Deep Generative Models

Lecture 4

Roman Isachenko

Moscow Institute of Physics and Technology
Yandex School of Data Analysis

2025, Autumn

Recap of Previous Lecture

Latent Variable Models (LVM)

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

MLE Problem for LVM

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

Naive Monte Carlo Estimation

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

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

Recap of Previous Lecture

ELBO Derivation 1 (Inequality)

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

ELBO Derivation 2 (Equality)

$$\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) = \int q(\mathbf{z}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} = \int q(\mathbf{z}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) p_{\boldsymbol{\theta}}(\mathbf{x})}{q(\mathbf{z})} d\mathbf{z} =$$
$$= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - \mathrm{KL}(q(\mathbf{z}) \| p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$$

Variational Decomposition

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

Recap of Previous Lecture

Variational Evidence Lower Bound (ELBO)

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

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

Log-likelihood Decomposition

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

- Rather than maximizing likelihood, maximize the ELBO:

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

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

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

Recap of Previous Lecture

$$\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_q \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(q(\mathbf{z})\|p(\mathbf{z})) =$$
$$= \mathbb{E}_q \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right] d\mathbf{z} \to \max_{q,\boldsymbol{\theta}}.$$

EM Algorithm (Block-Coordinate Optimization)

- Initialize $\boldsymbol{\theta}^*$;
- **E-step:** ($\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) \to \max_q$)
$$q^*(\mathbf{z}) = \arg\max_q \mathcal{L}_{q,\boldsymbol{\theta}^*}(\mathbf{x}) =$$
$$= \arg\min_q \mathrm{KL}(q(\mathbf{z})\|p_{\boldsymbol{\theta}^*}(\mathbf{z}|\mathbf{x})) = p_{\boldsymbol{\theta}^*}(\mathbf{z}|\mathbf{x});$$
- **M-step:** ($\mathcal{L}_{q,\boldsymbol{\theta}}(\mathbf{x}) \to \max_{\boldsymbol{\theta}}$)
$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}_{q^*,\boldsymbol{\theta}}(\mathbf{x});$$
- Repeat E-step and M-step until convergence.

Recap of Previous Lecture

EM-Algorithm

- E-Step:

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

- M-Step:

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

Amortized Variational Inference

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

Variational Bayes

- E-Step:

$$\boldsymbol{\phi}_k = \boldsymbol{\phi}_{k-1} + \eta \cdot \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}_{k-1}}(\mathbf{x})\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_{k-1}}$$

- M-Step:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\phi}_k,\boldsymbol{\theta}}(\mathbf{x})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k-1}}$$

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 ($\nabla_{\theta}\mathcal{L}_{\phi,\theta}(\mathbf{x})$)

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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

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

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

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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

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

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

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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

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

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

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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

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

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

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

Naive Monte Carlo Estimation

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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

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

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

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

Naive Monte Carlo Estimation

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

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

image credit: https://jmtomczak.github.io/blog/4/4_VAE.html

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 $q_\phi(\mathbf{z}|\mathbf{x})$ now depends on ϕ, so standard Monte Carlo estimation can't be applied:

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

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 $q_\phi(\mathbf{z}|\mathbf{x})$ now depends on ϕ, so standard Monte Carlo estimation can't be applied:

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

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 $q_\phi(\mathbf{z}|\mathbf{x})$ now depends on ϕ, so standard Monte Carlo estimation can't be applied:

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

Reparametrization Trick (LOTUS Trick)

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

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

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 $q_\phi(\mathbf{z}|\mathbf{x})$ now depends on ϕ, so standard Monte Carlo estimation can't be applied:

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

Reparametrization Trick (LOTUS Trick)

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

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

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

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

Reparametrization Trick (LOTUS Trick)

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

,

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

Reparametrization Trick (LOTUS Trick)

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

$$= \int p(\boldsymbol{\epsilon})\nabla_\phi \mathbf{f}(\mathbf{g}_\phi(\mathbf{x}, \boldsymbol{\epsilon}))d\boldsymbol{\epsilon} \approx \nabla_\phi \mathbf{f}(\mathbf{g}_\phi(\mathbf{x}, \boldsymbol{\epsilon}^*)),$$

where $\boldsymbol{\epsilon}^* \sim p(\boldsymbol{\epsilon})$.

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

Reparametrization Trick (LOTUS Trick)

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

$$= \int p(\epsilon)\nabla_\phi \mathbf{f}(\mathbf{g}_\phi(\mathbf{x},\epsilon))d\epsilon \approx \nabla_\phi \mathbf{f}(\mathbf{g}_\phi(\mathbf{x},\epsilon^*)),$$

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

Variational Assumption

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

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x})).$$

Here, $\boldsymbol{\mu}_\phi(\cdot)$ and $\boldsymbol{\sigma}_\phi(\cdot)$ are parameterized functions (outputs of a neural network).
Thus, we can write $q_\phi(\mathbf{z}|\mathbf{x}) = \mathrm{NN}_e(\mathbf{x}, \phi)$, the **encoder**.

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

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

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

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

Reconstruction Term

$$\nabla_\phi \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) d\mathbf{z} = \int p(\boldsymbol{\epsilon}) \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{g}_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon})) d\boldsymbol{\epsilon} \approx$$

$$\approx \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}} \left(\mathbf{x} | \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x}) \odot \boldsymbol{\epsilon}^* + \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})\right), \quad \text{where } \boldsymbol{\epsilon}^* \sim \mathcal{N}(0, \mathbf{I})$$

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

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

Reconstruction Term

$$\nabla_\phi \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} = \int p(\epsilon) \nabla_\phi \log p_\theta(\mathbf{x}|\mathbf{g}_\phi(\mathbf{x}, \epsilon)) d\epsilon \approx$$

$$\approx \nabla_\phi \log p_\theta \left(\mathbf{x}|\boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}^* + \boldsymbol{\mu}_\phi(\mathbf{x})\right), \quad \text{where } \boldsymbol{\epsilon}^* \sim \mathcal{N}(0, \mathbf{I})$$

The generative distribution $p_\theta(\mathbf{x}|\mathbf{z})$ can be implemented as a neural network.
We may write $p_\theta(\mathbf{x}|\mathbf{z}) = \mathrm{NN}_d(\mathbf{z}, \boldsymbol{\theta})$, called the **decoder**.

KL Term

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

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

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

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

Reconstruction Term

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

The generative distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ can be implemented as a neural network.
We may write $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathrm{NN}_d(\mathbf{z}, \boldsymbol{\theta})$, called the **decoder**.

KL Term

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

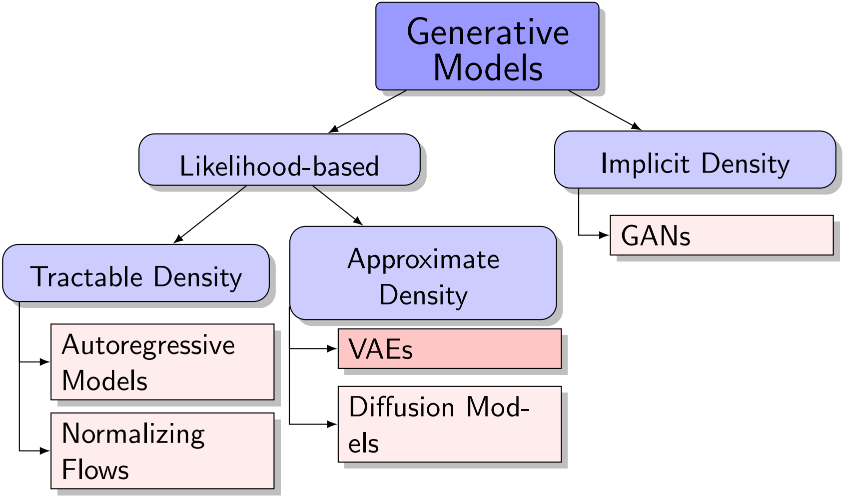
$$\nabla_{\boldsymbol{\phi}} \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) = \nabla_{\boldsymbol{\phi}} \mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x})) \| \mathcal{N}(0, \mathbf{I})\right)$$

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):

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

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

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):

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

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

- Update parameters via stochastic gradient steps with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ (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):

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

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

- Update parameters via stochastic gradient steps with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ (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):

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

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

- Update parameters via stochastic gradient steps with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ (as in autograd).

Inference

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

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):

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

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

- Update parameters via stochastic gradient steps with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ (as in autograd).

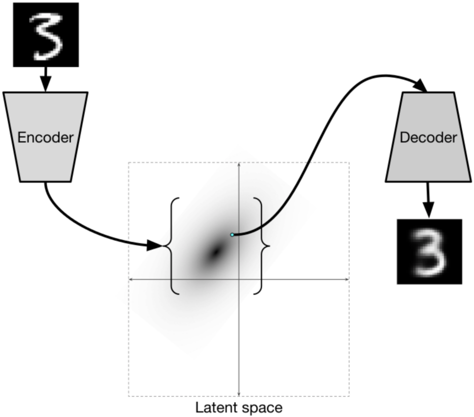
Inference

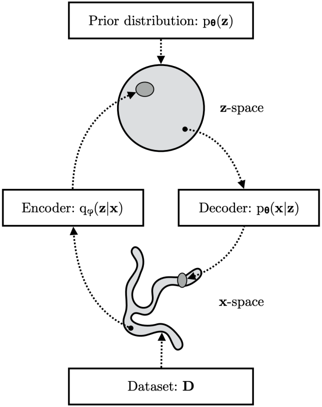
- Sample $\mathbf{z}^*$ from the prior $p(\mathbf{z})$ ($\mathcal{N}(0, \mathbf{I})$);
- Generate data from the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^*)$.

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

Variational Autoencoder

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



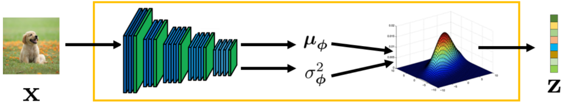


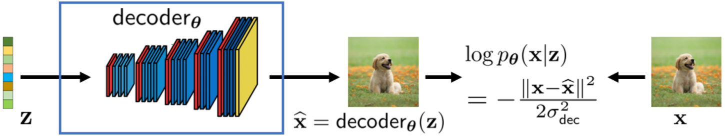
---

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 $q_\phi(\mathbf{z}|\mathbf{x}) = \mathrm{NN}_e(\mathbf{x}, \phi)$ outputs $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}_\phi(\mathbf{x})$.
- The decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathrm{NN}_d(\mathbf{z}, \boldsymbol{\theta})$ outputs parameters of the observed data distribution.





Chan S., Tutorial on Diffusion Models for Imaging and Vision, 2024

VAE vs Normalizing Flows

	VAE	**NF**
Objective	ELBO $\mathcal{L}$	Forward KL/MLE
Encoder	stochastic $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}\|\mathbf{x})$	deterministic $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ $q_{\boldsymbol{\theta}}(\mathbf{z}\|\mathbf{x}) = \delta(\mathbf{z} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}\|\mathbf{z})$	deterministic $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ $p_{\boldsymbol{\theta}}(\mathbf{x}\|\mathbf{z}) = \delta(\mathbf{x} - \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))$
Parameters	$\boldsymbol{\phi}, \boldsymbol{\theta}$	$\boldsymbol{\theta} \equiv \boldsymbol{\phi}$

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

VAE vs Normalizing Flows

	VAE	**NF**
Objective	ELBO $\mathcal{L}$	Forward KL/MLE
Encoder	stochastic $\mathbf{z} \sim q_{\phi}(\mathbf{z}\|\mathbf{x})$	deterministic $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ $q_{\boldsymbol{\theta}}(\mathbf{z}\|\mathbf{x}) = \delta(\mathbf{z} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}\|\mathbf{z})$	deterministic $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ $p_{\boldsymbol{\theta}}(\mathbf{x}\|\mathbf{z}) = \delta(\mathbf{x} - \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))$
Parameters	$\phi, \boldsymbol{\theta}$	$\boldsymbol{\theta} \equiv \phi$

Theorem

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

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

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

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

Outline

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

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

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

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

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

Discrete VAE Latents

Assumptions

- Let $c \sim \mathrm{Categorical}(\boldsymbol{\pi})$, where
$$\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^{K} \pi_k = 1.$$
- Suppose the VAE adopts a discrete latent variable c with prior $p(c) = \text{Uniform}\{1, \ldots, K\}$.

Discrete VAE Latents

Assumptions

- Let $c \sim \mathrm{Categorical}(\boldsymbol{\pi})$, 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

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(c|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|c) - \mathrm{KL}(q_{\boldsymbol{\phi}}(c|\mathbf{x}) \| p(c)) \to \max_{\boldsymbol{\phi},\boldsymbol{\theta}}.$$

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(c|\mathbf{x}) \| p(c)) = \sum_{k=1}^{K} q_{\boldsymbol{\phi}}(k|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(k|\mathbf{x})}{p(k)}$$

Discrete VAE Latents

Assumptions

- Let $c \sim \mathrm{Categorical}(\boldsymbol{\pi})$, 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

$$\mathcal{L}_{\phi,\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_\phi(c|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|c) - \mathrm{KL}(q_\phi(c|\mathbf{x}) \| p(c)) \to \max_{\phi,\boldsymbol{\theta}}.$$

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

$$= \sum_{k=1}^{K} q_\phi(k|\mathbf{x}) \log q_\phi(k|\mathbf{x}) - \sum_{k=1}^{K} q_\phi(k|\mathbf{x}) \log p(k)$$

Discrete VAE Latents

Assumptions

- Let $c \sim \mathrm{Categorical}(\boldsymbol{\pi})$, 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

$$\mathcal{L}_{\boldsymbol{\phi}, \boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(c|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|c) - \mathrm{KL}(q_{\boldsymbol{\phi}}(c|\mathbf{x}) \| p(c)) \to \max_{\boldsymbol{\phi}, \boldsymbol{\theta}}.$$

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(c|\mathbf{x}) \| p(c)) = \sum_{k=1}^{K} q_{\boldsymbol{\phi}}(k|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(k|\mathbf{x})}{p(k)} =$$

$$= \sum_{k=1}^{K} q_{\boldsymbol{\phi}}(k|\mathbf{x}) \log q_{\boldsymbol{\phi}}(k|\mathbf{x}) - \sum_{k=1}^{K} q_{\boldsymbol{\phi}}(k|\mathbf{x}) \log p(k) =$$

$$= -\mathrm{H}(q_{\boldsymbol{\phi}}(c|\mathbf{x})) + \log K.$$

Discrete VAE Latents

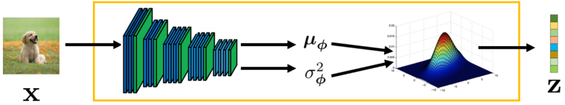
$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(c|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|c) + \mathrm{H}(q_{\boldsymbol{\phi}}(c|\mathbf{x})) - \log K \to \max_{\boldsymbol{\phi},\boldsymbol{\theta}}.$$

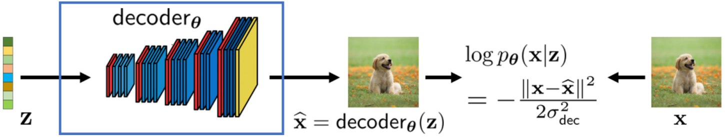
Chan S., Tutorial on Diffusion Models for Imaging and Vision, 2024

Discrete VAE Latents

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(c|\mathbf{x})} \log p_\theta(\mathbf{x}|c) + \mathrm{H}(q_\phi(c|\mathbf{x})) - \log K \to \max_{\phi,\theta}.$$

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





Chan S., Tutorial on Diffusion Models for Imaging and Vision, 2024

Summary

- The reparametrization trick provides unbiased gradients with respect to the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$.

- The VAE model is a latent variable model parameterized by two neural networks: a stochastic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and a stochastic decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

- NF models can be interpreted as VAEs with deterministic encoder and decoder functions.

- Discrete VAE latents offer a natural class of latent variable models.