UNIT 3

Variational Autoencoders

Johannes Brandstetter
Institute for Machine Learning

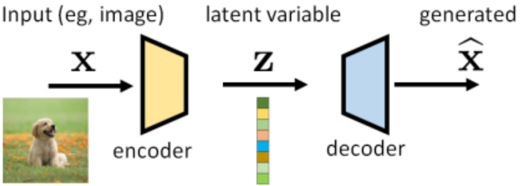
JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright statement:

This material, no matter whether in printed or electronic form, may be used for personal and non-commercial educational use only. Any reproduction of this material, no matter whether as a whole or in parts, no matter whether in printed or in electronic form, requires explicit prior acceptance of the authors.

Autoencoders

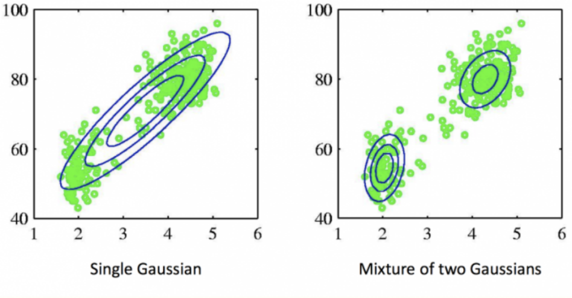


- The autoencoder has an input variable x and a latent variable z
- **Variational autoencoder**: we use probability distributions to describe x and z
- Instead of deterministic procedure of converting x to z, we ensure that the distribution $p(x)$ can be mapped to a desired distribution $p(z)$, and backwards to $p(x)$

Variational Autoencoder

- $p(\boldsymbol{x})$: The distribution of $\boldsymbol{x}$. It is **never known**. Deep generative modeling tries to find ways to draw samples from $p(\boldsymbol{x})$
- $p(\boldsymbol{z})$: The distribution of the latent variable. E.g., zero-mean unit-variance Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $p(\boldsymbol{z}|\boldsymbol{x})$: The conditional distribution associated with the **encoder**, which tells us the likelihood of $\boldsymbol{z}$ when given $\boldsymbol{x}$. We have no access to it, we are going to learn it
- $p(\boldsymbol{x}|\boldsymbol{z})$: The conditional distribution associated with the **decoder**, which tells us the posterior probability of getting $\boldsymbol{x}$ given $\boldsymbol{z}$. We have no access to it, we are going to learn it

Mixture of Gaussians (MOG)



- Mixture of Gaussians allows to model complex distributions:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Mixture of Gaussians

- Mixture of Gaussians allows to model complex distributions:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Integration wrt. $\boldsymbol{x}$ leads to $\sum_{k=1}^{K} \pi_k = 1$
- Since $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \geq 0 \Rightarrow \pi_k \geq 0$ thus $0 \leq \pi_k \leq 1 \Rightarrow$ mixing coefficients are required to be probabilities
- Moreover: $p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}\ , k) = \sum_{k=1}^{K} p(k) p(\boldsymbol{x} \mid k)$ where $p(k) = \pi_k$ and $p(\boldsymbol{x} \mid k) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

MOG through the lense of VAE

- Let's assume number of clusters $k = 2$, i.e., $\boldsymbol{z} \in \{1, 2\}$, and mean and variance $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ are known and are fixed
- **Encoder**:

$$\overset{\text{class 1}}{p_{\boldsymbol{z}|\boldsymbol{x}}(1|\boldsymbol{x})} \gtrless \overset{\text{class 2}}{p_{\boldsymbol{z}|\boldsymbol{x}}(2|\boldsymbol{x})}$$

- **Decoder**:

$$p_{\boldsymbol{x}|\boldsymbol{z}}(\boldsymbol{x}|k) = \mathcal{N}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$

- This example is trivial, but we realize: if we want to find the magical encoder and decoder, we must have a way to find the two conditional distributions

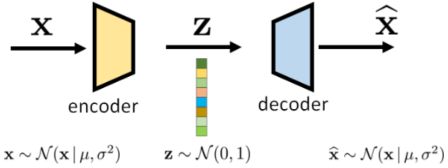
Gaussian distribution

- Random variable $\boldsymbol{x}$ and latent variable $\boldsymbol{z}$:

$$\boldsymbol{x} \sim \mathcal{N}(\mu, \sigma^2)$$
$$\boldsymbol{z} \sim \mathcal{N}(0, I)$$

- This problem has a trivial solution where $\boldsymbol{z} = (\boldsymbol{x} - \mu)/\sigma$ and $\boldsymbol{x} = \mu + \sigma \boldsymbol{z}$

$$\boldsymbol{z} = \text{Encode}(\boldsymbol{x}) = a\boldsymbol{x} + b \qquad \text{such that } \phi = [a, b]$$
$$\boldsymbol{x} = \text{Decode}(\boldsymbol{z}) = c\boldsymbol{z} + d \qquad \text{such that } \theta = [c, d]$$



Gaussian distribution

$$\boldsymbol{z} = \text{Encode}(\boldsymbol{x}) = a\boldsymbol{x} + b \qquad \text{such that } \phi = [a, b]$$

$$\boldsymbol{x} = \text{Decode}(\boldsymbol{z}) = c\boldsymbol{z} + d \qquad \text{such that } \theta = [c, d]$$

- Construct proxy distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(a\boldsymbol{x} + b, 1)$$

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(c\boldsymbol{z} + d, c)$$

- $q_\phi(\boldsymbol{z}|\boldsymbol{x})$: Natural choice for $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is to have the mean $a\boldsymbol{x} + b$ and $\sigma = 1$
- $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: similarly, if we are given $\boldsymbol{z}$, the decoder must take the form of $c\boldsymbol{z} + d$ (setup of the decoder). The variance we need to figure out.

Evidence Lower Bound (ELBO)

- Remember:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(a\boldsymbol{x} + b, 1)$$
$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(c\boldsymbol{z} + d, c)$$

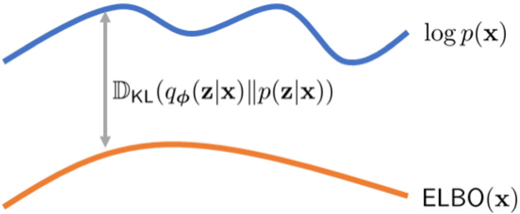
- How do we use these two proxy distributions to determine the encoder and the decoder?
- If we treat ϕ and θ as optimization variables, then we need a loss so that we can optimize ϕ and θ through training $\Rightarrow$ **Evidence Lower Bound (ELBO)**

$$\text{ELBO}(\boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]$$

$$
\begin{aligned}
\log p(\boldsymbol{x}) &= \log p(\boldsymbol{x}) \times \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} \\
&= \int \log p(\boldsymbol{x}) \times q_\phi(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p(\boldsymbol{x})\big] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} \times \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\text{ELBO}} + \underbrace{\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\mathbb{D}_{\mathsf{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x}))}
\end{aligned}
$$

ELBO in a nutshell

$$
\begin{aligned}
\log p(\boldsymbol{x}) = \ldots &= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&\geq \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&\stackrel{\text{def}}{=} \mathsf{ELBO}(\boldsymbol{x})
\end{aligned}
$$



Interlude: Kullback-Leibler divergence

- For discrete probability distributions P and Q defined on the same sample space $\mathcal{X}$, the KL divergence (or relative entropy) from Q to P is defined:

$$\mathbf{D}_{\mathsf{KL}}(P||Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) = -\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

- Relation KL divergence, cross entropy, entropy:

$$\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) = \sum_{x\in\mathcal{X}} P(x)\log P(x) - \sum_{x\in\mathcal{X}} P(x)\log Q(x)$$

$$\underbrace{\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)}_{\mathbf{D}_{\mathsf{KL}}(P||Q)} \underbrace{-\sum_{x\in\mathcal{X}} P(x)\log P(x)}_{H(P)} = \underbrace{-\sum_{x\in\mathcal{X}} P(x)\log Q(x)}_{H(P,Q)}$$

- Analogous relations hold for continuous probability distributions

Interlude: Kullback-Leibler divergence

- Divergences are defined as **statistical distance** on probability distributions
- Given a differentiable manifold M, a divergence D on M satisfies:
  - $D(u, v) \geq 0$
  - $D(u, v) = 0$ only if $u = v$
  - $D(u, u + du)$ is a positive definite quadratic form
- Usually a divergence is not symmetric $\Rightarrow$ divergence $\neq$ distance!! (triangle inequality not fulfilled)
- Family of f-divergences:

$$D_f(u, v) = \int u(x) f(\frac{v(x)}{u(x)})$$

- KL divergence: $f = \log\left(\frac{1}{\frac{q(x)}{p(x)}}\right)$

$$
\begin{aligned}
\text{ELBO}(\boldsymbol{x}) &= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]}_{\text{Reconstruction term}} - \underbrace{\mathbb{D}_{\mathsf{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))}_{\text{Prior matching term}}
\end{aligned}
$$

Reconstruction

- The term $\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$ is about the **decoder**
- The decoder should produce a "good" image x if we feed a latent z into the decoder
- We maximize $\log p_\theta(\boldsymbol{x}|\boldsymbol{z})$
- This is similar to maximum likelihood where we want to find the model parameter to maximize the likelihood of observing the image
- The expectation is taken with respect to the samples z (**conditioned on x**)
- The samples z cannot be an arbitrary noise vector, but need to be a meaningful latent vector $\Rightarrow$ sampled from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

Prior Matching

- The term $\mathbb{D}_{\mathsf{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$ is the KL divergence for the encoder
- The task of the encoder is to turn $\boldsymbol{x}$ into a latent vector $\boldsymbol{z}$ such that the latent vector will follow our chosen distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$
- $p(\boldsymbol{z})$ is the target distribution
- KL increases when two distributions become dissimilar:
    - KL measures how similar the learned variational distribution is to a prior belief held over latent variables

Gaussian distribution - revisited

$$\boldsymbol{z} = \text{Encode}(\boldsymbol{x}) = a\boldsymbol{x} + b \qquad \text{such that } \phi = [a, b]$$

$$\boldsymbol{x} = \text{Decode}(\boldsymbol{z}) = c\boldsymbol{z} + d \qquad \text{such that } \theta = [c, d]$$

- Construct proxy distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(a\boldsymbol{x} + b, 1)$$

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(c\boldsymbol{z} + d, c)$$

- $q_\phi(\boldsymbol{z}|\boldsymbol{x})$: Natural choice for $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is to have the mean $a\boldsymbol{x} + b$ and $\sigma = 1$
- $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: similarly, if we are given $\boldsymbol{z}$, the decoder must take the form of $c\boldsymbol{z} + d$ (setup of the decoder). The variance we need to figure out.

Gaussian distribution - prior matching

- To determine θ and ϕ, we need to **minimize the prior matching error** and **maximize the reconstruction term**
- For the prior matching, we know that:

$$\mathbb{D}_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = \mathbb{D}_{\text{KL}}\left(\mathcal{N}(\boldsymbol{z}|a\boldsymbol{x}+b,1)||\mathcal{N}(0,1)\right)$$

- Remember: $\mathrm{E}(\boldsymbol{x}) = \mu$ and $\text{var}(\boldsymbol{x}) = \sigma^2$
- The KL-divergence is minimized when $a = \dfrac{1}{\sigma}$ and $b = -\dfrac{\mu}{\sigma}$
- $a\boldsymbol{x} + b = \dfrac{\boldsymbol{x}-\mu}{\sigma} \Rightarrow \mathrm{E}(a\boldsymbol{x}+b) = 0,\ \text{var}(a\boldsymbol{x}+b) = 1$

Reconstruction

- For the reconstruction term, we need to evaluate the reconstruction loss:

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) &= \log \mathcal{N}(\mathsf{decode}(\boldsymbol{z}), \sigma^2_{\mathsf{dec}}) \\
&= \log \left(\frac{1}{(\sqrt{2\pi\sigma^2_{\mathsf{dec}}})} \exp - \frac{\|\boldsymbol{x} - \mathsf{decode}(\boldsymbol{z})\|^2}{2\sigma^2_{\mathsf{dec}}} \right) \\
&= -\log \sqrt{(2\pi\sigma^2_{\mathsf{dec}})} - \frac{\|\boldsymbol{x} - \mathsf{decode}(\boldsymbol{z})\|^2}{2\sigma^2_{\mathsf{dec}}}
\end{aligned}$$

- Therefore, we need to maximize:

$$\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[- \frac{(c\boldsymbol{z} + d - \mu)^2}{2c^2} \right]$$

- We use $\mathrm{E}(\boldsymbol{z}) = 0$ and var$(\boldsymbol{z}) = 1$ to get $d = \mu$ and $c = \sigma$

Summary

- In the simple Gaussian distribution example, the encoder and decoder parameters are:

$$\boldsymbol{z} = \text{encode}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mu}{\sigma}$$

$$\boldsymbol{x} = \text{decode}(\boldsymbol{z}) = \sigma \boldsymbol{z} + \mu$$

- During training, we assume access to both $\boldsymbol{z}$ and $\boldsymbol{x}$
- $\boldsymbol{z}$ needs to be sampled from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
- For reconstruction, we estimate θ to maximize $p_\theta(\boldsymbol{x}|\boldsymbol{z})$
- For prior matching, we find ϕ to minimize the KL divergence
- The optimization can be challenging, because if you update ϕ, the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ changes

VAE training

- In VAE training, the ELBO is optimized jointly over parameters ϕ and θ
- The encoder is commonly chosen to model a multivariate Gaussian with diagonal covariance:

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

- The prior is often selected to be a standard multivariate Gaussian:

$$p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Objective

- The objective can be written as:

$$\underset{\phi,\theta}{\operatorname{argmax}}\, \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] - \mathbb{D}_{\mathsf{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})|p(\boldsymbol{z}))$$

- The KL divergence term of the ELBO can be computed analytically
- The reconstruction term can be approximated using a Monte Carlo estimate:

$$\underset{\phi,\theta}{\operatorname{argmax}}\, \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] = \underset{\phi,\theta}{\operatorname{argmax}} \sum_{l} \log p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(l)})$$

- For every $\boldsymbol{x}$ latents $\{\boldsymbol{z}^{(l)}\}_{l=1}^{L}$ are sampled

Reparameterization trick

- A problem arises in this default setup: each $\boldsymbol{z}^{(l)}$ that our loss is computed on is generated by a stochastic sampling procedure $\Rightarrow$ non-differentiable
- Reparameterization trick on $q_\phi(\boldsymbol{z}|\boldsymbol{x})$:
  - Samples from a normal distribution $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ can be rewritten as:

    $$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{I} \cdot \boldsymbol{\epsilon}\,, \quad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

    (This holds for one and multi-dimensional normal distributions; we write diagonal matrices $\boldsymbol{\Sigma} = \boldsymbol{\sigma}\mathbf{I}$)
- By the reparameterization trick, sampling from an arbitrary Gaussian distribution can be performed by sampling from a standard Gaussian, scaling the result by the target standard deviation, and shifting it by the target mean

Gradient of expectation under change of variable

- If we express the random variable z as a deterministic variable $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{I} \cdot \epsilon$, we get an gradient of the expectation:

$$
\begin{aligned}
\nabla_\phi \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[f(\boldsymbol{z})\big] &= \nabla_\phi \mathrm{E}_{p(\boldsymbol{\epsilon})}\big[f(\boldsymbol{z})\big] \\
&= \mathrm{E}_{p(\boldsymbol{\epsilon})}\big[\nabla_\phi f(\boldsymbol{z})\big] \\
&\simeq \nabla_\phi f(\boldsymbol{z})
\end{aligned}
$$

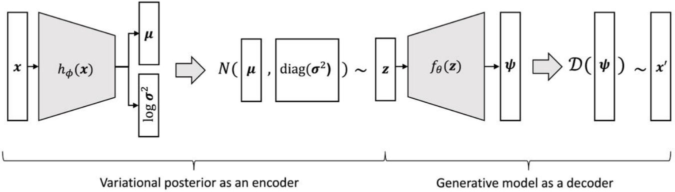
- In a VAE, each z is thus computed as a deterministic function of input $\boldsymbol{x}$ and auxiliary noise variable ϵ:

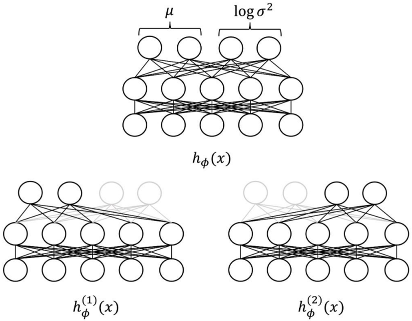
$$
\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}\ , \quad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
$$

VAE inference

- Generating new data can be performed by **sampling directly from the latent space $p(z)$** and then running it through the decoder
- The encoder is not involved in data generation!
- VAEs are particularly interesting when the dimensionality of z is less than that of input $x \Rightarrow$ we might then be learning **compact, useful representations**
- Furthermore, when a semantically meaningful latent space is learned, **latent vectors can be edited** before being passed to the decoder to more precisely control the data generated

VAE implementation

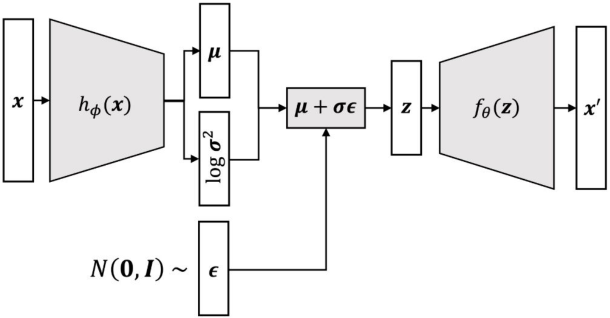




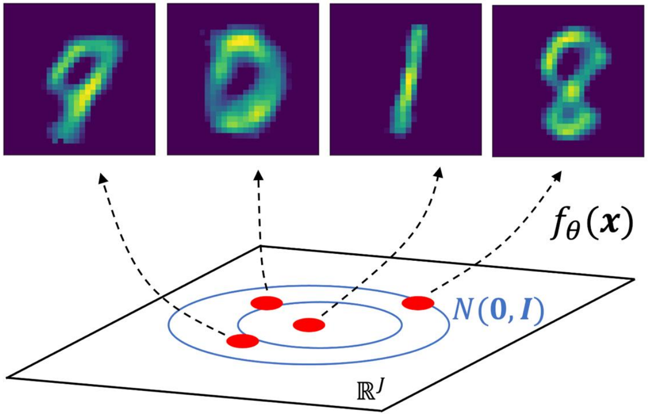
VAE implementation

```
class VAE(nn.Module):
    def __init__(
          self,
          x_dim,
          hidden_dim,
          z_dim=10
        ):
        super(VAE, self).__init__()

        # Define autoencoding layers
        self.enc_layer1 = nn.Linear(x_dim, hidden_dim)
        self.enc_layer2_mu = nn.Linear(hidden_dim, z_dim)
        self.enc_layer2_logvar = nn.Linear(hidden_dim, z_dim)

        # Define autoencoding layers
        self.dec_layer1 = nn.Linear(z_dim, hidden_dim)
        self.dec_layer2 = nn.Linear(hidden_dim, x_dim)

    def encoder(self, x):
        x = F.relu(self.enc_layer1(x))
        mu = F.relu(self.enc_layer2_mu(x))
        logvar = F.relu(self.enc_layer2_logvar(x))
        return mu, logVar

    def reparameterize(self, mu, logvar):
        std = torch.exp(logvar/2)
        eps = torch.randn_like(std)
        z = mu + std * eps
        return z

    def decoder(self, z):
        # Define decoder network
        output = F.relu(self.dec_layer1(z))
        output = F.relu(self.dec_layer2(output))
        return x

    def forward(self, x):
        mu, logvar = self.encoder(x)
        z = self.reparameterize(mu, logVar)
        output = self.decoder(z)
        return output, z, mu, logvar
```

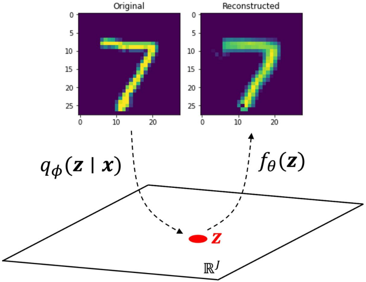
VAE implementation

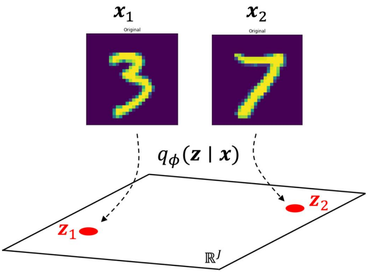


VAE latent space example



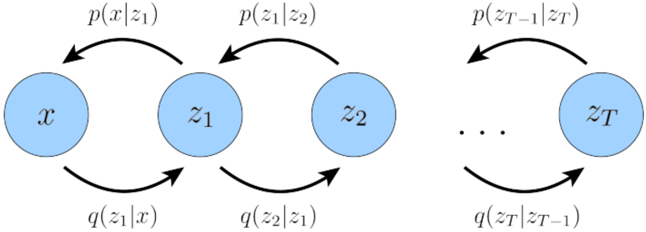
VAE latent space example





Hierarchical VAE

- HVAE extends to multiple hierarchies over latent variables
- Latent variables are interpreted as generated from other higher-level, i.e., more abstract latents
- Special case: **Markovian HVAE** $\Rightarrow$ generative process is a Markov chain



Markovian HVAE

- The generative process is modeled as a Markov chain: each latent $\boldsymbol{z}_t$ is generated only from the previous latent $\boldsymbol{z}_{t+1}$
- Joint distribution:

$$p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T) p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

- The encoder needs to model:

$$q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

ELBO of Hierarchical VAE

$$
\begin{aligned}
\log p(\boldsymbol{x}) &= \log p(\boldsymbol{x}) \times \int q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) d\boldsymbol{z}_{1:T} \\
&= \int \log p(\boldsymbol{x}) \times q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) d\boldsymbol{z}_{1:T} \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log p(\boldsymbol{x})\right] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{p(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right] \\
&= \mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{p(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \times \frac{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right]}_{\text{ELBO}} + \underbrace{\mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}{p(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right]}_{\mathbb{D}_{\text{KL}}(q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) || p(\boldsymbol{z}_{1:T}|\boldsymbol{x}))}
\end{aligned}
$$

ELBO of Hierarchical VAE

$$\mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right] =$$

$$\mathrm{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z}_T)p_\theta(\boldsymbol{x}|\boldsymbol{z}_1)\prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_1|\boldsymbol{x})\prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}\right]$$

ELBO of HVAE (diffusion models)

$\boldsymbol{z}_{1:T} \to \boldsymbol{x}_{1:T}$ we end up with ELBO for diffusion models

$$
\begin{aligned}
\log p(\boldsymbol{x}) &\geq \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)\prod_{t=1}^{T} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=1}^{T-1} p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] + \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \prod_{t=1}^{T-1} \frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] \\
&\quad + \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\sum_{t=1}^{T-1} \log \frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right]
\end{aligned}
$$

ELBO of HVAE (diffusion models)

$$
\begin{aligned}
\ldots &= \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] \\
&\quad + \sum_{t=1}^{T-1} \mathrm{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathrm{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathrm{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] \\
&\quad + \sum_{t=1}^{T-1} \mathrm{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \underbrace{\mathrm{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{\mathrm{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\mathbb{D}_{\mathsf{KL}}\Big(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})||p(\boldsymbol{x}_T)\Big)\right]}_{\text{prior matching term}} \\
&\quad + \underbrace{\sum_{t=1}^{T-1} \mathrm{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right]}_{\text{consistency term}}
\end{aligned}
$$

ELBO of HVAE (diffusion models)

- $\mathrm{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]$ can be interpreted as a reconstruction term, predicting the log probability of the original data sample given the first-step latent. This term also appears in a vanilla VAE, and can be trained similarly.
- $\mathrm{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\mathbb{D}_{\mathsf{KL}}\left(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})||p(\boldsymbol{x}_T)\right)\right]$ is a prior matching term; it is minimized when the final latent distribution matches the (Gaussian) prior.
- $\sum_{t=1}^{T-1}\mathrm{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right]$ is a consistency term; it endeavors to make the distribution at x_t consistent between priors.