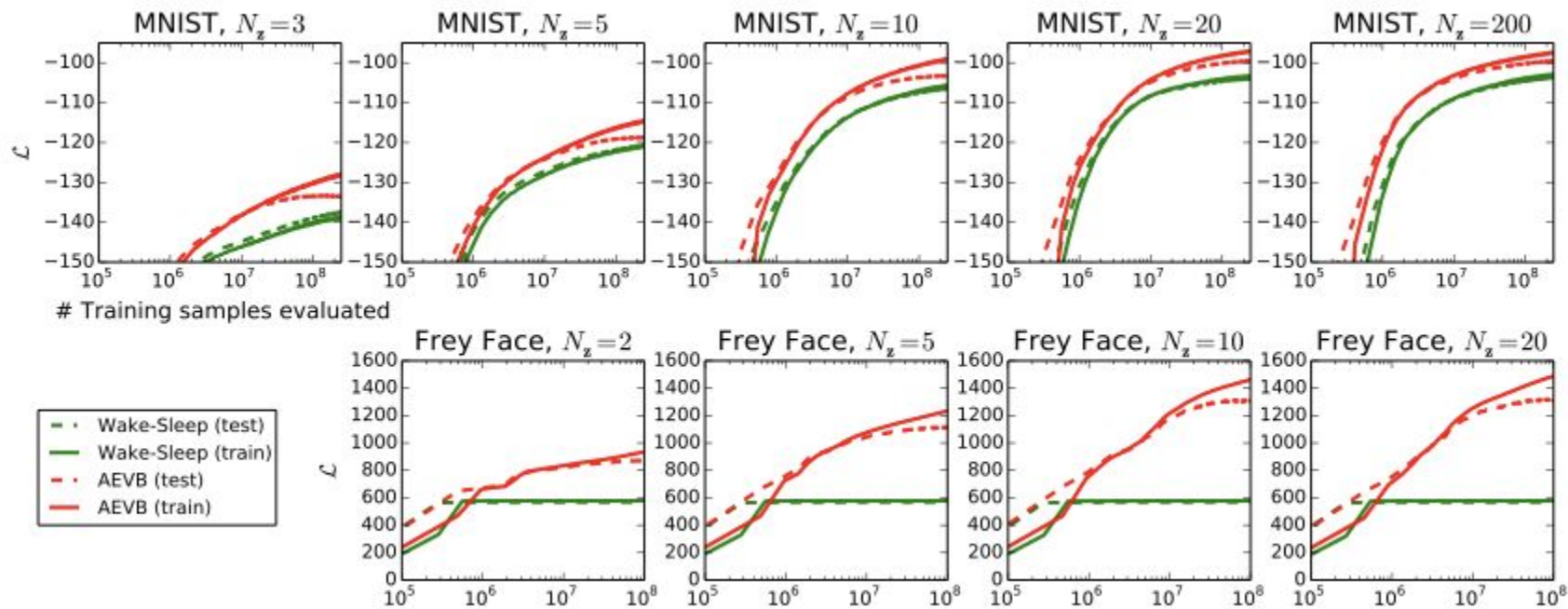
$$\begin{aligned}\text{VLB} &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} [\log p_\theta(x|z) - \log q_\phi(z|x) + \log p(z)] \\ &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} [\log p_\theta(x|z)] - KL(q_\phi(z|x) || p(z))\end{aligned}$$

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

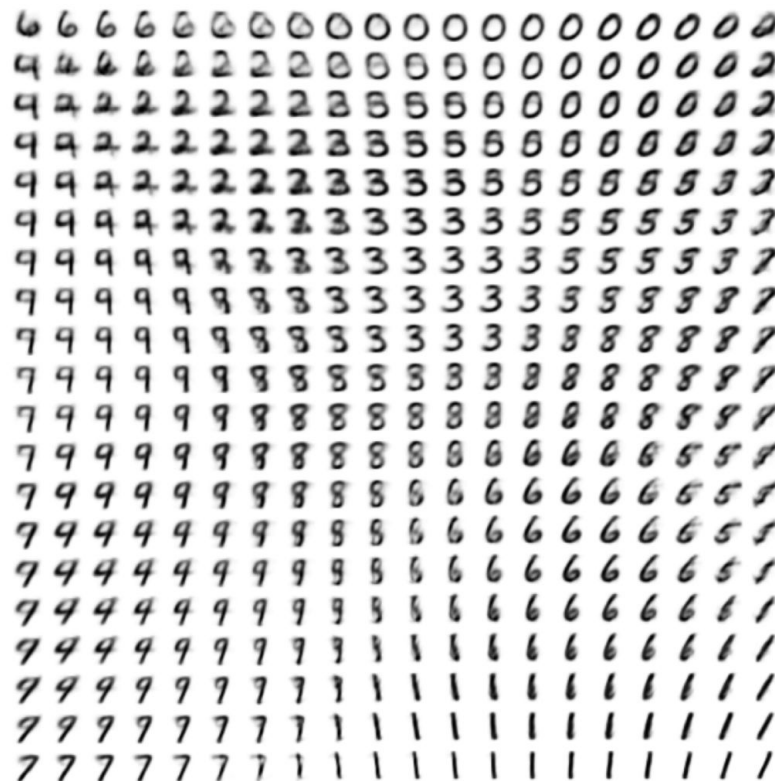
VAE



VAE

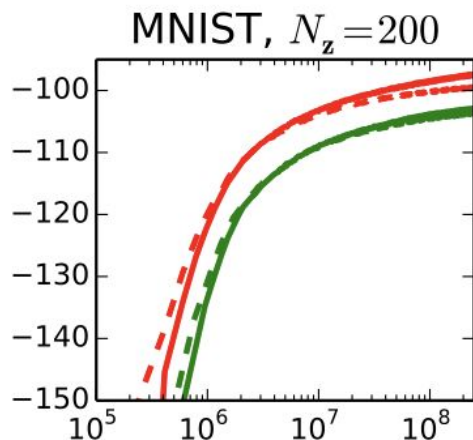


VAE



Compared to AR [skip]

- 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

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”?

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

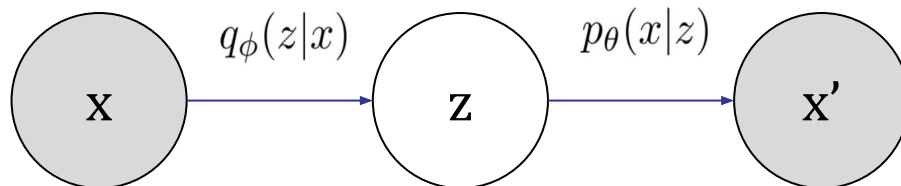
$L(\theta, \phi)$ - VAE objective

Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower Bound (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- **Variations:**
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- **Related ideas:**
 - Mutual Information Estimation

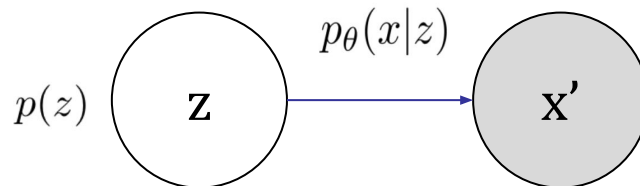
Parts of the VAE

Training:



$$\underbrace{\left(E_{z \sim q_x(z)} \log p_{\theta}(x|z) \right)}_{\text{Reconstruction loss}}$$
$$\underbrace{KL(q_{\phi}(z|x) || p(z))}_{\text{Regularization}}$$

Sampling:



VAE = Generative Model



<https://github.com/houxianxu/DFC-VAE>

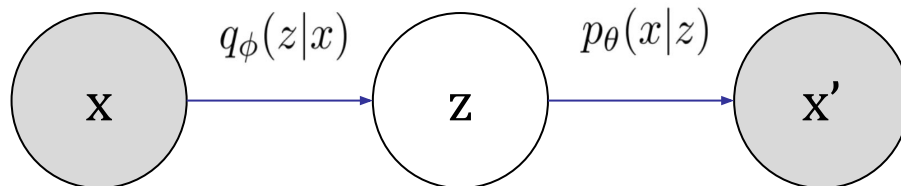
Latent Interpolation



<https://github.com/houxianxu/DFC-VAE>

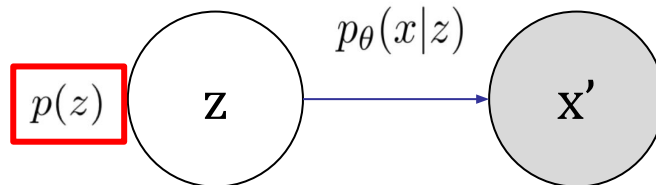
Varieties on $p(z)$

Training:



$$\underbrace{(E_{z \sim q_x(z)} \log p_\theta(x|z))}_{\text{Reconstruction loss}}$$
$$\underbrace{KL(q_\phi(z|x) || p(z))}_{\text{Regularization}}$$

Sampling:



$p(z)$ is a prior design choice.

Varieties on $p(z)$

- $p(z)$ = Unit Normal
 - Nice analytic form. Can compute $KL(q(z|x) || p(z))$ easily.
 - Can linearly interpolate, since approx. closed under convex combos.
 - All dimensions are independent -- a limiting assumption!

Varieties on $p(z)$

- $p(z) = \text{Unit Normal}$
 - Nice analytic form. Can compute $\text{KL}(q(z|x) \parallel p(z))$ easily.
 - Can linearly interpolate, since approx. closed under convex combos.
 - All dimensions are independent -- a limiting assumption!
- $p(z) = p(z_i \mid z_{\{i-1\}})$
 - $q(z_i \mid z_{\{i-1\}}, x)$
 - i.e., sample z autoregressively. Learn both $p(z)$ and $q(z|x)$.
 - How to handle KL between two learned distributions?

NVAE: A Deep Hierarchical Variational Autoencoder (Vahdat and Kautz, 2020)

Residual Normal

- **Insight:** Define $q(z|x)$ as a residual over $p(z)$.

$$p(z_l^i | \mathbf{z}_{<l}) := \mathcal{N}(\mu_i(\mathbf{z}_{<l}), \sigma_i(\mathbf{z}_{<l}))$$
$$q(z_l^i | \mathbf{z}_{<l}, \mathbf{x}) := \mathcal{N}(\mu_i(\mathbf{z}_{<l}) + \Delta\mu_i(\mathbf{z}_{<l}, \mathbf{x}), \sigma_i(\mathbf{z}_{<l}) \cdot \Delta\sigma_i(\mathbf{z}_{<l}, \mathbf{x}))$$

- We get a nice analytic KL again!

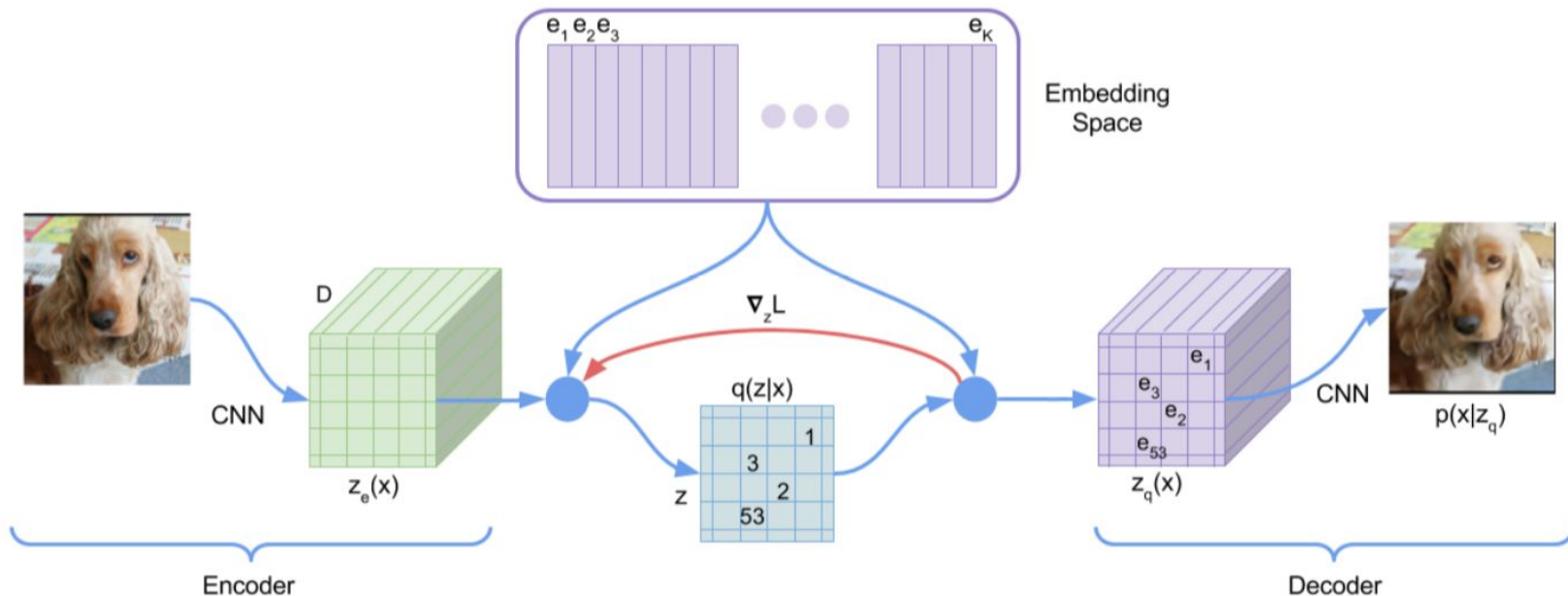
$$\text{KL}(q(z^i | \mathbf{x}) || p(z^i)) = \frac{1}{2} \left(\frac{\Delta\mu_i^2}{\sigma_i^2} + \Delta\sigma_i^2 - \log \Delta\sigma_i^2 - 1 \right)$$

NVAE: A Deep Hierarchical Variational Autoencoder (Vahdat and Kautz, 2020)

Varieties on $p(z)$

- What if the space of z is discrete?
 - The channel capacity of z is bounded.
 - It's simple to define expressive categoricals $p(z)$

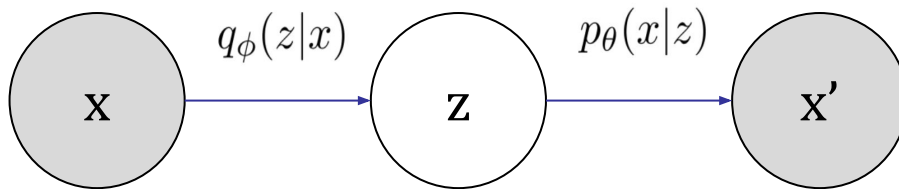
VQ-VAE



Neural Discrete Representation Learning (van den Oord et al, 2018)

Parts of the VAE

Training:

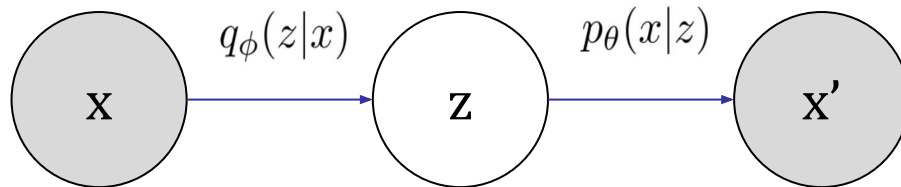


$$\underbrace{\left(E_{z \sim q_x(z)} \log p_{\theta}(x|z) \right)}_{\text{Reconstruction loss}}$$
$$\underbrace{KL(q_{\phi}(z|x) || p(z))}_{\text{Regularization}}$$

Parts of the VAE

Training:

*Deterministic distribution
of nearest token.*



Z = Set of tokens.

$$\underbrace{\left(E_{z \sim q_x(z)} \log p_{\theta}(x|z) \right)}_{\text{Reconstruction loss}}$$
$$\underbrace{KL(q_{\phi}(z|x) || p(z))}_{\text{Regularization}}$$
$$= \log K$$

How to optimize?

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2$$

Reconstruction Loss.

Embedding Loss.

Commitment Loss.

- Use the *straight-through* optimizer.
 - I.e., pass the gradient of $p(x|z)$ through into $z_q(x)$.
 - Embedding Loss: move embedding vectors closer to $z_q(x)$.
 - Commitment Loss: enforce $z_q(x)$ to not fluctuate.

Learning $p(z)$

- During training, $p(z)$ is assumed to be uniform.
- After training, we want $p(z)$ to match $q(z|x)$.
 - We can use our favorite generative models here.
 - In the paper: PixelCNN
 - These days, it's a transformer -- model $p(z)$ autoregressively.
 - Can also be a diffusion model! More on this in later lectures.

VQ-VAE -- Experiments

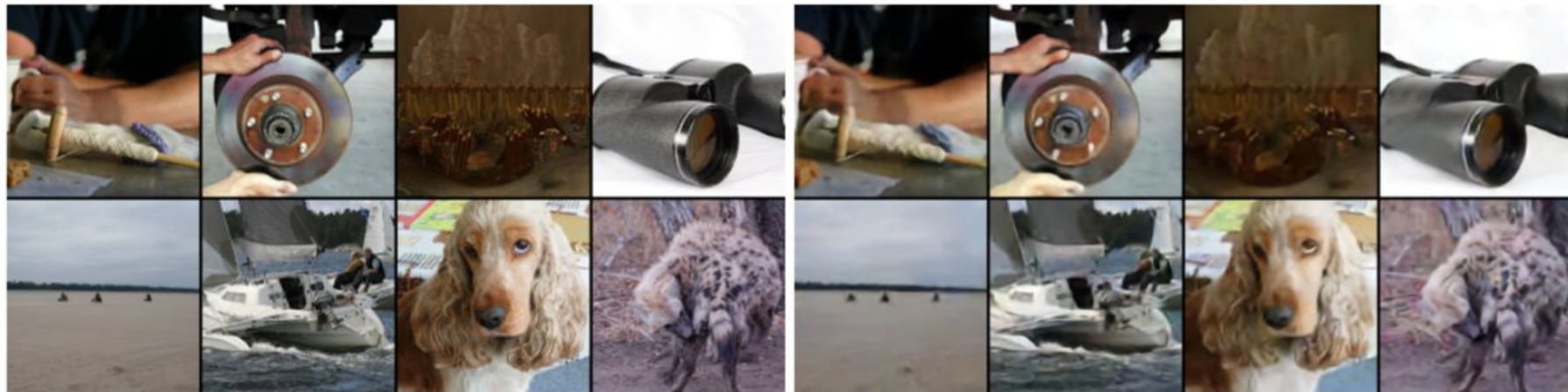


Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with $K=512$.

VQ-VAE -- Experiments

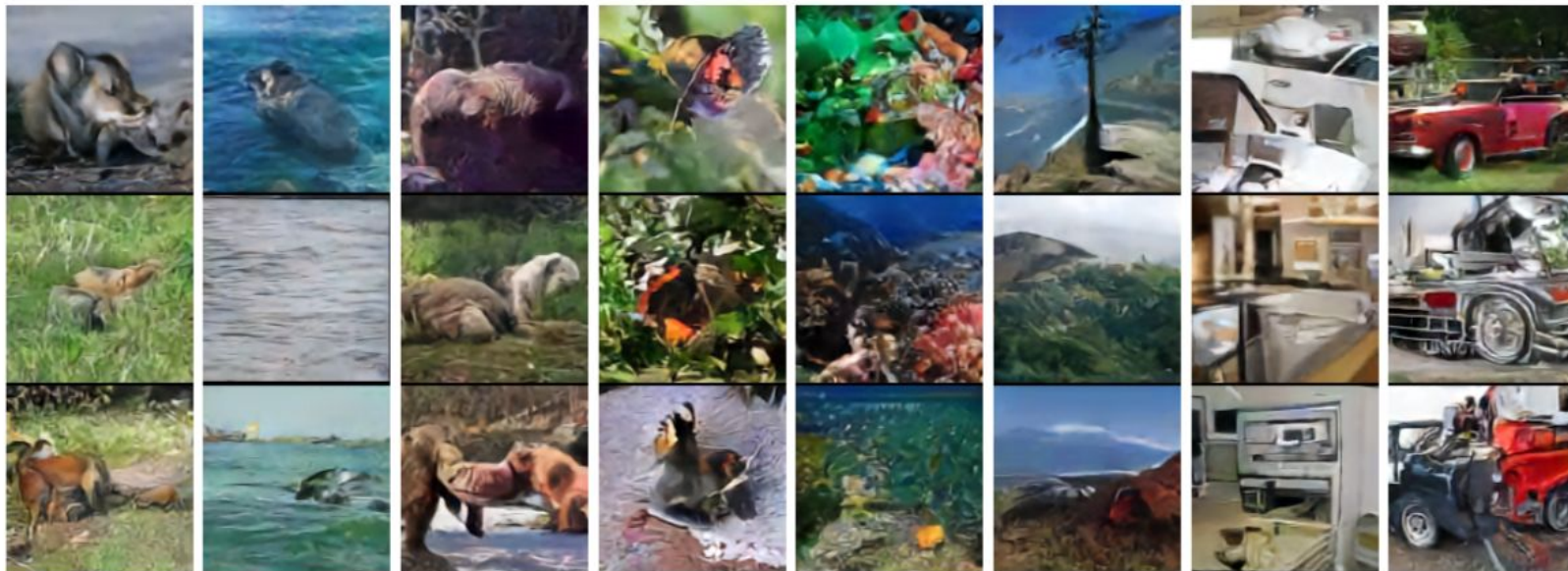
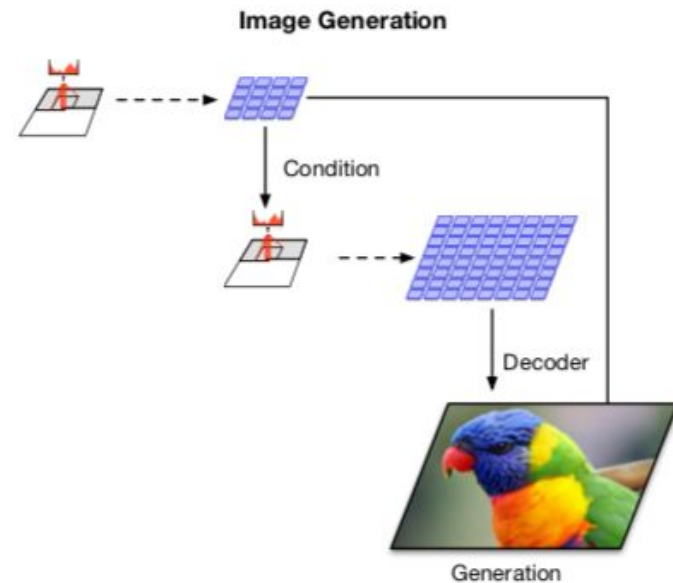
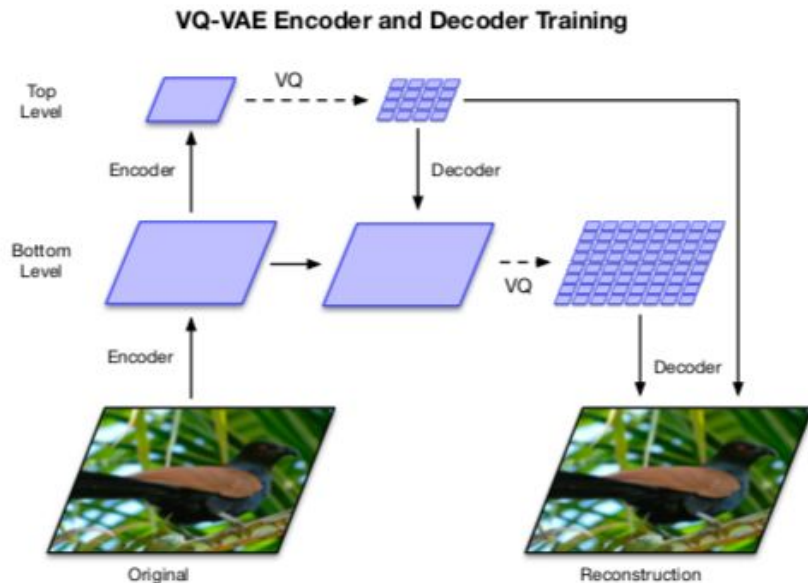


Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

VQ-VAE 2.0



VQ-VAE 2.0 -- Experiments



Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

VQ-VAE 2.0 -- Coverage



VQ-VAE 2.0



BigGAN Deep

VQ-VAE 2.0 -- Coverage



VQ-VAE 2.0

BigGAN Deep

VQ-VAE 2.0 -- Coverage



VQ-VAE 2.0

BigGAN Deep

VAEs as a building block.

- VAE high-level outline:
 - Learn a representation z st. $p(z|x)$, $q(x|z)$ exists.
 - If we can model $p(z)$, we get $p(x) = q(x|z)p(z)$.
- SOTA image generation methods depend on VAE backbones.
 - VQGAN -- extension of VQVAE (will be covered next lecture)
 - Stable Diffusion
 - CogView (Text-to-Image Transformer)

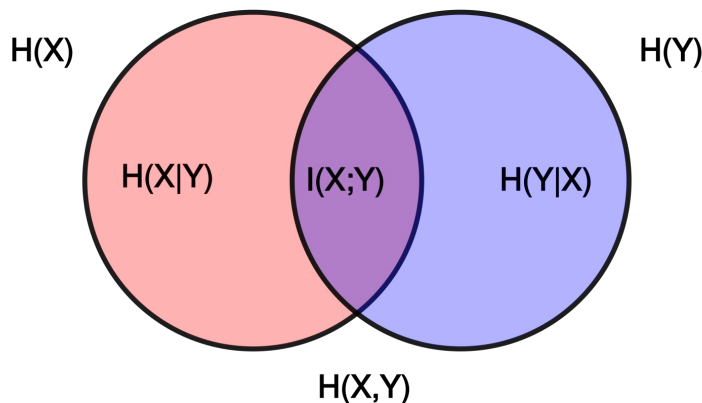
Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower Bound (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - Different priors $p(z)$
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - Connection to other generative models
- Related ideas:
 - **Mutual Information Estimation**

Mutual Information

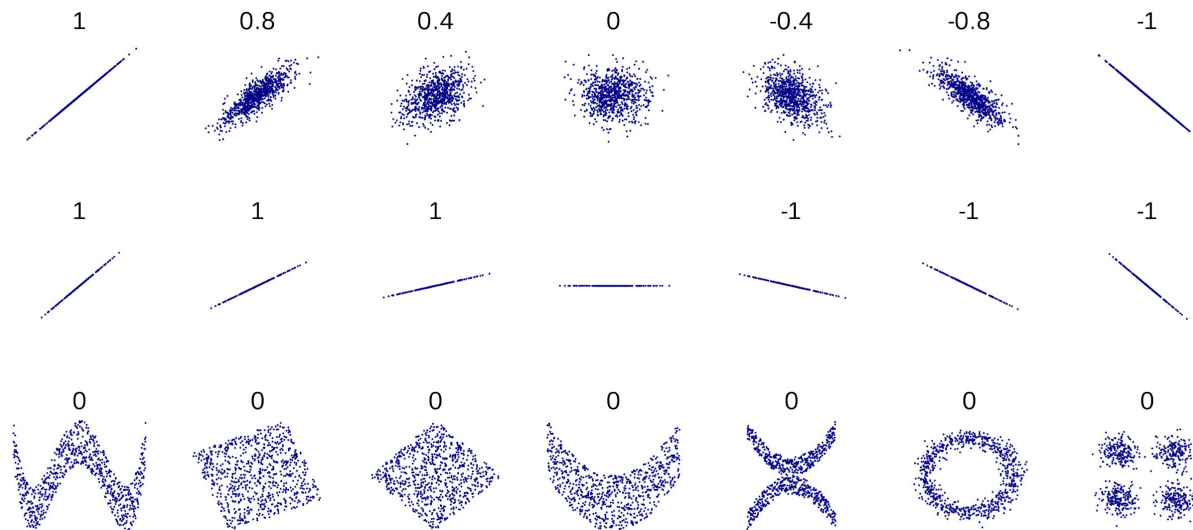
- Mutual information between two random variables X, Y : $I(X; Y)$ is defined as

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$



Mutual Information

- Mutual Information is a general way to measure dependency between two random variables
 - Unlike the more commonly used covariance



Mutual Information

- Useful in a lot of settings where one wants to maximize dependency between two variables or estimate their dependencies:
 - Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning
 - InfoGAN
 - CPC
 - ...

Estimating Mutual Information

- We can try to estimate the mutual information between z and x in a latent variable model

$$\begin{aligned} I(z; x) &= H(z) - H(z|x) \\ &= H(z) - \mathbb{E}_{(z,x) \sim p(z,x)} [-\log p(z|x)] \\ &= H(z) + \mathbb{E}_{(z,x) \sim p(z,x)} [\log p(z|x) - \log q(z|x) + \log q(z|x)] \\ &\geq H(z) + \mathbb{E}_{(z,x) \sim p(z,x)} [\log q(z|x)] \end{aligned}$$

- Has intractable posterior $p(z|x)$ but we can estimate by introducing a variational distribution $q(z|x)$

Decoder distribution

- So far all models use simple distribution for $p(x|z)$
- Due to lack of expressivity itself, all entropy is pushed to z and z needs to convey a lot of information

Powerful decoder

- What's the maximum VLB?

$$\begin{aligned}\mathbb{E}_{x \sim p_{\text{data}}(x)} [VLB] &\leq \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log p_{\theta}(x)] \\ &\leq \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log p_{\text{data}}(x)]\end{aligned}$$

- What if $p(x|z) = p_{\text{data}}(x)$?

$$\begin{aligned}\mathbb{E}_{x \sim p_{\text{data}}(x)} [VLB] &= \mathbb{E}_{x \sim p_{\text{data}}(x), z \sim q(z|x)} [\log p(x|z) + \log p(z) - \log q(z|x)] \\ &= \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log p_{\text{data}}(x) + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log p(z) - \log q(z|x)]]\end{aligned}$$

- $q(z|x)$ would be set to $p(z)$ -> z has no information

Powerful decoder

- Having information in z incurs vlb penalty of $KL(q \parallel p)$ which is usually non-zero
- “Ignoring latent code” problems well documented in literature
 - (Fabiou & van Amersfoort, 2014; Chung et al., 2015; Bowman et al., 2015; Serban et al., 2016; Fraccaro et al., 2016; Xu & Sun, 2016)
 - Many proposed solutions

Weakening models

- Adding dropout in autoregressive conditioning (Bowman et al., 2015)
- PixelCNN with limited receptive field (Chen et al., 2016)
- Constant bit rate $D_{\text{KL}}(q_{\phi}(z|x) \parallel p_{\theta}(z)) = c$ (Guu et al., 2017), (Xu & Durrett, 2018), (Davidson et al., 2018)
- Minimum bit rate $D_{\text{KL}}(q_{\phi}(z|x) \parallel p_{\theta}(z)) \geq \delta$ (Razavi et al., 2019)

Changing training dynamics

- $D_{\text{KL}}(q_{\phi}(z|x) \parallel p_{\theta}(z))$ warmup (Bowman et al., 2015); (Yang et al., 2017); (Kim et al., 2018); (Gulrajani et al., 2016)
- “Free-bits” (Kingma et al., 2016); (Chen et al., 2016)
- More training updates to $q(z|x)$ (He et al., 2019)

Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower Bound (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- **Variations:**
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
 - **Disentanglement: Beta VAE**
- Related ideas:
 - Variational Dequantization (flow++)
 - Mutual Information Estimation

Beta VAE

The Beta-VAE objective is identical to the VAE objective when $\beta = 1$

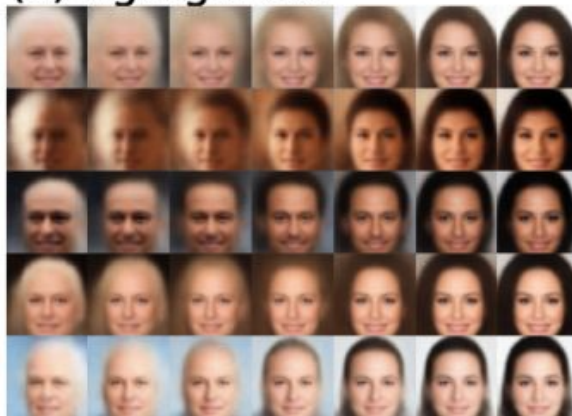
$$\mathcal{L} = \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - \beta D_{KL}(q_{\phi}(z|x) || p(z))$$

Beta VAE -- Experiments

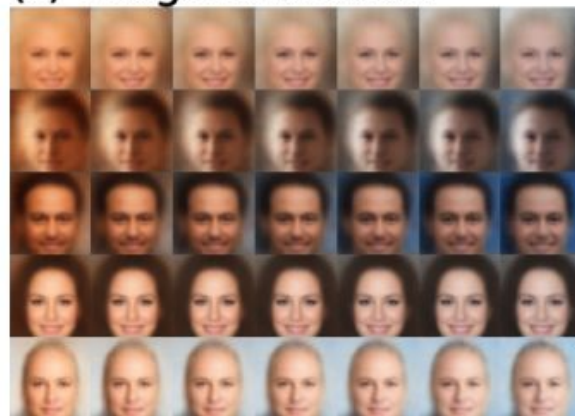
(a) Skin colour



(b) Age/gender



(c) Image saturation



Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
 - Disentanglement: Beta VAE
- ***Related ideas:***
 - ***Variational Dequantization (flow++)***
 - Mutual Information Estimation

Recap: Uniform Dequantization

- **Uniform Dequantization.** Add noise to data.
 - $\mathbf{x} \in \{0, 1, 2, \dots, 255\}$
 - We draw noise $\mathbf{u}$ uniformly from $[0, 1)^D$

$$\begin{aligned}\mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} [\log p_{\text{model}}(\mathbf{y})] &= \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \int_{[0,1)^D} \log p_{\text{model}}(\mathbf{x} + \mathbf{u}) d\mathbf{u} \\ &\leq \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \log \int_{[0,1)^D} p_{\text{model}}(\mathbf{x} + \mathbf{u}) d\mathbf{u} \\ &= \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} [\log P_{\text{model}}(\mathbf{x})]\end{aligned}$$

[Theis, Oord, Bethge, 2016]

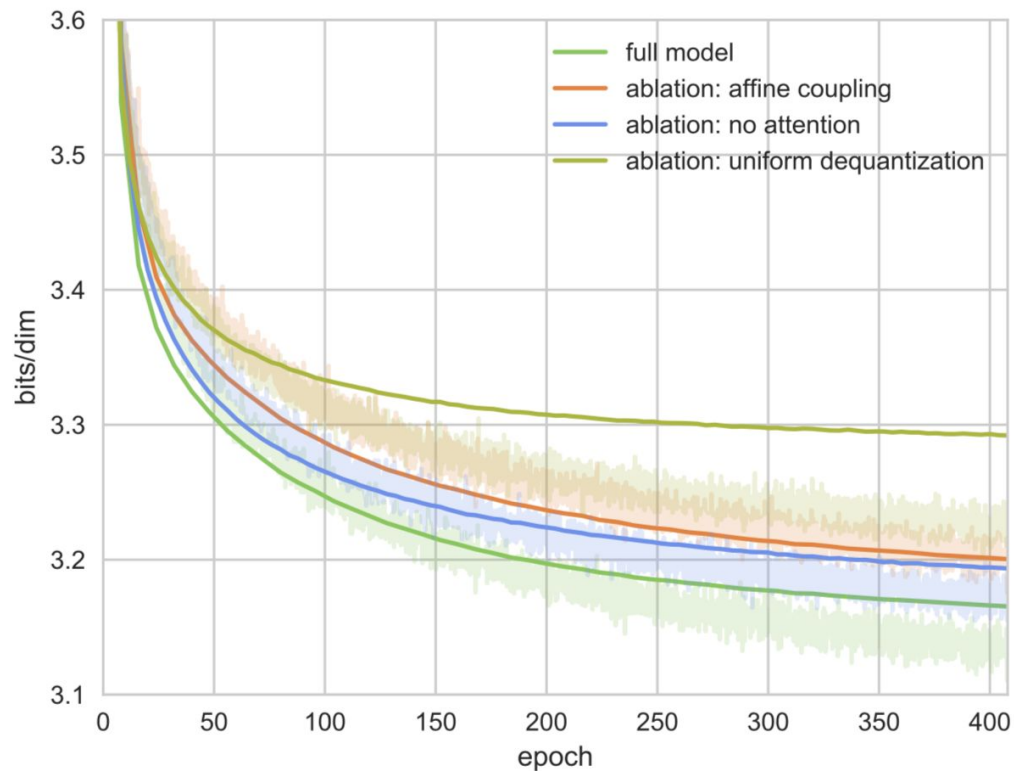
Variational Dequantization

- **Variational Dequantization.** Add a learnable noise q to data.

$$\begin{aligned}\mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} [\log P_{\text{model}}(\mathbf{x})] &= \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\log \int_{[0,1]^D} q(\mathbf{u}|\mathbf{x}) \frac{p_{\text{model}}(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} d\mathbf{u} \right] \\ &\geq \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\int_{[0,1]^D} q(\mathbf{u}|\mathbf{x}) \log \frac{p_{\text{model}}(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} d\mathbf{u} \right] \\ &= \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} \mathbb{E}_{\mathbf{u} \sim q(\cdot|\mathbf{x})} \left[\log \frac{p_{\text{model}}(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} \right]\end{aligned}$$

[Ho et al., 2019]

Variational Dequantization on CIFAR



[Ho et al., 2019]

Outline

- Motivation
- Training Latent Variable Models (including VAE and IWAE)
 - Objective
 - Exact
 - Prior Sampling
 - Importance Sampling
 - Variational Lower Bound (VLB) / Evidence Lower BOund (ELBO)
 - Optimization
 - Likelihood Ratio Gradients vs. Reparameterization Trick Gradients
 - Optimizing the VLB/ELBO
- Variations:
 - SOTA: VQ-VAE, VQ-VAE 2.0
 - AR + VAE: Variational Lossy AutoEncoder, PixelVAE
 - Disentanglement: Beta VAE
- ***Related ideas:***
 - Variational Dequantization (flow++)
 - ***Mutual Information Estimation***

Bibliography

VAE: Auto-Encoding Variational Bayes, D. Kingma and M. Welling, ICLR 2014, <https://arxiv.org/pdf/1312.6114.pdf>

IWAE: Importance Weighted Autoencoders, Y. Burda, R. Grosse and R. Salakhutdinov, ICRL 2015, <https://arxiv.org/pdf/1509.00519.pdf>

VQ-VAE: Neural Discrete Representation Learning, A. van den Oord, O. Vinyals and K. Kavukcuoglu, NeurIPS 2017, <https://arxiv.org/pdf/1711.00937.pdf>

VQ-VAE 2.0: Generating Diverse High-Fidelity Images with VQ-VAE-2, A. Razavi, A. van den Oord and O. Vinyals, NeurIPS 2018, <https://arxiv.org/pdf/1906.00446.pdf>

VLAE: Variational Lossy Autoencoder, X. Chen et al, ICRL 2017, <https://arxiv.org/pdf/1611.02731.pdf>

PixelVAE: A Latent Variable Model for Natural Images, I. Gulrajani et al, <https://arxiv.org/pdf/1611.05013.pdf>

IAF-VAE: Improving Variational Inference with Inverse Autoregressive Flow, D. Kingma et al, NeurIPS 2016, <https://arxiv.org/pdf/1606.04934.pdf>

beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, I. Higgins et al, ICLR 2017,, <https://openreview.net/pdf?id=Sy2fzU9ql>

Wake-Sleep: The wake-sleep algorithm for unsupervised neural networks, G. Hinton et al, <https://www.cs.toronto.edu/~hinton/csc2535/readings/ws.pdf>

Variational Dequantization (flow++) (Flow++): Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design, J. Ho et al, ICML 2019, <https://arxiv.org/pdf/1902.00275.pdf>

Extra Slides

Wake-Sleep algorithm

- Note: VLB was derived from $\min \text{KL}[q_x(z) || p(z|x)]$, hard to optimize because z drawn from $q_x(z)$
- What if we instead minimize $\text{KL}[p(z|x) || q_x(z)]$ 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 $\mathbf{x} \sim p_{\text{model}}$ instead of $\mathbf{x} \sim p_{\text{data}}$
- Problem?
 - VLB is maximized with x drawn from p_{data}
 - minimizing $\text{kl}(p || q)$ 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

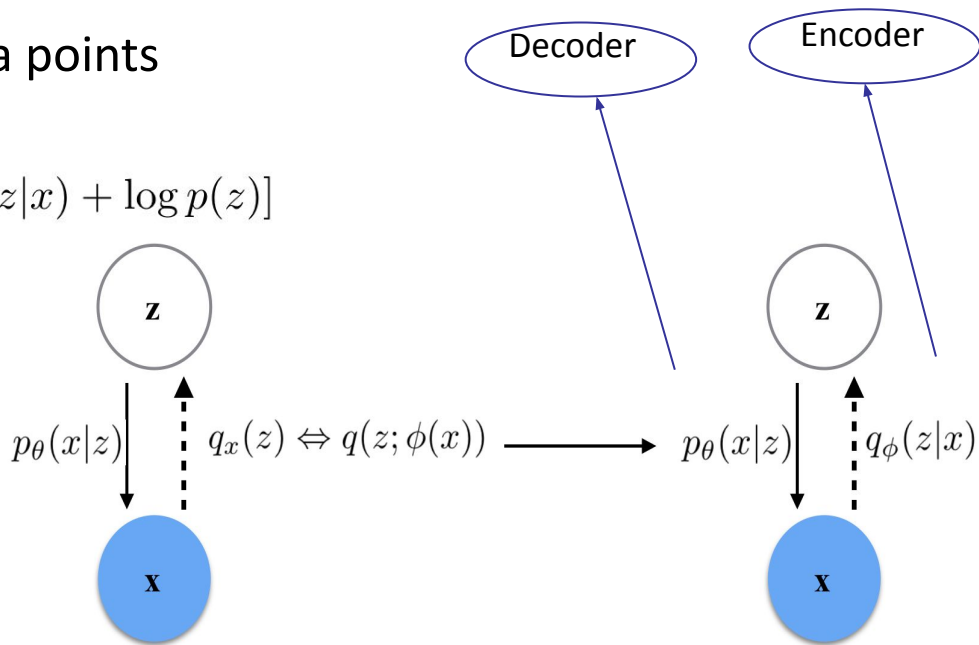
$$KL(p(z|x)||q(z; \phi(x))) = E_{z \sim p(z|x)} \left[\underbrace{\log p(z|x)}_{\text{independent of } \phi} - \log q(z; \phi(x)) \right]$$

Amortized Inference

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

- Amortized Inference

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



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