Interpretable Factor Models of Single-cell RNA-Seq via Variational Autoencoders

2019/10/09

Ping-Han Hsieh

Outline

- Background
  - Single-cell RNA-Seq
  - Neural network
  - Autoencoder
  - Variational autoencoder (VAE)
- Methods
  - Linearly decoded variational autoencoder (LDVAE)
- Results
- Discussion

Background

Single-cell RNA-Seq

- Problem
  - scRNA-Seq is useful to analyze relationship between genes depend on cell types.
  - In order to investigate such interaction, we need to learn the manifold of gene expression in different cell types.

- Existing Method
  - Principle Component Analysis: Gaussian likelihood not suitable for RNA-Seq data (Negative Binomial Distribution)
  - ZINB-WaVE: not scalable to big data
  - Variational Autoencoder (scVI): hard to interpret the result

Neural Network – Representation (1)

$$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix} \Big\downarrow \text{Samples}$$

$\xrightarrow{\hspace{6em}}$ Features

$$x_k \rightarrow \begin{bmatrix} X_{1,k} & X_{2,k} & \cdots & X_{n,k} \end{bmatrix}$$

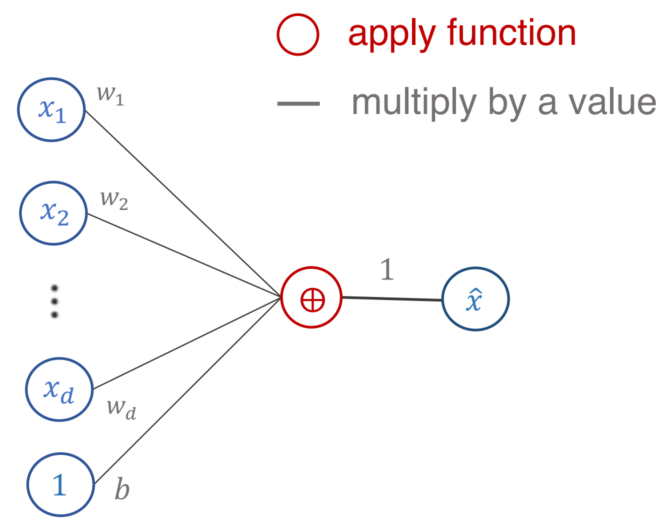
$$x^{(i)} \rightarrow \begin{bmatrix} X_{i,1} & X_{i,2} & \cdots & X_{i,d} \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_d \end{bmatrix}$$

Linear Combination

$$\hat{\mathbf{x}} = \boldsymbol{X}\boldsymbol{w}^{\boldsymbol{T}} + b = w_1\mathbf{x_1} + w_2\mathbf{x_2} + \ldots w_d\mathbf{x_d} + b$$

$$w_1 \overset{x_1}{\begin{bmatrix} X_{1,1} \\ X_{2,1} \\ \vdots \\ X_{n,1} \end{bmatrix}} + w_2 \overset{x_2}{\begin{bmatrix} X_{1,2} \\ X_{2,2} \\ \vdots \\ X_{n,2} \end{bmatrix}} + \cdots + w_d \overset{x_d}{\begin{bmatrix} X_{1,d} \\ X_{2,d} \\ \vdots \\ X_{n,d} \end{bmatrix}} + b$$



Neural Network – Representation (2)

$$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}$$

Observations

Features

$$x_k \rightarrow \begin{bmatrix} X_{1,k} & X_{2,k} & \cdots & X_{n,k} \end{bmatrix}$$

$$x^{(i)} \rightarrow \begin{bmatrix} X_{i,1} & X_{i,2} & \cdots & X_{i,d} \end{bmatrix}$$

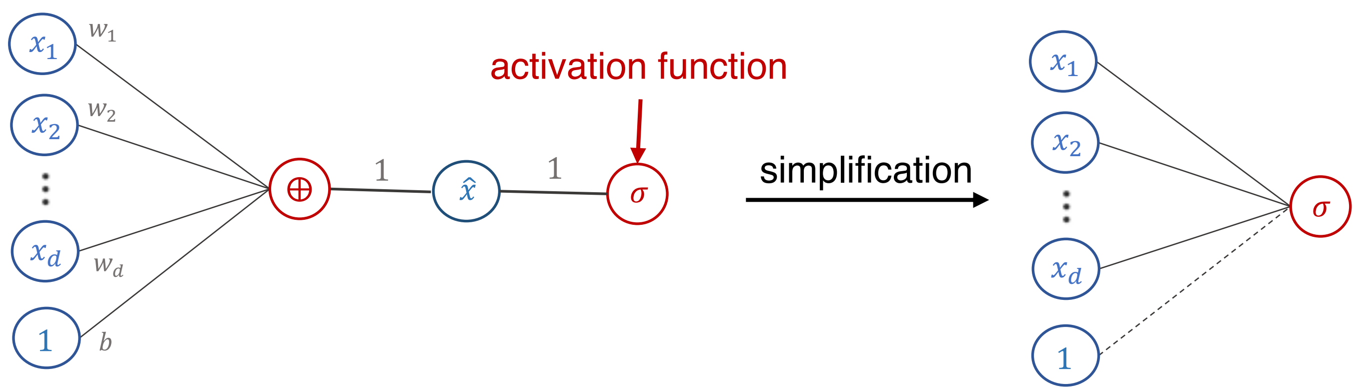
$$\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_d \end{bmatrix}$$

Linear combination + activation function

$$\sigma(w_1\mathbf{x_1} + w_2\mathbf{x_2} + \ldots w_d\mathbf{x_d} + b)$$

apply function

multiply by a value



Neural Network – Gradient Descent

1. Define the loss function

$$L = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$
$$= \sum_{i=1}^{n}(w_1 X_{i,1} + w_2 X_{i,2} + \ldots w_d X_{i,d} - y_i)^2$$

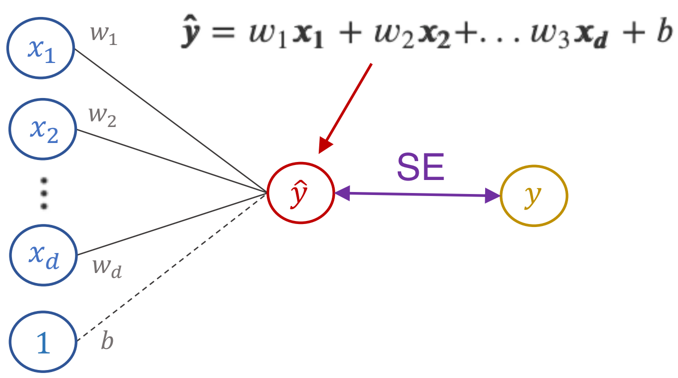
2. The loss function is actually a function of w

3. Rewrite the loss function to $\mathrm{L}(w)$

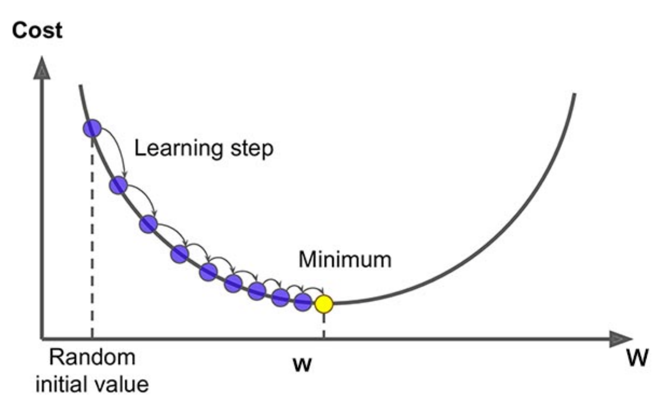
4. We can use the gradient to approximate the minimum loss

$$w^1 = w^0 - \eta \nabla_w L|_{w^0}$$
$$w^2 = w^1 - \eta \nabla_w L|_{w^1}$$
$$w^3 = w^2 - \eta \nabla_w L|_{w^2}$$
$$\vdots$$

Linear regression



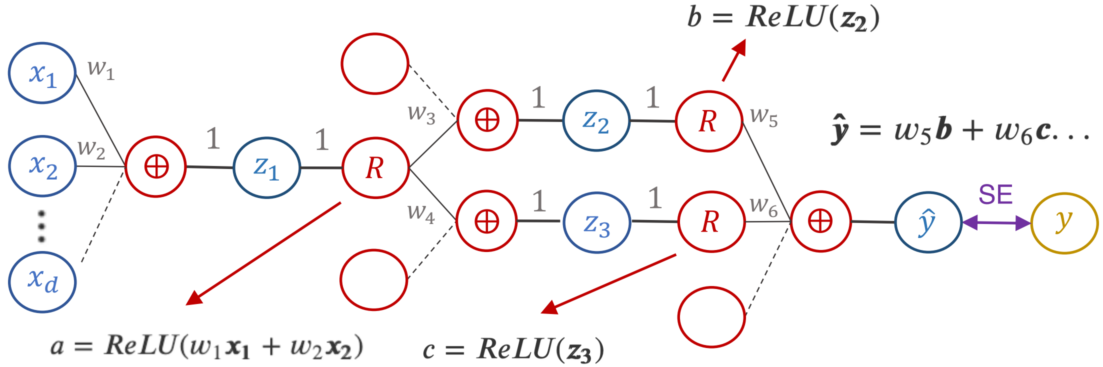
<u>Gradient descent</u>

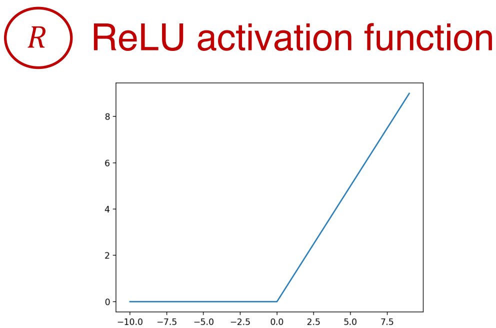


from Saugat Bhattarai

Neural Network – Backpropagation (1)

Consider more than one hidden layer





1. Feed forward, compute the derivative for $\boldsymbol{z}$ w.r.t w

$$\frac{\partial \boldsymbol{z_1}}{\partial w_1} = \boldsymbol{x_1} \qquad \frac{\partial \boldsymbol{z_2}}{\partial w_3} = \boldsymbol{a}$$
$$\frac{\partial \boldsymbol{z_1}}{\partial w_2} = \boldsymbol{x_2} \qquad \frac{\partial \boldsymbol{z_3}}{\partial w_4} = \boldsymbol{a}$$

The derivative is the corresponding input of the node

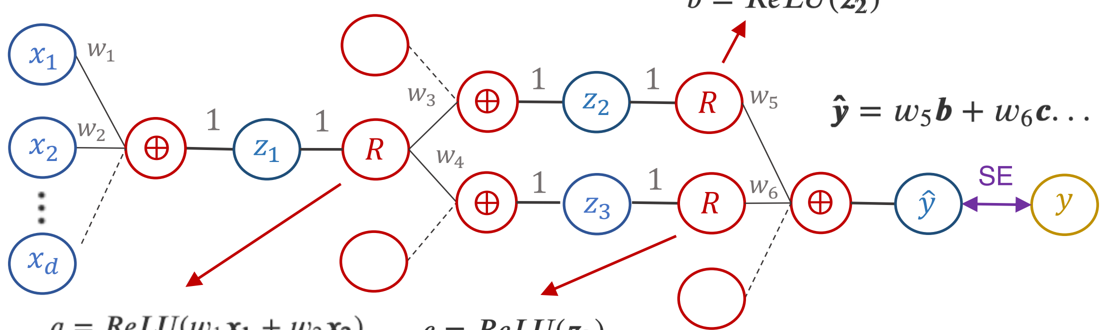
2. Backward pass, compute the derivative for L w.r.t $\boldsymbol{z}$

$$\frac{\partial L}{\partial \boldsymbol{z_3}} = \frac{\partial L}{\partial \boldsymbol{c}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{z_3}} = \underbrace{\frac{\partial L}{\partial \hat{\boldsymbol{y}}}}_{2\hat{y}}\underbrace{\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{c}}}_{w_6}\underbrace{\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{z3}}}_{I\{z_3>0\}} \qquad \frac{\partial L}{\partial \boldsymbol{z_2}} = \frac{\partial L}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z_2}} = \underbrace{\frac{\partial L}{\partial \hat{\boldsymbol{y}}}}_{2\hat{y}}\underbrace{\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{b}}}_{w_5}\underbrace{\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z2}}}_{I\{z_2>0\}}$$

Neural Network – Backpropagation (2)

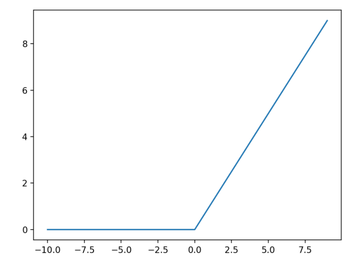
Consider more than one hidden layer

$b = ReLU(\boldsymbol{z_2})$



$a = ReLU(w_1\boldsymbol{x_1} + w_2\boldsymbol{x_2})$ $c = ReLU(\boldsymbol{z_3})$

R ReLU activation function



$$\frac{\partial \boldsymbol{z_1}}{\partial w_1} = \boldsymbol{x_1} \quad \frac{\partial \boldsymbol{z_2}}{\partial w_3} = \boldsymbol{a} \qquad \frac{\partial L}{\partial \boldsymbol{z_3}} = \frac{\partial L}{\partial \boldsymbol{c}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{z_3}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{c}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{z3}} \qquad \frac{\partial L}{\partial \boldsymbol{z_2}} = \frac{\partial L}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z_2}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{z2}}$$

$$\frac{\partial \boldsymbol{z_1}}{\partial w_2} = \boldsymbol{x_2} \quad \frac{\partial \boldsymbol{z_3}}{\partial w_4} = \boldsymbol{a}$$

2. Backward pass, compute the derivative for L w.r.t $\boldsymbol{z}$

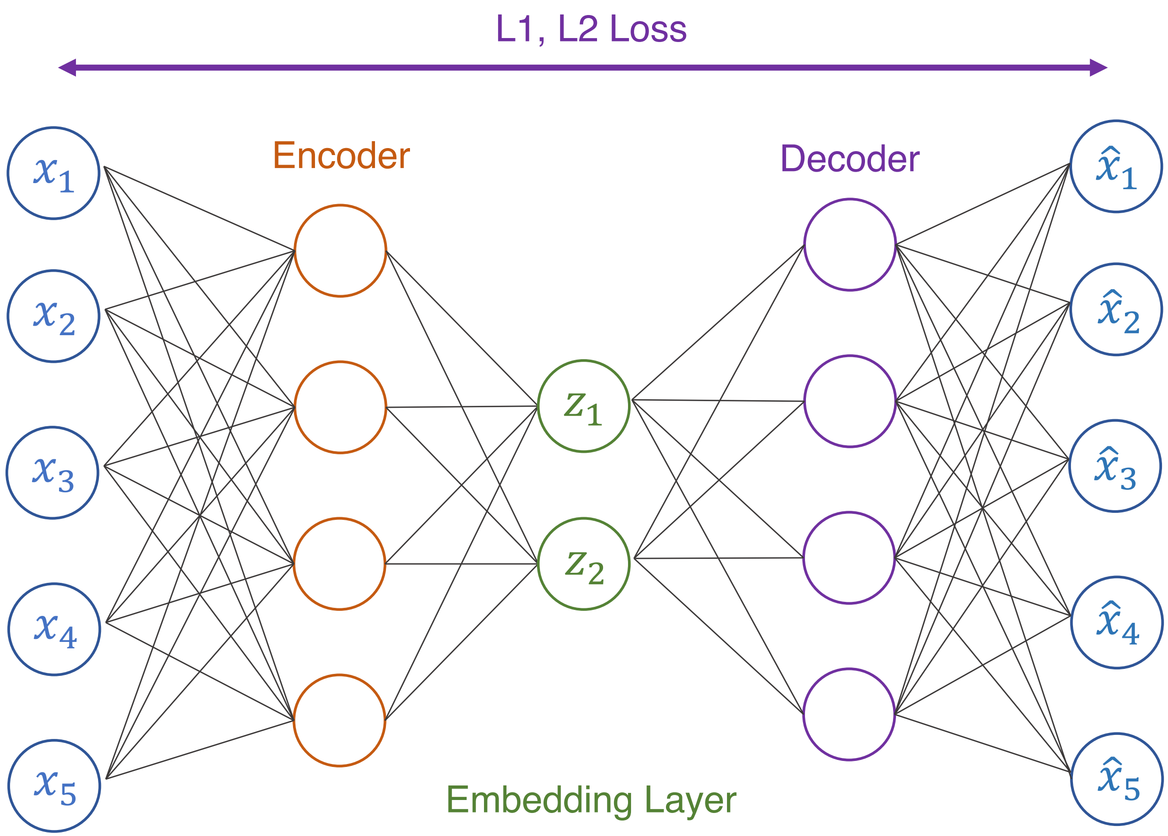
$$\frac{\partial L}{\partial \boldsymbol{z_1}} = \frac{\partial L}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z_1}} = \left(\frac{\partial L}{\partial \boldsymbol{z_2}}\underbrace{\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{a}}}_{w_3} + \frac{\partial L}{\partial \boldsymbol{z_3}}\underbrace{\frac{\partial \boldsymbol{z_3}}{\partial \boldsymbol{a}}}_{w_4}\right)\underbrace{\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z_1}}}_{I\{z_1 > 0\}}$$

Because L is a function of z_2 and z_3, and z_2 and z_3 are functions of a

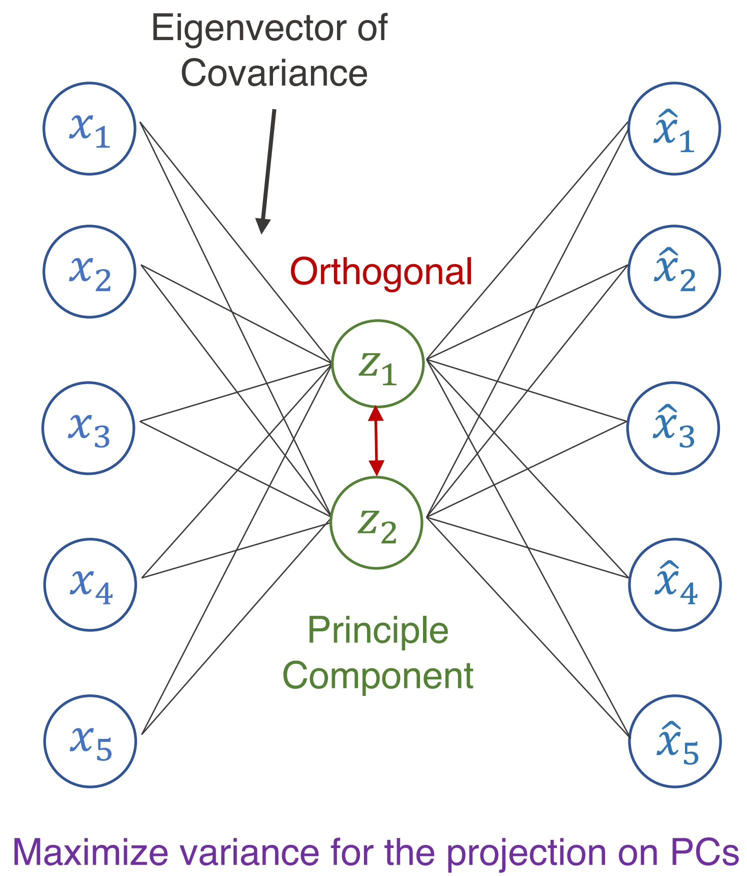
3. Now we can compute the derivative for L w.r.t $\boldsymbol{w}$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w}$$

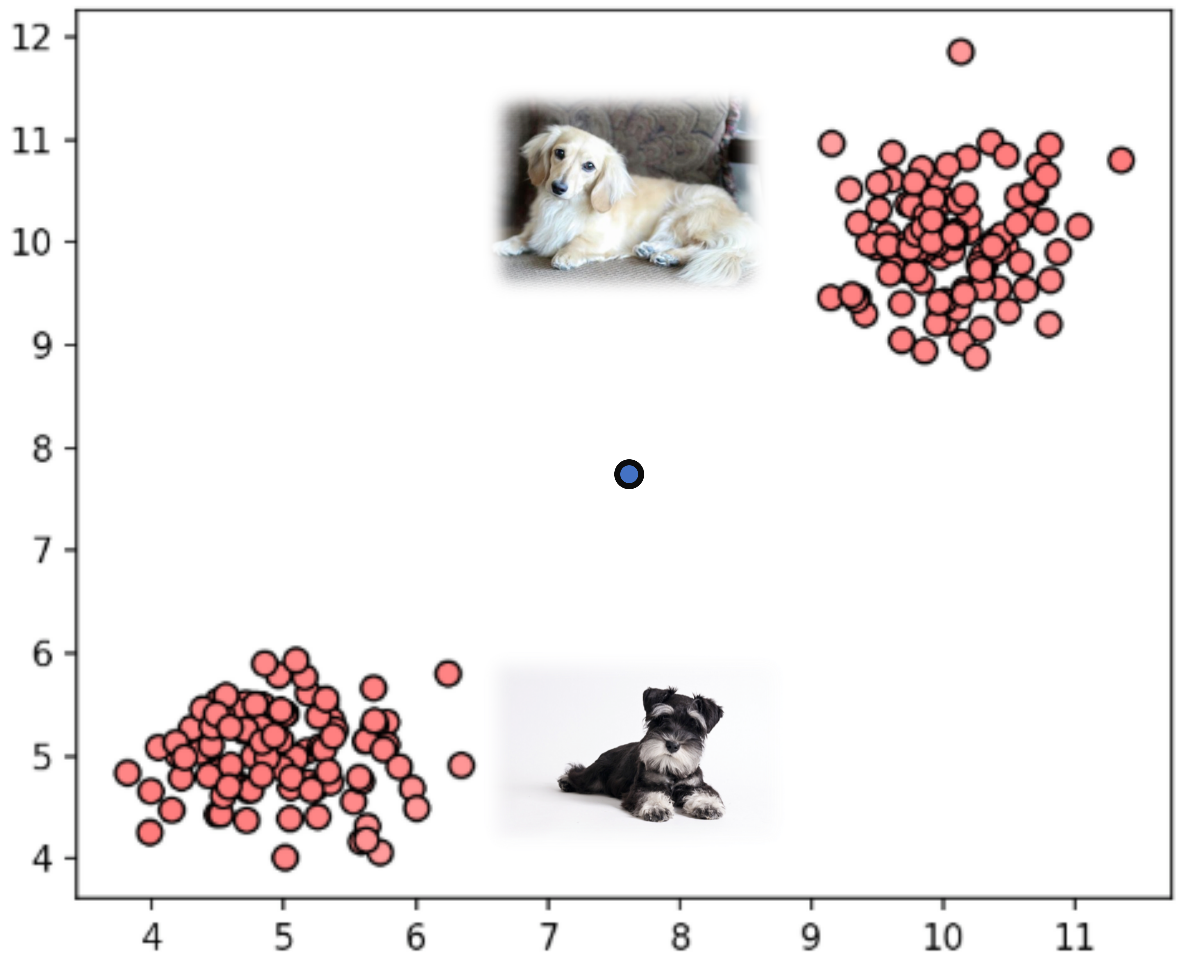
Autoencoder



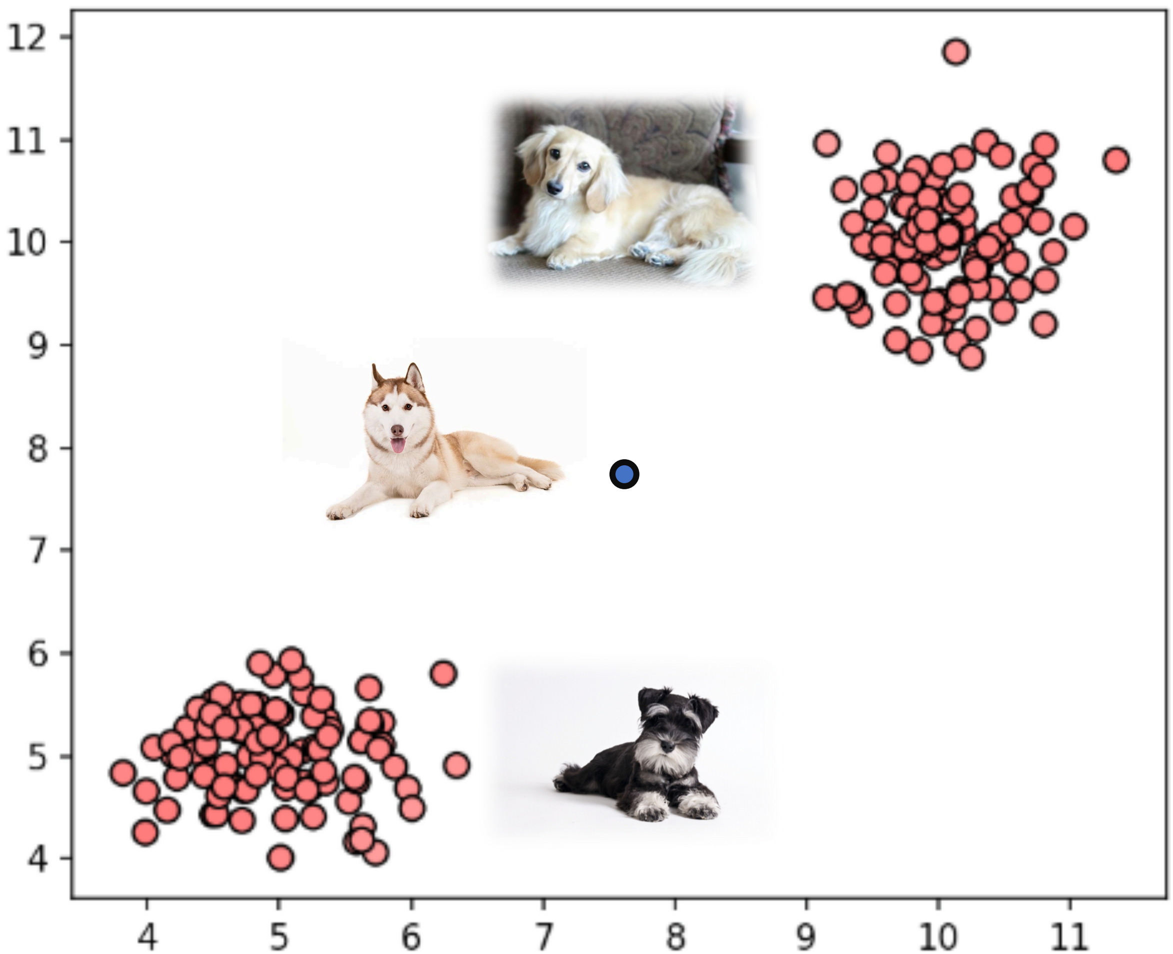
Principle Component Analysis



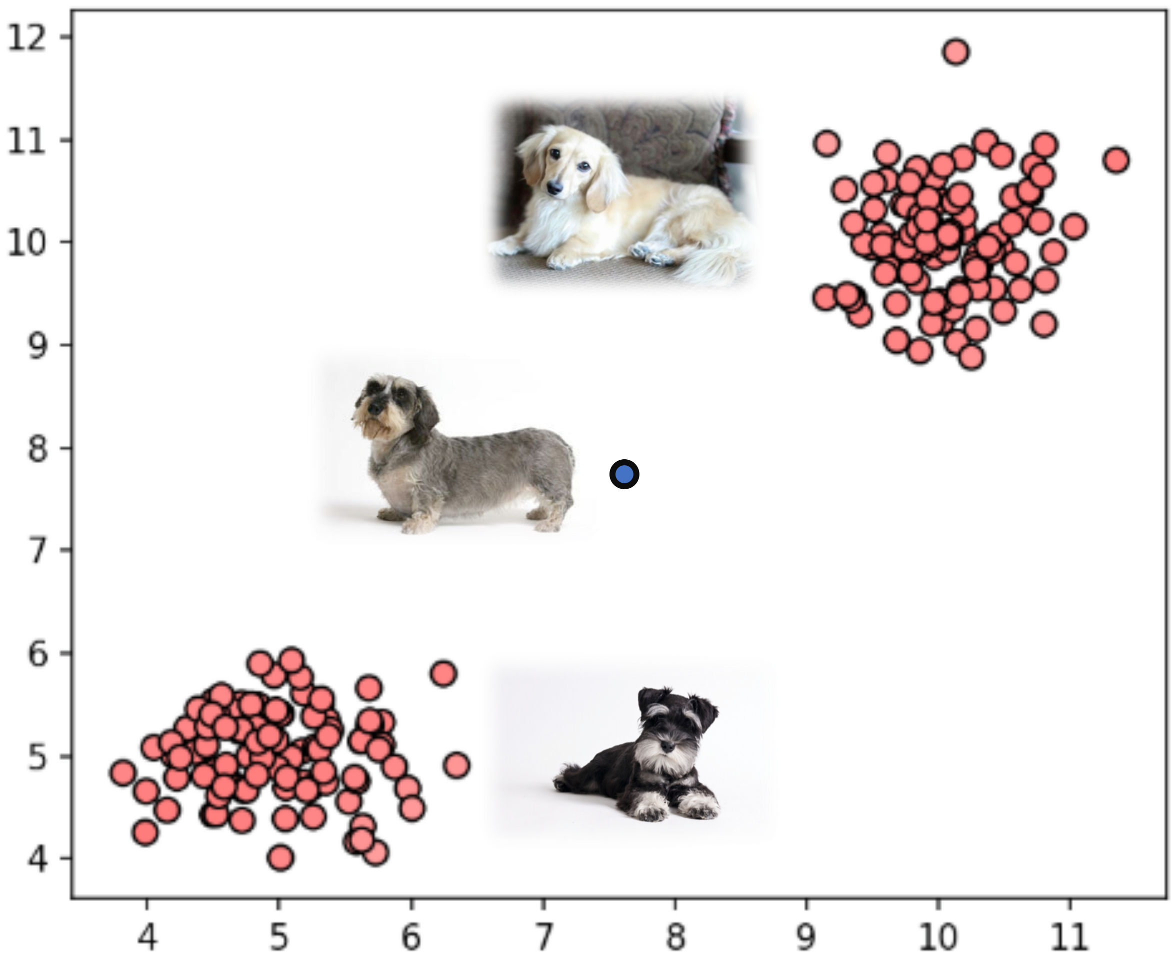
Limitation of Autoencoder



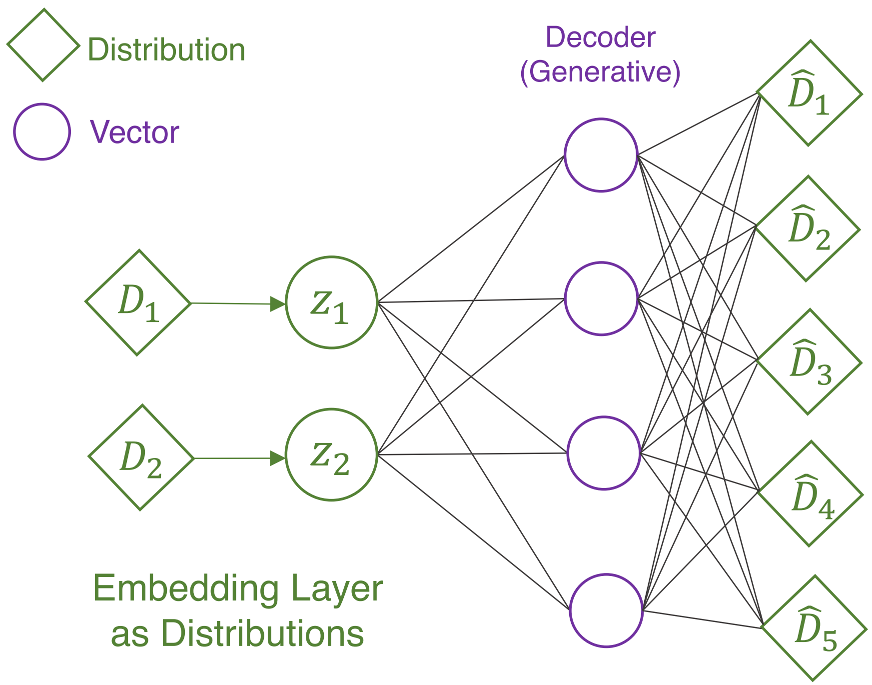
Limitation of Autoencoder



Limitation of Autoencoder



Reconstruct Data from Code

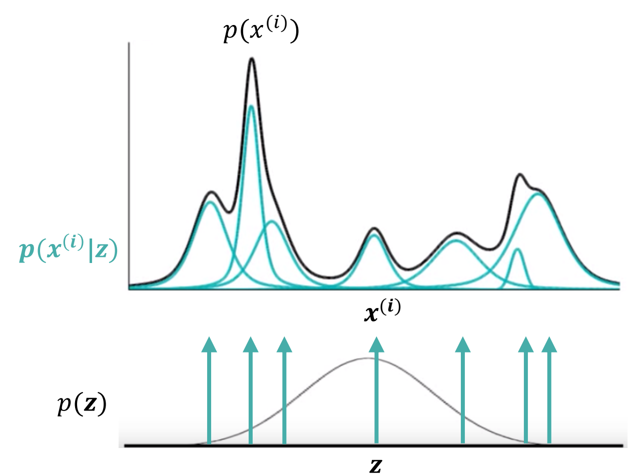


Auto-encoding Variational Bayes
Problem Formation

- Assumptions
  - Suppose $\boldsymbol{x}^{(\boldsymbol{i})}$ is generated from a random process.
  - This random process involve an unobserved continuous random variable $\boldsymbol{z}$
  - $\boldsymbol{z}$ is generated through a random (Gaussian) process $p_{\theta^*}(\boldsymbol{z})$
  - $\boldsymbol{x}^{(\boldsymbol{i})}$ is generated from $p_{\theta^*}(x|z)$ (The distribution of x, given the latent variable z)
  - $\boldsymbol{z}$ and $\boldsymbol{\theta}^*$ are unknown
  - $p_\theta(x) = \int p_\theta(z)\, p_\theta(x|z)dz$ is intractable since we have infinite possibility of $\boldsymbol{z}$
  - Therefore the posterior $p_\theta(z|x) = \int p_\theta(x|z)p_\theta(z)\,/p_\theta(x)dz$ is also intractable

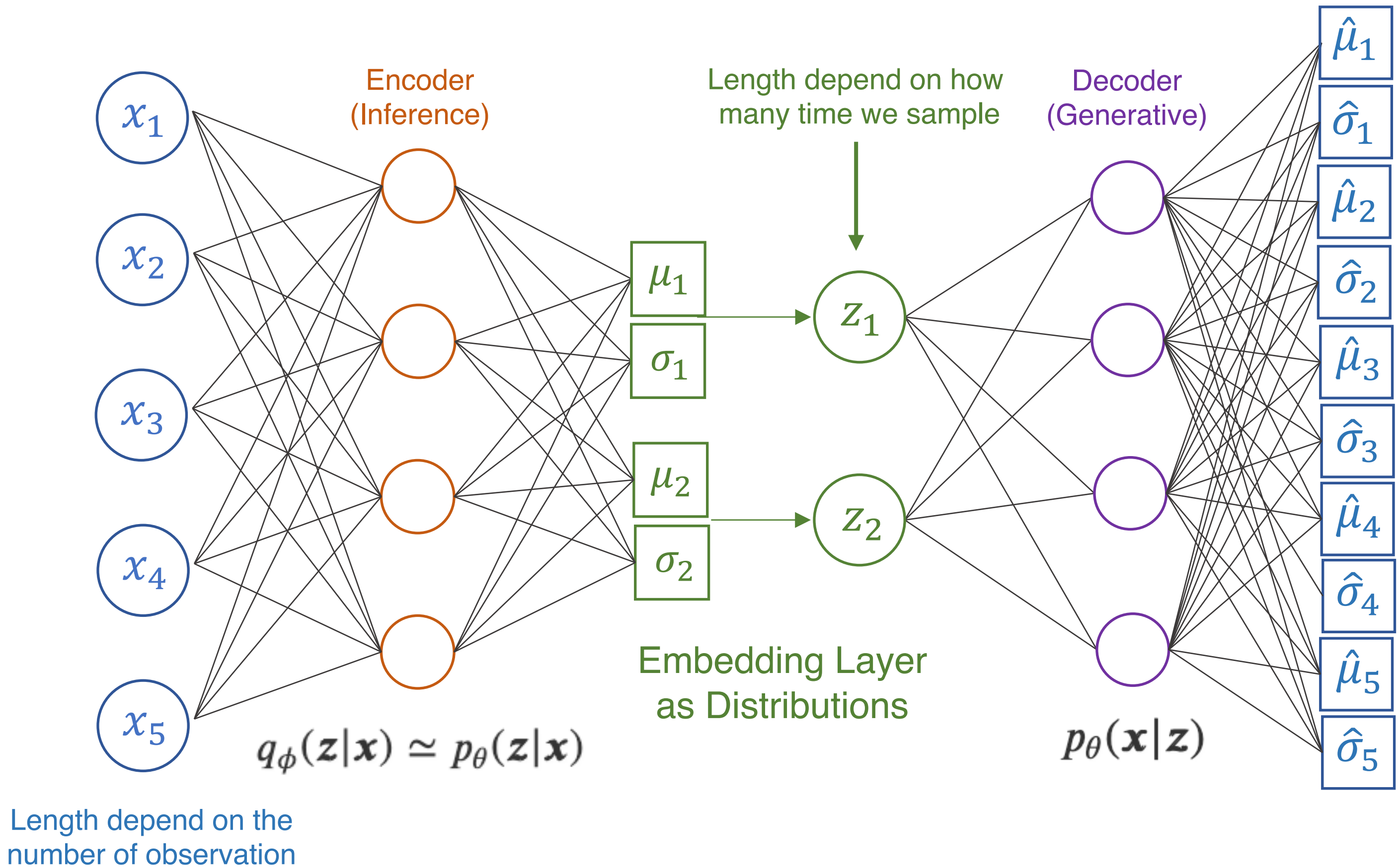
- Problems
  - Estimate the true parameter of θ^* efficiently
  - But we don't know how to optimize the problem

- Consider using autoencoder
  - Let encoder function to be $q_\varphi \cong p_\theta(z|x)$
  - Let decoder function to be $p_\theta(x|z)$



from Hung-Yi Lee

Embedding Layer as Distributions



Auto-encoding Variational Bayes

Methods (1)

- For any PDF, we can maximize the likelihood of this function with available data

$$
\begin{aligned}
\log p(x) &= \log p(x) \int_z q(z|x)dz \quad \text{(Integral of any PDF equals to 1)} \\
&= \int_z q(z|x)\log p(x)dz \\
&= \int_z q(z|x)\log(\frac{p(z,x)}{p(z|x)})dz \quad \text{(Bayes theorem)} \\
&= \int_z q(z|x)\log(\frac{p(z,x)}{q(z|x)}\frac{q(z|x)}{p(z|x)})dz \\
&= \int_z q(z|x)\log(\frac{p(z,x)}{q(z|x)})dz + \int_z q(z|x)\log(\frac{q(z|x)}{p(z|x)})dz
\end{aligned}
$$

- Let $L_b = \int_z q(z|x)\log(\frac{p(z,x)}{q(z|x)})dz$

  $D_{KL}(q(z|x) \mid\mid p(z|x)) = \int_z q(z|x)\log(\frac{q(z|x)}{p(z|x)})dz$ ← Intractable, but D_{KL} always ≥ 0

- Then $\log p(x) = L_b + D_{KL}(q(z|x) \mid\mid p(z|x)) \geq L_b$ Evidence lower bound (ELBO)

- Rewrite

$$
\begin{aligned}
L_b &= \int_z q(z|x)\log(\frac{p(z,x)}{q(z|x)})dz \\
&= \int_z q(z|x)\log(\frac{p(x|z)p(z)}{q(z|x)})dz \\
&= \int_z q(z|x)\log p(x|z)dz + \int_z q(z|x)\log(\frac{p(z)}{q(z|x)})dz \\
&= \int_z q(z|x)\log p(x|z)dz - D_{KL}(q(z|x) \mid\mid p(z))
\end{aligned}
$$

Auto-encoding Variational Bayes

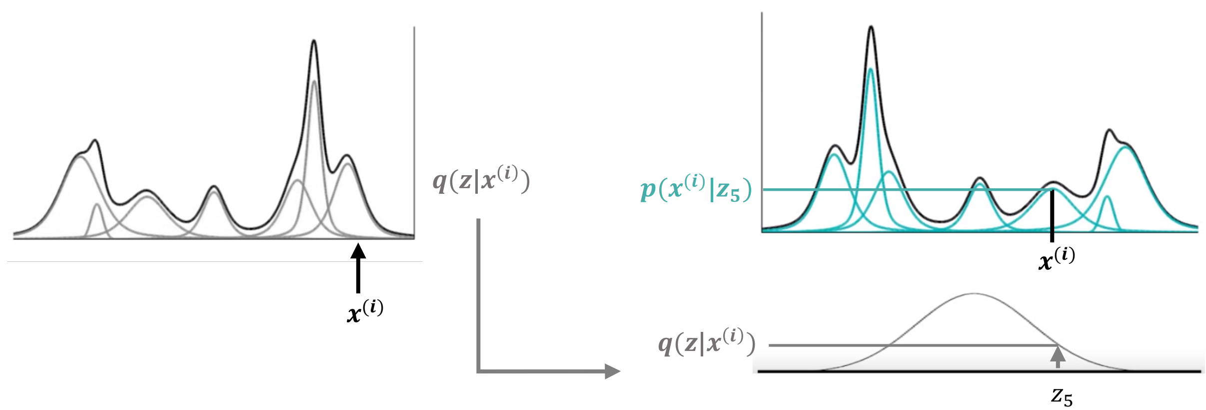
Methods (2)

Evidence lower bound

- Rewrite $\log p(x) = \int_z q(z|x) \log p(x|z) dz - D_{KL}(q(z|x) \,||\, p(z)) + D_{KL}(q(z|x) \,||\, p(z|x))$

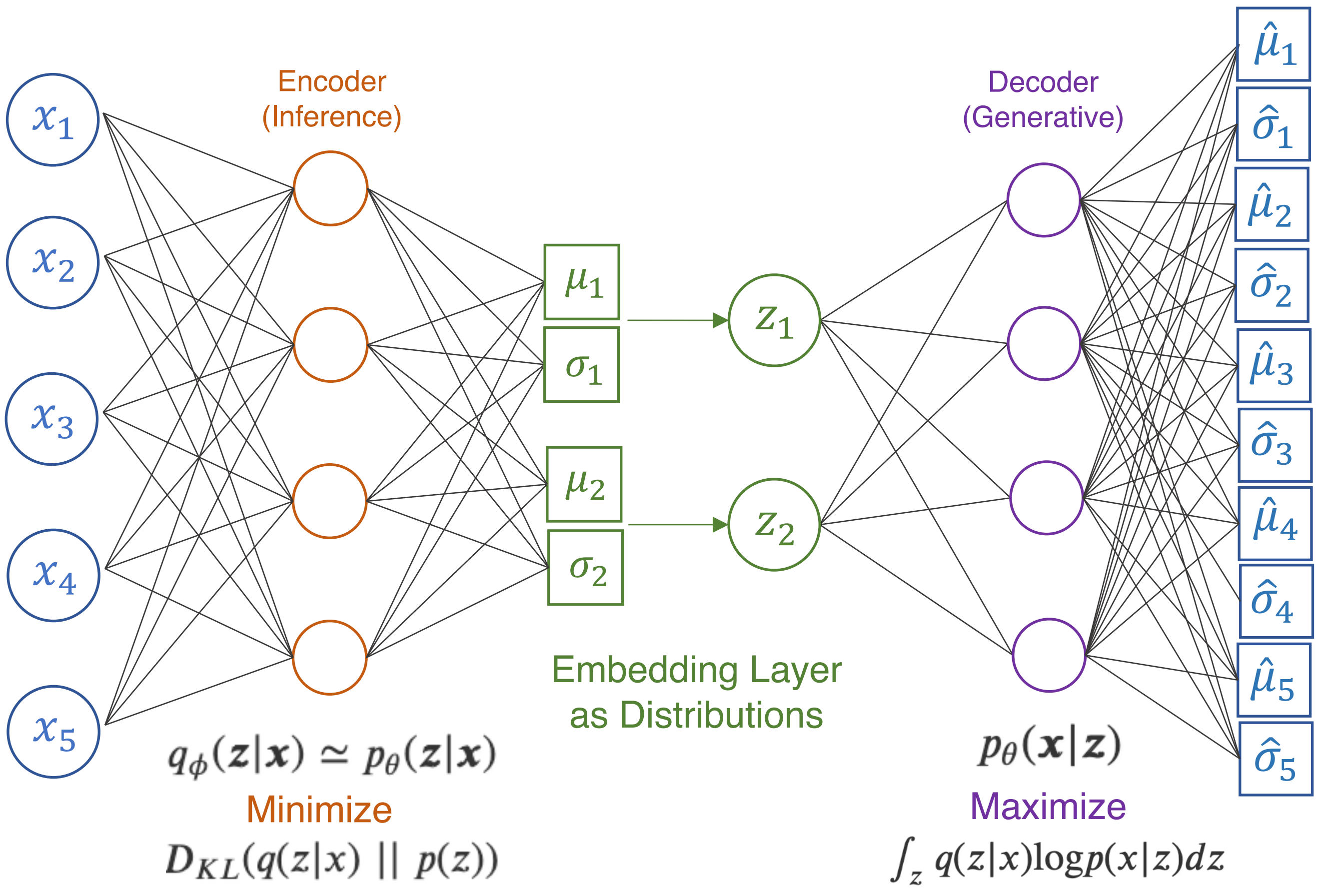
The contribution of this paper Has solution (*) Intractable

- We can maximize the evidence lower bound and "attempt to" maximize the likelihood of $p(x)$
- $-D_{KL}(q(z|x) \,||\, p(z)) = \sum_{j=1}^{J} (\sigma_j)^2 - (1 + \log(\sigma_j)) + (\mu_j)^2$ (Appendix B)
- Try Monte Carlo estimator to maximize $\int_z q(z|x) \log p(x|z) dz$ ⟶ Exhibit high variance



Auto-encoding Variational Bayes

Methods (3)



Auto-encoding Variational Bayes

Methods (4)

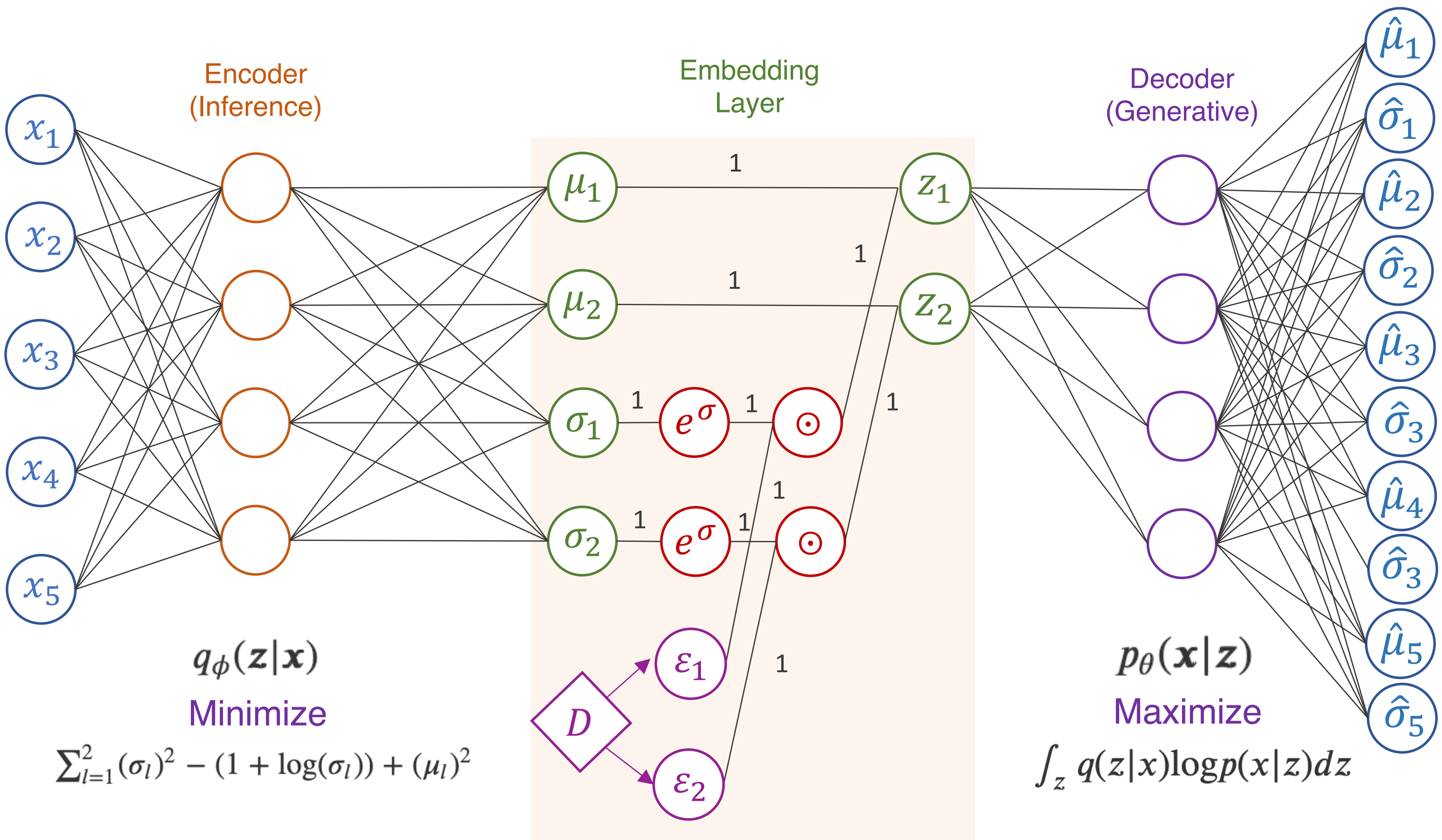
- Instead of output the conditional distribution from the encoder function, we can output a function for $\boldsymbol{z}$.

$$\boldsymbol{z}_{\boldsymbol{j}}^{(\boldsymbol{l})} = \boldsymbol{\varepsilon}_{\boldsymbol{j}}^{(\boldsymbol{l})} \odot \exp(\boldsymbol{\sigma}_{\boldsymbol{j}}) + \boldsymbol{\mu}_{\boldsymbol{j}}, \text{ where } \varepsilon \sim N(0, I)$$

- The variables σ_j and μ_j are the output from the encoder function. In other word, they are a function of x
- ε is a random variable sampled from a standard distribution (now it is independent from the parameter φ)
- l is the index of the sampling process.
- The reparameterization alleviate the problem of high variance using Monte Carlo estimator.
- The authors did not mention why it help.

Auto-encoding Variational Bayes

Methods (5)



$$\boldsymbol{z}_j^{(l)} = \boldsymbol{\varepsilon}_j^{(l)} \odot \exp(\boldsymbol{\sigma}_j) + \boldsymbol{\mu}_j$$

In practice we only sample ε once for each observation

Methods

Methods

$$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}$$

Cell types

Genes

- Model the observed expression with a conditional negative binomial distribution

- Recall that negative binomial distribution can be written as Gamma-Poisson (mixture) distribution:

$$NB(k|r,p) = \int \underbrace{f_{Poisson(\lambda)}(k)}_{\text{sampling}} \underbrace{f_{Gamma\left(r,\frac{1-p}{p}\right)}(\lambda)}_{\text{technical noise}} d\lambda$$

- Combine $y \sim Poisson(v, s)$ and $v \sim Gamma(\exp(\mu), \frac{1}{\varphi})$

$$y \sim NB(\exp(\mu), s, \frac{1}{\varphi})$$

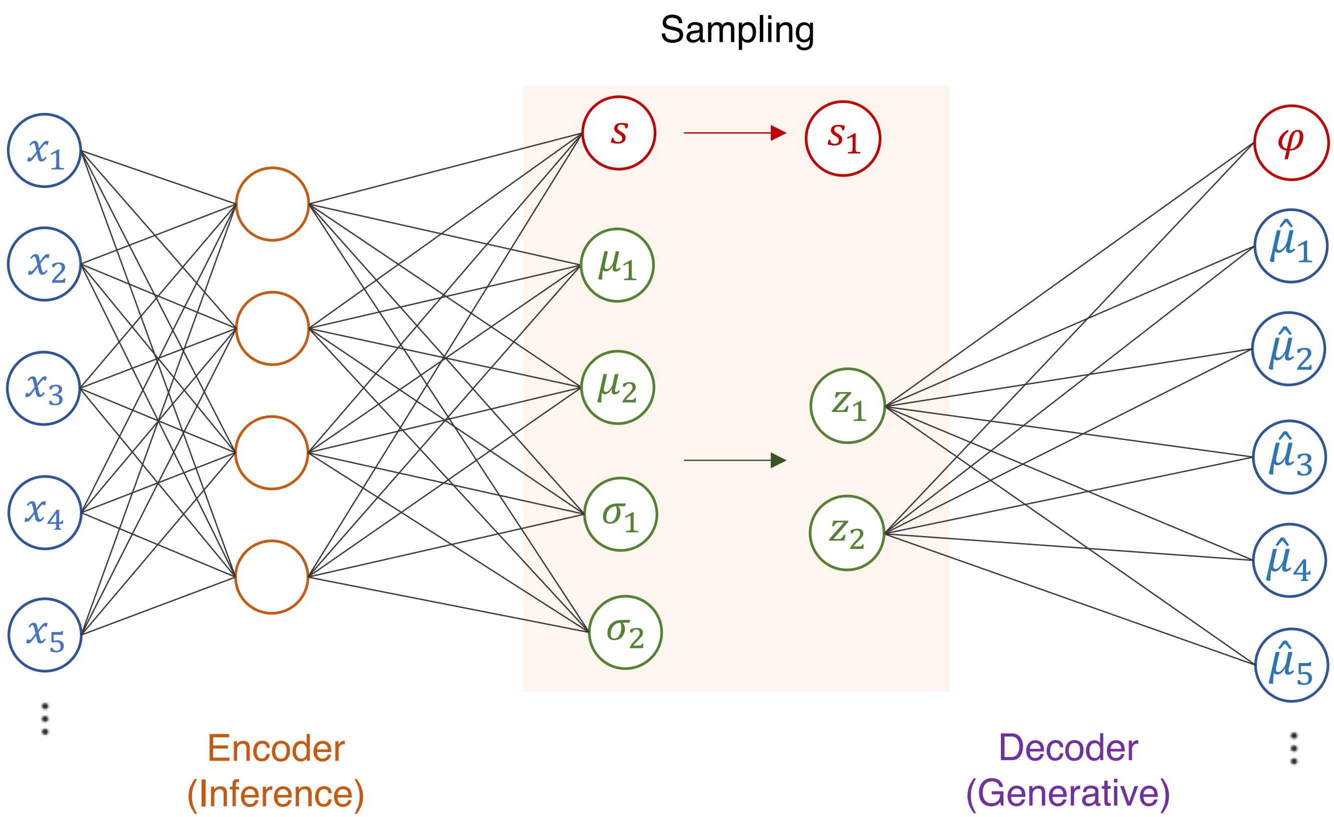
- Here, $r = \exp(\mu_n^g)$ and $p = \varphi$ are latent variables

- Therefore, the generative model condition on latent variable μ_n^g, φ^g, s_n^g

- Also, the authors set the decoder to be a linear function $\mu_n^g = z_n W^T$

W is the weight in decoder function

Methods

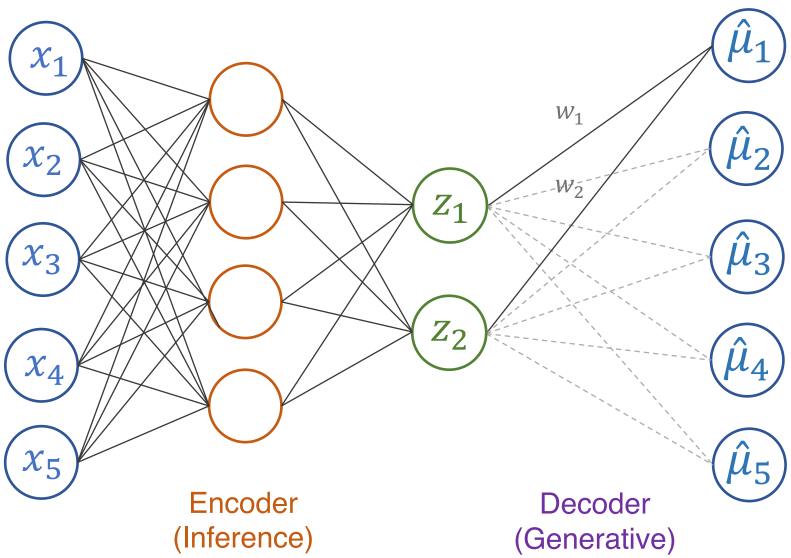


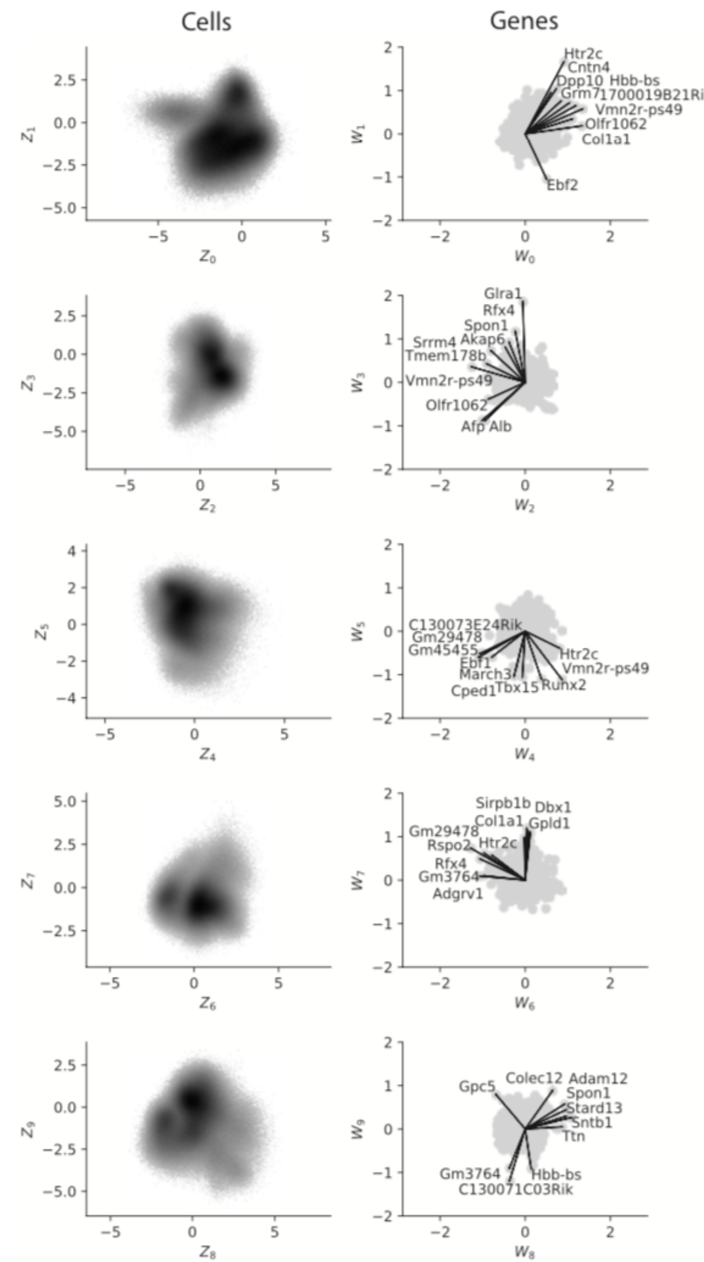
Linear Decoder Variational Autoencoder

Results

Results

- Apply on developing mouse embryos in different stages of development (Cao *et al.*, 2019)
- The variation in latent variable ***z*** can directly related to variation in gene expression





Discussion

Discussion

- Manuscripts
  - This is an additional feature added to scVI package
  - Promote the concept of extend functions on existing frameworks in bioinformatics
  - Enable interpretable analysis of data at massive scale
- Something more
  - The effect of using linear function as decoder
  - Improve the generative model with InfoGAN?

References and Resources

- Neural Network
  - Numerical Optimization by Ben Frederickson
  - Back Propagation by 3Blue1Brown
- Variational Autoencoder
  - Auto-Encoding Variational Bayes, arXiv 2014.
  - Stanford CS231n: Convolutional Neural Networks for Visual Recognition, Generative Model
  - Stanford CS228: Probabilistic Graphical Models, Variational Autoencoder
  - The Reparameterization Tricks by JP Zhang
- Generative Model on scRNA
  - Deep generative modeling for single-cell transcriptomics, Nature Methods 2018.

Thanks