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)

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

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

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

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)

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

$$= \log p(\mathbf{x}|\boldsymbol{\theta}) - \mathrm{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}))$$

Variational Decomposition

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

Recap of Previous Lecture

Variational Evidence Lower Bound (ELBO)

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

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

Log-likelihood Decomposition

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

- 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:

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

Recap of Previous Lecture

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

EM Algorithm (Block-Coordinate Optimization)

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

Outline

1. EM-Algorithm
   - Amortized Inference
   - ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

Outline

1. EM-Algorithm
   Amortized Inference
   ELBO Gradients, Reparametrization Trick
2. Variational Autoencoder (VAE)
3. Discrete VAE Latent Representations

Outline

Amortized Variational Inference

E-step

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

Amortized Variational Inference

E-step

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

$q(\mathbf{z})$ approximates the true posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$, hence it is called **variational posterior**.

Amortized Variational Inference

E-step

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

$q(\mathbf{z})$ approximates the true posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$, hence it is called **variational posterior**.

- $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$ may be **intractable**;
- $q(\mathbf{z})$ is individual for each data point $\mathbf{x}$.

Amortized Variational Inference

E-step

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

$q(\mathbf{z})$ approximates the true posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$, hence it is called **variational posterior**.

- $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$ may be **intractable**;
- $q(\mathbf{z})$ is individual for each data point $\mathbf{x}$.

Variational Bayes

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

Amortized Variational Inference

E-step

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

$q(\mathbf{z})$ approximates the true posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$, hence it is called **variational posterior**.

- $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^*)$ may be **intractable**;
- $q(\mathbf{z})$ is individual for each data point $\mathbf{x}$.

Variational Bayes

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

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

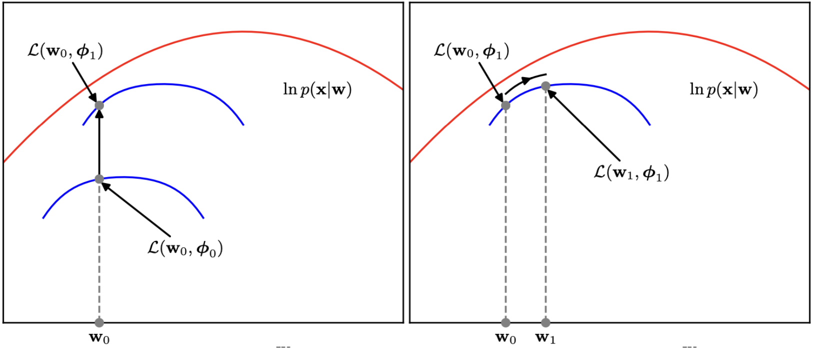
Variational EM Illustration

- 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:

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



Bishop C., Deep Learning: Foundations and Concepts, 2024

Variational EM Algorithm

ELBO

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

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

Variational EM Algorithm

ELBO

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

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

- **E-step:**

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

  where ϕ denotes the parameters of the variational posterior $q(\mathbf{z}|\mathbf{x},\phi)$.
- **M-step:**

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

  where $\boldsymbol{\theta}$ represents the parameters of the generative model $p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})$.

Variational EM Algorithm

ELBO

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

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

- **E-step:**

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

  where ϕ denotes the parameters of the variational posterior $q(\mathbf{z}|\mathbf{x},\phi)$.

- **M-step:**

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

  where $\boldsymbol{\theta}$ represents the parameters of the generative model $p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})$.

The remaining step is to obtain **unbiased** Monte Carlo estimates of the gradients: $\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})$ and $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})$.

Outline

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Naive Monte Carlo Estimation

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

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

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

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

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

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

Naive Monte Carlo Estimation

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

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

---

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

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:

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

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:

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

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

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

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:

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

Reparametrization Trick (LOTUS Trick)

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

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

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

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

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:

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

Reparametrization Trick (LOTUS Trick)

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

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

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 $p(\boldsymbol{\epsilon})$.

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

Reparametrization Trick (LOTUS Trick)

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

,

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

Reparametrization Trick (LOTUS Trick)

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

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

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

Reparametrization Trick (LOTUS Trick)

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

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

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

Variational Assumption

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

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

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

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

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

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

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

Reconstruction Term

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

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

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

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

Reconstruction Term

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

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

KL Term

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

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

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

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

Reconstruction Term

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

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

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

KL Term

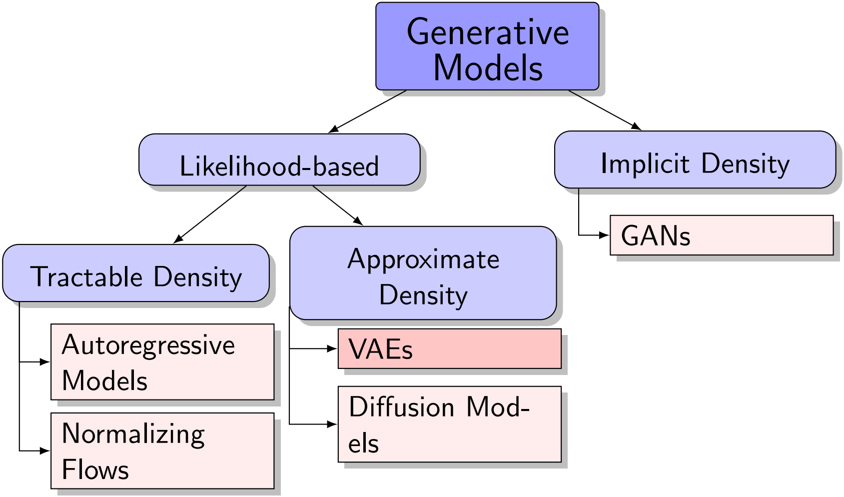
$p(\mathbf{z})$ is the prior over latents $\mathbf{z}$, typically $p(\mathbf{z}) = \mathcal{N}(0,\mathbf{I})$.

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

This expression admits a closed-form analytic solution.

Outline

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):

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

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

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):

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

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

- 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):

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

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

- 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):

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

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

- 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 $p(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\theta})$.

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):

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

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

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

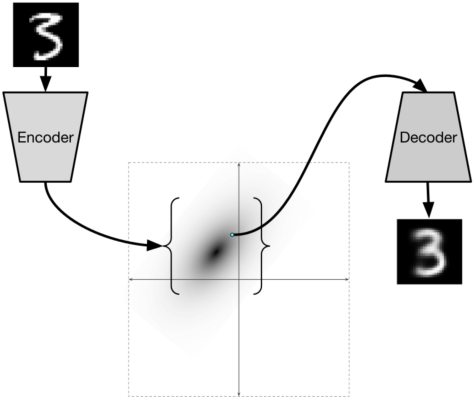
Inference

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

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

Variational Autoencoder

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



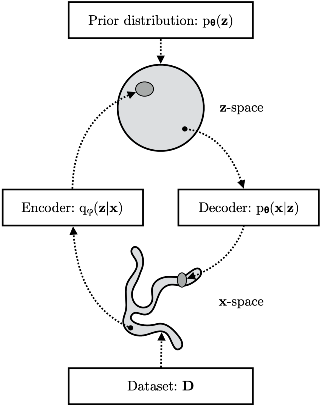
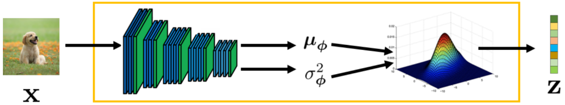
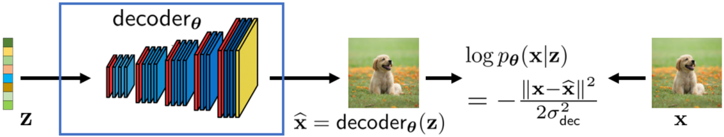


image credit: http://ijdykeman.github.io/ml/2016/12/21/cvae.html
Kingma D. P., Welling M., An Introduction to Variational Autoencoders, 2019

Variational Autoencoder

- The encoder $q(\mathbf{z}|\mathbf{x}, \phi) = \text{NN}_e(\mathbf{x}, \phi)$ outputs $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}_\phi(\mathbf{x})$.
- The decoder $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \text{NN}_d(\mathbf{z}, \boldsymbol{\theta})$ 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 $\mathbf{z} \sim q(\mathbf{z}\|\mathbf{x}, \phi)$	deterministic $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ $q(\mathbf{z}\|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta})$	deterministic $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ $p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))$
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 $\mathbf{z} \sim q(\mathbf{z}\|\mathbf{x}, \phi)$	deterministic $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ $q(\mathbf{z}\|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$
Decoder	stochastic $\mathbf{x} \sim p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta})$	deterministic $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ $p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))$
Parameters	$\boldsymbol{\phi}, \boldsymbol{\theta}$	$\boldsymbol{\theta} \equiv \boldsymbol{\phi}$

Theorem

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

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

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

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

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

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

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

- Apply the reparametrization trick to obtain unbiased gradients.
- Use Gaussian distributions for $q(\mathbf{z}|\mathbf{x},\boldsymbol{\phi})$ 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 $c \sim \text{Categorical}(\boldsymbol{\pi})$, 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

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

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

Discrete VAE Latents

Assumptions

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

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

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

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

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

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

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

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

$$= -\text{H}(q(c|\mathbf{x}, \phi)) + \log K.$$

Discrete VAE Latents

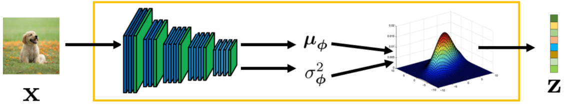
$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q(c|\mathbf{x},\phi)} \log p(\mathbf{x}|c,\boldsymbol{\theta}) + \mathrm{H}(q(c|\mathbf{x},\phi)) - \log K \to \max_{\boldsymbol{\phi},\boldsymbol{\theta}} .$$

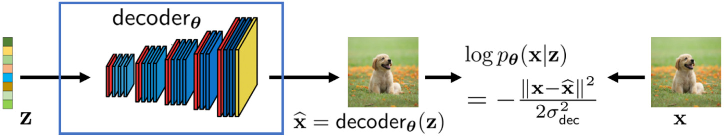
Chan S., Tutorial on Diffusion Models for Imaging and Vision, 2024

Discrete VAE Latents

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

- 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 $p(\mathbf{x}|c,\boldsymbol{\theta})$ must take a discrete random variable c as input.





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

Summary

- Amortized variational inference enables efficient estimation of the ELBO via Monte Carlo estimation.
- 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 $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$.
- NF models can be interpreted as VAEs with deterministic encoder and decoder functions.
- Discrete VAE latents offer a natural class of latent variable models.