From Autoencoder to Variational Autoencoder

Hao Dong

Peking University
From Autoencoder to Variational Autoencoder

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)
• Vanilla Autoencoder
• Denoising Autoencoder
• Sparse Autoencoder
• Contractive Autoencoder
• Stacked Autoencoder
• Variational Autoencoder (VAE)
Vanilla Autoencoder

• What is it?

Reconstruct high-dimensional data using a neural network model with a narrow bottleneck layer.

The bottleneck layer captures the compressed latent coding, so the nice by-product is dimension reduction.

The low-dimensional representation can be used as the representation of the data in various applications, e.g., image retrieval, data compression ...
Vanilla Autoencoder

• How it works?

\[ \mathbf{X} \xrightarrow{E} \mathbf{Z} \xrightarrow{G} \hat{\mathbf{X}} \]

Ideally the input and reconstruction are identical

The encoder network is for dimension reduction, just like PCA

Latent code: the compressed low dimensional representation of the input data
Vanilla Autoencoder

- Training

The hidden units are usually less than the number of inputs
- Dimension reduction --- Representation learning

The distance between two data can be measure by Mean Squared Error (MSE):

\[ \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (x^i - G(E(x^i)))^2 \]

where \( n \) is the number of variables

- It is trying to learn an approximation to the identity function so that the input is “compress” to the “compressed” features, discovering interesting structure about the data.
Vanilla Autoencoder

- **Testing/Inferencing**

  - Autoencoder is an *unsupervised learning* method if we considered the latent code as the “output”.
  - Autoencoder is also a *self-supervised (self-taught) learning* method which is a type of *supervised learning* where the training labels are determined by the input data.
  - Word2Vec (from RNN lecture) is another unsupervised, self-taught learning example.

Autoencoder for MNIST dataset (28×28×1, 784 pixels)
Vanilla Autoencoder

- Example:
  - Compress MNIST (28x28x1) to the latent code with only 2 variables
Vanilla Autoencoder

• **Power of Latent Representation**
  • t-SNE visualization on MNIST: PCA vs. Autoencoder

2006 Science paper by Hinton and Salakhutdinov
Vanilla Autoencoder

- Discussion

  - Hidden layer is overcomplete if greater than the input layer
Vanilla Autoencoder

• Discussion

  • Hidden layer is overcomplete if greater than the input layer
  
    • No compression
  
    • No guarantee that the hidden units extract meaningful feature
• Vanilla Autoencoder
• **Denoising Autoencoder**
• Sparse Autoencoder
• Contractive Autoencoder
• Stacked Autoencoder
• Variational Autoencoder (VAE)
Denoising Autoencoder (DAE)

- Why?
  - Avoid overfitting
  - Learn robust representations
Denoising Autoencoder

• Architecture

Applying dropout between the input and the first hidden layer

• Improve the robustness
Denoising Autoencoder

• Feature Visualization

One neuron == One feature extractor

Visualizing the learned features
Denoising Autoencoder

• **Denoising Autoencoder & Dropout**

Denoising autoencoder was proposed in 2008, 4 years before the dropout paper (Hinton, et al. 2012).

Denoising autoencoder can be seem as applying dropout between the input and the first layer.

Denoising autoencoder can be seem as one type of data augmentation on the input.
• Vanilla Autoencoder
• Denoising Autoencoder
• **Sparse Autoencoder**
• Contractive Autoencoder
• Stacked Autoencoder
• Variational Autoencoder (VAE)
Sparse Autoencoder

- **Why?**

  Even when the number of hidden units is large (perhaps even greater than the number of input pixels), we can still discover interesting structure, by imposing other constraints on the network.

  In particular, if we impose a "sparsity" constraint on the hidden units, then the autoencoder will still discover interesting structure in the data, even if the number of hidden units is large.
Sparse Autoencoder

• Recap: KL Divergence

\[
KL(p(x)\|q(x)) = \int p(x) \ln \frac{p(x)}{q(x)} dx = \mathbb{E}_{x \sim p(x)} \left[ \ln \frac{p(x)}{q(x)} \right]
\]

Smaller == Closer

\[
KL(p(x)\|q(x)) = 0 \iff p(x) = q(x)
\]

\[
KL(p(x)\|q(x)) = 0 \iff p(x) = q(x)
\]

\[
D_B(p(x), q(x)) = -\ln \int \sqrt{p(x)q(x)} dx
\]
Sparse Autoencoder

- Sparsity Regularization

Given $M$ data samples (batch size) and Sigmoid activation function, the active ratio of a neuron $a_j$:

$$\hat{\rho}_j = \frac{1}{M} \sum_{m=1}^{M} a_j$$

To make the output "sparse", we would like to enforce the following constraint, where $\rho$ is a "sparsity parameter", such as 0.2 (20% of the neurons)

$$\hat{\rho}_j = \rho$$

The penalty term is as follow, where $s$ is the number of activation outputs.

$$L_\rho = \sum_{j=1}^{s} KL(\rho || \hat{\rho}_j)$$

$$= \sum_{j=1}^{s} (\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j})$$

The total loss:

$$L_{total} = L_{MSE} + \lambda L_\rho$$

The number of hidden units can be greater than the number of input variables.
Sparse Autoencoder

- Sparsity Regularization

Smaller $\rho$ $==$ More sparse

Autoencoders for MNIST dataset

Input $\mathbf{x}$

Autoencoder $\hat{\mathbf{x}}$

Sparse Autoencoder $\tilde{\mathbf{x}}$
Sparse Autoencoder

• Different regularization loss

<table>
<thead>
<tr>
<th>Method</th>
<th>Hidden Activation</th>
<th>Reconstruction Activation</th>
<th>Loss Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>Sigmoid</td>
<td>Sigmoid</td>
<td>$\mathcal{L}<em>{total} = \mathcal{L}</em>{MSE} + \mathcal{L}_\rho$</td>
</tr>
<tr>
<td>Method 2</td>
<td>ReLU</td>
<td>Softplus</td>
<td>$\mathcal{L}<em>{total} = \mathcal{L}</em>{MSE} + |a|$</td>
</tr>
</tbody>
</table>

$L_1$ on the hidden activation output
Sparse Autoencoder

- Sparse Autoencoder vs. Denoising Autoencoder

Feature Extractors of Sparse Autoencoder

Feature Extractors of Denoising Autoencoder
Sparse Autoencoder

• Autoencoder vs. Denoising Autoencoder vs. Sparse Autoencoder
• Vanilla Autoencoder
• Denoising Autoencoder
• Sparse Autoencoder
• **Contractive Autoencoder**
• Stacked Autoencoder
• Variational Autoencoder (VAE)
Contractive Autoencoder

**Why?**

- Denoising Autoencoder and Sparse Autoencoder overcome the overcomplete problem via the input and hidden layers.

- Could we add an explicit term in the loss to avoid uninteresting features?

  We wish the features that ONLY reflect variations observed in the training set.
Contractive Autoencoder

• How

  • Penalize the representation being too sensitive to the input
  • Improve the robustness to small perturbations
  • Measure the sensitivity by the Frobenius norm of the Jacobian matrix of the encoder activations
Contractive Autoencoder

• Recap: Jocobian Matrix

\[ z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]

\[ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_1 + z_2 \\ 2z_1 \end{bmatrix} = f \left( \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \right) \]

\[ x = f(z) \quad z = f^{-1}(x) \]

\[ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_2/2 \\ x_1 - x_2/2 \end{bmatrix} = f^{-1} \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \]

\[ J_f = \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_1}{\partial z_2} \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} \end{bmatrix} \quad J_{f^{-1}} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} \end{bmatrix} \]

\[ J_f^{-1} = \begin{bmatrix} \frac{1}{2} & 1/2 \\ 1 & -1/2 \end{bmatrix} \]

\[ J_f J_f^{-1} = I \]
Contractive Autoencoder

- **Jacobian Matrix**

\[
y = \begin{bmatrix}
f_1(x) \\
f_2(x) \\
f_3(x) \\
\vdots \\
f_n(x)
\end{bmatrix}
\]

\[
y_1 = \begin{bmatrix}f_1(x_1, x_2, x_3, \ldots x_n) \\
f_2(x_1, x_2, x_3, \ldots x_n) \\
f_3(x_1, x_2, x_3, \ldots x_n) \\
\vdots \\
f_n(x_1, x_2, x_3, \ldots x_n)
\end{bmatrix}
\]

\[
y(x) \approx y(p) + J \cdot (x - p)
\]

\[
J(x_1, x_2, x_3, \ldots x_n) = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \ldots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \ldots & \frac{\partial f_2}{\partial x_n} \\
\frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \ldots & \frac{\partial f_3}{\partial x_n} \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \ldots & \frac{\partial f_n}{\partial x_n}
\end{bmatrix}
\]
Contractive Autoencoder

• New Loss

\[ \|J_f(x)\|_F^2 = \sum_{ij} \left( \frac{\partial h_j(x)}{\partial x_i} \right)^2 \]

\[ J_{CAE}(\theta) = \sum_{x \in D_n} (L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2) \]

reconstruction  new regularization
Contractive Autoencoder

• vs. Denoising Autoencoder

• Advantages
  • CAE can better model the distribution of raw data

• Disadvantages
  • DAE is easier to implement
  • CAE needs second-order optimization (conjugate gradient, LBFGS)
• Vanilla Autoencoder
• Denoising Autoencoder
• Sparse Autoencoder
• Contractive Autoencoder
• **Stacked Autoencoder**
• Variational Autoencoder (VAE)
Stacked Autoencoder

- **Start from Autoencoder: Learn Feature From Input**

The feature extractor for the input data

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights
Stacked Autoencoder

- **2\textsuperscript{nd} Stage: Learn 2\textsuperscript{nd} Level Feature From 1\textsuperscript{st} Level Feature**

```
\begin{align*}
\mathbf{x}! & \rightarrow \mathbf{a}! \\
\mathbf{x}! & \rightarrow \mathbf{a}!
\end{align*}
```

The feature extractor for the first feature extractor

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights
Stacked Autoencoder

• 3rd Stage: Learn 3rd Level Feature From 2nd Level Feature

The feature extractor for the second feature extractor

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights
Stacked Autoencoder

- **4th Stage: Learn 4th Level Feature From 3rd Level Feature**

- The feature extractor for the third feature extractor

- Red color indicates the trainable weights

- Red lines indicate the trainable weights

- Black lines indicate the fixed/nontrainable weights
Stacked Autoencoder

- Use the Learned Feature Extractor for Downstream Tasks

Learn to classify the input data by using the labels and high-level features

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights
Stacked Autoencoder

- **Fine-tuning**

  Fine-tune the entire model for classification

  Red color indicates the trainable weights

  Red lines indicate the trainable weights

  Black lines indicate the fixed/non trainable weights

  Supervised

input  hidden 1  hidden 2  hidden 3  hidden 4  output
Stacked Autoencoder

• Discussion

• Advantages
  • ...

• Disadvantages
  • ...
- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)
  - From Neural Network Perspective
  - From Probability Model Perspective
Before we start

• Question?

• Are the previous Autoencoders generative model?

• Recap: We want to learn a probability distribution $p(x)$ over $x$

  o **Generation (sampling):** $x_{new} \sim p(x)$
    (NO, The compressed latent codes of autoencoders are not prior distributions, autoencoder cannot learn to represent the data distribution)

  o **Density Estimation:** $p(x)$ high if $x$ looks like a real data
    NO

  o **Unsupervised Representation Learning:**
    Discovering the underlying structure from the data distribution (e.g., ears, nose, eyes ...)
    (YES, Autoencoders learn the feature representation)
• Vanilla Autoencoder
• Denoising Autoencoder
• Sparse Autoencoder
• Contractive Autoencoder
• Stacked Autoencoder
• Variational Autoencoder (VAE)
  • From Neural Network Perspective
  • From Probability Model Perspective
Variational Autoencoder

- How to perform generation (sampling)?

Can the hidden output be a prior distribution, e.g., Normal distribution?

\[ p(X) = \sum_Z p(X|Z)p(Z) \]

Decoder (Generator) maps \( N(0, 1) \) to data space

Variational Autoencoder

• Quick Overview

\[ \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{kl}} \]
Variational Autoencoder

• The neural net perspective
  • A variational autoencoder consists of an encoder, a decoder, and a loss function

Variational Autoencoder

• Encoder, Decoder

\[ Z \]

\[ \text{Encoder } q_\theta(z|x) \]

Data: \( x \)

\[ \text{Decoder } p_\phi(x|z) \]

Reconstruction: \( \tilde{x} \)

Variational Autoencoder

- **Loss function**

\[
l_i(\theta, \phi) = -\mathbb{E}_{z \sim q(\theta \mid x_i)} \left[ \log p(\hat{x}_i \mid z) \right] + \text{KL}(q(\theta \mid x_i) \mid \mid p(z))
\]

Can be represented by MSE regularization

Variational Autoencoder

- Why $\text{KL}(Q || P)$ not $\text{KL}(P || Q)$

\[ q^* = \arg\min_q D_{KL}(p || q) \quad \text{and} \quad q^* = \arg\min_q D_{KL}(q || p) \]

![Probability Density](image1)

- Which direction of the KL divergence to use?
  - Some applications require an approximation that usually places high probability anywhere that the true distribution places high probability: left one
  - VAE requires an approximation that rarely places high probability anywhere that the true distribution places low probability: right one

If:

\[
D_{KL}(P || Q) = \mathbb{E}_{x \sim P} \left[ \log \frac{P(x)}{Q(x)} \right] = \mathbb{E}_{x \sim P} \left[ \log P(x) - \log Q(x) \right].
\]
Variational Autoencoder

- Reparameterization Trick

1. Encode the input
2. Predict means
3. Predict standard derivations
4. Use the predicted means and standard derivations to sample new latent variables individually
5. Reconstruct the input

Variational Autoencoder

- **Reparameterization Trick**
  - $z \sim N(\mu, \sigma)$ is not differentiable
  - To make sampling $z$ differentiable
  - $z = \mu + \sigma \times \epsilon \quad \epsilon \sim N(0, 1)$

\[
\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)dz
\]

\[
= \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2} \left(\frac{z-\mu}{\sigma}\right)^2\right]d\left(\frac{z-\mu}{\sigma}\right)
\]

Variational Autoencoder

- Reparameterization Trick

\[
\log P(X|z) - \mathcal{D}[Q(z|X)\|P(z)]
\]

\[
E_{X\sim\mathcal{D}}[E_{z\sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)\|P(z)]]
\]

Variational Autoencoder

• Loss function

\[ KL(q_\theta(z \mid x_i) \mid\mid p(z)) \]
Variational Autoencoder

• Where is ‘variational’?

\[ \text{KL}(q_\theta(z | x_i) \mid \mid p(z)) \]
• Vanilla Autoencoder
• Denoising Autoencoder
• Sparse Autoencoder
• Contractive Autoencoder
• Stacked Autoencoder
• Variational Autoencoder (VAE)
  • From Neural Network Perspective
  • From Probability Model Perspective
Variational Autoencoder

• Problem Definition
  Goal: Given $X = \{x_1, x_2, x_3 \ldots, x_n\}$, find $p(X)$ to represent $X$

  How: It is difficult to directly model $p(X)$, so alternatively, we can ...

  $$p(X) = \sum_Z p(X|Z)p(Z)$$

  where $p(Z) = N(0,1)$ is a prior/known distribution

  i.e., sample $X$ from $Z$
Variational Autoencoder

• **The probability model perspective**
  • \( P(X) \) is hard to model

\[
p(X) = \sum_z p(X|Z)p(Z)
\]

• Alternatively, we learn the joint distribution of \( X \) and \( Z \)

\[
p(X, Z) = p(Z)p(X|Z)
\]

\[
p(X) = \sum_z p(X, Z)
\]

Variational Autoencoder

- **Assumption**

\[ P(z) = \mathcal{N}(z|0, I) \]
Variational Autoencoder

- Assumption

\[ P(z) = \mathcal{N}(z|0, I) \]

\[ g(z) = \frac{z}{10} + \frac{z}{||z||} \]
Variational Autoencoder

- **Monte Carlo?**
  - $n$ might need to be extremely large before we have an accurate estimation of $P(X)$

\[
P(X) = \int P(X|z; \theta)P(z)dz.
\]

\[
P(X) \approx \frac{1}{n} \sum_{i} P(X|z_i)
\]
Variational Autoencoder

- Monte Carlo?
  - Pixel difference is different from perceptual difference

Variational Autoencoder

- **Monte Carlo?**
  - VAE alters the sampling procedure
Variational Autoencoder

- Recap: Variational Inference
  - VI turns inference into optimization

\[
p(x) = \int p(x, z) dz
\]

\[
\mathcal{D} = \{q_\theta(z)\}
\]

\[
\theta^* = \arg\min_\theta \text{KL}(q_\theta(z) \| p(z|x))
\]

\[
p(z|x) = \frac{p(x, z)}{p(x)} \propto p(x, z)
\]

\[
\theta^* = \arg\max_\theta \mathbb{E}_q[\log p(x, z) - \log q_\theta(z)]
\]

Variational Autoencoder

- **Variational Inference**
  - VI turns inference into optimization
Variational Autoencoder

- **Setting up the objective**
  - Maximize $P(X)$
  - Set $Q(z)$ to be an arbitrary distribution

\[
D [Q(z) \| P(z | X)] = E_{z \sim Q} [\log Q(z) - \log P(z | X)]
\]

\[
D [Q(z) \| P(z | X)] = E_{z \sim Q} [\log Q(z) - \log P(X | z) - \log P(z)] + \log P(X)
\]

\[
\log P(X) - D [Q(z) \| P(z | X)] = E_{z \sim Q} [\log P(X | z)] - D [Q(z) \| P(z)]
\]

\[
\log P(X) - D [Q(z | X) \| P(z | X)] = E_{z \sim Q} [\log P(X | z)] - D [Q(z | X) \| P(z)]
\]

Goal: maximize this $\log P(x)$

Variational Autoencoder

- Setting up the objective

\[
\log P(X) - D [Q(z|X) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z|X) \| P(z)]
\]

Goal: maximize this

\[\mathcal{L}_{MSE} \]

Goal becomes: optimize this

\[\mathcal{L}_{total} = \mathcal{L}_{MSE} + \mathcal{L}_{kl}\]
Variational Autoencoder

• Setting up the objective: ELBO

\[
p(z|X) = \frac{p(X,z)}{p(X)}
\]

- ELBO

\[
KL(q_\theta(z | x) \mid \mid p(z | x)) = E_q[\log q_\theta(z | x)] - E_q[\log p(x, z)] + \log p(x)
\]

\[
q^*_\theta(z | x) = \arg \min_\theta KL(q_\theta(z | x) \mid \mid p(z | x))
\]

\[
ELBO(\theta) = E_q[\log p(x, z)] - E_q[\log q_\theta(z | x)]
\]

Variational Autoencoder

- Setting up the objective: ELBO

\[
ELBO(\theta) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\theta(z | x)]
\]

\[
\log p(x) = ELBO(\theta) + KL(q_\theta(z | x) \| p(z | x))
\]

The ELBO for a single datapoint in the variational autoencoder is:

\[
ELBO_i(\theta) = \mathbb{E}_{q \theta}(z | x_i)[\log p(x_i | z)] - KL(q_\theta(z | x_i) \| p(z))
\]

Variational Autoencoder

- Recap: The KL Divergence Loss

\[
KL(\mathcal{N}(\mu, \sigma^2) || \mathcal{N}(0, 1)) = \int \mathcal{N}(\mu, \sigma^2) \log \frac{\mathcal{N}(\mu, \sigma^2)}{\mathcal{N}(0, 1)} \, dx
\]

\[
= \int \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left( \log \frac{1}{\sqrt{2\pi} e^{\frac{-\mu^2}{2}}} \right) \, dx \\
= \int \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \log \left( \frac{1}{\sqrt{\sigma^2} e^{\frac{-x^2}{2}}} \right) \, dx \\
= \frac{1}{2} \int \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left[ -\log \sigma^2 + x^2 - \frac{(x - \mu)^2}{\sigma^2} \right] \, dx
\]

Variational Autoencoder

- Recap: The KL Divergence Loss

\[
KL(\mathcal{N}(\mu, \sigma^2) \mid \mid \mathcal{N}(0,1)) = \frac{1}{2} \int \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left[-\log \sigma^2 + x^2 - \frac{(x-\mu)^2}{\sigma^2}\right] dx
\]

\[
= \frac{1}{2} (-\log \sigma^2 + \mu^2 + \sigma^2 - 1)
\]
Variational Autoencoder

- Recap: The KL Divergence Loss

\[
\mathcal{D}[\mathcal{N}(\mu_0, \Sigma_0) \| \mathcal{N}(\mu_1, \Sigma_1)] = \\
\frac{1}{2} \left( \text{tr} \left( \Sigma_1^{-1} \Sigma_0 \right) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) \right) - k + \log \left( \frac{\text{det} \Sigma_1}{\text{det} \Sigma_0} \right)
\]
Variational Autoencoder

• Optimizing the objective

\[
\log P(X) - \mathcal{D}[Q(z|X)\|P(z|X)] = E_{z \sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)\|P(z)]
\]

\[
E_{X \sim D}[\log P(X) - \mathcal{D}[Q(z|X)\|P(z|X)]] =
E_{X \sim D}[E_{z \sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)\|P(z)]],
\]
Variational Autoencoder

- VAE is a Generative Model

\[ p(Z|X) \text{ is not } N(0,1) \]

Can we input \( N(0,1) \) to the decoder for sampling?

YES: the goal of KL is to make \( p(Z|X) \) to be \( N(0,1) \)
Variational Autoencoder

- VAE vs. Autoencoder
  - VAE: distribution representation, $p(z|x)$ is a distribution
  - AE: feature representation, $h = E(x)$ is deterministic
Variational Autoencoder

- Challenges
  - Low quality images
  - ...

Summary: Take Home Message

- Autoencoders learn data representation in an unsupervised/ self-supervised way.
- Autoencoders learn data representation but cannot model the data distribution $p(X)$.
- Different with vanilla autoencoder, in sparse autoencoder, the number of hidden units can be greater than the number of input variables.
- VAE
- ...
- ...
- ...
- ...
- ...
- ...
- ...
Thanks