Deep Learning
lecture 5
Variational Autoencoder

Yi Wu, IIIS

Spring 2025

Mar-17

Logistics

- Coding Project 2 due tonight!
  - Be aware of submission format and evaluation script!
  - Please make sure your model can be properly evaluated!

Today’s Topic

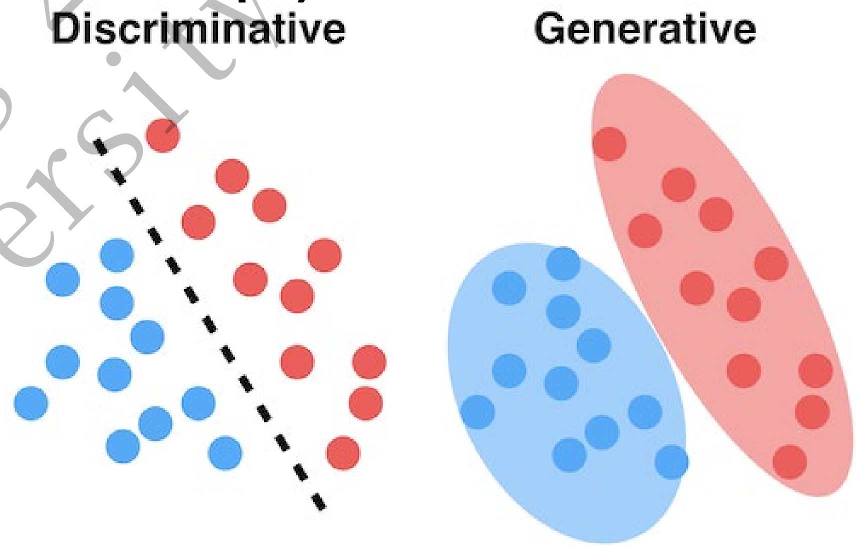
- Latent Variable Model
  - Variational inference

- Variational Autoencoder

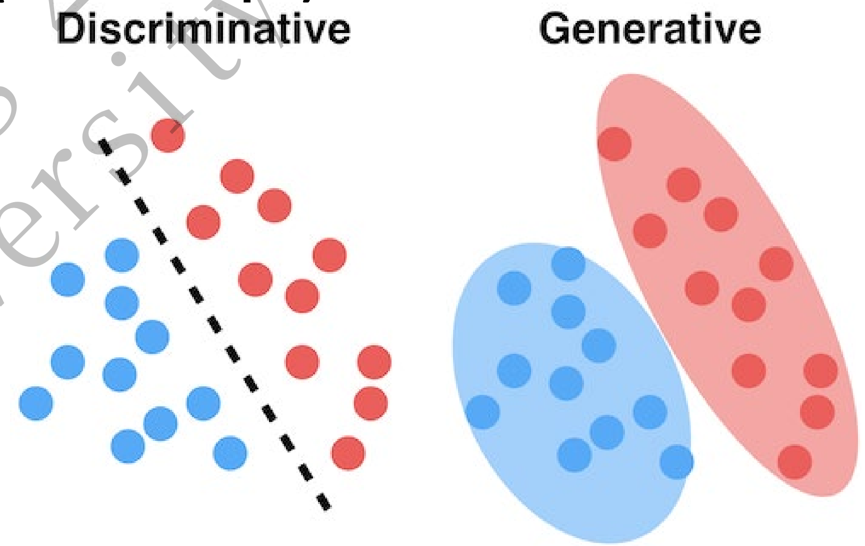
Discriminative v.s. Generative (Recap)

- Discriminative model (lecture 2-3)
  - Objective: $p(y|x)$
  - Simple problem and feedforward networks

- Generative model
  - Objective: $p(x)$ or $p(x,y)$
  - The problem itself is hard (need to model high-dimensional data distribution)

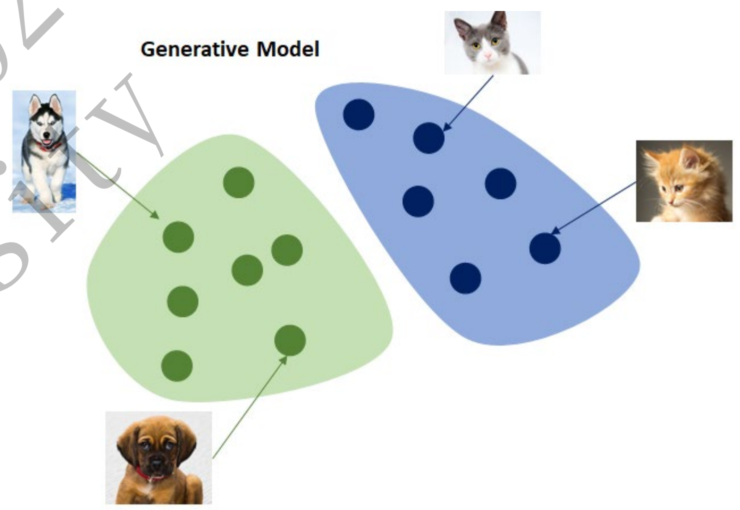


Discriminative v.s. Generative (Recap)

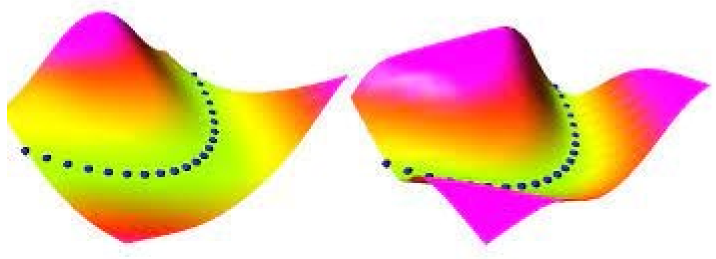


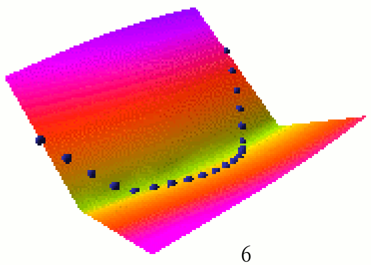
- Discriminative model (lecture 2-3)
  - Objective: $p(y|x)$
  - Simple problem and feedforward networks
  - Typically require labels (supervised learning)
- Generative model
  - Objective: $p(x)$ or $p(x, y)$
  - The problem itself is hard (need to model high-dimensional data distribution)
  - Generating high-dimensional data in a flexible way
    - But sampling can be non-trivial (e.g., Energy-based model)

Energy-Based Model



- Goal: learn $p(x;\theta)$
- General formulation: $p(x) = \frac{1}{Z}\exp(-E(x))$
  - Non-trivial sampling: MCMC
    - Gradients! (Stochastic Gradient MCMC)
  - Z: partition function (key challenge)
    - No closed-form density for $p(x)$
  - Learning: Contrastive Divergence
    - No closed form for $p(x)$
    - Run Monte Carlo sampling to estimate gradient
    - Decrease $E(x)$ on data samples & increase $E(x')$ on non-data samples
- Can we design an easy-to-sample model?





Intuition

- Goal: design $p(x)$ s.t.
  - Easy to sample
  - Easy to compute likelihood (density function)

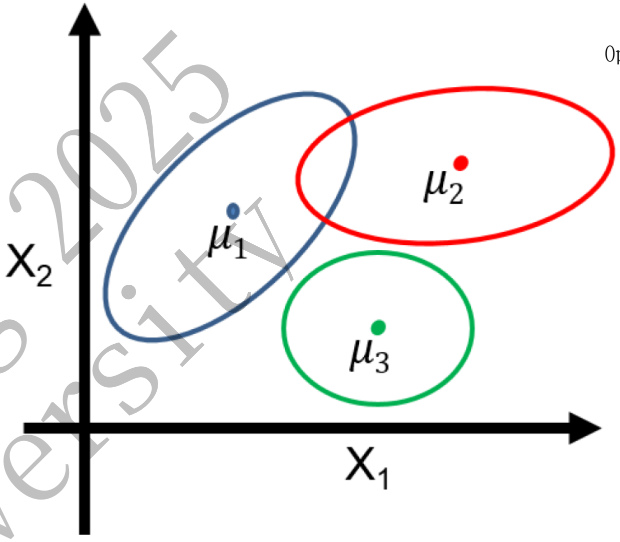
Intuition

- Goal: design $p(x)$ s.t.
  - Easy to sample
  - Easy to compute likelihood (density function)
- Easy to sample?
  - Recap: how do we sample from a complex random variable $p(x)$?
    - Easy to hard sampling: $z \to x$
    - Uniform & Gaussian! $z \sim \text{Unif}(0,1)$ or $z \sim N(0,1)$
  - Principle: first sample z and then sample x based on z
    - First $z \sim N(0,1)$
    - Then $p(x|z)$: design a sampling process for x from z
      - E.g. $x \sim N(f(z), g(z))$

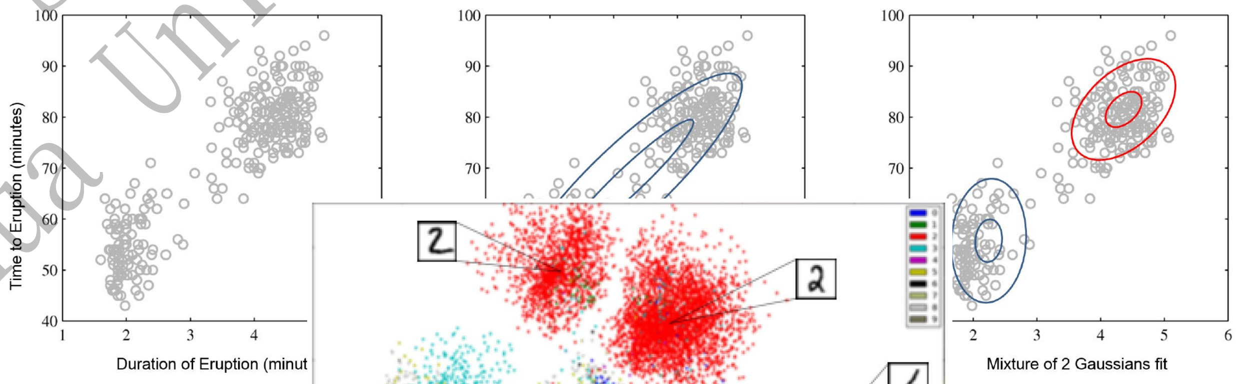
Latent Variable Model

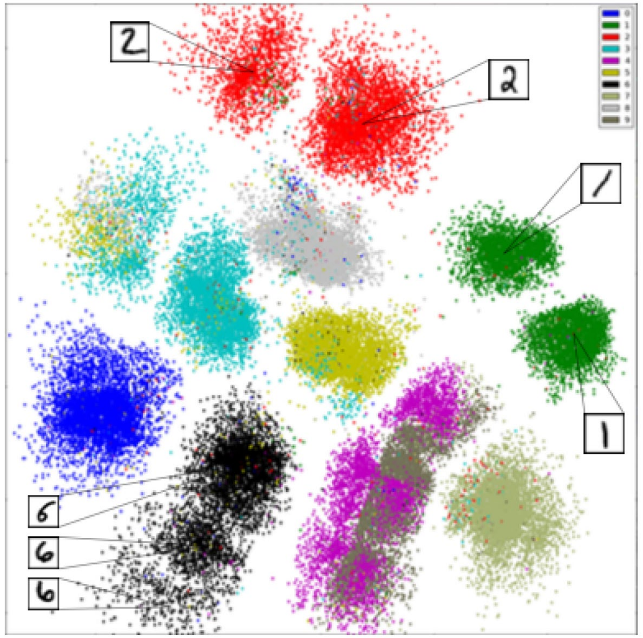
- Formulation $p(x, z) = p(z)p(x|z)$
  - z: latent variable
  - x: observed variable (data)
- Key Idea: from a simple variable z to high-dimensional data x
  - $p(z)$: prior distribution
  - $p(x|z)$: conditional distribution (conditional likelihood)
  - $p(z|x)$: posterior distribution
  - $p(x) = \int_z p(x|z)p(z)$: marginal distribution (marginal likelihood)
    - $p(x)$ remains non-trivial
- Generation is straightforward
  - Assume $p(z)$ and $p(x|z)$ are easy to sample from

Latent Variable Model



- $p(x, z) = p(z)p(x|z)$
  - x data; z latent variable
- Example: Gaussian Mixture Model
  - $z \sim \text{Categorical}(w_1, \dots, w_K)$
  - $x \sim N(\mu_z, \Sigma_z)$
  - Parameters: $\forall 1 \leq k \leq K, w_k, \mu_k, \Sigma_k$
  - Generation process
    - Sampling a cluster index z
    - Generate x according to the cluster distribution
  - Training process
    - Given data, learn cluster centers and co-variances
    - This is called ***clustering***





Latent Variable Model: Training

- Learning the latent variable model
  - Joint probability: $p(x, z; \theta)$ for random variable X and Z
    - $p(x, z; \theta) = p(z; \theta)p(x|z; \theta)$
  - Dataset $D = \{x^{(i)}\}$ for X, variable Z is never observed
- Maximal Likelihood Learning

$$L(\theta) = \log \prod_{x \in D} p(x; \theta) = \sum_{x \in D} \log \sum_{z} p(x, z; \theta)$$

  - Marginal probability can be expensive to compute!
    - When z is continuous, the objective even becomes intractable

Latent Variable Model: Training

- Learning the latent variable model
  - Joint probability: $p(x, z; \theta)$ for random variable X and Z
    - $p(x, z; \theta) = p(z; \theta)p(x|z; \theta)$
  - Dataset $D = \{x^{(i)}\}$ for X, variable Z is never observed
- Maximal Likelihood Learning

$$L(\theta) = \log \prod_{x \in D} p(x; \theta) = \sum_{x \in D} \log \sum_{z} p(x, z; \theta)$$

  - Marginal probability can be expensive to compute!
    - When z is continuous, the objective even becomes intractable
    - Remark: $\nabla L(\theta)$ can be tractable when $p(x, z) \propto \exp(-E(x, z))$

Latent Variable Model: Training

- Learning the latent variable model
  - Joint probability: $p(x, z; \theta)$ for random variable X and Z
    - $p(x, z; \theta) = p(z; \theta)p(x|z; \theta)$
  - Dataset $D = \{x^{(i)}\}$ for X, variable Z is never observed
- Maximal Likelihood Learning

$$L(\theta) = \log \prod_{x \in D} p(x; \theta) = \sum_{x \in D} \log \sum_{z} p(x, z; \theta)$$

  - Marginal probability can be expensive to compute!
    - When z is continuous, the objective even becomes intractable
    - Remark: $\nabla L(\theta)$ can be tractable when $p(x, z) \propto \exp(-E(x, z))$
- Goal: a fast approximation of the marginal probability

Latent Variable Model: Training

- Goal: approximation of $\log \sum_z p(x, z; \theta)$
- Idea#1: Importance Sampling
  - Proposal distribution $q(z)$

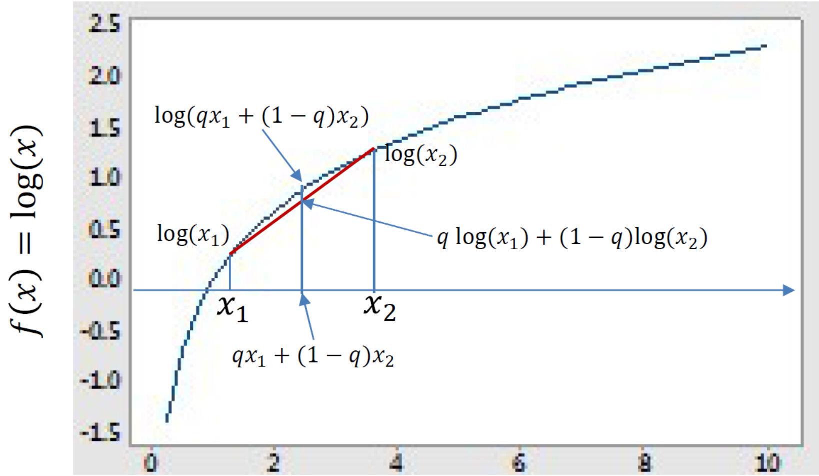
$$p(x) = \sum_z q(z) \cdot \frac{p(x, z; \theta)}{q(z)}$$

    - The probability can be approximated by drawing samples from $q(z)$
  - Learning objective $L(x; \theta)$

$$L(x; \theta) = \log \sum_z q(z) \cdot \frac{p(x, z; \theta)}{q(z)}$$

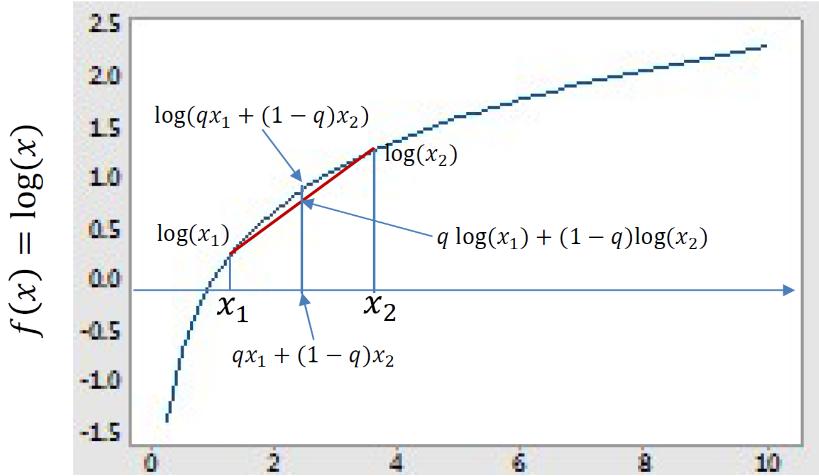
Latent Variable Model: Training

- Goal: approximation of $\log \sum_z q(z) \cdot \frac{p(x,z;\theta)}{q(z)}$
- Idea#2: concavity of $\log(\cdot)$
  - $\log \sum_z q(z) \cdot \frac{p(x,z;\theta)}{q(z)}$
  - For any $0 < x_1 \le x_2 \le 1$,
    - $\log(\alpha x_1 + (1-\alpha)x_2) \ge \alpha \log(x_1) + (1-\alpha)\log(x_2)$
  - More general, for any weights $\alpha_i > 0$ & $\sum_i \alpha_i = 1$,
    - $\log(\sum_i \alpha_i x_i) \ge \sum_i \alpha_i \log(x_i)$



Latent Variable Model: Training

- Goal: approximation of $\log \sum_z q(z) \cdot \frac{p(x,z;\theta)}{q(z)}$
- Idea#2: concavity of $\log(\cdot)$
  - $\boxed{\log \sum_z q(z) \cdot \frac{p(x,z;\theta)}{q(z)} \geq \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}}$
  - For any $0 < x_1 \leq x_2 \leq 1$,
    - $\log(\alpha x_1 + (1-\alpha)x_2) \geq \alpha \log(x_1) + (1-\alpha)\log(x_2)$
  - More general, for any weights $\alpha_i > 0$ & $\sum_i \alpha_i = 1$,
    - $\log(\sum_i \alpha_i x_i) \geq \sum_i \alpha_i \log(x_i)$



Latent Variable Model: Training

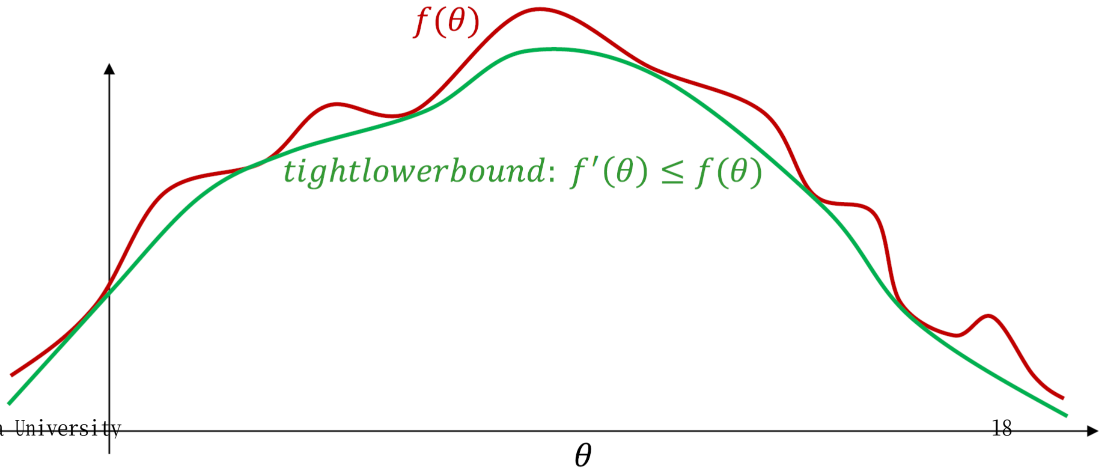
- Goal: approximate $\log \sum_z p(x,z;\theta)$
  - Ideas: importance sampling & concavity of $\log(\cdot)$
- Evidence Lower Bound (ELBO)

$$\log p(x;\theta) = \log \sum_z p(x,z;\theta) \geq \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$$

  - A tractable lower bound of the true objective
    - Easy to optimize
- When will the equality hold?
  - i.e., a tight lower bound
  - Sol: $q(z) \leftarrow p(z|x;\theta)$

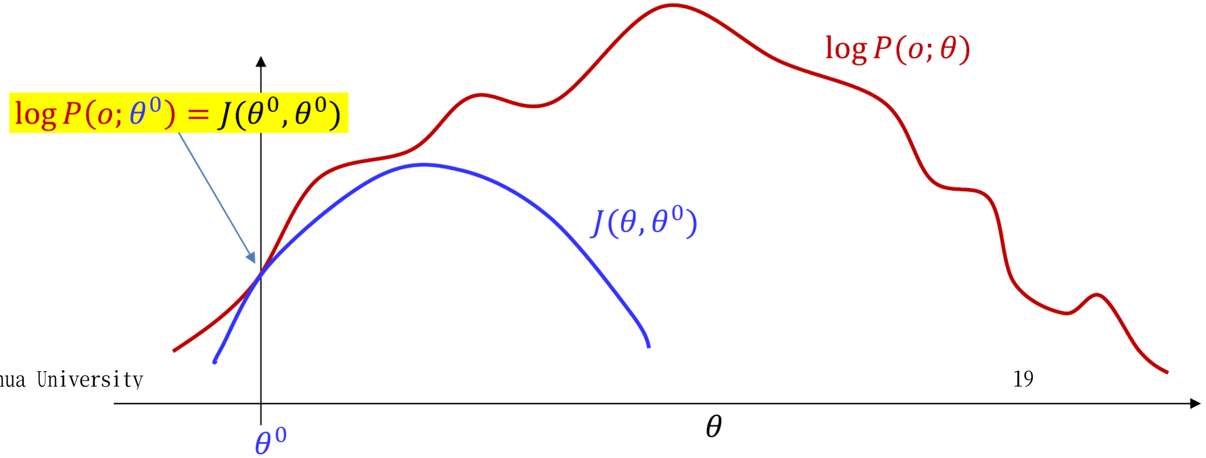
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta)$
  - Repeat



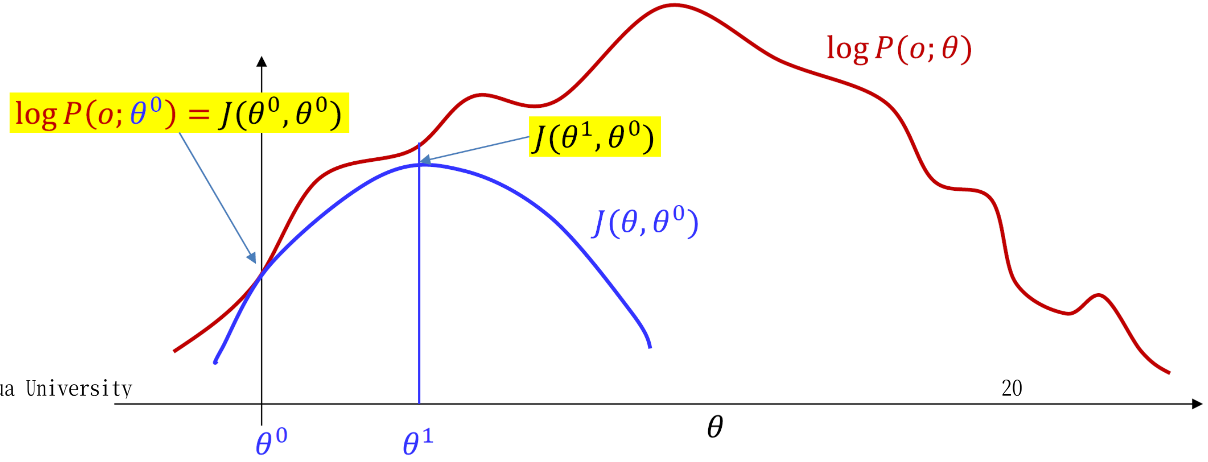
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta^0)$
  - Repeat



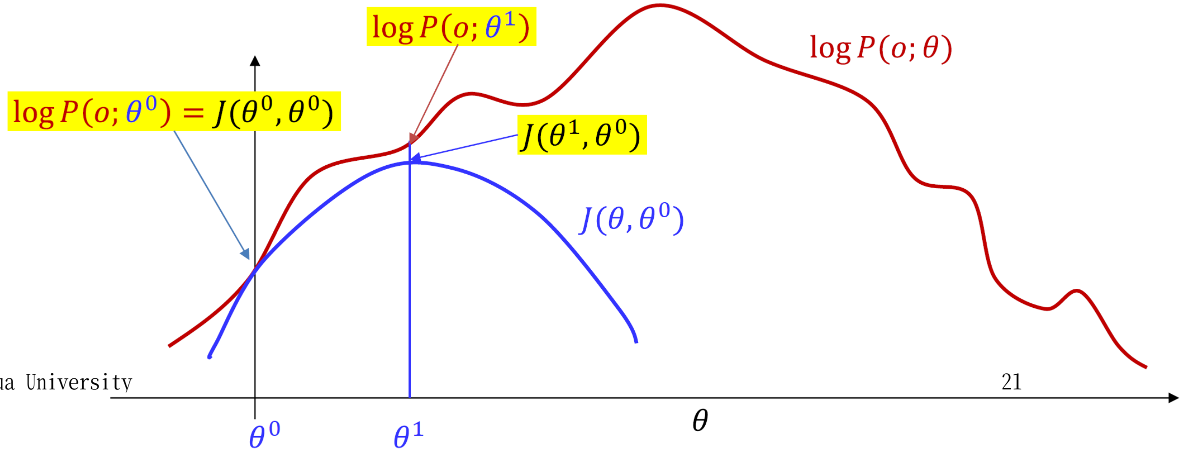
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z;\theta^0)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta)$
  - Repeat



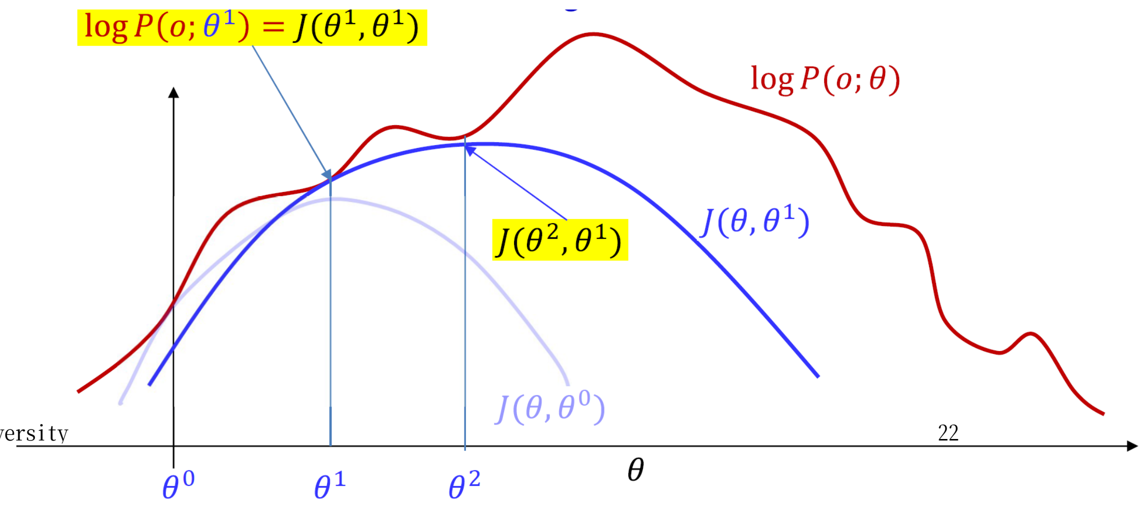
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta^1)$
  - Repeat



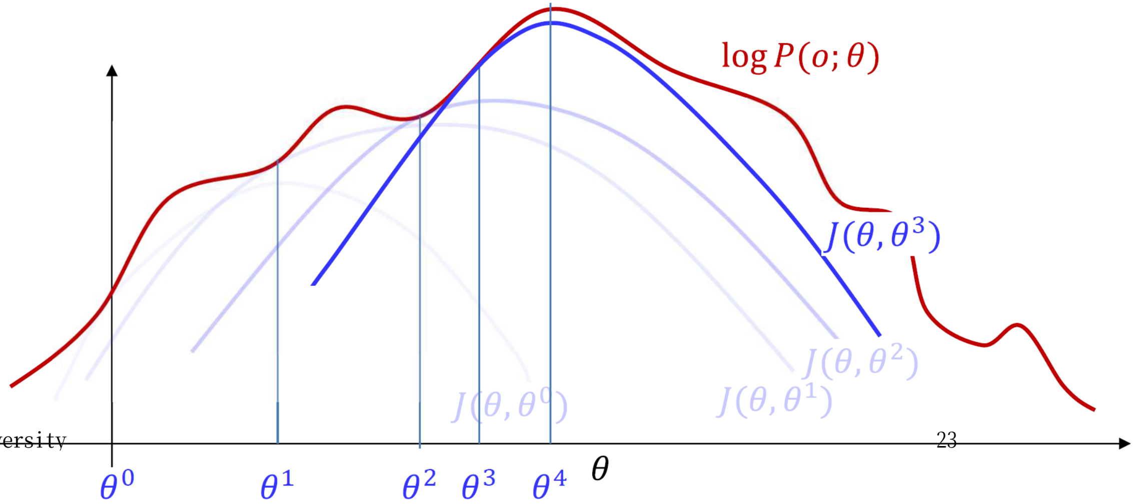
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z;\theta^1)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta^2)$
  - Repeat



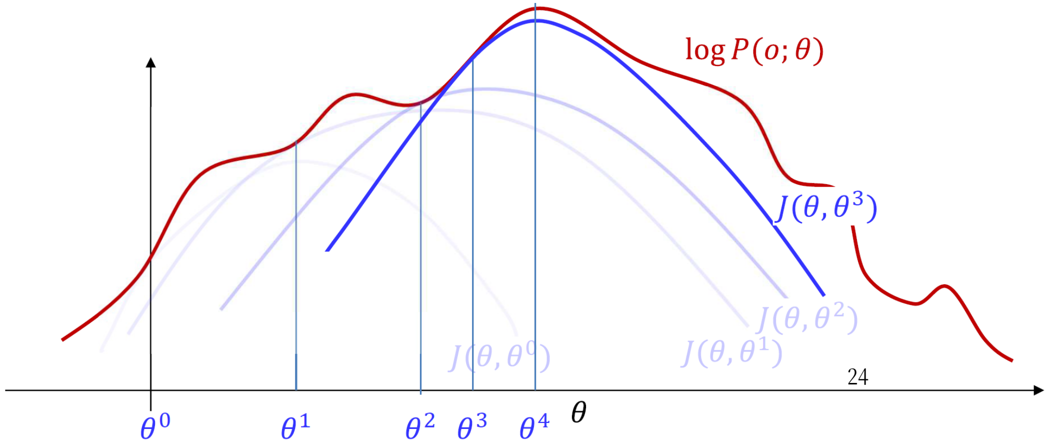
Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z)$
    - $J(\theta) = \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)}$
  - Set $q(z) \leftarrow p(z|x;\theta)$
  - Repeat
  - Converge to a local optimum



Evidence Lower Bound

- ELBO becomes exact when $q(z) = p(z|x;\theta)$
  - $\sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = \sum_z q(z) \log \frac{p(x,z;\theta)}{p(z|x;\theta)}$
  - $= \sum_z q(z) \log p(x;\theta)$
  - $= \log p(x;\theta)$
- We can optimize a tight lower bound by setting $q(z) = p(z|x;\theta)$
- An iterative process
  - Optimize $p(x,z;\theta)$ w.r.t. fixed $q(z)$ (M-step)
  - Set $q(z) \leftarrow p(z|x;\theta)$ (E-step)
  - ***An EM algorithm***
- How to set $q(z) \leftarrow p(z|x;\theta)$?



Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. a fixed θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_\phi KL(q||p)$
  - Remark: pay attention to the order of KL (reverse KL)!
  - Mean-field variational inference
    - A factored proposal: $q(z) = \prod_i q_i(z_i|x)$
    - By calculus of variation （变分法，泛函分析领域）

$$\log q_i^\star(z_i|x) = \mathrm{E}_{z_{j\neq i}}[\log p(z,x)] + constant$$

    - Repeatedly update the distribution of $q_i(z_i)$ using the expectation of $p(z,x)$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
    - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
        - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
    - Distance metric between $q(z;\phi)$ and $p(z|x)$
        - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$
    - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x)}$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$
  - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x)} = \sum_z q(z;\phi) \log \frac{q(z;\phi)p(x)}{p(z,x)}$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_\phi KL(q||p)$
  - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x)} = \sum_z q(z;\phi) \log \frac{q(z;\phi)p(x)}{p(z,x)}$
  - $= \sum_z q(z;\phi) \log p(x) - \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$
  - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x)} = \sum_z q(z;\phi) \log \frac{q(z;\phi)p(x)}{p(z,x)}$
  - $= \underbrace{\log p(x)}_{\textbf{Constant}} - \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$
  - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x)} = \sum_z q(z;\phi) \log \frac{q(z;\phi)p(x)}{p(z,x)}$
  - $= \log p(x) - \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$
  - $L(\phi) = \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$

Evidence Lower Bound (ELBO)!!!

Also called variational lower bound in VI

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z)\log\frac{q(z)}{p(z)}$
- Variational Inference: $\min_\phi KL(q||p)$
  - $KL(q(z;\phi)||p(z|x)) = \sum_z q(z;\phi)\log\frac{q(z;\phi)}{p(z|x)} = \sum_z q(z;\phi)\log\frac{q(z;\phi)p(x)}{p(z,x)}$
  - $= \log p(x) - \sum_z q(z;\phi)\log\frac{p(z,x)}{q(z;\phi)}$
  - $L(\phi) = \sum_z q(z;\phi)\log\frac{p(z,x)}{q(z;\phi)}$

Variational Inference

- Goal: $q(z;\phi) \leftarrow p(z|x)$
  - Find a parameterized distribution $q(z;\phi)$ to approximate the posterior
    - In our case, we want to learn ϕ to approximate $p(z|x;\theta)$ w.r.t. ***a fixed*** θ
  - Distance metric between $q(z;\phi)$ and $p(z|x)$
    - $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
- Variational Inference: $\min_{\phi} KL(q||p)$
  - $\log p(x) = KL\big(q(z;\phi)||p(z|x)\big) + \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$
  - $=$ approximate error $+$ ELBO $\geq$ ELBO
  - $L(\phi) = \sum_z q(z;\phi) \log \frac{p(z,x)}{q(z;\phi)}$

Variational Inference (Explained)

- General Formulation of Bayesian Inference
  - Dataset $D = \{x\}$
  - Model $p(x; \theta)$ with parameter θ
  - Goal $p(\theta|x)$
    - Remark: optimization learns a single θ^* while BI learns a distribution
- Variational Inference as a Mean of Approximate Bayesian Inference
  - Use $q(\theta; \phi)$ to approximate $p(\theta|x)$
  - VI Objective: $KL(q||p) = C + ELBO$
  - Interpretation: VI objective is a *lower bound* of $\log p(x)$

$$\log p(x; \theta) = \textit{approximation error} + \ ELBO$$

- VAE naturally inherits all the nice mathematical properties of VI ☺
  - Further read: black-box variational inference https://arxiv.org/abs/1401.0118

Latent Variable Model: Training

- Latent Variable Model: $p(z,x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z,x;\theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(z,x;\theta)}{q(z)} = L(\theta;q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$

Latent Variable Model: Training

- Latent Variable Model: $p(z,x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z,x;\theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(z,x;\theta)}{q(z)} = L(\theta;q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$
- Variational Inference
  - Approximate $p(z|x;\theta)$ by a tractable distribution $q(z;\phi)$

Latent Variable Model: Training

- Latent Variable Model: $p(z,x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z,x;\theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(z,x;\theta)}{q(z)} = L(\theta;q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$
- Variational Inference
  - Approximate $p(z|x;\theta)$ by a tractable distribution $q(z;\phi)$

$$L(\phi;\theta) = \sum_z q(z;\phi) \log \frac{p(z,x;\theta)}{q(z;\phi)}$$

Latent Variable Model: Training

- Latent Variable Model: $p(z, x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z, x; \theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(z,x;\theta)}{q(z)} = L(\theta; q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$
- Variational Inference
  - Approximate $p(z|x;\theta)$ by a tractable distribution $q(z;\phi)$

$$L(\phi;\theta) = \sum_z q(z;\phi) \log \frac{p(z, x; \theta)}{q(z;\phi)}$$

Use variational inference to learn a separate $q(z;\phi)$ for each possible x?

Latent Variable Model: Training

- Latent Variable Model: $p(z, x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z, x; \theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(z,x;\theta)}{q(z)} = L(\theta; q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$
- Amortized Variational Inference
  - Approximate $p(z|x;\theta)$ by a conditional tractable distribution $q(z|x;\phi)$

$$L(\phi;\theta) = \sum_z q(z|x;\phi) \log \frac{p(z,x;\theta)}{q(z|x;\phi)}$$

  - A universal approximator q for any x and $p(z|x)$

Latent Variable Model: Training

- Latent Variable Model: $p(z,x) = p(z)p(x|z)$
  - MLE objective: $p(x;\theta) = \sum_z p(z,x;\theta)$
- ELBO: $p(x;\theta) \geq \sum_z q(z) \log \frac{p(x,z;\theta)}{q(z)} = L(\theta;q)$
  - Iterative learning: (1) optimize θ w.r.t. $q(z)$ and (2) $q(z) \leftarrow p(z|x;\theta)$
- Amortized Variational Inference
  - Approximate $p(z|x;\theta)$ by a conditional tractable distribution $q(z|x;\phi)$

$$L(\phi;\theta) = \sum_z q(z|x;\phi) \log \frac{p(z,x;\theta)}{q(z|x;\phi)}$$

- Joint Learning $J(\theta,\phi;x)$

$$J(\theta,\phi;x) = \sum_z q(z|x;\phi) \log \frac{p(z,x;\theta)}{q(z|x;\phi)}$$

Latent Variable Model: Training

- Learning objective $J(\theta, \phi; x)$
  - $J(\theta, \phi; x) = \sum_z q(z|x; \phi)(\log p(z, x; \theta) - \log q(z|x; \phi))$

Latent Variable Model: Training

- Learning objective $J(\theta, \phi; x)$
  - $J(\theta, \phi; x) = \sum_z q(z|x;\phi)(\log p(z, x; \theta) - \log q(z|x;\phi))$
  - $= \sum_z q(z|x;\phi)(\log p(x|z;\theta) - \log q(z|x;\phi) + \log p(z;\theta))$

Latent Variable Model: Training

- Learning objective $J(\theta, \phi; x)$
  - $J(\theta, \phi; x) = \sum_z q(z|x;\phi)(\log p(z, x; \theta) - \log q(z|x;\phi))$
  - $\quad = \sum_z q(z|x;\phi)(\log p(x|z;\theta) - \log q(z|x;\phi) + \log p(z;\theta))$
  - $\quad = \sum_z q(z|x;\phi) \log p(x|z;\theta) - \sum_z q(z|x;\phi) \log \frac{q(z|x;\phi)}{p(z;\theta)}$

Latent Variable Model: Training

- Learning objective $J(\theta, \phi; x)$
  - $J(\theta, \phi; x) = \sum_z q(z|x;\phi)(\log p(z, x; \theta) - \log q(z|x;\phi))$
  - $= \sum_z q(z|x;\phi)(\log p(x|z;\theta) - \log q(z|x;\phi) + \log p(z;\theta))$
  - $= \sum_z q(z|x;\phi) \log p(x|z;\theta) - \sum_z q(z|x;\phi) \log \frac{q(z|x;\phi)}{p(z;\theta)}$
  - $= E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z;\theta))$

Expectation of log likelihood (reconstruction) KL divergence

- Design of $p(z, x; \theta)$ and $q(z|x;\phi)$
  - Principle: easy to compute!
  - Gaussian prior: $p(z) \sim N(0, I)$
  - Gaussian likelihood: $p(x_{ij}|z;\theta) \sim N\left(f_{ij}(z;\theta), 1\right)$
  - Isomorphic Gaussian: $q(z|x;\phi) \sim N\left(\mu(x;\phi), \mathrm{diag}\left(\exp\left(\sigma(x;\phi)\right)\right)\right)$

Latent Variable Model: Training

- Learning objective $J(\theta, \phi; x)$
  - $J(\theta, \phi; x) = \sum_z q(z|x;\phi)(\log p(z,x;\theta) - \log q(z|x;\phi))$
  - $\quad = \sum_z q(z|x;\phi)(\log p(x|z;\theta) - \log q(z|x;\phi) + \log p(z;\theta))$
  - $\quad = \sum_z q(z|x;\phi) \log p(x|z;\theta) - \sum_z q(z|x;\phi) \log \frac{q(z|x;\phi)}{p(z;\theta)}$
  - $\quad = E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z;\theta))$

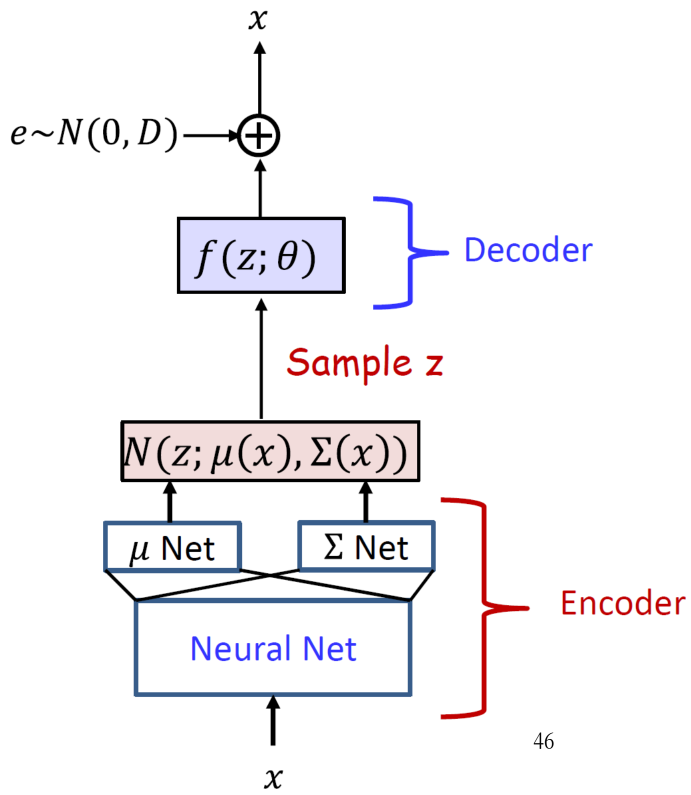
Expectation of log likelihood (reconstruction) KL divergence

- Design of $p(z,x;\theta)$ and $q(z|x;\phi)$
  - Principle: easy to compute!
  - Gaussian prior: $p(z) \sim N(0, I)$
  - Gaussian likelihood: $p(x_{ij}|z;\theta) \sim N\left(f_{ij}(z;\theta), 1\right)$
  - Isomorphic Gaussian: $q(z|x;\phi) \sim N\left(\mu(x;\phi), \mathrm{diag}\left(\exp\left(\sigma(x;\phi)\right)\right)\right)$

Neural networks!

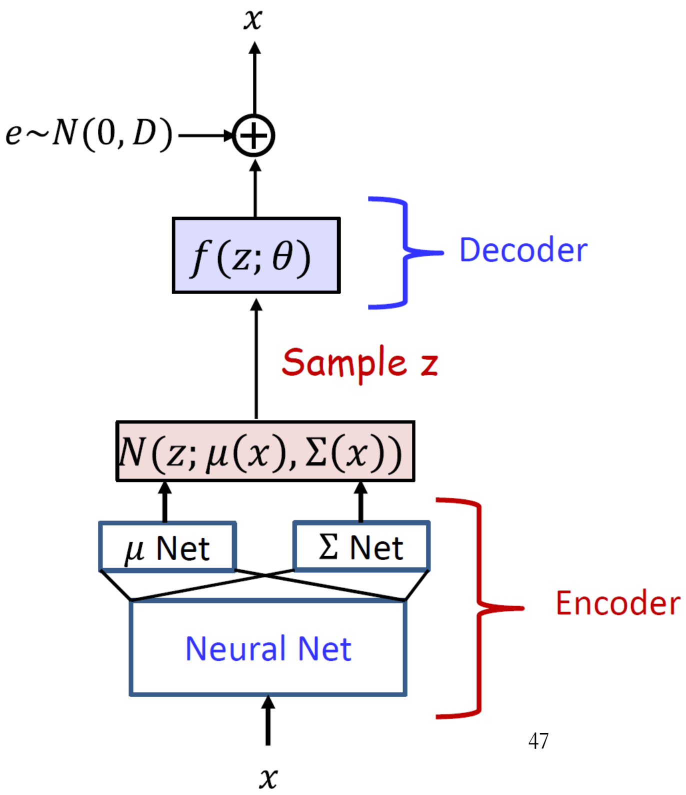
Variational Autoencoder

- VAE Architecture
  - Isomorphic Gaussian: $q(z|x;\phi) \sim N\left(\mu(x;\phi), \text{diag}\left(\exp\left(\sigma(x;\phi)\right)\right)\right)$
  - Gaussian prior: $p(z) \sim N(0, I)$
  - Gaussian likelihood: $p(x|z;\theta) \sim N(f(z;\theta), I)$
- Autoencoder $x \rightarrow z \rightarrow x$
  - Unsupervised learning (data to data, z never observed)
  - Encoder $q(z|x;\phi)$: $x \rightarrow z$
  - Decoder $p(x|z;\theta)$: $z \rightarrow x$
  - Remark
    - $p(x|z)$ is the actual generative model
    - $q(z|x)$ is only the proposal
      - but optimized to approximate $p(z|x)$



Variational Autoencoder

- Training via jointly optimizing ELBO
  - $J(\phi,\theta;x) = \mathrm{E}_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z))$
  - Two terms: likelihood term & KL term

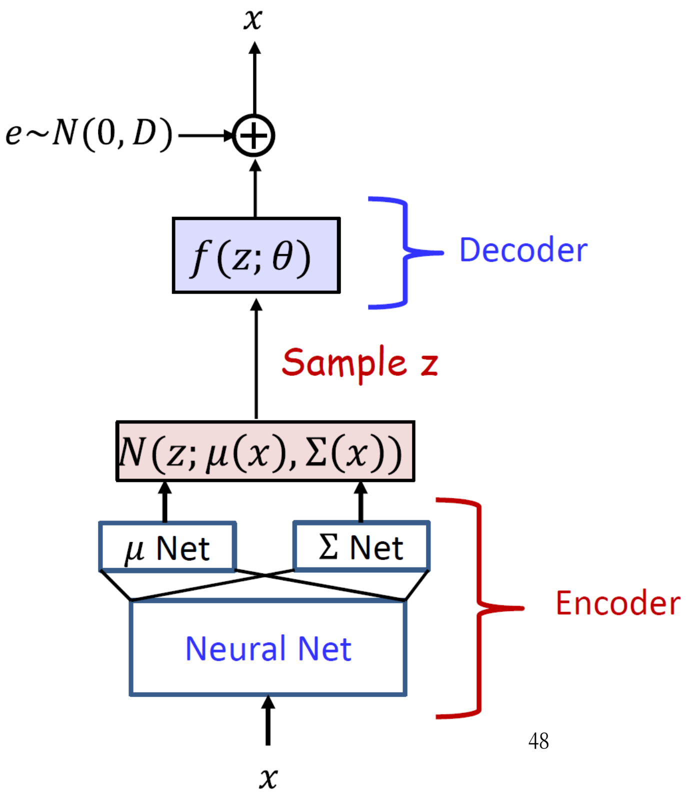


Variational Autoencoder

- Training via jointly optimizing ELBO
  - $J(\phi, \theta; x) = \mathrm{E}_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z))$
- KL penalty
  - $q(z|x;\phi) \sim N\left(\mu(x;\phi), \mathrm{diag}\left(\exp\left(\sigma(x;\phi)\right)\right)\right)$
  - $p(z) \sim N(0, I)$
  - Closed-form!

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left\{\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - k + \ln\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right\}$$

- Implement it in your coding project ☺

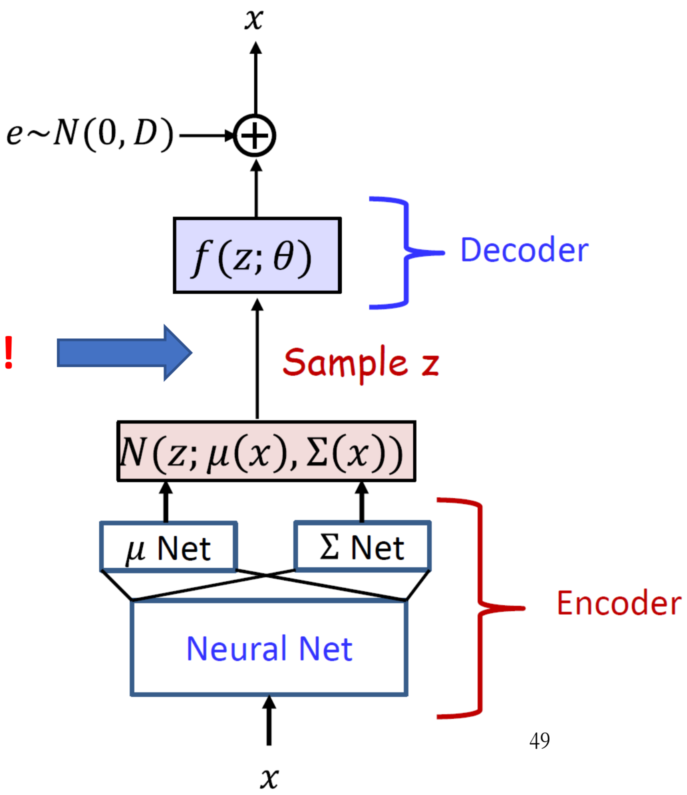


Variational Autoencoder

- Training via jointly optimizing ELBO
  - $J(\phi,\theta;x) = \mathrm{E}_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z))$
- Likelihood term (reconstruction loss)
  - Monte-Carlo estimate!
    - Draw samples from $q(z|x;\phi)$
    - Compute gradient of θ: $L(\theta) \propto \sum_z |x - f(z;\theta)|^2$
      - $x \sim N(f(z;\theta);I)$
      - $p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}|x - f(z;\theta)|^2\right)$
    - How to get **the gradient of ϕ** through $q(z;\phi)$??

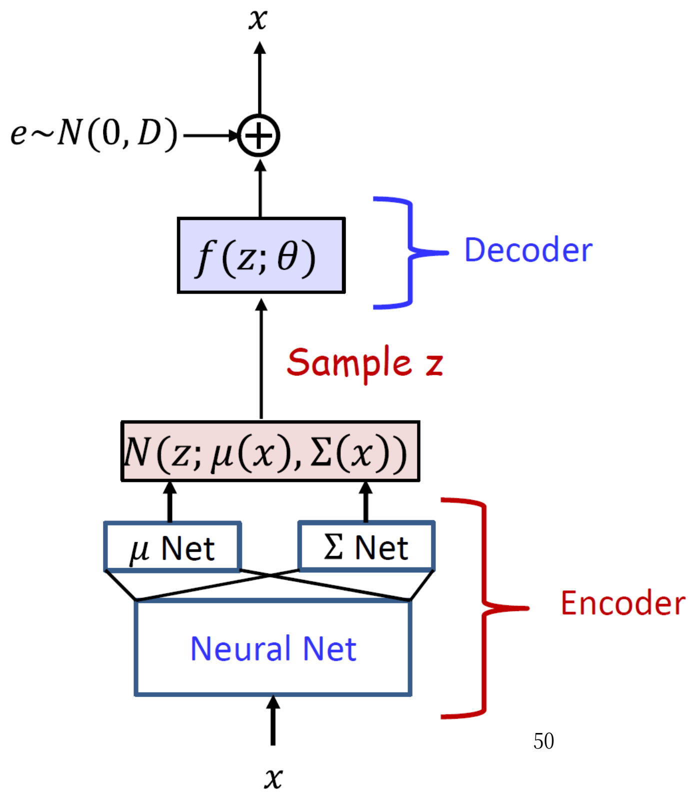
$$L(\phi) = E_{z\sim q(z;\phi)}[\log p(x|z)]$$

No gradient!



Variational Autoencoder

- Training via jointly optimizing ELBO
  - $J(\phi, \theta; x) = \mathrm{E}_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - KL(q(z|x;\phi)||p(z))$
- Likelihood term (reconstruction loss)
  - Monte-Carlo estimate!
    - Draw samples from $q(z|x;\phi)$
    - Compute gradient of θ: $L(\theta) \propto \sum_z |x - f(z;\theta)|^2$
  - Re-parameterization trick
    - Recap the sampling method for Gaussian
      - $z\sim N(\mu, \sigma^2) \iff z = \mu + \sigma \cdot \epsilon,\ \epsilon\sim N(0,1)$
    - $L(\phi) \propto \mathrm{E}_{z\sim q(z|\phi)}[|f(z) - x|^2]$
    - $\propto \mathrm{E}_{\epsilon\sim N(0,I)}[|f(\mu(x;\phi) + \sigma(x;\phi)\cdot\epsilon) - x|^2]$
      - Monte-Carlo estimate for $\nabla L(\phi)$!
      - 1 sample for ϵ is sufficient for stable training



Variational Autoencoder

- Variational Autoencoder (VAE)
  - Encoder $q(z|x;\phi)$
  - Decoder $p(x|z;\theta)$
  - End-to-end unsupervised learning ($x \rightarrow z \sim q(z|x) \rightarrow x$)
    $J(\phi, \theta; x) = \mathrm{E}_{\epsilon \sim N(0,I)}[\log p(x|\mu(x;\phi) + \sigma(x;\phi) \cdot \epsilon; \theta)] - KL(q(z;\phi)||p(z))$
  - By Kingma & Welling, ICLR 2013 (43k citation, ICLR 2023 test-of-time award)

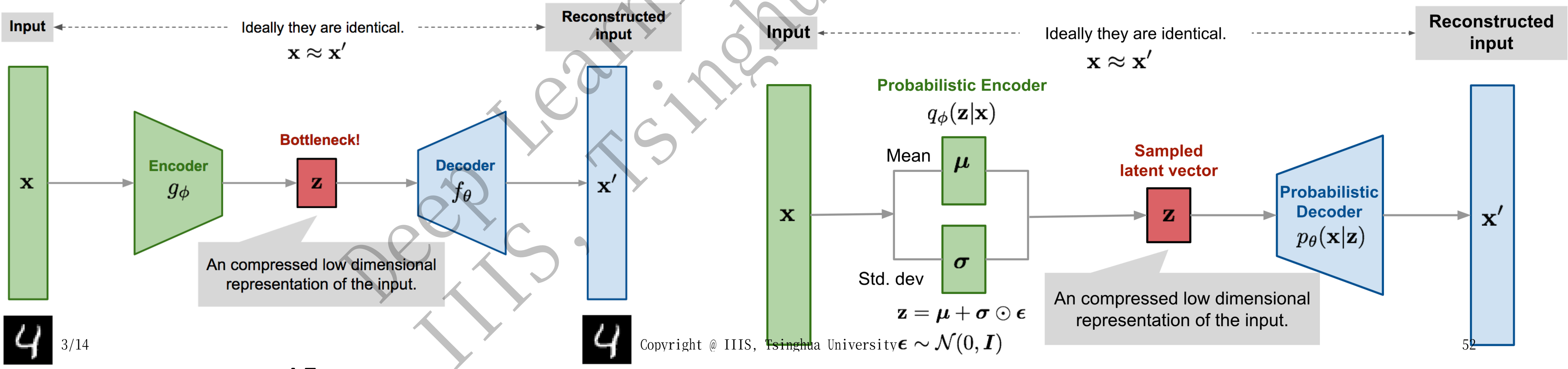
Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

VAE v.s. Standard Autoencoder

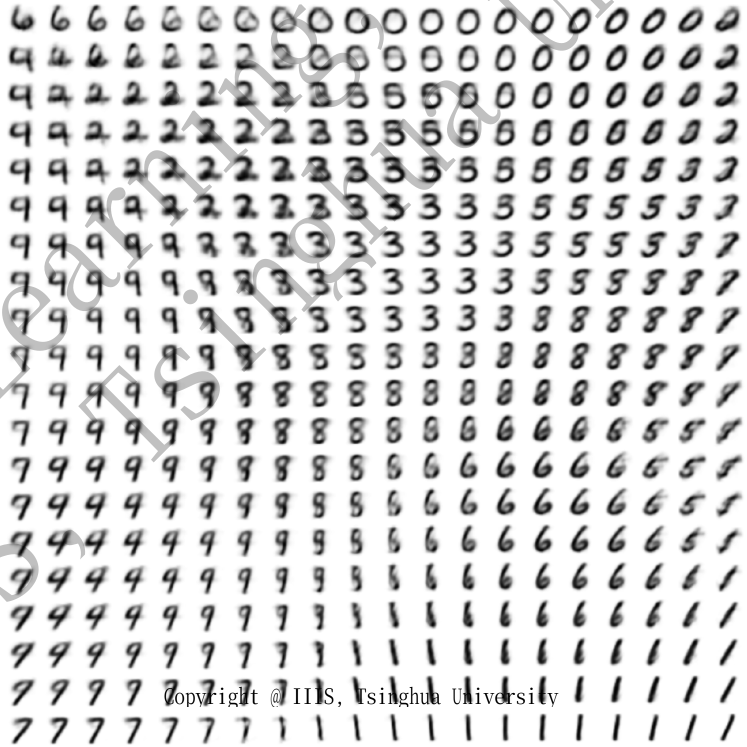
- Autoencoder
  - A classical unsupervised learning method for representation learning
- VAE: a simple generative extension of AE
  - Generative model: AE + Gaussian noise on z
  - KL penalty: L2 constraint on the latent vector z

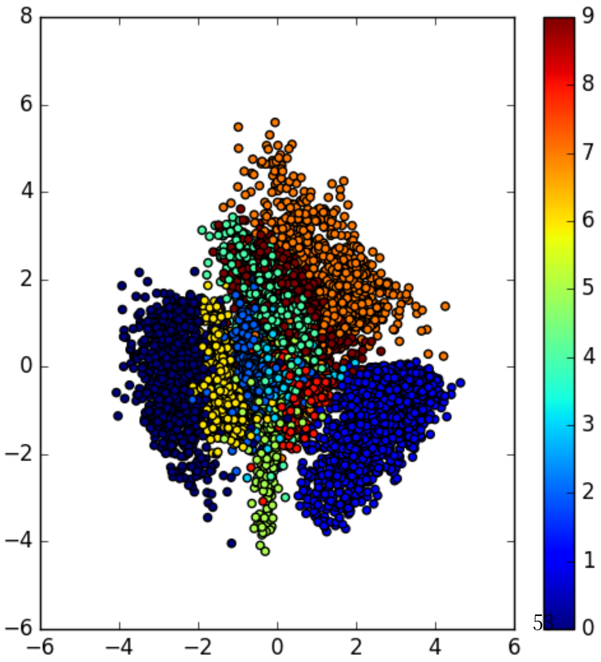


AE VAE

Variational Autoencoder

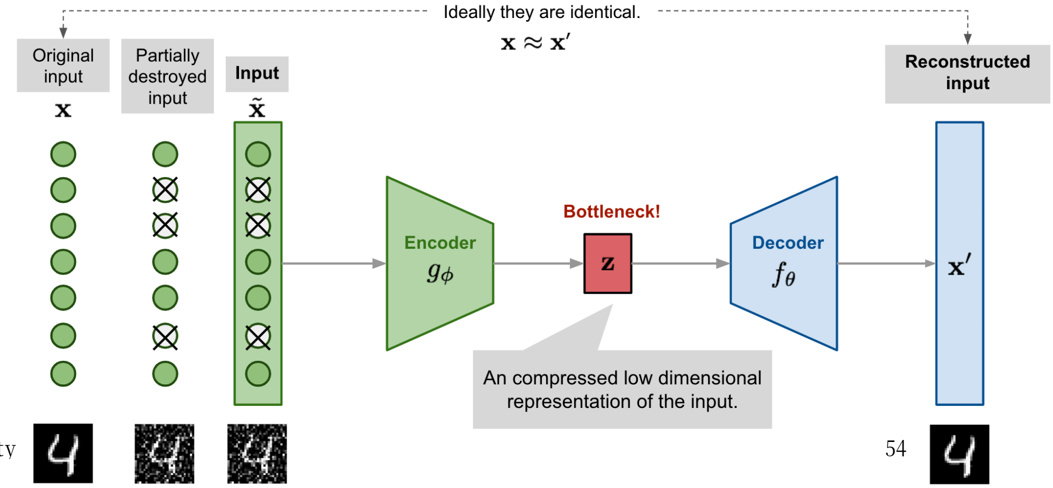
- Interpretable latent space
  - By interpolating z, we can observe how the generated samples vary
  - Automatic clustering in the (low-dimensional) latent space





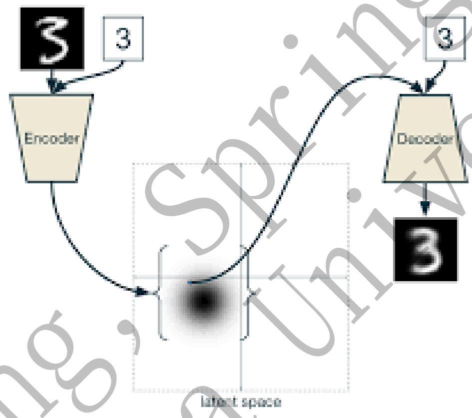
Inpainting with VAE

- Inference the mixing pixels?
  - Fully observable training data $D = \{x^{(i)}\}$
  - Standard VAE: $q(z|x)$ & $p(x|z)$
  - Corrupted data: $\bar{x} = x \odot mask$
  - Goal: $q(z|\bar{x}) \approx q(z|x)$
    - We do not need to change the generator $p(x|z)$
- Randomized mask in training!
  - $x \odot mask \to z \to x$
  - Better encoder architecture
    - Masked convolution
    - Idea: convolution only on unmasked pixels
    - Image Inpainting for Irregular Holes Using Partial Convolutions (ECCV 2018)

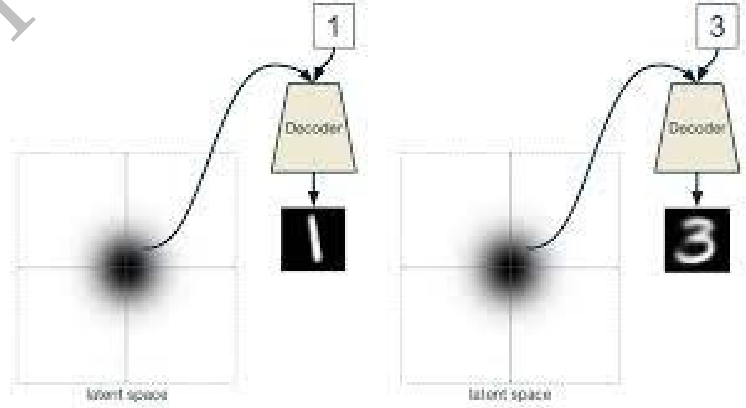


Conditioned VAE

- Include label in VAE
  - $D = \{(x^{(i)}, y^{(i)})\}$
  - Encoder: $q(z|x, y; \phi)$
  - Decoder: $p(x|y, z; \theta)$
  - Conditioned generation!



conditioned VAE

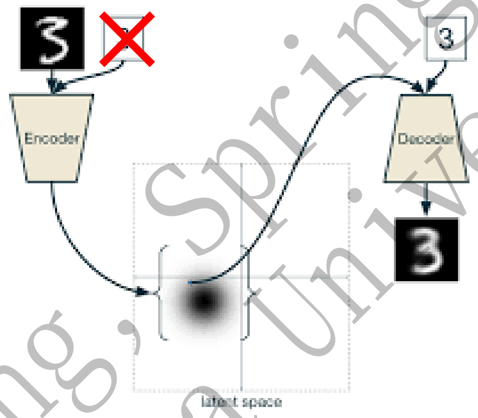


conditioned generation

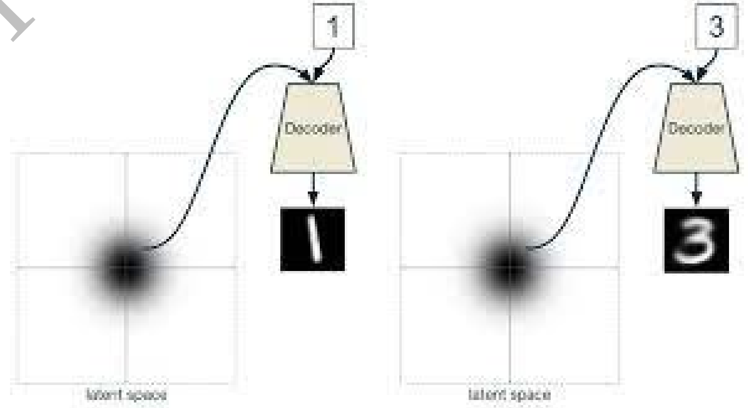
- What if we have both labeled data and unlabeled data?

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder?
    - $q(z, y|x; \phi)$
    - In practice, we assume independent variables $q(z, y|x; \phi) = q(z|x; \phi) \cdot q(y|x; \phi)$



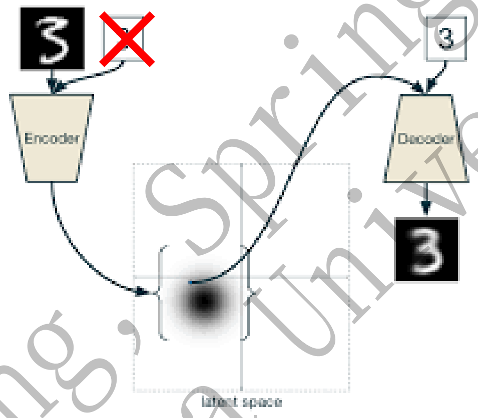
conditioned VAE



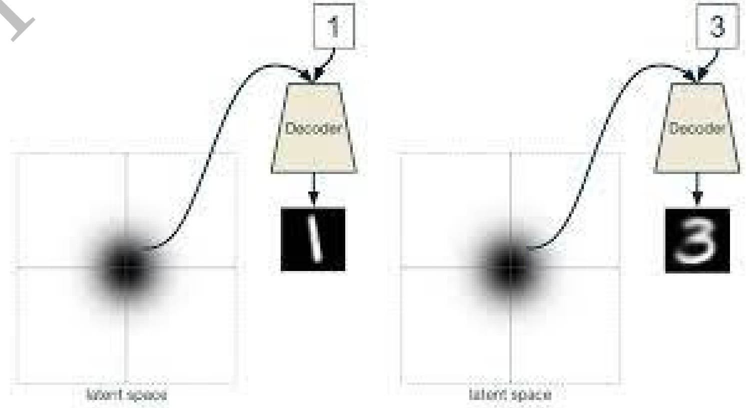
conditioned generation

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$
- Training
  - Easy on supervised data
    - Standard VAE training for $q(z|x; \phi)$
    - Cross-entropy loss on $q(y|x; \phi)$ on labeled data
  - What about unlabeled data?
    - How to train $q(y|x; \phi)$?



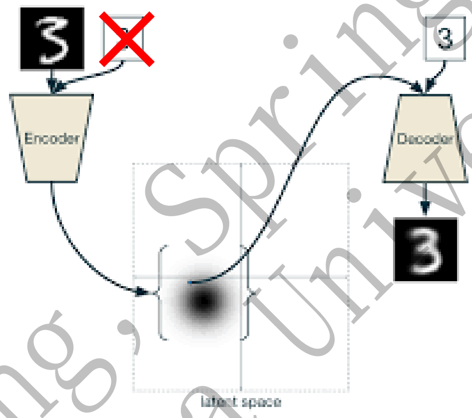
conditioned VAE



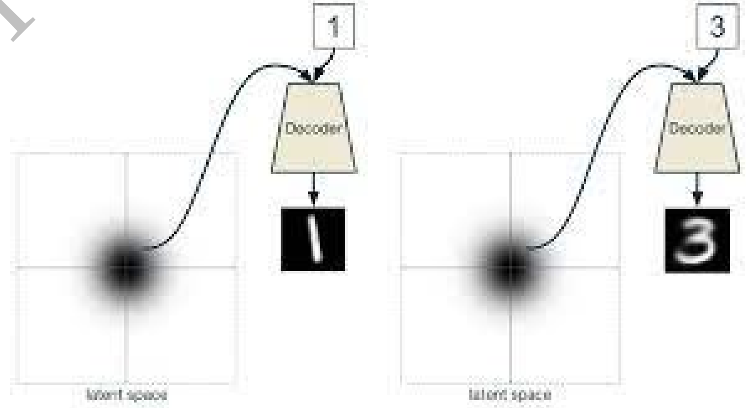
conditioned generation

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$
- Training on unlabeled data D_u
  - Loss = reconstruction + KL penalty
  - KL penalty: $KL(q(z)||p(z)) + KL(q(y)||p(y))$ ($p(y)$~uniform)
  - Reconstruction loss: $L = E_{z,y\sim q(z,y)}[\log p(x|z, y; \theta)]$



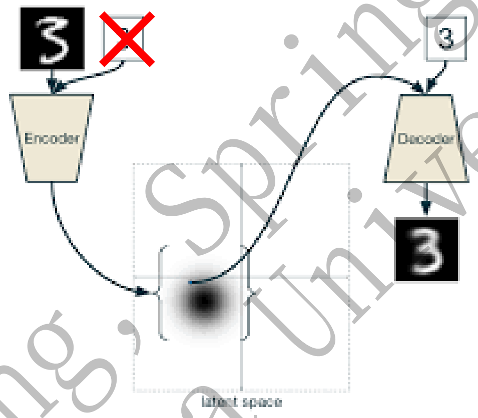
conditioned VAE



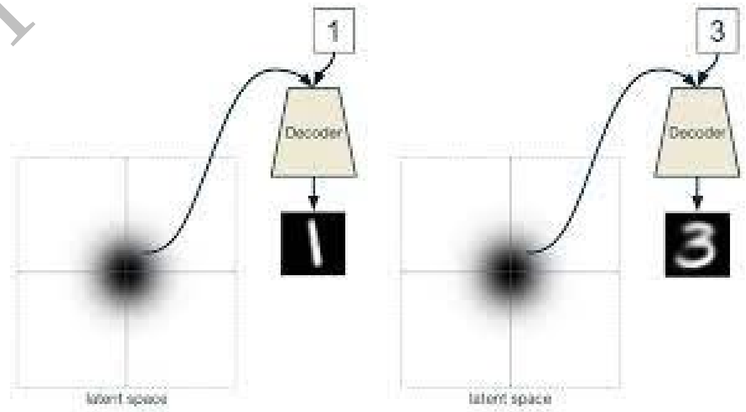
conditioned generation

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$
- Training on unlabeled data D_u
  - Loss = reconstruction + KL penalty
  - KL penalty: $KL(q(z)||p(z)) + KL(q(y)||p(y))$ ($p(y)$~uniform)
  - Reconstruction loss: $L = E_{\epsilon\sim N(0,I), y\sim q(y)}[\log p(x|\mu(x) + \sigma(x) \cdot \epsilon, y; \theta)]$
    - Reparameterization trick for z



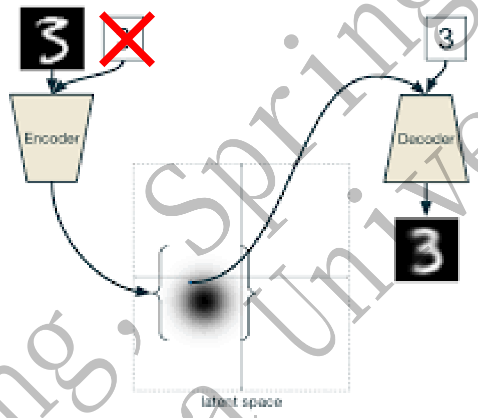
conditioned VAE



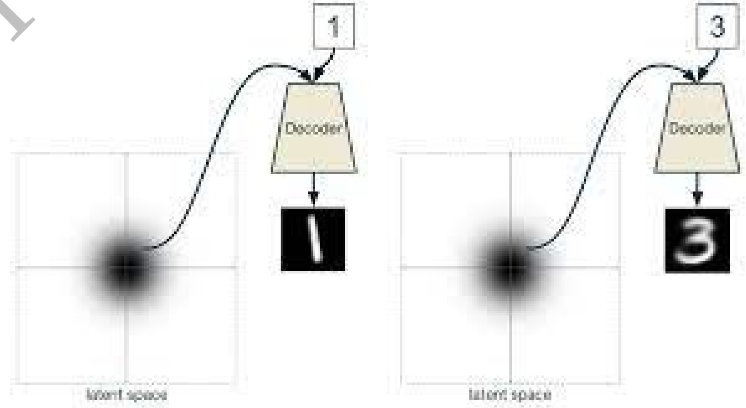
conditioned generation

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$



conditioned VAE

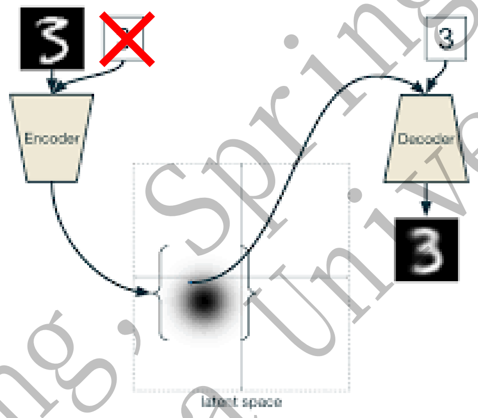


conditioned generation

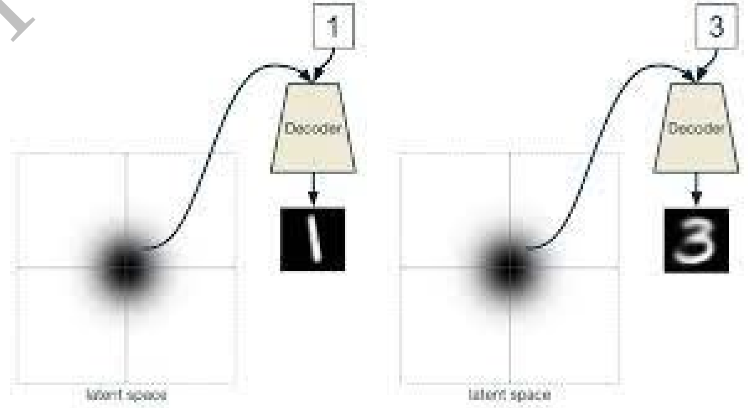
- Training on unlabeled data D_u
  - Loss = reconstruction + KL penalty
  - KL penalty: $KL(q(z)||p(z)) + KL(q(y)||p(y))$ ($p(y)$~uniform)
  - Reconstruction loss: $L = E_{\epsilon\sim N(0,I), y\sim q(y)}[\log p(x|\mu(x) + \sigma(x)\cdot\epsilon, y; \theta)]$
    - Reparameterization trick for z
    - What about y ?
      - …. although we do have tricks in lecture 11 :P

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$



conditioned VAE

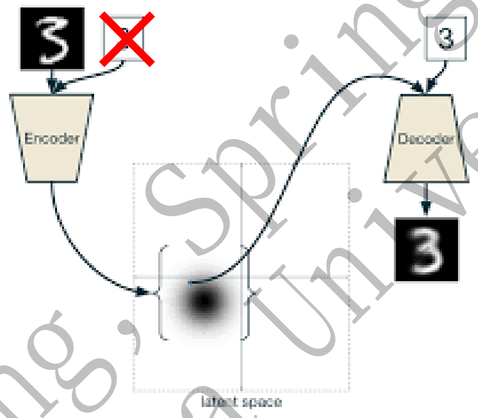


conditioned generation

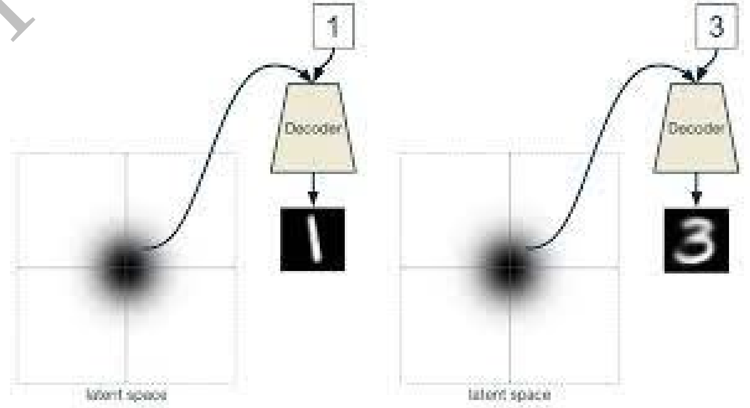
- Training on unlabeled data D_u
  - Loss = reconstruction + KL penalty
  - KL penalty: $KL(q(z)||p(z)) + KL(q(y)||p(y))$ ($p(y)$~uniform)
  - Reconstruction loss: $L = E_{\epsilon \sim N(0,I), y \sim q(y)}[\log p(x|\mu(x) + \sigma(x) \cdot \epsilon, y; \theta)]$
    - Reparameterization trick for z
    - **We only have a few labels! Expand the expectation!**

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$
- Training on unlabeled data D_u
  - Loss = reconstruction + KL penalty
  - KL penalty: $KL(q(z)||p(z)) + KL(q(y)||p(y))$ ($p(y)$~uniform)
  - Reconstruction loss:
    - $L = E_{\epsilon\sim N(0,I), y\sim q(y)}[\log p(x|\mu(x) + \sigma(x) \cdot \epsilon, y; \theta)]$
    - $\quad = E_{\epsilon\sim N(0,I)}[\sum_c q(y = c) \cdot \log p(x|\mu(x) + \sigma(x) \cdot \epsilon, y; \theta)]$



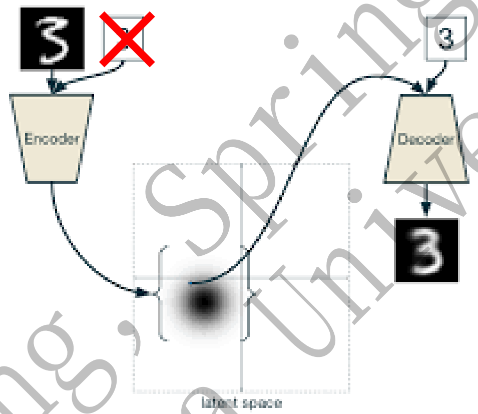
conditioned VAE



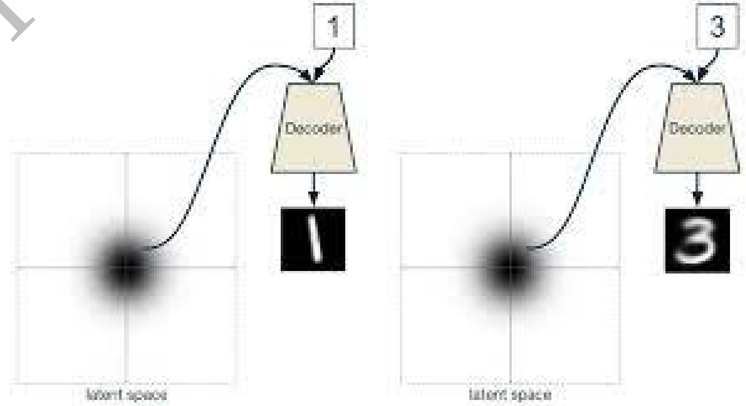
conditioned generation

Conditioned VAE

- Semi-supervised learning
  - $D_l = \{(x^{(i)}, y^{(i)})\}$
  - $D_u = \{(x^{(i)})\}$
  - Decoder: $p(x|y, z; \theta)$
  - Encoder: $q(z, y|x; \phi)$
- Training on the entire dataset D
  - Supervised loss L^l
    - Cross entropy for $q(y)$; VAE loss for $q(z)$& $p(x|z, y)$
  - Unsupervised loss L^u
    - Expanded likelihood over y for reconstruction loss
  - Combined loss: $J(\theta, \phi) = L^l + \beta L^u$
    - Leverage massive unlabeled data!



conditioned VAE



conditioned generation

Semi-supervised Learning with Deep Generative Models

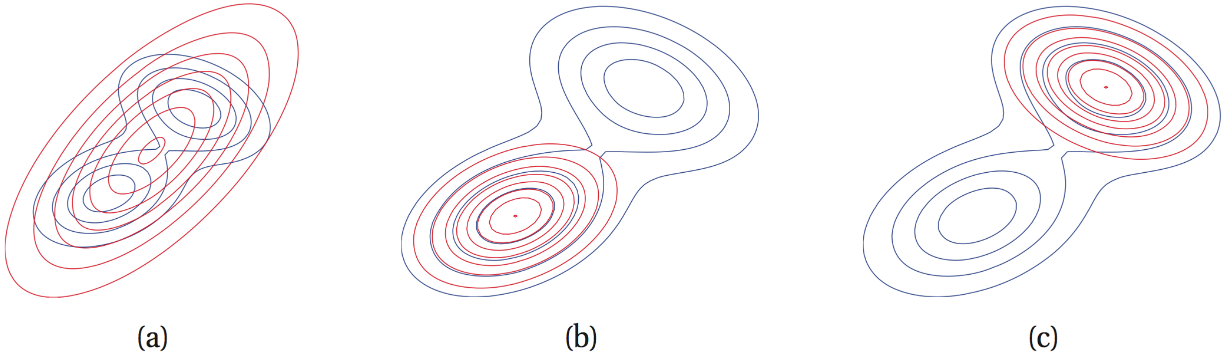
Diederik P. Kingma*, Danilo J. Rezende†, Shakir Mohamed†, Max Welling*
*Machine Learning Group, Univ. of Amsterdam, {D.P.Kingma, M.Welling}@uva.nl
†Google Deepmind, {danilor, shakir}@google.com

Variational Autoencoder: Summary

- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference

Variational Autoencoder: Summary

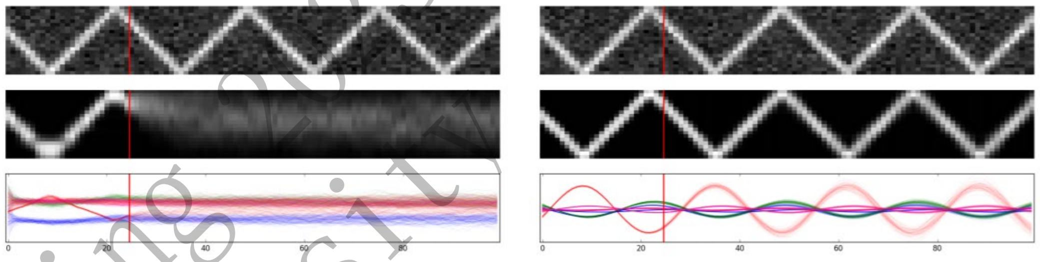
- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference
    - Intrinsic issue of KL divergence in VI
      - KL is asymmetric
        - VI: $KL(q||p) = \sum_z q(z) \log \frac{q(z)}{p(z)}$
      - $KL(q||p)$: reverse (exclusive) KL
      - $KL(p||q)$: forward (inclusive) KL
      - The mode collapse issue
        - Use forward KL?
        - Further reading of interest
        - https://arxiv.org/abs/2202.01841



inclusive KL   exclusive KL

Variational Autoencoder:

- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference
    - Intrinsic issue of KL divergence in VI
    - Assumed density of $q(z|x)$ & $p(z)$
      - $p(z) \sim N(0, I)$ for computation reason
        - We can have a more powerful prior (later in lecture 10)
        - E.g., structured VAE; VQ-VAE-2
      - $q(z|x) \sim N(\mu(x), \Sigma(x))$
        - What if $p(z|x)$ is multi-modal?
        - We need a more powerful proposal distribution
          - E.g., flow models as $q(z)$ (lecture 7(a))



(a) Predictions after 200 training epochs. (b) Predictions after 1100 training epochs.

Structured VAE
https://arxiv.org/abs/1603.06277

h_{top} | $h_{\text{top}}, h_{\text{middle}}$ | $h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$ | Original

VQ-VAE-2
https://arxiv.org/abs/1906.00446

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

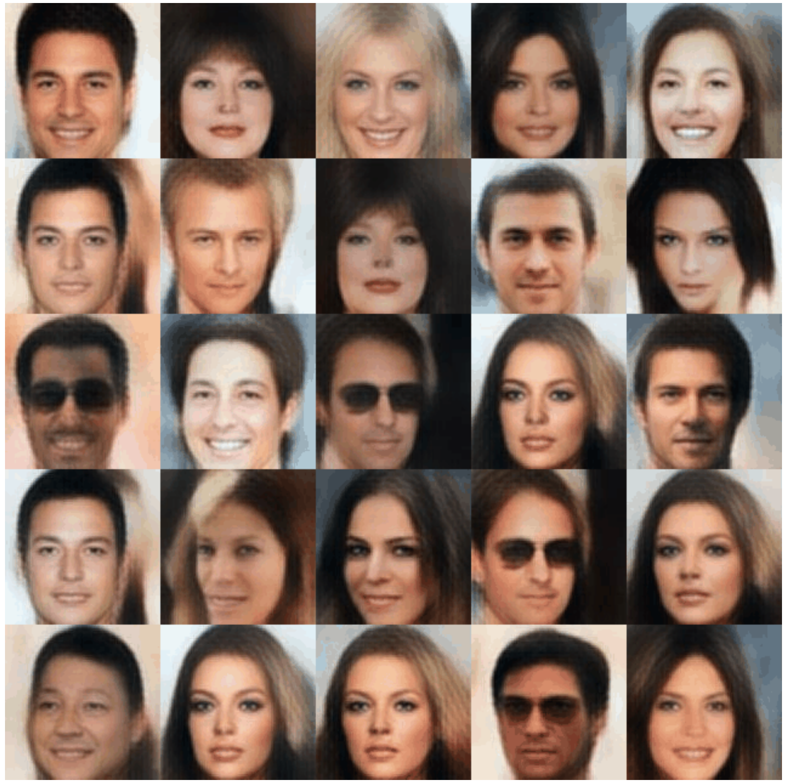
https://arxiv.org/abs/1505.05770

Variational Autoencoder: Summary

- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference
    - Intrinsic issue of KL divergence
    - Assumed density of $q(z)$ & $p(z)$
    - Variance due to single-step sampling
      - Importance-weighted autoencoder (Burda, Grosse & Ruslan, ICLR16)
        - https://arxiv.org/abs/1509.00519
      - Use more than one samples from $q(z|x)$ for a tighter lower-bound

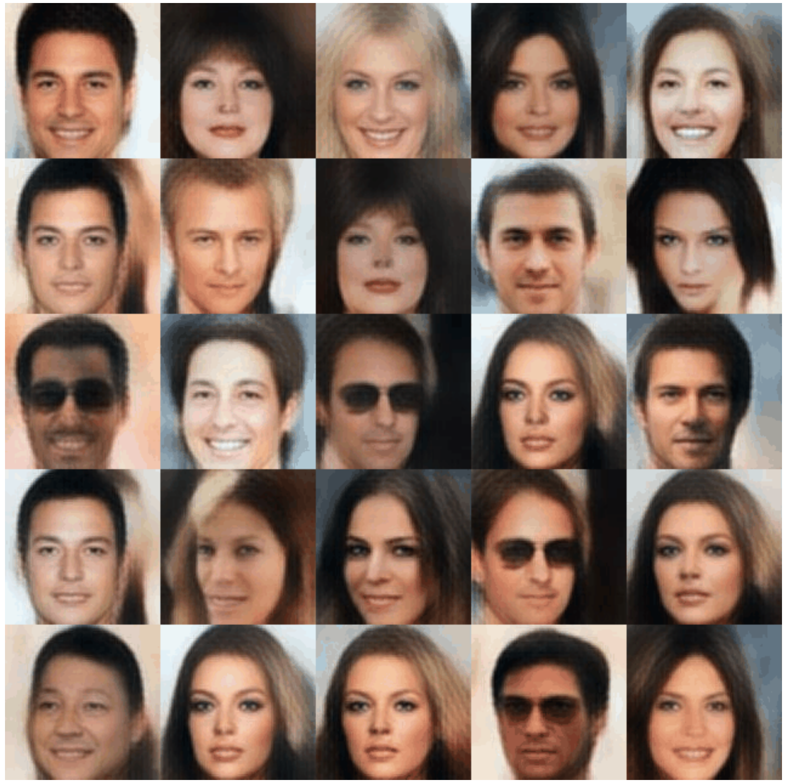
Variational Autoencoder: Summary

- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference
    - Intrinsic issue of KL divergence
    - Assumed density of $q(z)$ & $p(z)$
    - Variance due to single-step sampling
    - MLE as the reconstruction loss
      - $p(x|z;\theta) = N\left(\text{diag}\left(f(z;\theta)\right), I\right)$
      - Blurry samples!
        - Improve the decoder architecture (lecture 9)
        - Balancing the KL penalty and reconstruction loss
        - Gaussian latent to discrete latent (in lecture 11)
        - Change the loss! (next lecture ☺)



Variational Autoencoder: Summary

- Pros
  - Flexible architecture & stable training
- Cons
  - Approximate inference
    - Intrinsic issue of KL divergence
    - Assumed density of $q(z)$ & $p(z)$
    - Variance due to single-step sampling
    - MLE as the reconstruction loss
      - $p(x|z;\theta) = N(\text{diag}(f(z;\theta)), I)$
      - Blurry samples!
        - Improve the decoder architecture(lecture 9)
        - Balancing the KL penalty and reconstruction loss
        - Gaussian latent to discrete latent (in lecture 10)
        - Change the loss! (next lecture ☺)



VAE Variants

- VAE Objective (ELBO)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

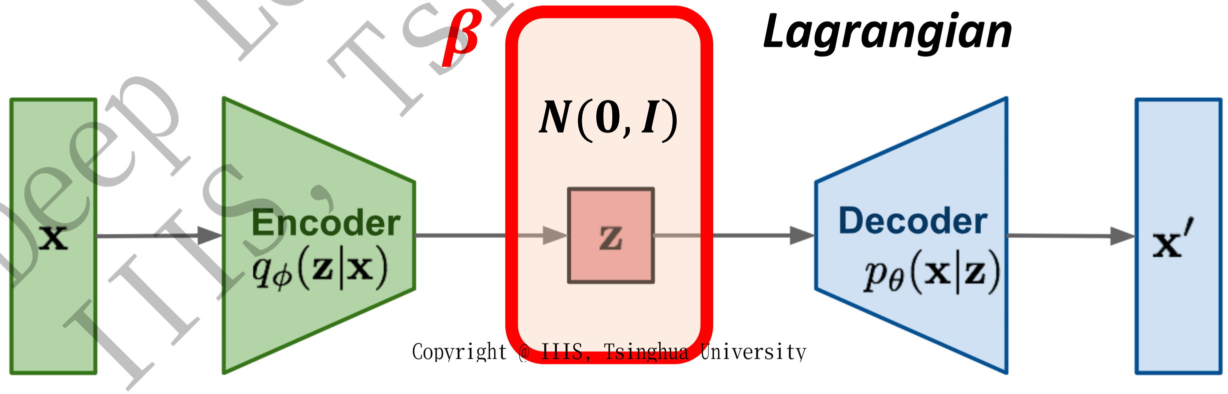
VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \boldsymbol{\beta}\, \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

- Interpretation

$$\max_{\theta,\phi} E_{x\sim D}\left[E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]\right]$$
$$\text{subject to } KL(q(z|x;\phi)||p(z)) < \epsilon$$



VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \boldsymbol{\beta}\, \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

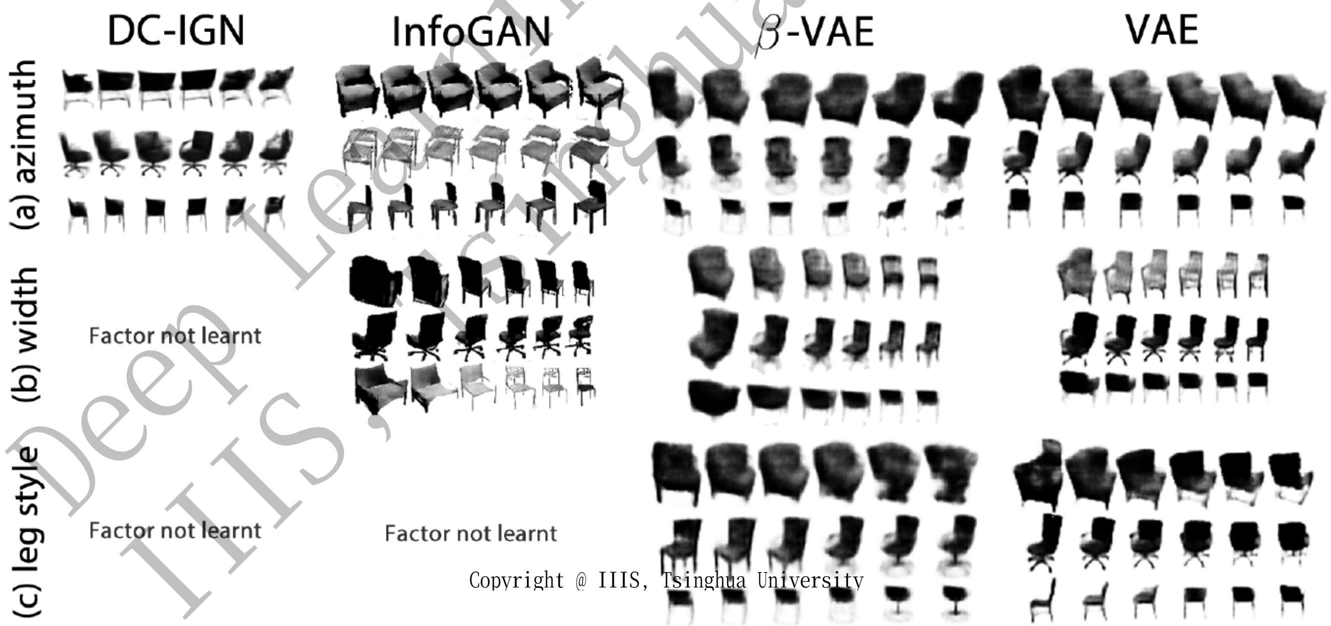
- Special cases
  - $\beta = 0$: standard AE
  - $\beta = 1$: standard VAE
  - $\beta > 1$: force the latent space closer to isomorphic Gaussian
    - Insight: each dimension of z are forced to be independent
    - Disentangle factors!

VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \boldsymbol{\beta}\, \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

- Learned factors in z



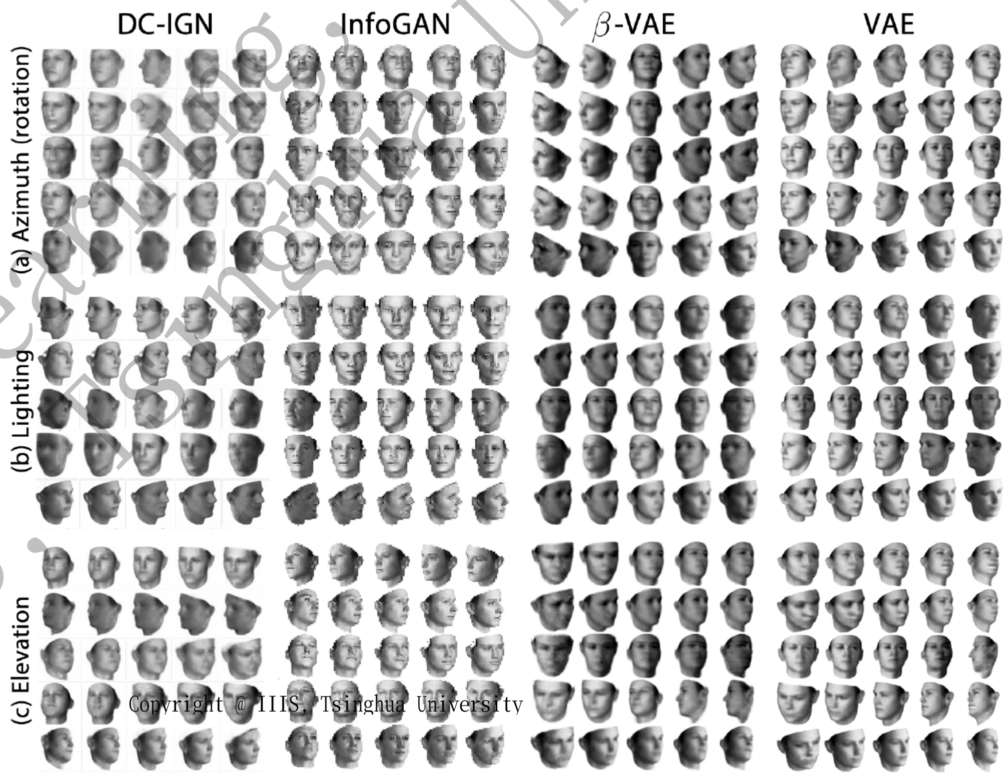
VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta, \phi; x) = E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - \boldsymbol{\beta}\, KL(q(z|x;\phi)||p(z;\theta))$$

Reconstruction KL penalty

- Learned factors in z



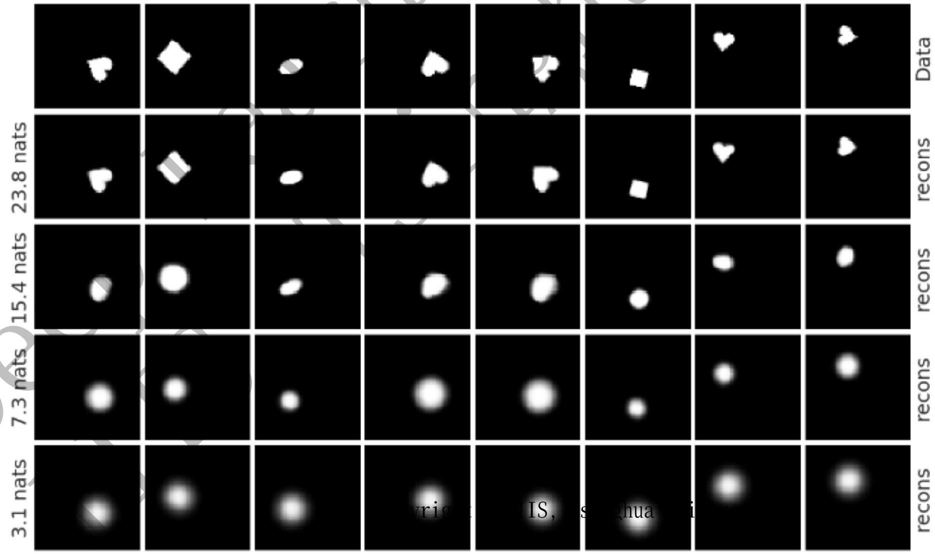
VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \boldsymbol{\beta}\, \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

- Learned factors in z
  - Trade-off between reconstruction and disentangle features!

$KL(q||N(0,I))$



***β* can be critical!**

VAE Variants

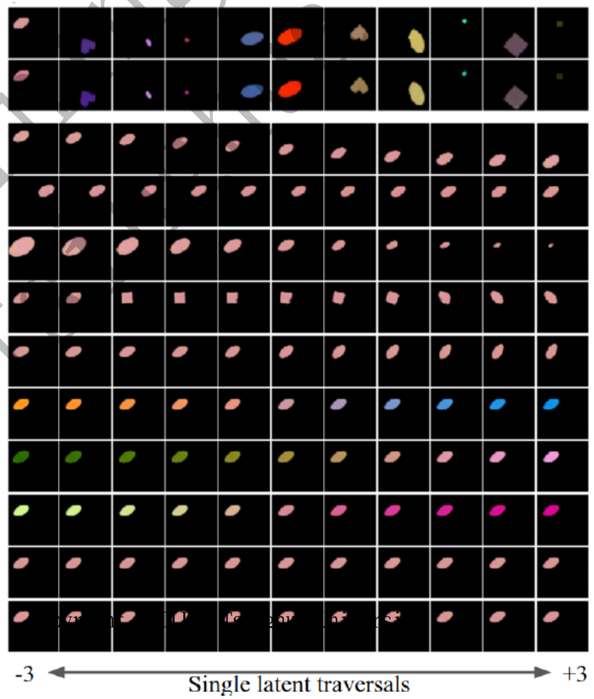
- Understanding disentangling in β-VAE (DeepMind, NIPS 2017)

$$J(\theta,\phi;x) = E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)] - \boldsymbol{\beta}\,|KL(q(z|x;\phi)||p(z;\theta)) - \boldsymbol{C}|$$

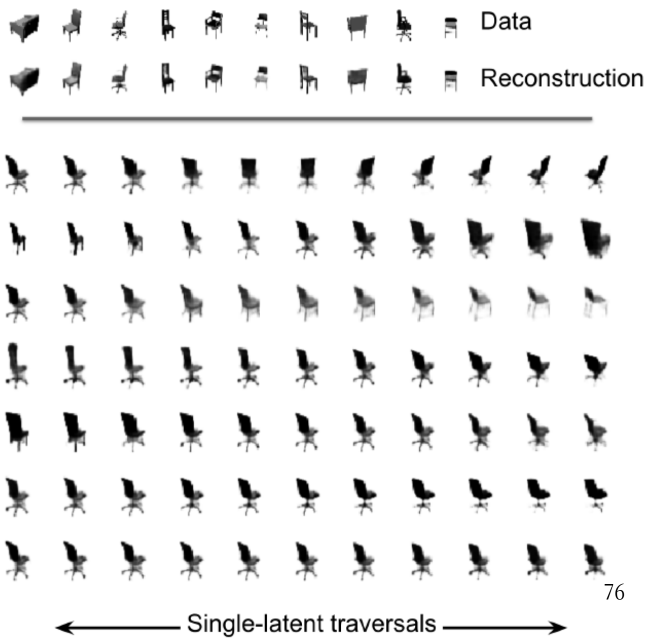
Reconstruction KL penalty **Controlled capacity**

- Learned factors in z
  - Gradually *increase* C!

(a) Coloured dSprites



(b) 3D Chairs



VAE Variants

- β-VAE (Higgins et. al, DeepMind, ICLR 2017)

$$J(\theta,\phi;x) = \underbrace{E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} - \boldsymbol{\beta}\, \underbrace{KL(q(z|x;\phi)||p(z;\theta))}_{\text{KL penalty}}$$

- Learned factors in z
  - A popular (unsupervised) approach for pretraining features
- No free lunch!
  - Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations, (Google Brain, ICML2019)
  - Disentangle features are fundamentally **impossible** without supervision or model inductive bias
    - Inductive bias or supervision is important (structured model)
    - Empirical successes can be highly random …
      - **Tune your model and your baseline hard!**

Summary

- Generative Model
  - Learn a probability distribution $p(x;\theta)$
  - Energy-based model: $p(x) = \frac{1}{Z}\exp(-E(x;\theta))$
  - Latent variable model: $p(x,z) = p(x|z)p(z)$
- Variational Autoencoder
  - A computation-efficient design of $p(x,z)$
    - Isomorphic Gaussian wherever possible
  - Variational inference for efficient and stable learning
    - ELBO & reparameterization trick
  - Flexible framework with nice mathematical property
    - But may suffer from blurry outputs… (next lecture!)

Thanks