

Deep Generative Models

Lecture 4

Roman Isachenko

Moscow Institute of Physics and Technology
Yandex School of Data Analysis

2025, Autumn

Recap of Previous Lecture

Posterior Distribution (Bayes' Theorem)

$$p(\theta|\mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\theta)p(\theta)}{\int p(\mathbf{x}|\theta)p(\theta)d\theta}$$

- ▶ \mathbf{x} – observed variables;
- ▶ θ – unobserved variables (latent parameters);
- ▶ $p(\mathbf{x}|\theta)$ – likelihood;
- ▶ $p(\mathbf{x}) = \int p(\mathbf{x}|\theta)p(\theta)d\theta$ – evidence;
- ▶ $p(\theta)$ – prior distribution;
- ▶ $p(\theta|\mathbf{x})$ – posterior distribution.

Recap of Previous Lecture

Latent Variable Models (LVM)

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})p(\mathbf{z})d\mathbf{z}.$$

MLE Problem for LVM

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{X}|\boldsymbol{\theta}) = \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log p(\mathbf{x}_i|\boldsymbol{\theta}) = \\ &= \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log \int p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta})p(\mathbf{z}_i)d\mathbf{z}_i.\end{aligned}$$

Naive Monte Carlo Estimation

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})p(\mathbf{z})d\mathbf{z} = \mathbb{E}_{p(\mathbf{z})} p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) \approx \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{z}_k, \boldsymbol{\theta}),$$

where $\mathbf{z}_k \sim p(\mathbf{z})$.

Recap of Previous Lecture

ELBO Derivation 1 (Inequality)

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} \geq \mathbb{E}_q \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} = \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$$

ELBO Derivation 2 (Equality)

$$\begin{aligned}\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) &= \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) p(\mathbf{x}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \\ &= \log p(\mathbf{x}|\boldsymbol{\theta}) - \text{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}))\end{aligned}$$

Variational Decomposition

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) + \text{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \geq \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}).$$

Recap of Previous Lecture

Variational Evidence Lower Bound (ELBO)

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) + \text{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \geq \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}).$$

$$\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) = \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$$

Log-likelihood Decomposition

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \text{KL}(q(\mathbf{z})\|p(\mathbf{z})) + \text{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

- ▶ Rather than maximizing likelihood, maximize the ELBO:

$$\max_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta}) \quad \rightarrow \quad \max_{q, \boldsymbol{\theta}} \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x})$$

- ▶ Maximizing the ELBO with respect to the variational distribution q is equivalent to minimizing the KL divergence:

$$\arg \max_q \mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) \equiv \arg \min_q \text{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

Recap of Previous Lecture

$$\begin{aligned}\mathcal{L}_{q,\theta}(\mathbf{x}) &= \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z})\|p(\mathbf{z})) = \\ &= \mathbb{E}_q \left[\log p(\mathbf{x}|\mathbf{z}, \theta) - \log \frac{q(\mathbf{z})}{p(\mathbf{z})} \right] d\mathbf{z} \rightarrow \max_{q,\theta}.\end{aligned}$$

EM Algorithm (Block-Coordinate Optimization)

- ▶ Initialize θ^* ;
- ▶ **E-step:** $(\mathcal{L}_{q,\theta}(\mathbf{x}) \rightarrow \max_q)$

$$\begin{aligned}q^*(\mathbf{z}) &= \arg \max_q \mathcal{L}_{q,\theta^*}(\mathbf{x}) = \\ &= \arg \min_q \text{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x}, \theta^*)) = p(\mathbf{z}|\mathbf{x}, \theta^*);\end{aligned}$$

- ▶ **M-step:** $(\mathcal{L}_{q,\theta}(\mathbf{x}) \rightarrow \max_\theta)$
$$\theta^* = \arg \max_\theta \mathcal{L}_{q^*,\theta}(\mathbf{x});$$

- ▶ Repeat E-step and M-step until convergence.

Recap of Previous Lecture

EM-Algorithm

- ▶ E-Step:

$$q^*(\mathbf{z}) = \arg \max_q \mathcal{L}_{q, \theta^*}(\mathbf{x}) = \arg \min_q \text{KL}(q(\mathbf{z}) \| p(\mathbf{z} | \mathbf{x}, \theta^*));$$

- ▶ M-Step:

$$\theta^* = \arg \max_{\theta} \mathcal{L}_{q^*, \theta}(\mathbf{x});$$

Amortized Variational Inference

Restrict the family of possible distributions $q(\mathbf{z})$ to a parameterized class $q(\mathbf{z} | \mathbf{x}, \phi)$, conditioned on samples \mathbf{x} and defined by ϕ .

Variational Bayes

- ▶ E-Step:

$$\phi_k = \phi_{k-1} + \eta \cdot \nabla_{\phi} \mathcal{L}_{\phi, \theta_{k-1}}(\mathbf{x}) \big|_{\phi=\phi_{k-1}}$$

- ▶ M-Step:

$$\theta_k = \theta_{k-1} + \eta \cdot \nabla_{\theta} \mathcal{L}_{\phi_k, \theta}(\mathbf{x}) \big|_{\theta=\theta_{k-1}}$$

Outline

1. ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

Outline

1. ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

M-step: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\theta} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z}$$

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

M-step: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\theta} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &= \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z}\end{aligned}$$

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

M-step: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\theta} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &= \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &\approx \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}^*, \theta), \quad \mathbf{z}^* \sim q(\mathbf{z}|\mathbf{x}, \phi).\end{aligned}$$

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

M-step: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\theta} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &= \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &\approx \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}^*, \theta), \quad \mathbf{z}^* \sim q(\mathbf{z}|\mathbf{x}, \phi).\end{aligned}$$

Naive Monte Carlo Estimation

$$p(\mathbf{x}|\theta) = \int p(\mathbf{x}|\mathbf{z}, \theta) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{z}_k, \theta), \quad \mathbf{z}_k \sim p(\mathbf{z}).$$

ELBO Gradients: M-Step ($\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\mathcal{L}_{q, \theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

M-step: $\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\theta} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &= \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \\ &\approx \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}^*, \theta), \quad \mathbf{z}^* \sim q(\mathbf{z}|\mathbf{x}, \phi).\end{aligned}$$

Naive Monte Carlo Estimation

$$p(\mathbf{x}|\theta) = \int p(\mathbf{x}|\mathbf{z}, \theta) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{z}_k, \theta), \quad \mathbf{z}_k \sim p(\mathbf{z}).$$

The variational posterior $q(\mathbf{z}|\mathbf{x}, \phi)$ typically concentrates more probability mass in a much smaller region than the prior $p(\mathbf{z})$.

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

E-step: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

Unlike the M-step, the density $q(\mathbf{z}|\mathbf{x}, \phi)$ now depends on ϕ , so standard Monte Carlo estimation can't be applied:

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

E-step: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

Unlike the M-step, the density $q(\mathbf{z}|\mathbf{x}, \phi)$ now depends on ϕ , so standard Monte Carlo estimation can't be applied:

$$\begin{aligned}\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) \\ &\neq \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))\end{aligned}$$

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

E-step: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

Unlike the M-step, the density $q(\mathbf{z}|\mathbf{x}, \phi)$ now depends on ϕ , so standard Monte Carlo estimation can't be applied:

$$\begin{aligned}\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) \\ &\neq \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))\end{aligned}$$

Reparametrization Trick (LOTUS Trick)

Assume $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)$ is generated by a random variable $\epsilon \sim p(\epsilon)$ via a deterministic mapping $\mathbf{z} = \mathbf{g}_{\phi}(\mathbf{x}, \epsilon)$. Then,

$$\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)} \mathbf{f}(\mathbf{z}) = \mathbb{E}_{\epsilon \sim p(\epsilon)} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon))$$

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

E-step: $\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$

Unlike the M-step, the density $q(\mathbf{z}|\mathbf{x}, \phi)$ now depends on ϕ , so standard Monte Carlo estimation can't be applied:

$$\begin{aligned}\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) \\ &\neq \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))\end{aligned}$$

Reparametrization Trick (LOTUS Trick)

Assume $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)$ is generated by a random variable $\epsilon \sim p(\epsilon)$ via a deterministic mapping $\mathbf{z} = \mathbf{g}_{\phi}(\mathbf{x}, \epsilon)$. Then,

$$\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)} \mathbf{f}(\mathbf{z}) = \mathbb{E}_{\epsilon \sim p(\epsilon)} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon))$$

Note: The LHS expectation is with respect to the parametric distribution $q(\mathbf{z}|\mathbf{x}, \phi)$, while the RHS is for the non-parametric $p(\epsilon)$.

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

Reparametrization Trick (LOTUS Trick)

$$\nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \mathbf{f}(\mathbf{z}) d\mathbf{z} = \nabla_{\phi} \int p(\epsilon) \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon$$

,

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

Reparametrization Trick (LOTUS Trick)

$$\begin{aligned}\nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \mathbf{f}(\mathbf{z}) d\mathbf{z} &= \nabla_{\phi} \int p(\epsilon) \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon = \\ &= \int p(\epsilon) \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon^*)),\end{aligned}$$

where $\epsilon^* \sim p(\epsilon)$.

ELBO Gradients: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

Reparametrization Trick (LOTUS Trick)

$$\begin{aligned}\nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \mathbf{f}(\mathbf{z}) d\mathbf{z} &= \nabla_{\phi} \int p(\epsilon) \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon = \\ &= \int p(\epsilon) \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \nabla_{\phi} \mathbf{f}(\mathbf{g}_{\phi}(\mathbf{x}, \epsilon^*)),\end{aligned}$$

where $\epsilon^* \sim p(\epsilon)$.

Variational Assumption

$$p(\epsilon) = \mathcal{N}(0, \mathbf{I}); \quad \mathbf{z} = \mathbf{g}_{\phi}(\mathbf{x}, \epsilon) = \sigma_{\phi}(\mathbf{x}) \odot \epsilon + \mu_{\phi}(\mathbf{x});$$

$$q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mu_{\phi}(\mathbf{x}), \sigma_{\phi}^2(\mathbf{x})).$$

Here, $\mu_{\phi}(\cdot)$ and $\sigma_{\phi}(\cdot)$ are parameterized functions (outputs of a neural network).

Thus, we can write $q(\mathbf{z}|\mathbf{x}, \phi) = \text{NN}_e(\mathbf{x}, \phi)$, the **encoder**.

ELBO Gradient: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

ELBO Gradient: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

Reconstruction Term

$$\begin{aligned} \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} &= \int p(\epsilon) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{g}_{\phi}(\mathbf{x}, \epsilon), \theta) d\epsilon \approx \\ &\approx \nabla_{\phi} \log p(\mathbf{x}|\sigma_{\phi}(\mathbf{x}) \odot \epsilon^* + \mu_{\phi}(\mathbf{x}), \theta), \quad \text{where } \epsilon^* \sim \mathcal{N}(0, \mathbf{I}) \end{aligned}$$

ELBO Gradient: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

Reconstruction Term

$$\begin{aligned} \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} &= \int p(\epsilon) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{g}_{\phi}(\mathbf{x}, \epsilon), \theta) d\epsilon \approx \\ &\approx \nabla_{\phi} \log p(\mathbf{x}|\sigma_{\phi}(\mathbf{x}) \odot \epsilon^* + \mu_{\phi}(\mathbf{x}), \theta), \quad \text{where } \epsilon^* \sim \mathcal{N}(0, \mathbf{I}) \end{aligned}$$

The generative distribution $p(\mathbf{x}|\mathbf{z}, \theta)$ can be implemented as a neural network.

We may write $p(\mathbf{x}|\mathbf{z}, \theta) = \text{NN}_d(\mathbf{z}, \theta)$, called the **decoder**.

KL Term

$p(\mathbf{z})$ is the prior over latents \mathbf{z} , typically $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

$$\nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) = \nabla_{\phi} \text{KL}(\mathcal{N}(\mu_{\phi}(\mathbf{x}), \sigma_{\phi}^2(\mathbf{x})) \| \mathcal{N}(0, \mathbf{I}))$$

ELBO Gradient: E-Step ($\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x})$)

$$\nabla_{\phi} \mathcal{L}_{\phi, \theta}(\mathbf{x}) = \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

Reconstruction Term

$$\begin{aligned} \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} &= \int p(\epsilon) \nabla_{\phi} \log p(\mathbf{x}|\mathbf{g}_{\phi}(\mathbf{x}, \epsilon), \theta) d\epsilon \approx \\ &\approx \nabla_{\phi} \log p(\mathbf{x}|\boldsymbol{\sigma}_{\phi}(\mathbf{x}) \odot \boldsymbol{\epsilon}^* + \boldsymbol{\mu}_{\phi}(\mathbf{x}), \theta), \quad \text{where } \boldsymbol{\epsilon}^* \sim \mathcal{N}(0, \mathbf{I}) \end{aligned}$$

The generative distribution $p(\mathbf{x}|\mathbf{z}, \theta)$ can be implemented as a neural network.

We may write $p(\mathbf{x}|\mathbf{z}, \theta) = \text{NN}_d(\mathbf{z}, \theta)$, called the **decoder**.

KL Term

$p(\mathbf{z})$ is the prior over latents \mathbf{z} , typically $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

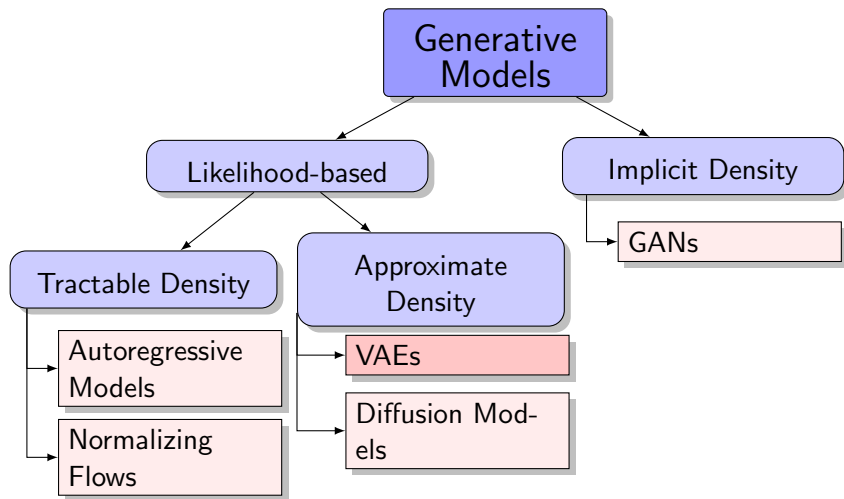
$$\nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) = \nabla_{\phi} \text{KL}(\mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{x}), \boldsymbol{\sigma}_{\phi}^2(\mathbf{x})) \| \mathcal{N}(0, \mathbf{I}))$$

This expression admits a closed-form analytic solution.

Outline

1. ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

Generative Models Zoo



Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).

Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).
- ▶ Compute the objective (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - \text{KL}(q(\mathbf{z}^*|\mathbf{x}, \phi) \| p(\mathbf{z}^*)).$$

Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).
- ▶ Compute the objective (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - \text{KL}(q(\mathbf{z}^*|\mathbf{x}, \phi) \| p(\mathbf{z}^*)).$$

- ▶ Update parameters via stochastic gradient steps with respect to ϕ and θ (as in autograd).

Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).
- ▶ Compute the objective (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - \text{KL}(q(\mathbf{z}^*|\mathbf{x}, \phi) \| p(\mathbf{z}^*)).$$

- ▶ Update parameters via stochastic gradient steps with respect to ϕ and θ (as in autograd).

Inference

- ▶ Sample \mathbf{z}^* from the prior $p(\mathbf{z})$ ($\mathcal{N}(0, \mathbf{I})$);

Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).
- ▶ Compute the objective (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - \text{KL}(q(\mathbf{z}^*|\mathbf{x}, \phi) \| p(\mathbf{z}^*)).$$

- ▶ Update parameters via stochastic gradient steps with respect to ϕ and θ (as in autograd).

Inference

- ▶ Sample \mathbf{z}^* from the prior $p(\mathbf{z})$ ($\mathcal{N}(0, \mathbf{I})$);
- ▶ Generate data from the decoder $p(\mathbf{x}|\mathbf{z}^*, \theta)$.

Variational Autoencoder (VAE)

Training (EM Algorithm)

- ▶ Select a random sample $\mathbf{x}_i, i \sim \text{Uniform}\{1, n\}$ (or a batch).
- ▶ Compute the objective (apply the reparametrization trick):

$$\epsilon^* \sim p(\epsilon); \quad \mathbf{z}^* = \mathbf{g}_\phi(\mathbf{x}, \epsilon^*);$$

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - \text{KL}(q(\mathbf{z}^*|\mathbf{x}, \phi) \| p(\mathbf{z}^*)).$$

- ▶ Update parameters via stochastic gradient steps with respect to ϕ and θ (as in autograd).

Inference

- ▶ Sample \mathbf{z}^* from the prior $p(\mathbf{z})$ ($\mathcal{N}(0, \mathbf{I})$);
- ▶ Generate data from the decoder $p(\mathbf{x}|\mathbf{z}^*, \theta)$.

Note: The encoder $q(\mathbf{z}|\mathbf{x}, \phi)$ isn't needed during generation.

Variational Autoencoder

$$\mathcal{L}_{q,\theta}(\mathbf{x}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z}))$$

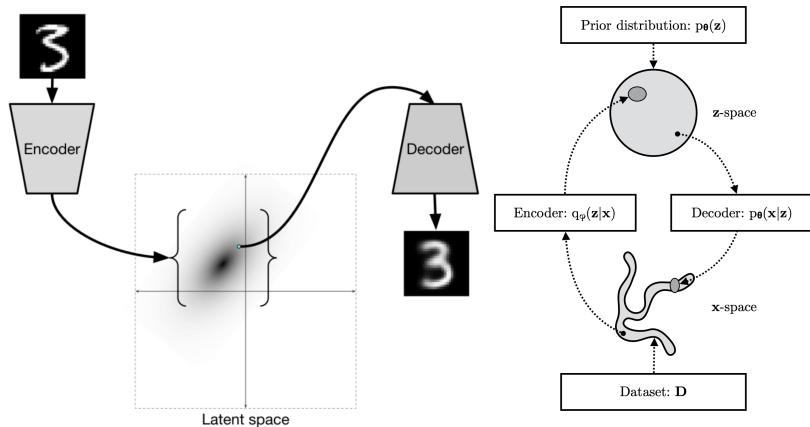
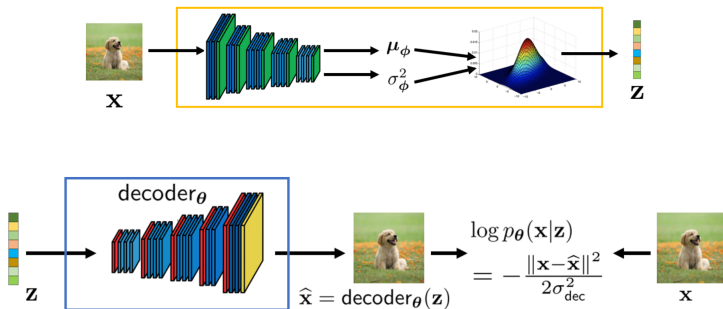


image credit: <http://ijdykeman.github.io/ml/2016/12/21/cvae.html>
Kingma D. P., Welling M., An Introduction to Variational Autoencoders, 2019

Variational Autoencoder

- ▶ The encoder $q(\mathbf{z}|\mathbf{x}, \phi) = \text{NN}_e(\mathbf{x}, \phi)$ outputs $\mu_\phi(\mathbf{x})$ and $\sigma_\phi(\mathbf{x})$.
- ▶ The decoder $p(\mathbf{x}|\mathbf{z}, \theta) = \text{NN}_d(\mathbf{z}, \theta)$ outputs parameters of the observed data distribution.



VAE vs Normalizing Flows

	VAE	NF
Objective	ELBO \mathcal{L}	Forward KL/MLE
Encoder	stochastic $\mathbf{z} \sim q(\mathbf{z} \mathbf{x}, \phi)$	deterministic $\mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x})$ $q(\mathbf{z} \mathbf{x}, \theta) = \delta(\mathbf{z} - \mathbf{f}_{\theta}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p(\mathbf{x} \mathbf{z}, \theta)$	deterministic $\mathbf{x} = \mathbf{g}_{\theta}(\mathbf{z})$ $p(\mathbf{x} \mathbf{z}, \theta) = \delta(\mathbf{x} - \mathbf{g}_{\theta}(\mathbf{z}))$
Parameters	ϕ, θ	$\theta \equiv \phi$

VAE vs Normalizing Flows

	VAE	NF
Objective	ELBO \mathcal{L}	Forward KL/MLE
Encoder	stochastic $\mathbf{z} \sim q(\mathbf{z} \mathbf{x}, \phi)$	deterministic $\mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x})$ $q(\mathbf{z} \mathbf{x}, \theta) = \delta(\mathbf{z} - \mathbf{f}_{\theta}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p(\mathbf{x} \mathbf{z}, \theta)$	deterministic $\mathbf{x} = \mathbf{g}_{\theta}(\mathbf{z})$ $p(\mathbf{x} \mathbf{z}, \theta) = \delta(\mathbf{x} - \mathbf{g}_{\theta}(\mathbf{z}))$
Parameters	ϕ, θ	$\theta \equiv \phi$

Theorem

MLE for a normalizing flow is equivalent to maximizing the ELBO for a VAE where:

$$p(\mathbf{x}|\mathbf{z}, \theta) = \delta(\mathbf{x} - \mathbf{f}_{\theta}^{-1}(\mathbf{z})) = \delta(\mathbf{x} - \mathbf{g}_{\theta}(\mathbf{z}));$$

$$q(\mathbf{z}|\mathbf{x}, \theta) = \delta(\mathbf{z} - \mathbf{f}_{\theta}(\mathbf{x})).$$

Nielsen D., et al., *SurVAE Flows: Surjections to Bridge the Gap Between VAEs and Flows*, 2020

Outline

1. ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

Discrete VAE Latents

Motivation

- ▶ Previous VAE models have used **continuous** latent variables \mathbf{z} .
- ▶ For some modalities, **discrete** representations \mathbf{z} may be a more natural choice.
- ▶ Advanced autoregressive models (e.g., PixelCNN) are highly effective for distributions over discrete variables.
- ▶ Current transformer-like models process discrete tokens.

Discrete VAE Latents

Motivation

- ▶ Previous VAE models have used **continuous** latent variables \mathbf{z} .
- ▶ For some modalities, **discrete** representations \mathbf{z} may be a more natural choice.
- ▶ Advanced autoregressive models (e.g., PixelCNN) are highly effective for distributions over discrete variables.
- ▶ Current transformer-like models process discrete tokens.

ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) \rightarrow \max_{\phi, \theta}.$$

Discrete VAE Latents

Motivation

- ▶ Previous VAE models have used **continuous** latent variables \mathbf{z} .
- ▶ For some modalities, **discrete** representations \mathbf{z} may be a more natural choice.
- ▶ Advanced autoregressive models (e.g., PixelCNN) are highly effective for distributions over discrete variables.
- ▶ Current transformer-like models process discrete tokens.

ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) \| p(\mathbf{z})) \rightarrow \max_{\phi, \theta}.$$

- ▶ Apply the reparametrization trick to obtain unbiased gradients.
- ▶ Use Gaussian distributions for $q(\mathbf{z}|\mathbf{x}, \phi)$ and $p(\mathbf{z})$ to compute the KL analytically.

Discrete VAE Latents

Assumptions

- ▶ Let $c \sim \text{Categorical}(\boldsymbol{\pi})$, where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$

- ▶ Suppose the VAE adopts a discrete latent variable c with prior $p(c) = \text{Uniform}\{1, \dots, K\}$.

Discrete VAE Latents

Assumptions

- ▶ Let $c \sim \text{Categorical}(\boldsymbol{\pi})$, where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$

- ▶ Suppose the VAE adopts a discrete latent variable c with prior $p(c) = \text{Uniform}\{1, \dots, K\}$.

ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) - \text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) \rightarrow \max_{\phi, \theta}.$$

$$\text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) = \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log \frac{q(k|\mathbf{x}, \phi)}{p(k)}$$

Discrete VAE Latents

Assumptions

- ▶ Let $c \sim \text{Categorical}(\boldsymbol{\pi})$, where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$

- ▶ Suppose the VAE adopts a discrete latent variable c with prior $p(c) = \text{Uniform}\{1, \dots, K\}$.

ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) - \text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) \rightarrow \max_{\phi, \theta}.$$

$$\begin{aligned} \text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) &= \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log \frac{q(k|\mathbf{x}, \phi)}{p(k)} = \\ &= \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log q(k|\mathbf{x}, \phi) - \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log p(k) \end{aligned}$$

Discrete VAE Latents

Assumptions

- ▶ Let $c \sim \text{Categorical}(\boldsymbol{\pi})$, where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$

- ▶ Suppose the VAE adopts a discrete latent variable c with prior $p(c) = \text{Uniform}\{1, \dots, K\}$.

ELBO

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) - \text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) \rightarrow \max_{\phi, \theta}.$$

$$\begin{aligned} \text{KL}(q(c|\mathbf{x}, \phi) \| p(c)) &= \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log \frac{q(k|\mathbf{x}, \phi)}{p(k)} = \\ &= \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log q(k|\mathbf{x}, \phi) - \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log p(k) = \\ &= -H(q(c|\mathbf{x}, \phi)) + \log K. \end{aligned}$$

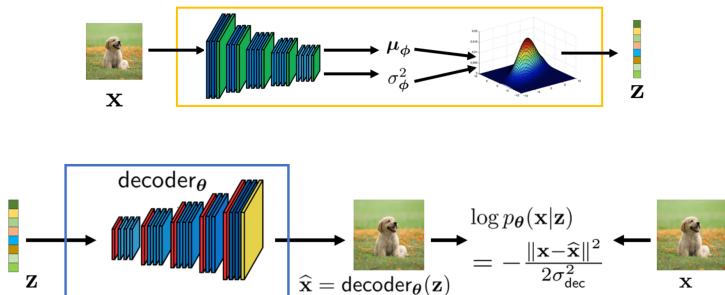
Discrete VAE Latents

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) + H(q(c|\mathbf{x}, \phi)) - \log K \rightarrow \max_{\phi, \theta}.$$

Discrete VAE Latents

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) + H(q(c|\mathbf{x}, \phi)) - \log K \rightarrow \max_{\phi, \theta}.$$

- ▶ The encoder should output a discrete distribution $q(c|\mathbf{x}, \phi)$.
- ▶ We need an analogue of the reparametrization trick for discrete $q(c|\mathbf{x}, \phi)$.
- ▶ The decoder $p(\mathbf{x}|c, \theta)$ must take a discrete random variable c as input.



Summary

- ▶ The reparametrization trick provides unbiased gradients with respect to the variational posterior $q(\mathbf{z}|\mathbf{x}, \phi)$.
- ▶ The VAE model is a latent variable model parameterized by two neural networks: a stochastic encoder $q(\mathbf{z}|\mathbf{x}, \phi)$ and a stochastic decoder $p(\mathbf{x}|\mathbf{z}, \theta)$.
- ▶ NF models can be interpreted as VAEs with deterministic encoder and decoder functions.
- ▶ Discrete VAE latents offer a natural class of latent variable models.