香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

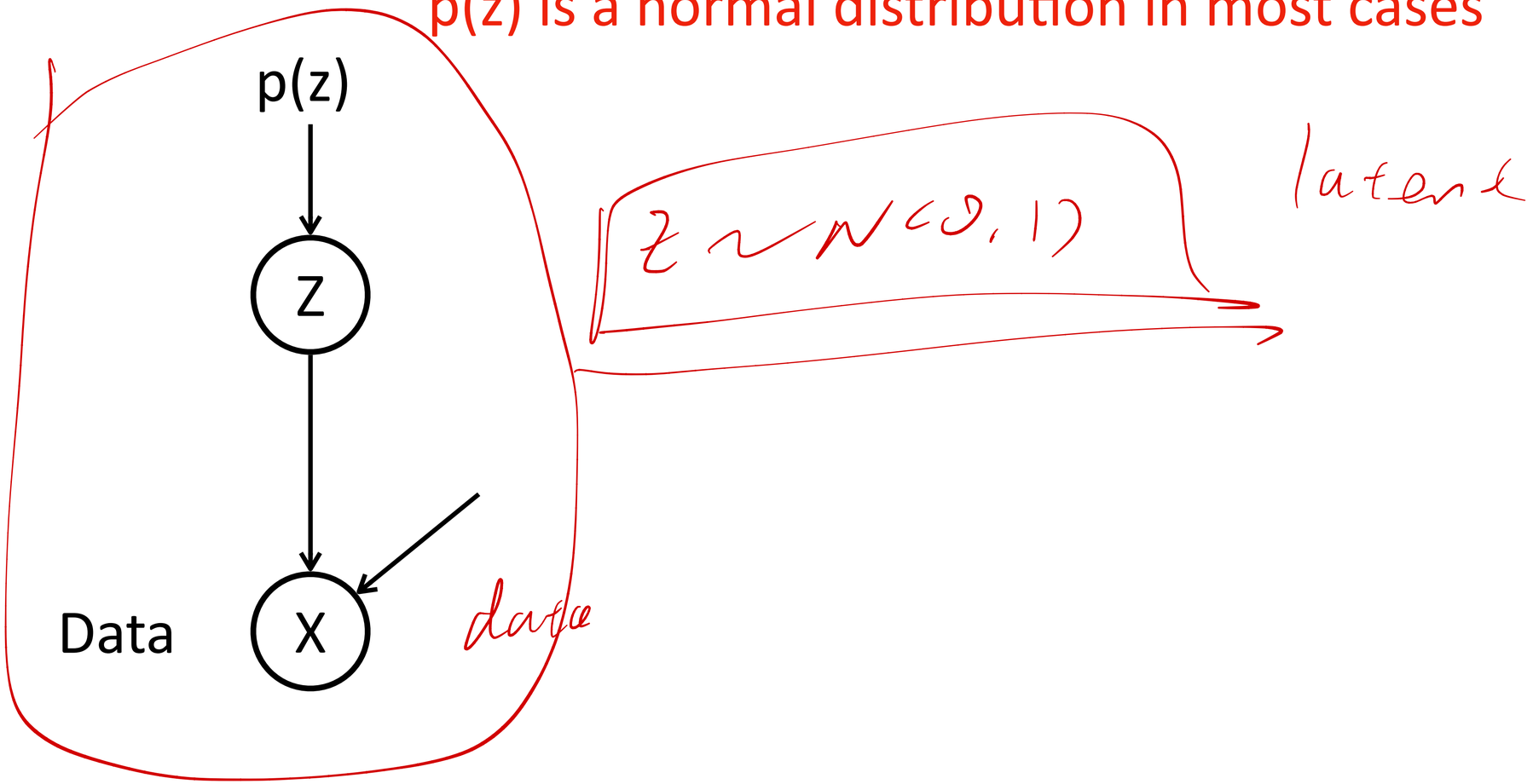
COMP 5212
Machine Learning
Lecture 14

Variational Autoencoders, Hidden Markov Models

Junxian He
Apr 2, 2026

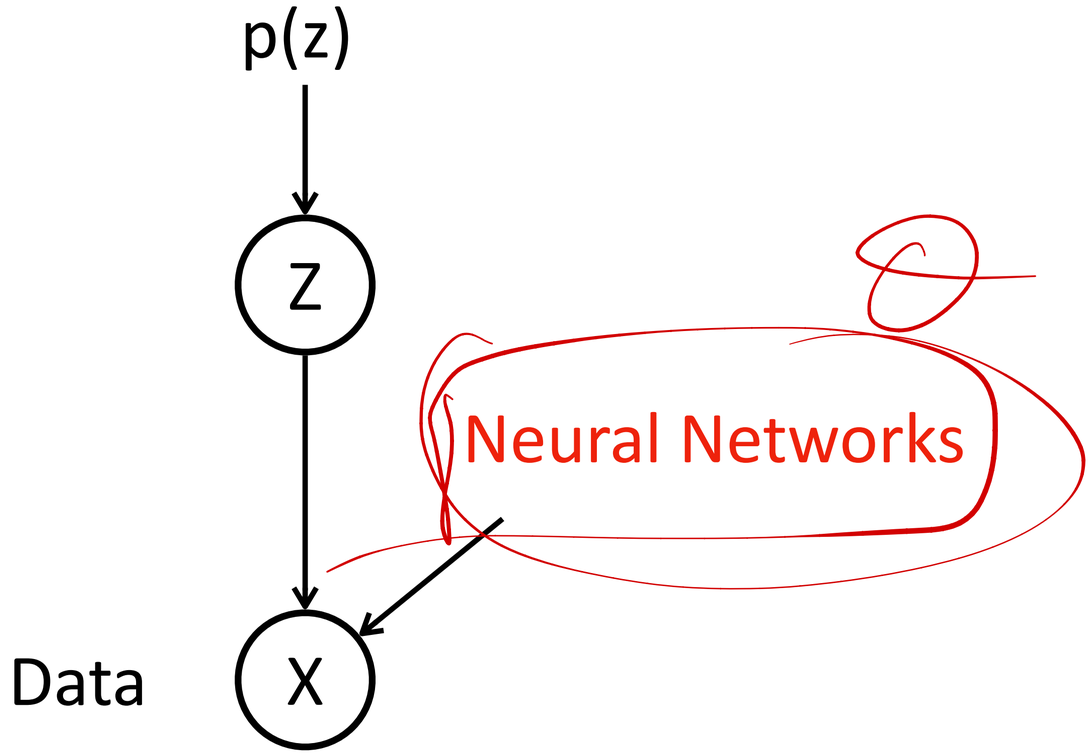
Recap: The VAE Model

p(z) is a normal distribution in most cases



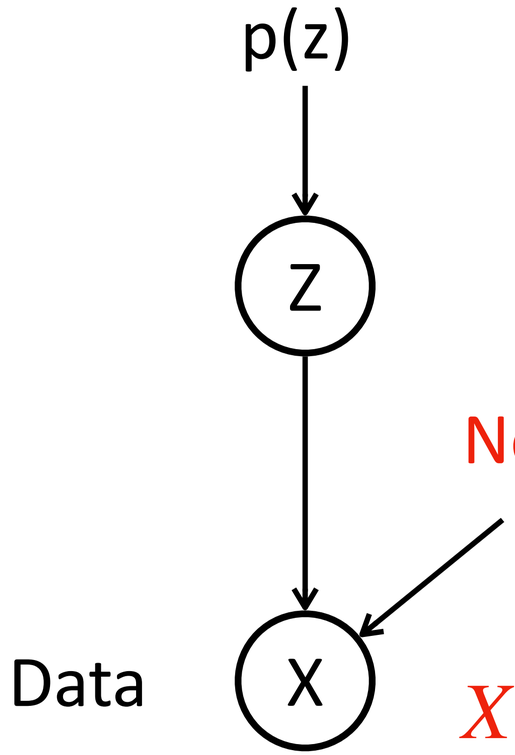
Recap: The VAE Model

p(z) is a normal distribution in most cases



Recap: The VAE Model

p(z) is a normal distribution in most cases



Neural Networks

$x \sim N(\mu = f(z;\theta), I)$

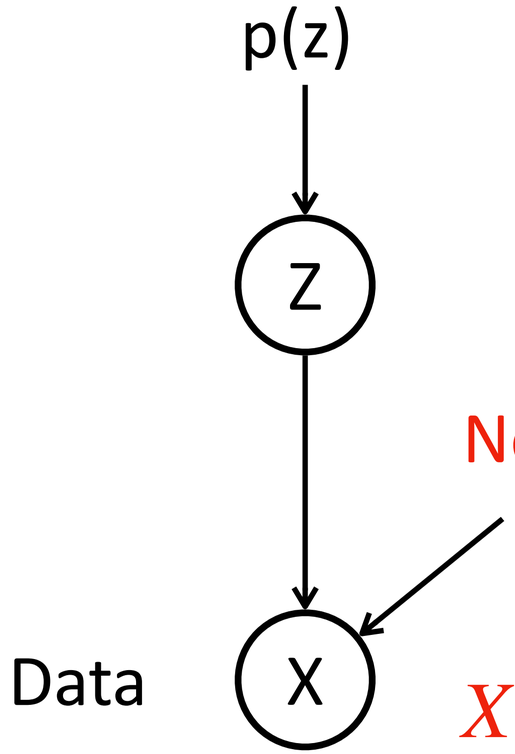
$f(z;\theta)$

$X \sim P(x, f(z;\theta))$

N
Categorical

Recap: The VAE Model

p(z) is a normal distribution in most cases

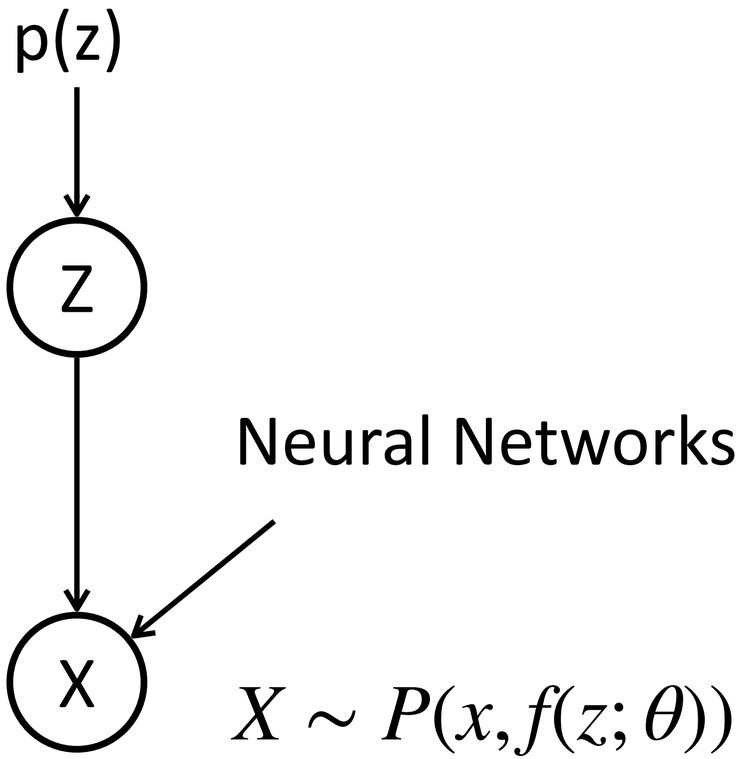


Neural Networks

$$X \sim P(x, f(z;\theta))$$

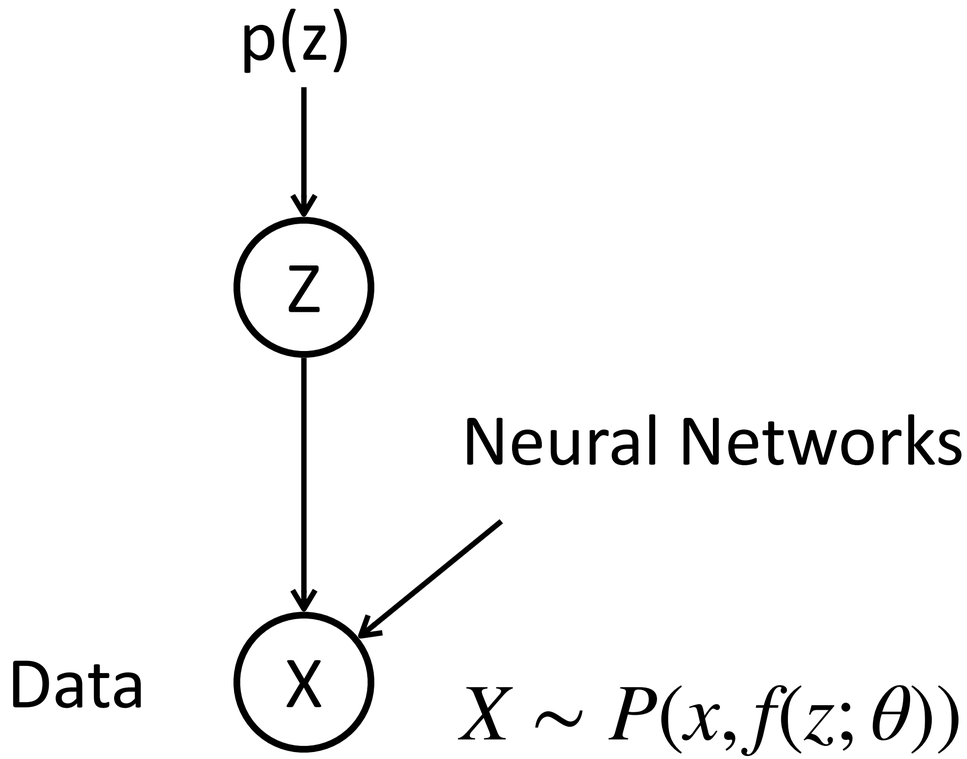
f is a neural network taking Z as input

Training



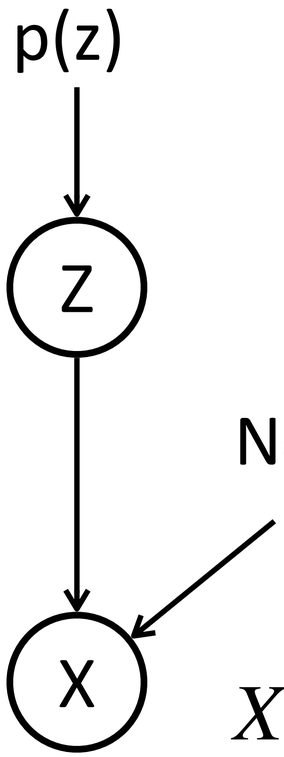
$$X \sim P(x, f(z; \theta))$$

Training



How to train the model? Can we do MLE?

Training



Neural Networks

Data

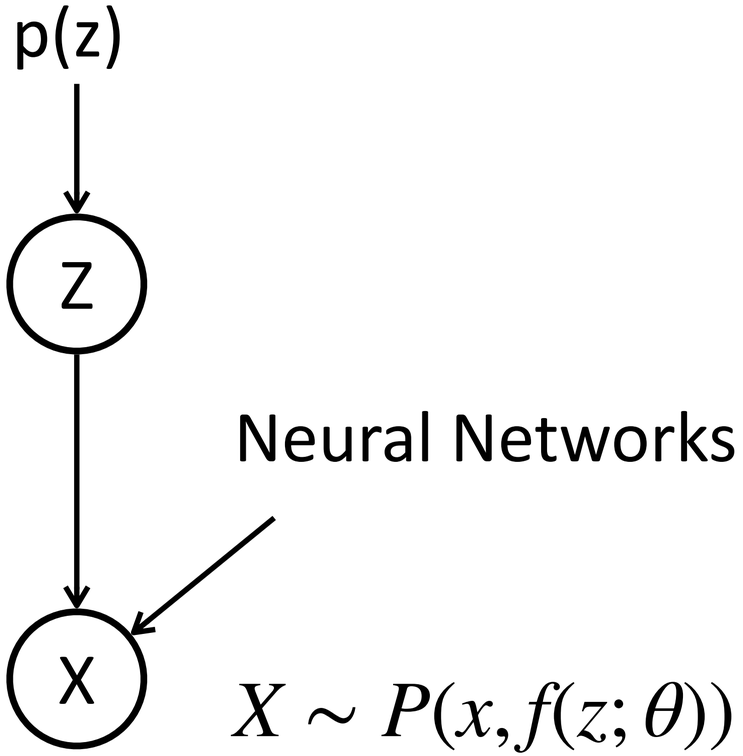
$$X \sim P(x, f(z;\theta))$$

How to train the model? Can we do MLE?

Intractable P(X), EM algorithm?

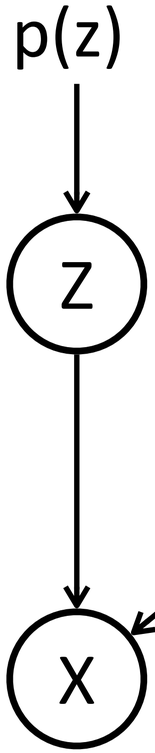
$$\int_z P(x,z)\,dz$$

Let's try EM



$$X \sim P(x, f(z; \theta))$$

Let's try EM



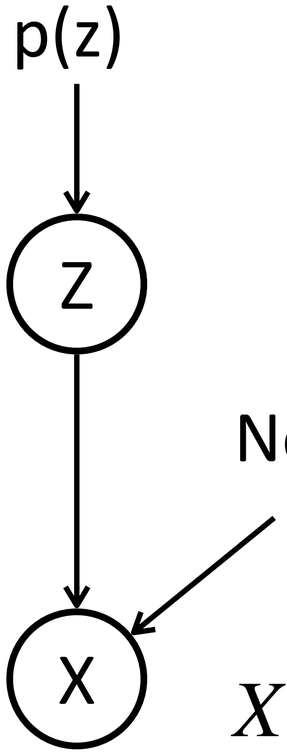
Neural Networks

$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

$$Q(z) = P(z \mid x) \propto P(z)P(x \mid z)$$

Let's try EM



Neural Networks

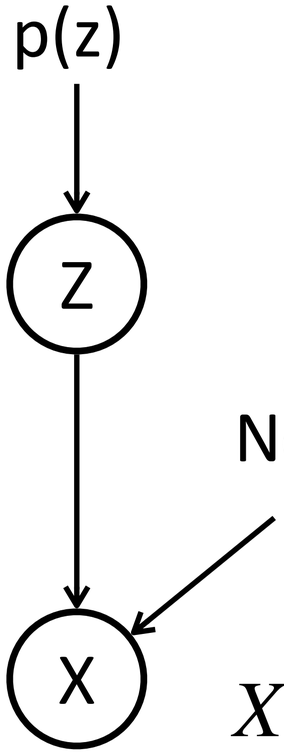
$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

$$Q(z) = P(z \mid x) \propto P(z)P(x \mid z)$$

This is ok?

Let's try EM



Neural Networks

$$X \sim P(x, f(z;\theta))$$

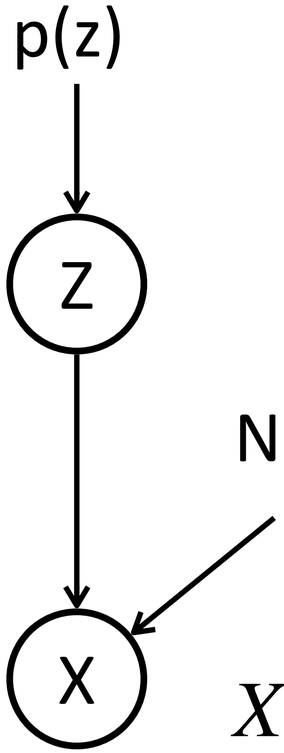
E-Step: compute P(z|x)

$$Q(z) = P(z \mid x) \propto P(z)P(x \mid z)$$ This is ok?

M-Step: the ELBO objective

$$\text{argmax}_\theta \sum_z Q(z) \log p(x, z; \theta) = \text{argmax}_\theta \mathbb{E}_{z \sim Q(z)} \log p(x, z; \theta)$$

Let's try EM



Neural Networks

$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

$$Q(z) = P(z|x) \propto P(z)P(x|z)$$

This is ok?

M-Step: the ELBO objective

$$\text{argmax}_\theta \sum_z Q(z) \log p(x, z;\theta) = \text{argmax}_\theta \mathbb{E}_{z\sim Q(z)} \log p(x, z;\theta)$$

In most cases, we cannot do the sum, and cannot easily sample from Q(z) either

Approximate Posterior

We need an easy-to-sample distribution to approximate $P(z|x)$

Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

$q(z \mid x; \phi)$ to approximate $p(z \mid x; \theta)$

Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

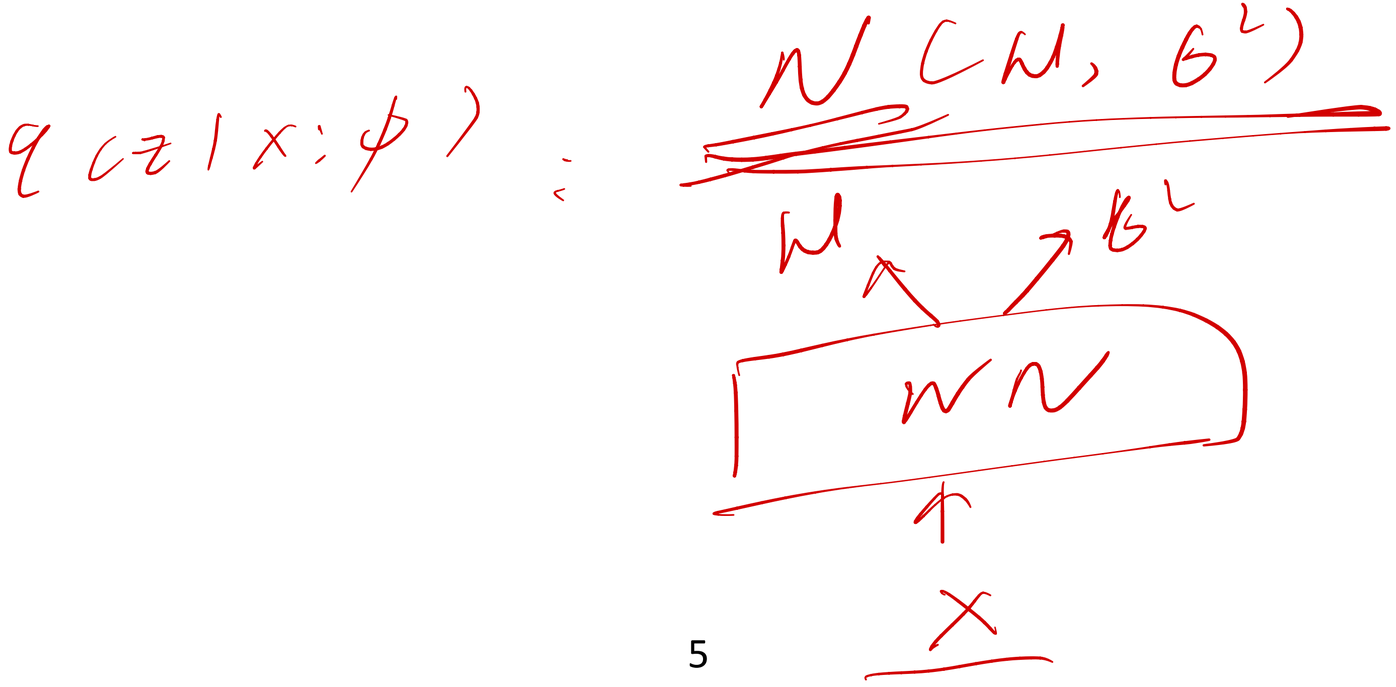
$q(z \mid x; \phi)$ to approximate $p(z \mid x; \theta)$ Why conditioned on x?

Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

$q(z|x;\phi)$ to approximate $p(z|x;\theta)$ Why conditioned on x?

ϕ is the parameter for the approximate function, θ is the generative model parameter



Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

$q(z|x;\phi)$ to approximate $p(z|x;\theta)$ Why conditioned on x?

ϕ is the parameter for the approximate function, θ is the generative model parameter

How to train $q(z|x;\phi)$, what would be the loss to find ϕ?

E step: $q(z|x;\phi) \longrightarrow P(z|x;\theta)$

Recap: ELBO

$$\mathrm{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\mathrm{argmax}_{Q(z)} \mathrm{ELBO}(x; Q, \theta)$?

Recap: ELBO

$$\text{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\text{argmax}_{Q(z)} \text{ELBO}(x; Q, \theta)$?

ELBO is maximized when Q(z) is equal to p(z|x)

$Q(z) = p(z|x)$

Recap: ELBO

$$\text{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\text{argmax}_{Q(z)} \text{ELBO}(x; Q, \theta)$?

ELBO is maximized when Q(z) is equal to p(z|x)

Therefore, we can approximate the true posterior by maximizing ELBO:

$$\text{argmax}_\phi \sum_z q(z|x;\phi) \log \frac{p(x, z; \theta)}{q(z|x;\phi)}$$

constant, not depend Q

$$ELBO = \log P(x;\theta) - KL(Q \| P(z|x;\theta))$$

$$\arg\max_{Q} ELBO = \arg\min_{Q} KL(Q \| P(z|x;\theta))$$

Recap: ELBO

$$\mathrm{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\text{argmax}_{Q(z)} \mathrm{ELBO}(x; Q, \theta)$?

ELBO is maximized when Q(z) is equal to p(z|x)

Therefore, we can approximate the true posterior by maximizing ELBO:

$$\text{argmax}_\phi \sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Variational Inference

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Same objective, different parameters to optimize

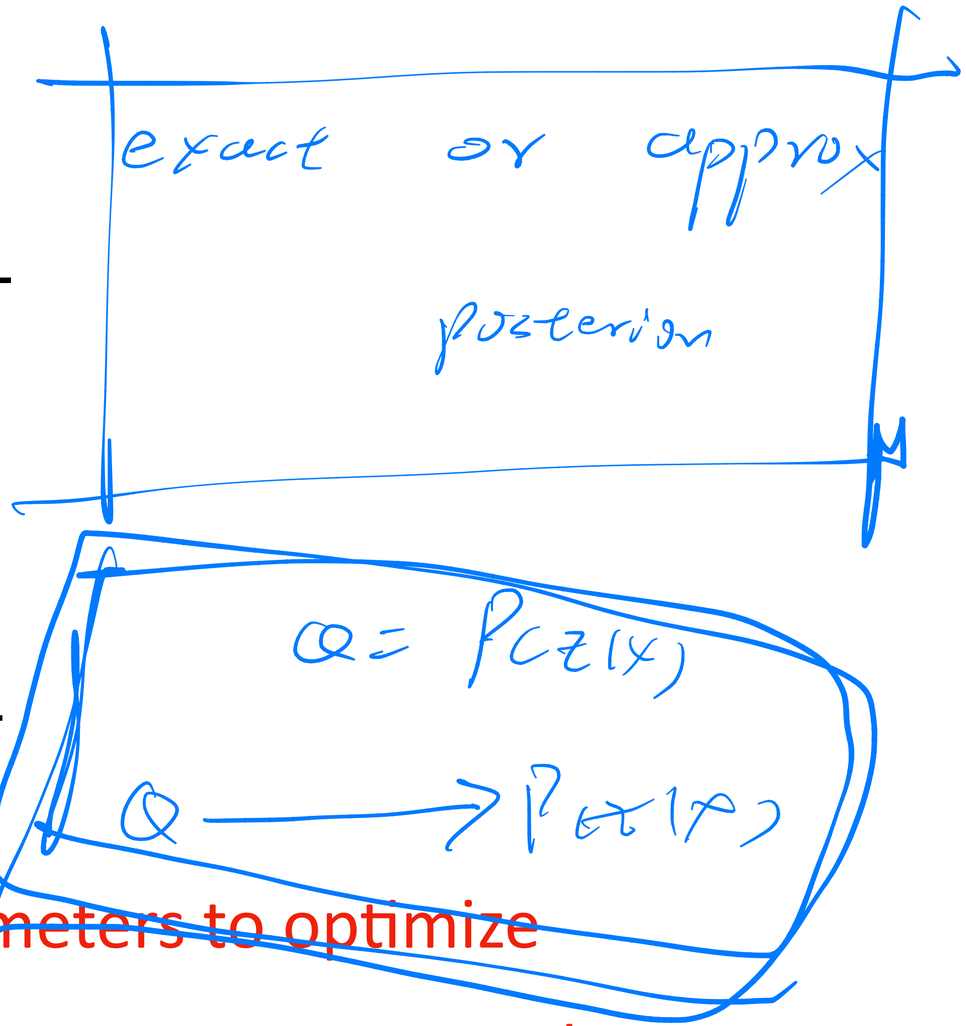
Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$



Same objective, different parameters to optimize

Because we use approximate rather than exact posterior, it is also called Variational EM

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Same objective, different parameters to optimize

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Same objective, different parameters to optimize

Because we use approximate rather than exact posterior, it is also called Variational EM

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

We use MC sampling to approximate expectation and use gradient descent to optimize θ

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Can we do gradient descent over ϕ?

M-Step:

$$\text{argmax}_{\theta} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

We use MC sampling to approximate expectation and use gradient descent to optimize θ

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

First, we cannot do sum, but we can sample z_i from $q(z \mid x; \phi)$, which depends on ϕ, how do we propagate gradients to ϕ?

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

First, we cannot do sum, but we can sample z_i from $q(z \mid x; \phi)$, which depends on ϕ, how do we propagate gradients to ϕ?

Try to express z as a deterministic function $z = g_{\phi}(\epsilon, x)$, where ϵ is an auxiliary random variable

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

First, we cannot do sum, but we can sample z_i from $q(z \mid x; \phi)$, which depends on ϕ, how do we propagate gradients to ϕ?

Try to express z as a deterministic function $z = g_{\phi}(\epsilon, x)$, where ϵ is an auxiliary random variable

$$z \sim N(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim N(0,1)$$

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

First, we cannot do sum, but we can sample z_i from $q(z \mid x; \phi)$, which depends on ϕ, how do we propagate gradients to ϕ?

Try to express z as a deterministic function $z = g_{\phi}(\epsilon, x)$, where ϵ is an auxiliary random variable

$$z \sim N(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim N(0,1)$$

Can you verify z in this equation is Gaussian?

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

1. Randomly sample $\epsilon^{(i)} \sim N(0,1)$

Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

1. Randomly sample $\epsilon^{(i)} \sim N(0,1)$
2. Obtain z sample as $z^{(i)} = \mu + \sigma \odot \epsilon^{(i)}$

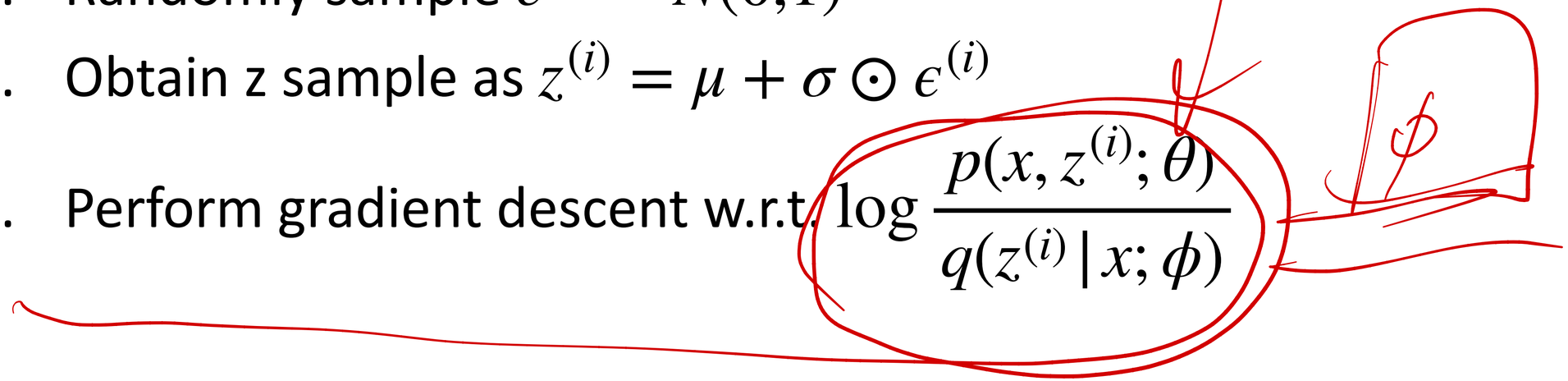
Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

1. Randomly sample $\epsilon^{(i)} \sim N(0,1)$
2. Obtain z sample as $z^{(i)} = \mu + \sigma \odot \epsilon^{(i)}$
3. Perform gradient descent w.r.t. $\log \frac{p(x, z^{(i)}; \theta)}{q(z^{(i)} \mid x; \phi)}$



Reparameterization Trick

E-Step:

$$\text{argmax}_{\phi} \sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

For every gradient step (assuming batch size=1):

1. Randomly sample $\epsilon^{(i)} \sim N(0,1)$
2. Obtain z sample as $z^{(i)} = \mu + \sigma \odot \epsilon^{(i)}$
3. Perform gradient descent w.r.t. $\log \frac{p(x, z^{(i)}; \theta)}{q(z^{(i)} \mid x; \phi)}$

We can now propagate gradients from z to ϕ

ELBO

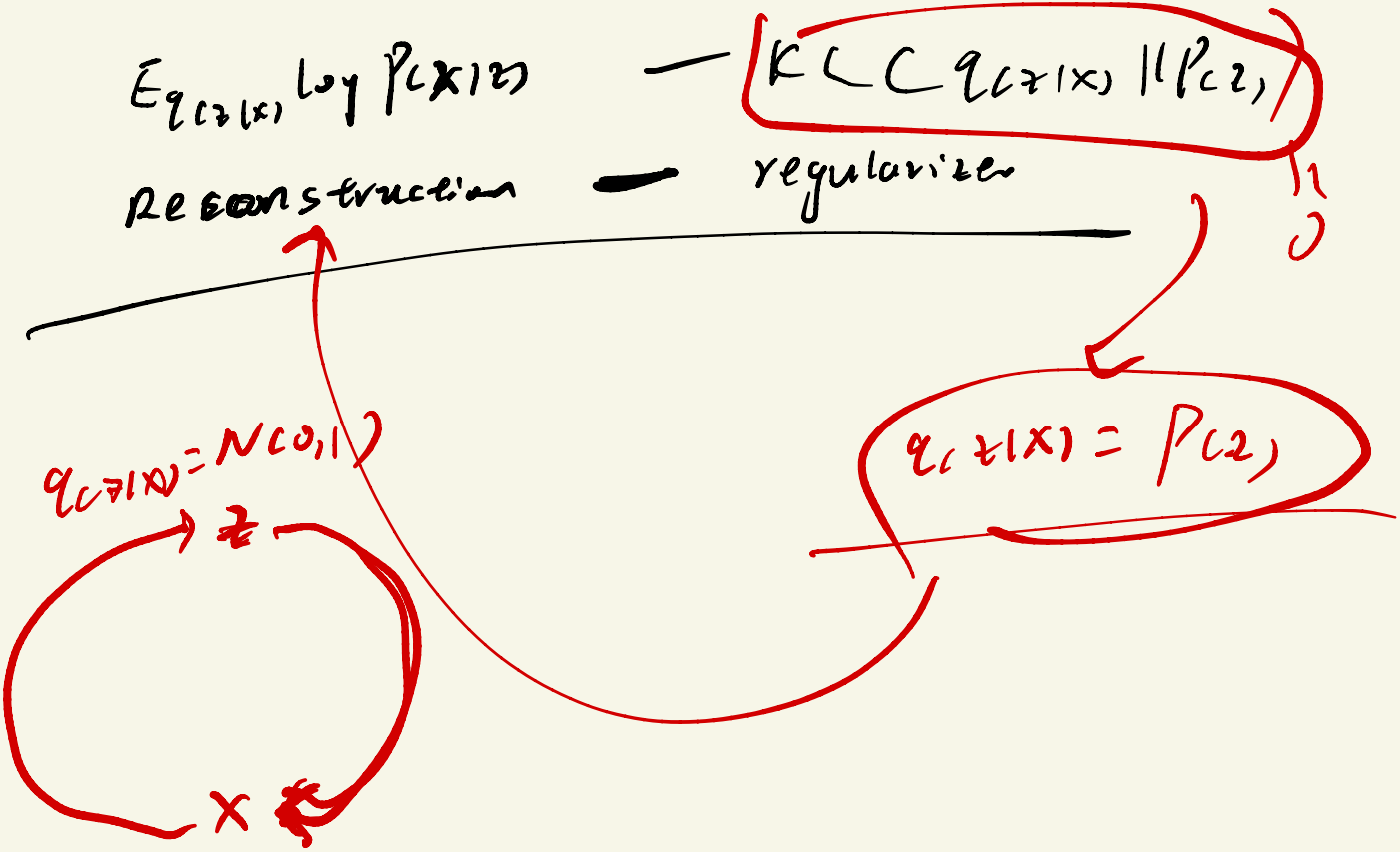
$$\sum_{z} q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)} = \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)]$$

ELBO

$$\sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)} = \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)]$$

ELBO is implemented with the following form:

$$\underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{KL Regularizer}}$$


$E_{q(z|x)} \log p(x|z)$
$KL(q(z|x) \| p(z))$
Reconstruction
regularizer
$q(z|x) = N(0,1)$
$q(z|x) = p(z)$
z
x

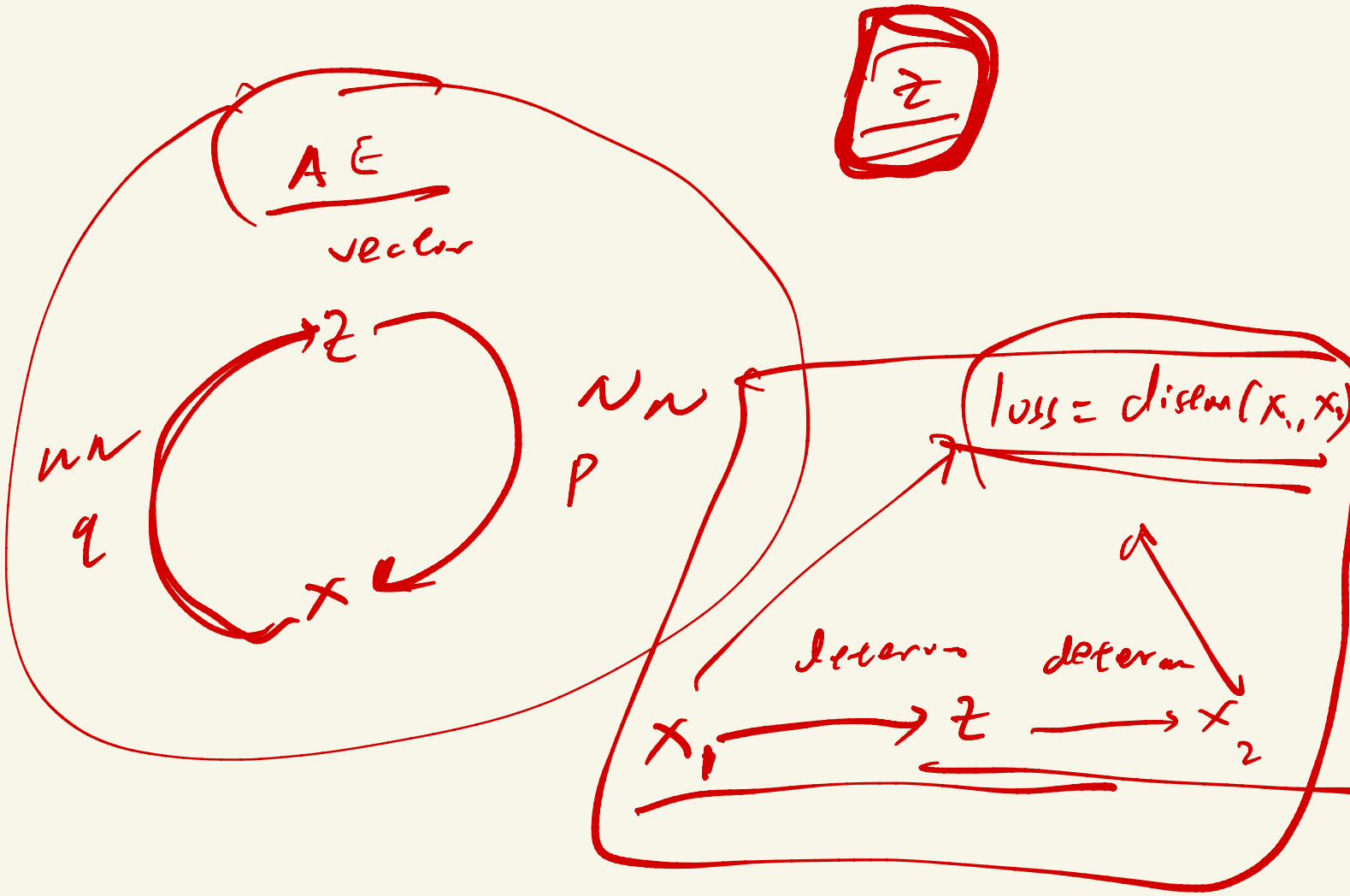
ELBO

$$\sum_z q(z\,|\,x;\phi)\log\frac{p(x,z;\theta)}{q(z\,|\,x;\phi)} = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z\,|\,x)]$$

ELBO is implemented with the following form:

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder


z
AE
vector
z
NN
q
NN
p
x
loss = distance(x1, x2)
x1
deterministic
z
deterministic
x2

ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder Loss

ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder Loss

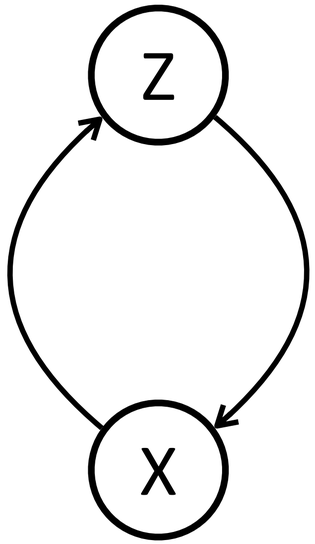
q(z|x) and p(z) are both Gaussian, there is a closed-form for this

ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder Loss

q(z|x) and p(z) are both Gaussian, there is a closed-form for this

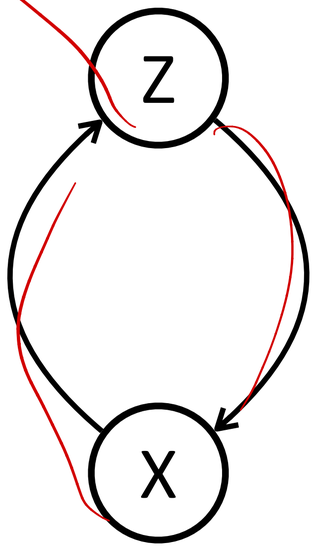


ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Autoencoder Loss

q(z|x) and p(z) are both Gaussian, there is a closed-form for this



$$ELBO = \log P(x;\theta) - KL(q(z|x) \,||\, p(z|x))$$

This is why it is called variational "autoencoder"

ELBO

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

ELBO

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z}$$

$$= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)$$

J is the dimensionality of z

ELBO

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z}$$

$$= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)$$

J is the dimensionality of z

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log q_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z}$$

$$= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2)$$

ELBO

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z}$$

$$= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2)$$

J is the dimensionality of z

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log q_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z}$$

$$= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2)$$

$$-D_{KL}((q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z})) = \int q_{\boldsymbol{\theta}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\right)\, d\mathbf{z}$$

$$= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)$$

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

$$\text{argmax}_{\theta} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Training VAEs

E-Step:

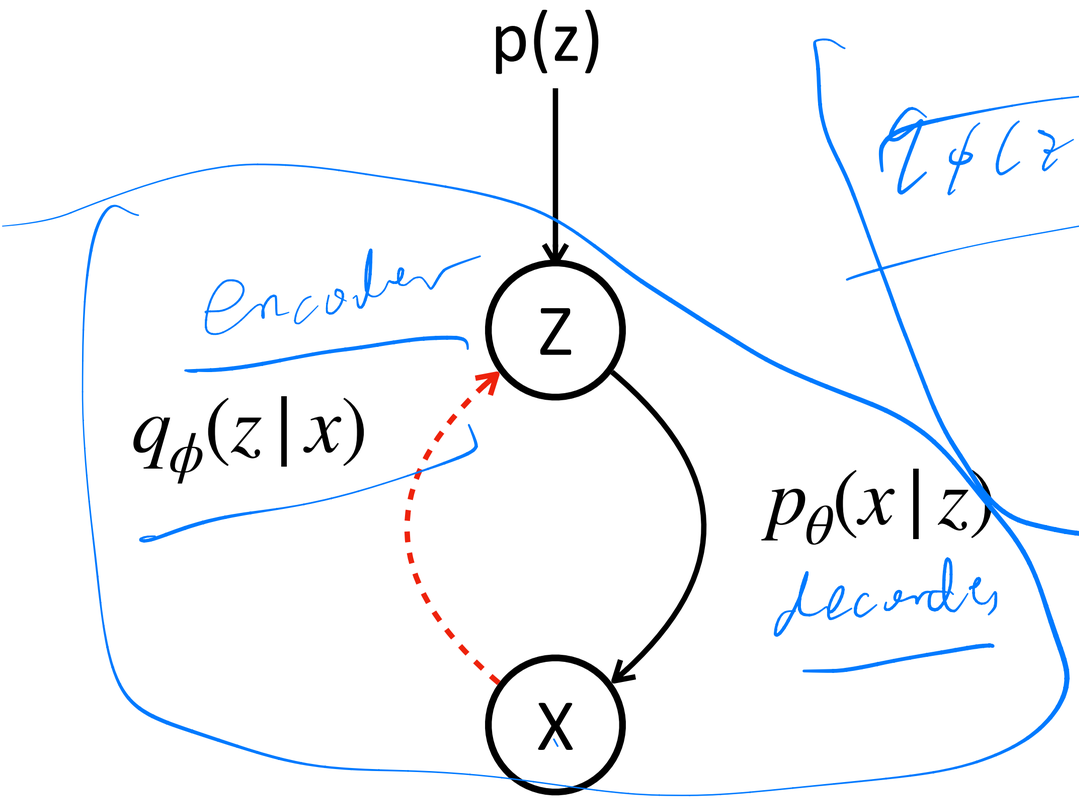
$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

$$\text{argmax}_{\theta} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Intuitively we hope to approximate p(z|x) with q(z|x) accurately in the E-step, to approximate the true EM algorithm

Review VAE



ELBO Loss

ϕ θ

$q_\phi(z|x)$ NN Inference Net

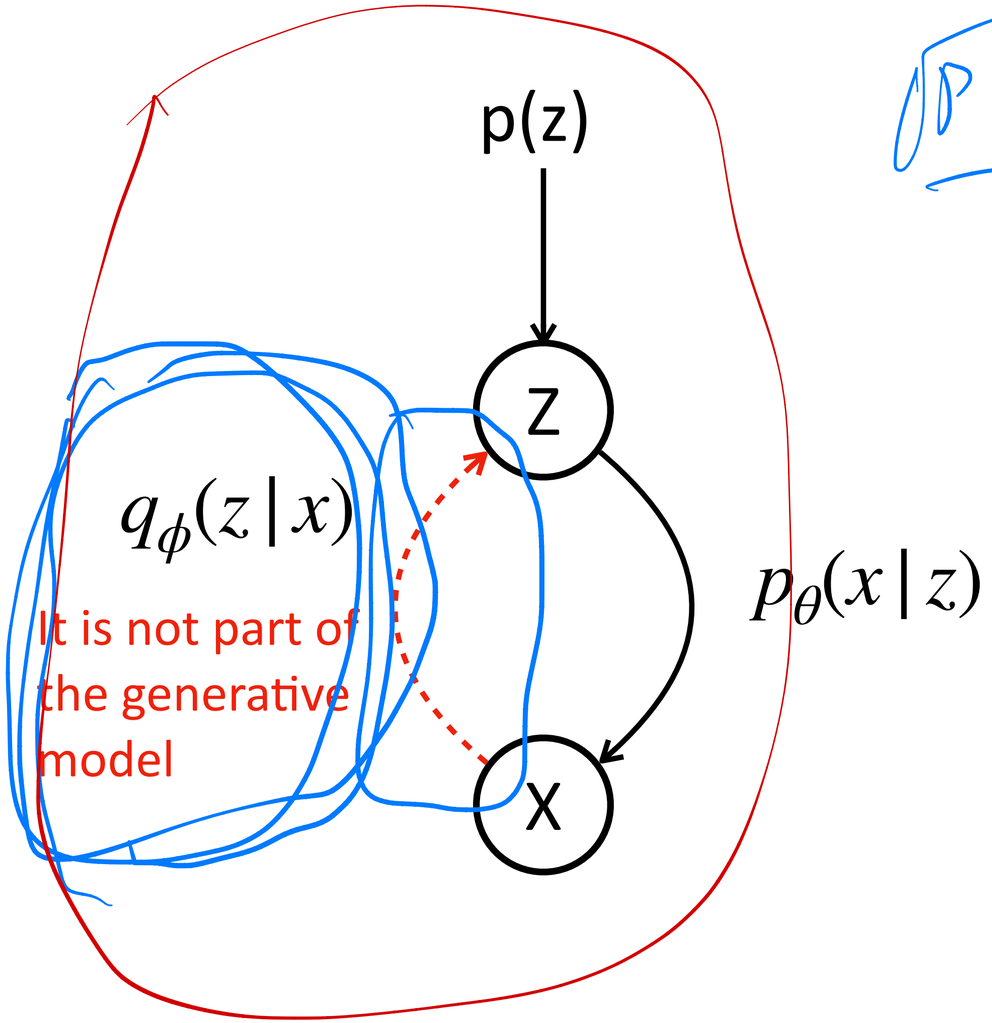
$p_\theta(x|z)$ NN generative net

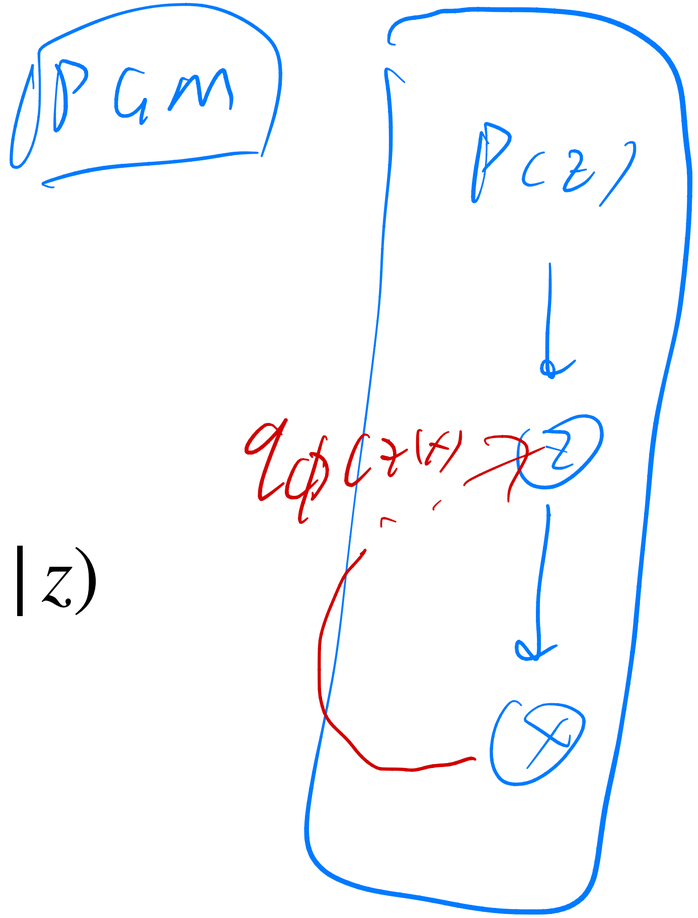
Encoder

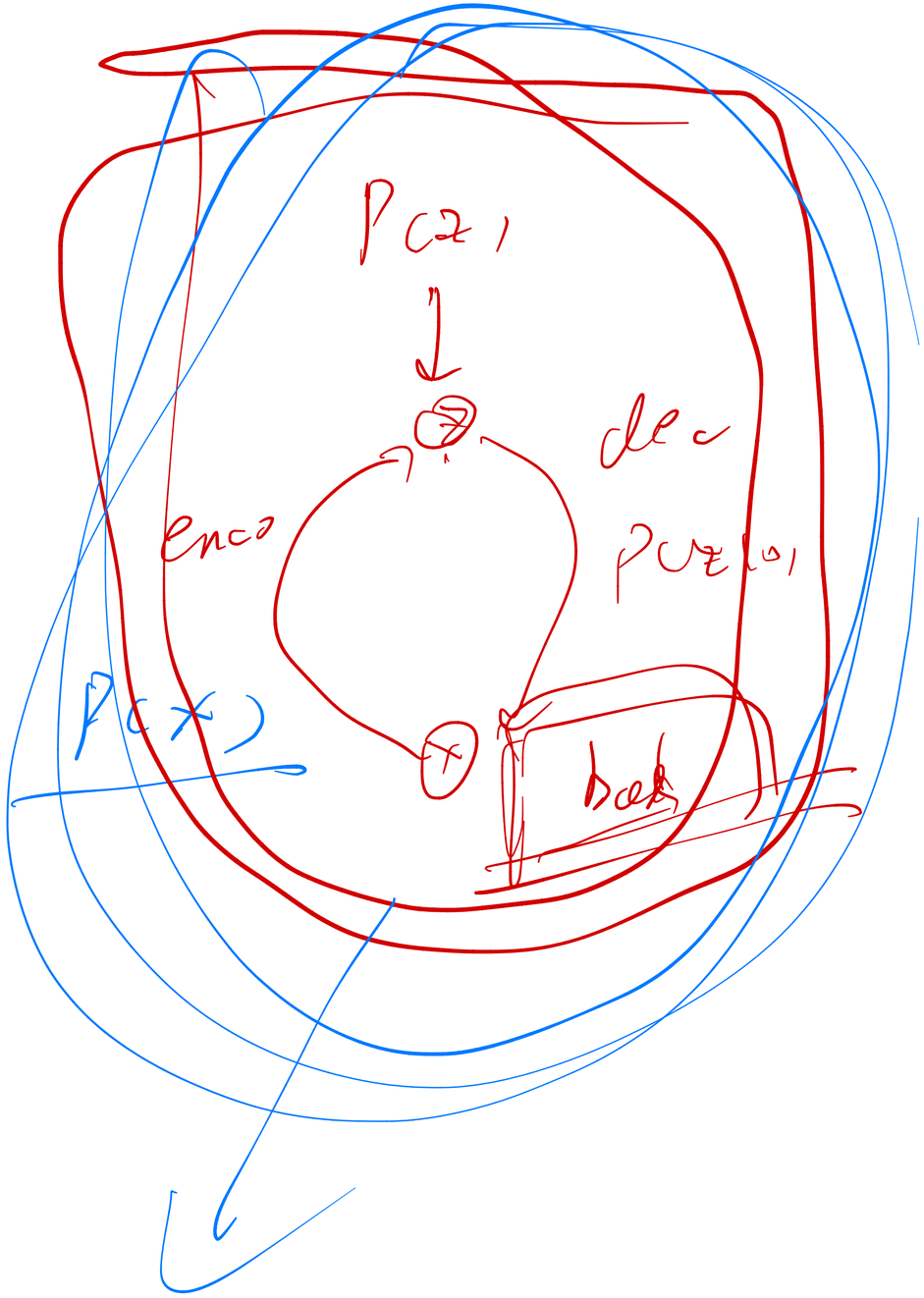
decoder

VAE model

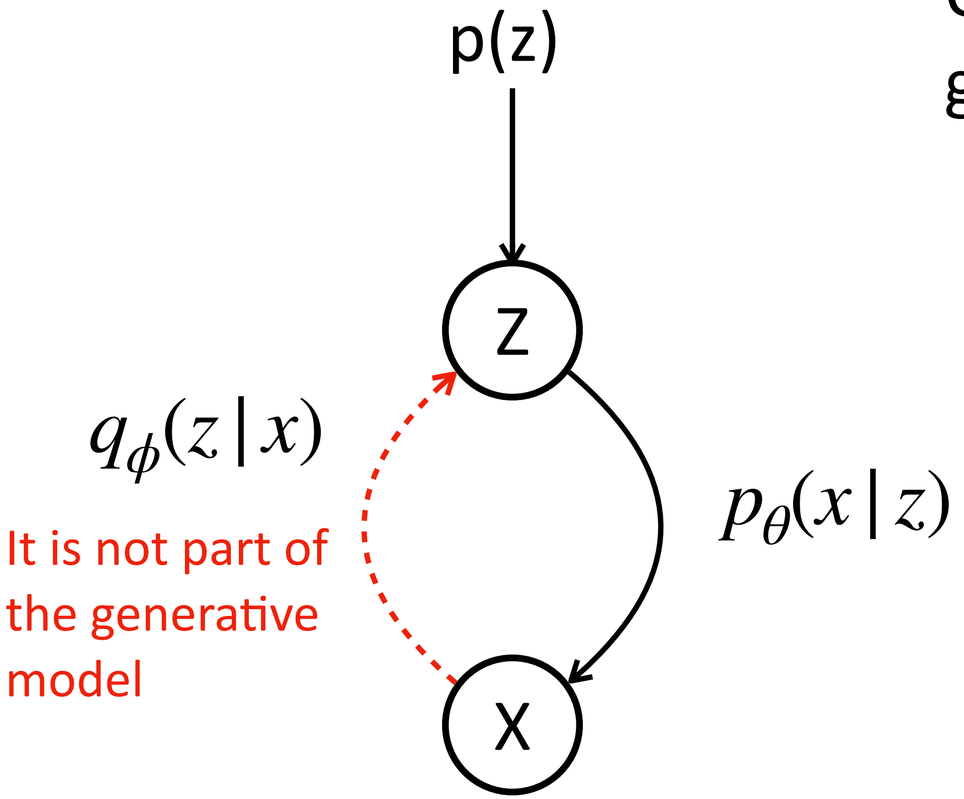
Review VAE


p(z)
z
x
$q_\phi(z \mid x)$
$p_\theta(x \mid z)$
It is not part of the generative model


PGM
p(z)
$q_\phi(z \mid x)$
z
x

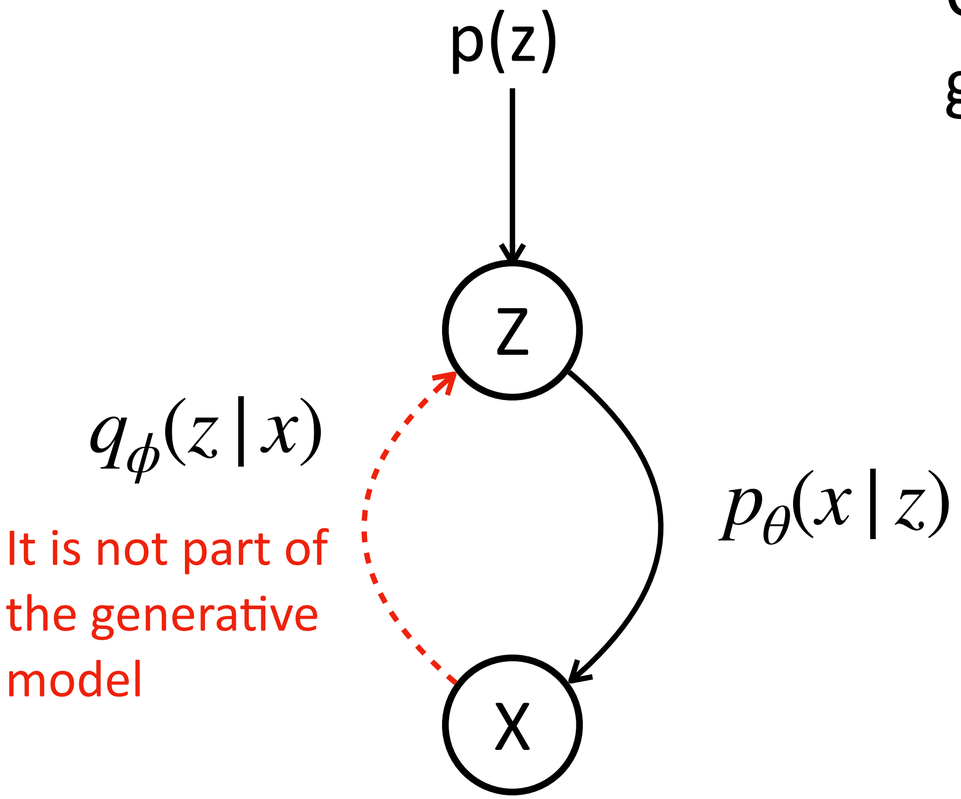

p(z)
enco
dec
p(x|z)
p(x)
x
bce

Review VAE



Only the right (black) part defines the generative model, and the distribution

Review VAE

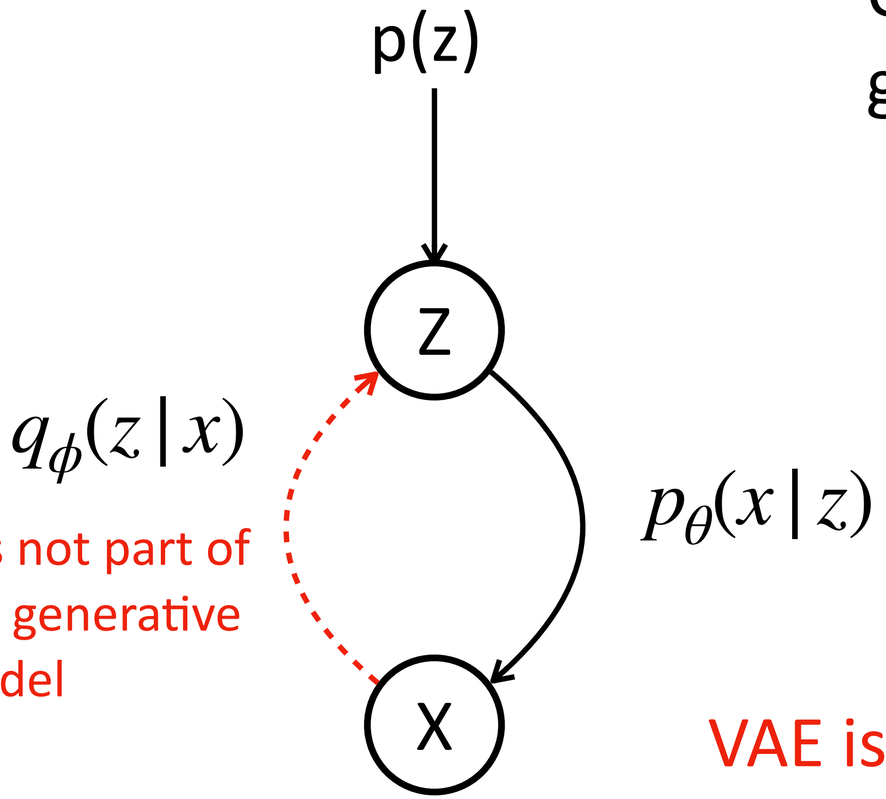


Only the right (black) part defines the generative model, and the distribution

$p_\theta(x \mid z)$: generative network/decoder

$q_\phi(z \mid x)$: inference network/encoder

Review VAE



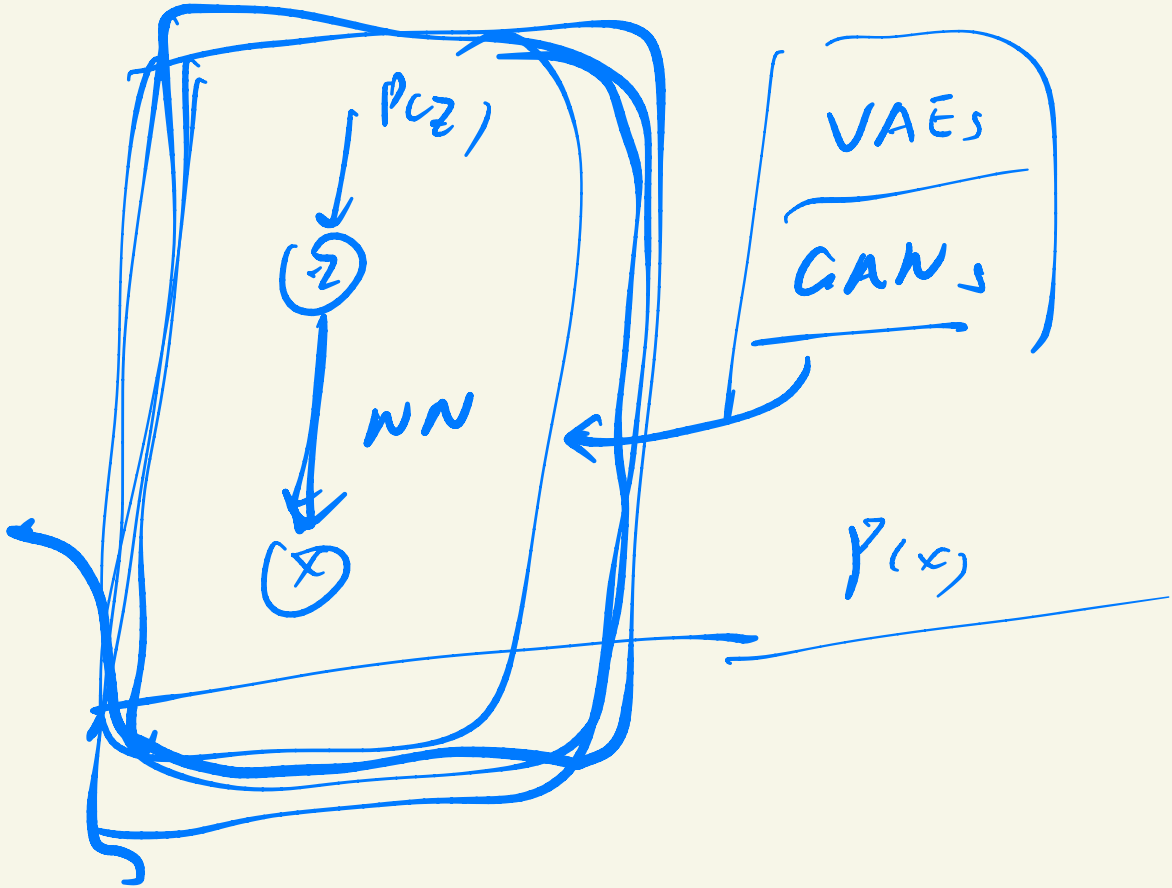
Only the right (black) part defines the generative model, and the distribution

$p_\theta(x \mid z)$: generative network/decoder

$q_\phi(z \mid x)$: inference network/encoder

VAE: { model + traing method

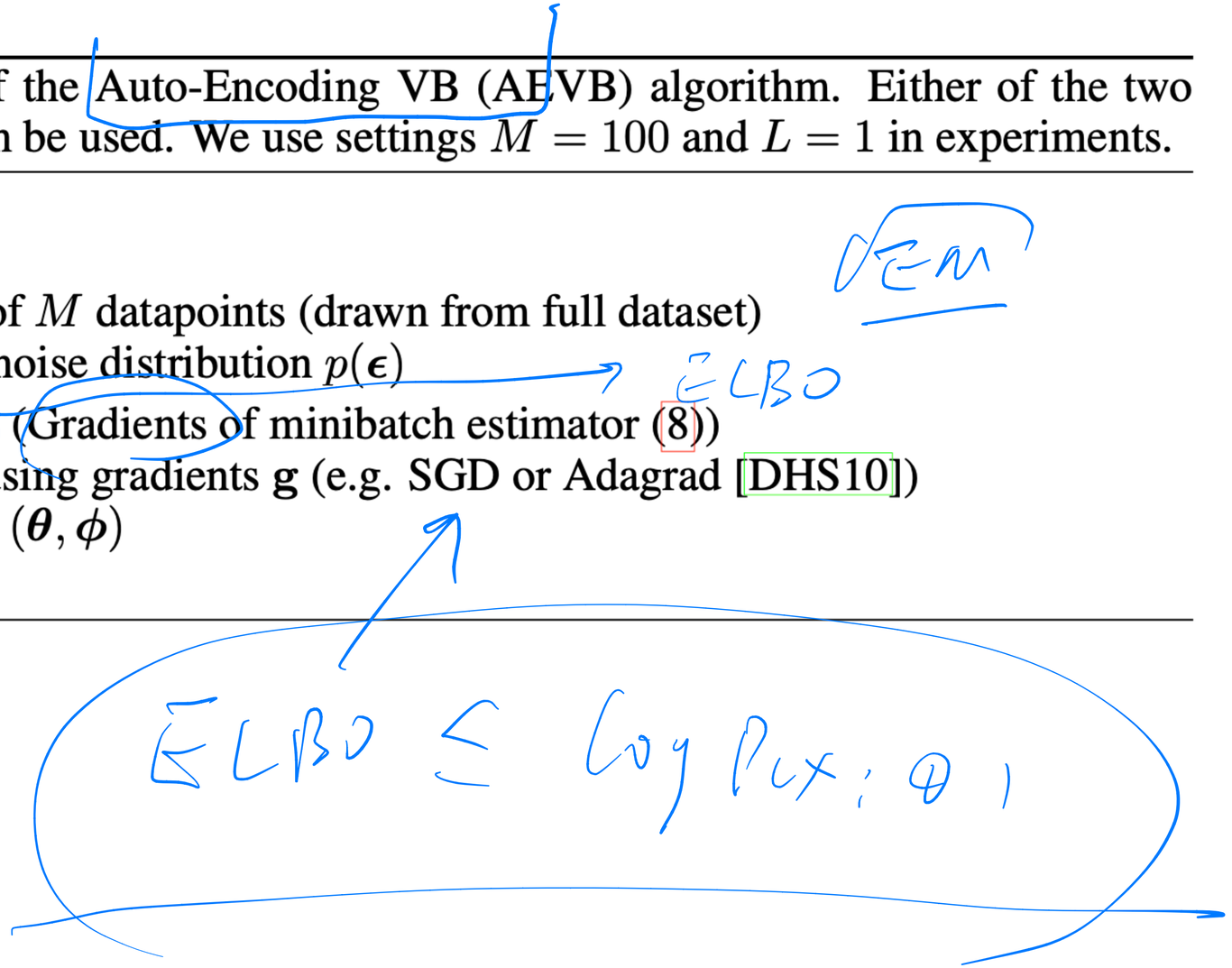
VAE is a name to represent both the model p(x) and the inference network that is used to train the model, but do not confuse them together


P(z)
z
NN
x
VAEs
GANs
P(x)

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$



Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

End-to-end, because the objectives are the same (ELBO)

Training VAEs

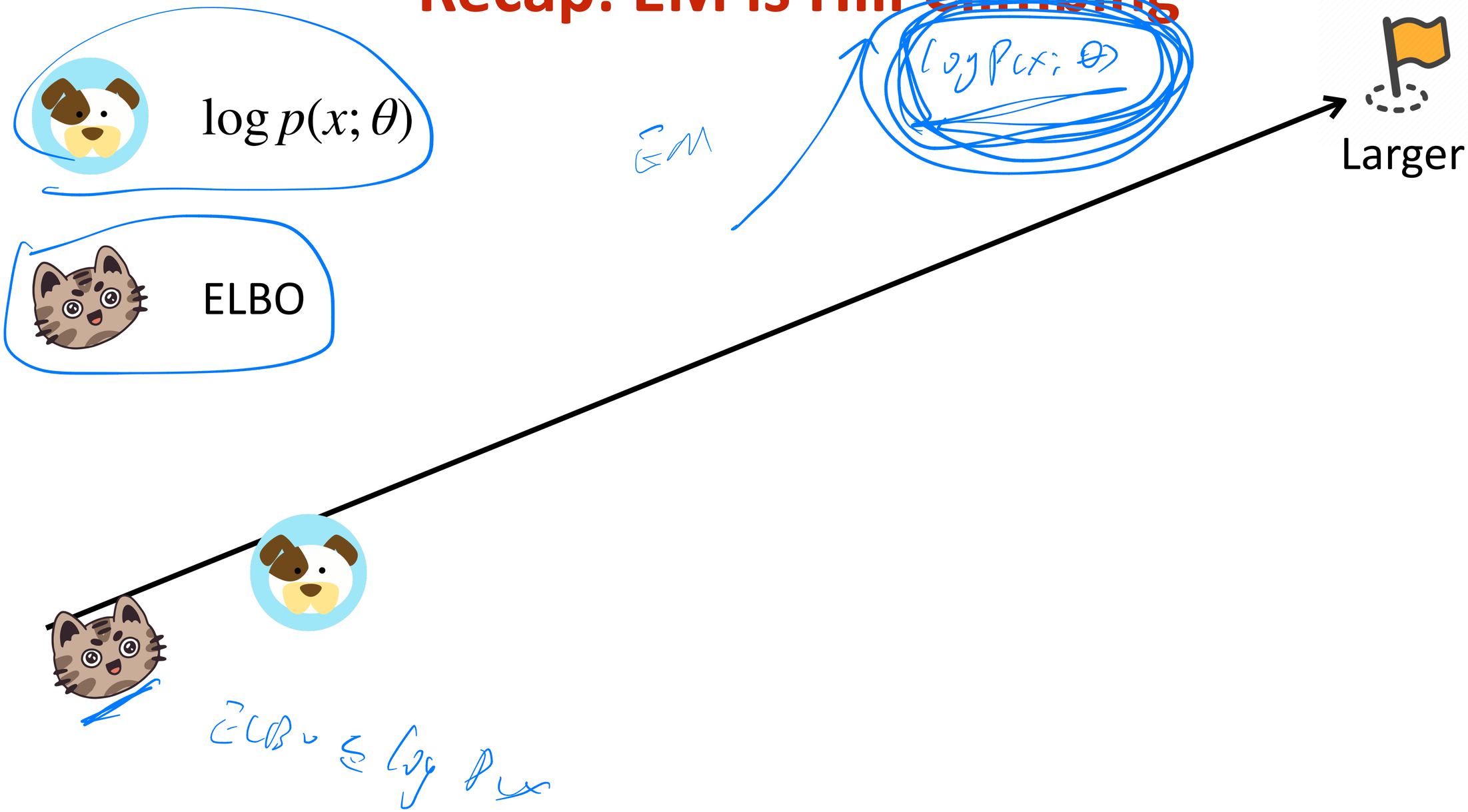
Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

End-to-end, because the objectives are the same (ELBO)

VAE training is optimizing ELBO with gradient descent

Recap: EM is Hill Climbing



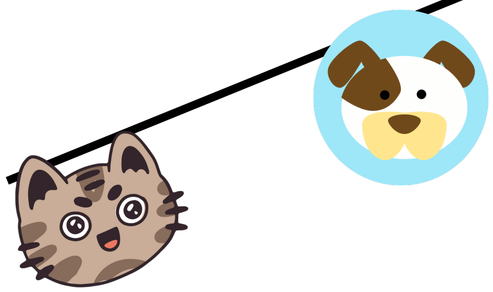
Recap: EM is Hill Climbing

$\log p(x;\theta)$

Only related to θ, no z

ELBO

Larger

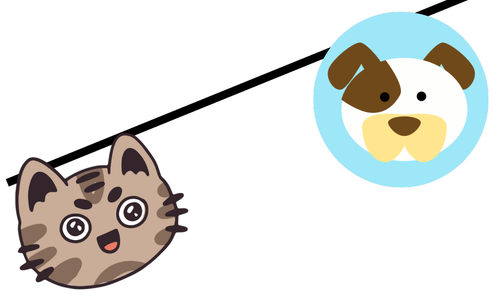


Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger

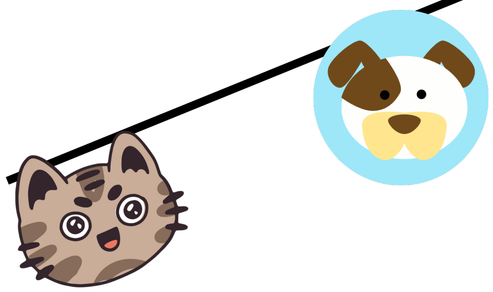


Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger



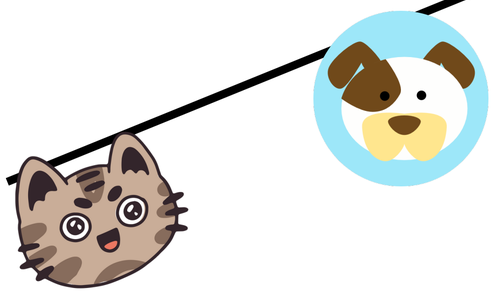
E-step: $Q(z) = p(z \mid x;\theta)$, making ELBO tight

Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger



E-step: $Q(z) = p(z \mid x;\theta)$, making ELBO tight

"dog" doesn't change, because θ does not change

Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger

E-step: $Q(z) = p(z \mid x;\theta)$, making ELBO tight

"dog" doesn't change, because θ does not change

Recap: EM is Hill Climbing

$\log p(x;\theta)$ $\geq$ ELBO

ELBO

Larger

M-step: $\max_{\theta} ELBO$

$\Rightarrow$

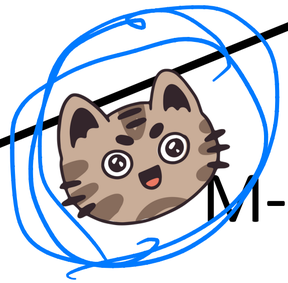
ELBO becomes larger, and it is not tight anymore because posterior changes

Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger



M-step: $\max_{\theta} ELBO$

ELBO becomes larger, and it is not tight anymore because posterior changes

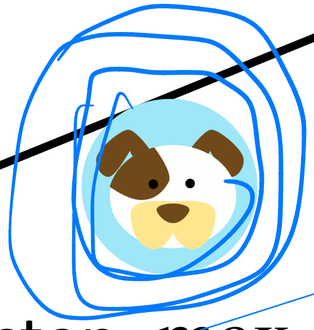
Recap: EM is Hill Climbing

$\log p(x;\theta)$

increase

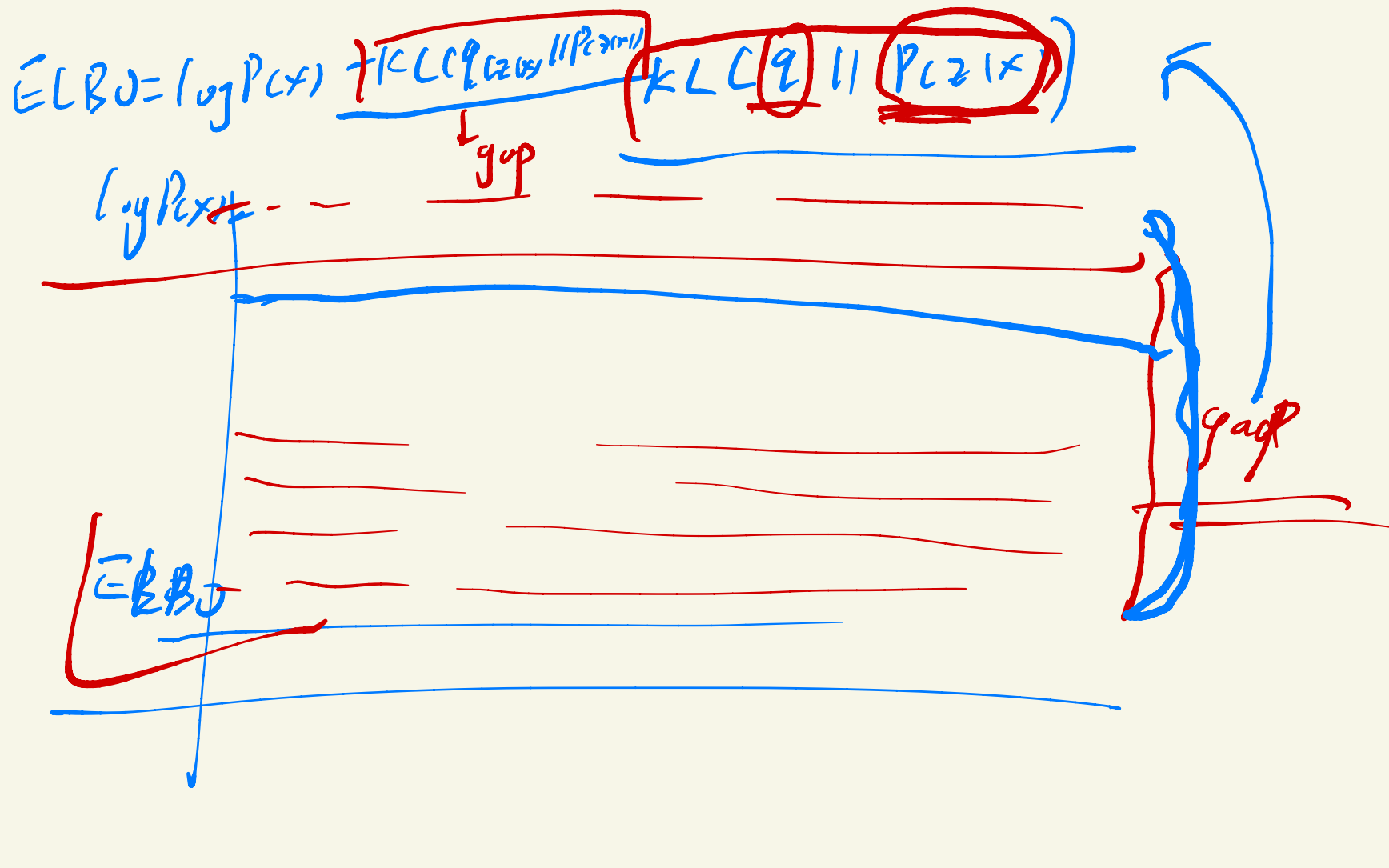
Larger

ELBO

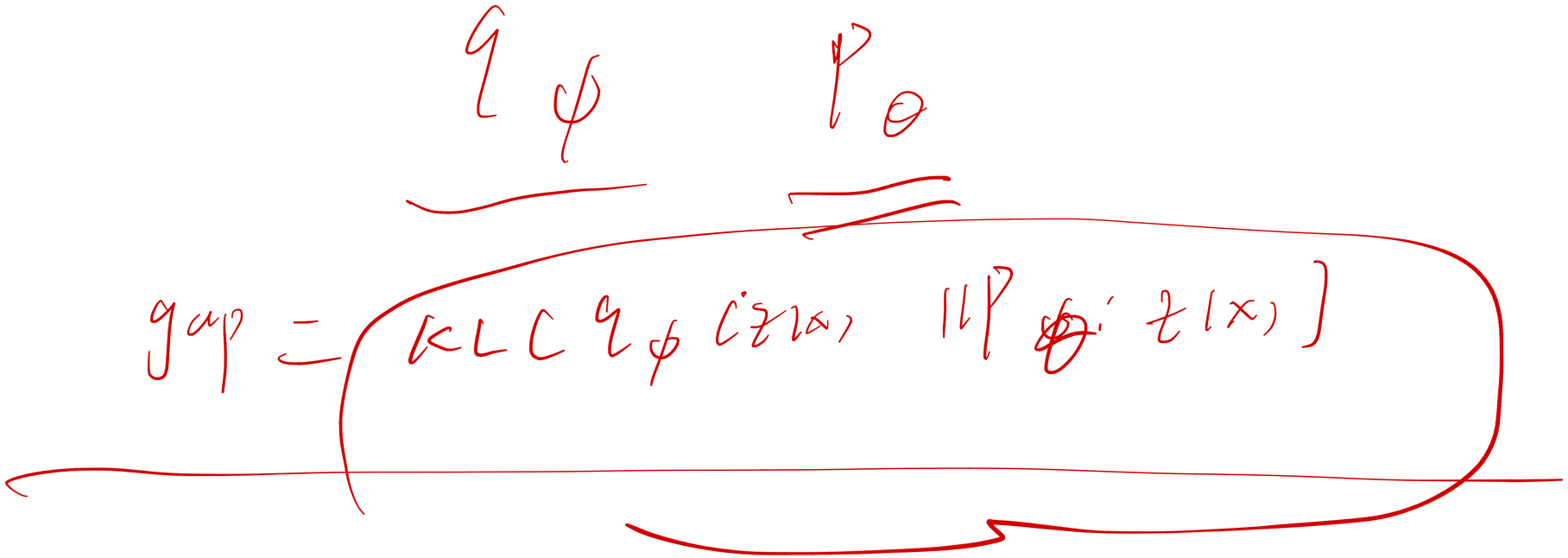


M-step: $\max_{\theta} ELBO$

ELBO becomes larger, and it is not tight anymore because posterior changes


ELBO = log P(x) - KL(q(z|x) || p(z|x)) + KL(q(z|x) || p(z))
gap
log P(x)
ELBO
gap

Is VAE training still Hill Climbing?



Is VAE training still Hill Climbing?

It is not, because $q(z|x)$ may not be accurate to approximate $p(z|x)$

Is VAE training still Hill Climbing?

It is not, because q(z|x) may not be accurate to approximate p(z|x)

In VAE training, there is no guarantee that log p(x) is monotonically increasing

Is VAE training still Hill Climbing?

It is not, because q(z|x) may not be accurate to approximate p(z|x)

In VAE training, there is no guarantee that log p(x) is monotonically increasing

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, ϕ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not

Is VAE training still Hill Climbing?

It is not, because q(z|x) may not be accurate to approximate p(z|x)

In VAE training, there is no guarantee that log p(x) is monotonically increasing

It just works in many cases

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, ϕ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not

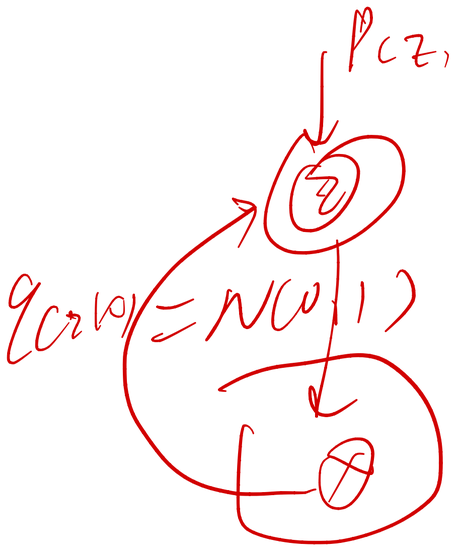
The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z \mid x) = p(z)$ and z and x becomes independent (especially in applications of NLP)



The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z\,|\,x) = p(z)$ and z and x becomes independent (especially in applications of NLP)

Z does not affect x, the model degenerates to a generative model without latent variables

The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z\,|\,x) = p(z)$ and z and x becomes independent (especially in applications of NLP)

Z does not affect x, the model degenerates to a generative model without latent variables

Researchers commonly blame that the KL regularizer is too strong for this and use a weight $0 < \lambda < 1$ to control it:

The Posterior Collapse Issue

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z\,|\,x) = p(z)$ and z and x becomes independent (especially in applications of NLP)

Z does not affect x, the model degenerates to a generative model without latent variables

Researchers commonly blame that the KL regularizer is too strong for this and use a weight $0 < \lambda < 1$ to control it:

$0 < \lambda < 1$

Reconstruction Loss - λ * KL regularizer

The Posterior Collapse Issue

ELBO

$$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

In practice, it is often found that after training, $q_\phi(z\,|\,x) = p(z)$ and z and x becomes independent (especially in applications of NLP)

Z does not affect x, the model degenerates to a generative model without latent variables

Researchers commonly blame that the KL regularizer is too strong for this and use a weight $0 < \lambda < 1$ to control it:

could be > log P(x)

Reconstruction Loss - λ * KL regularizer

> ELBO

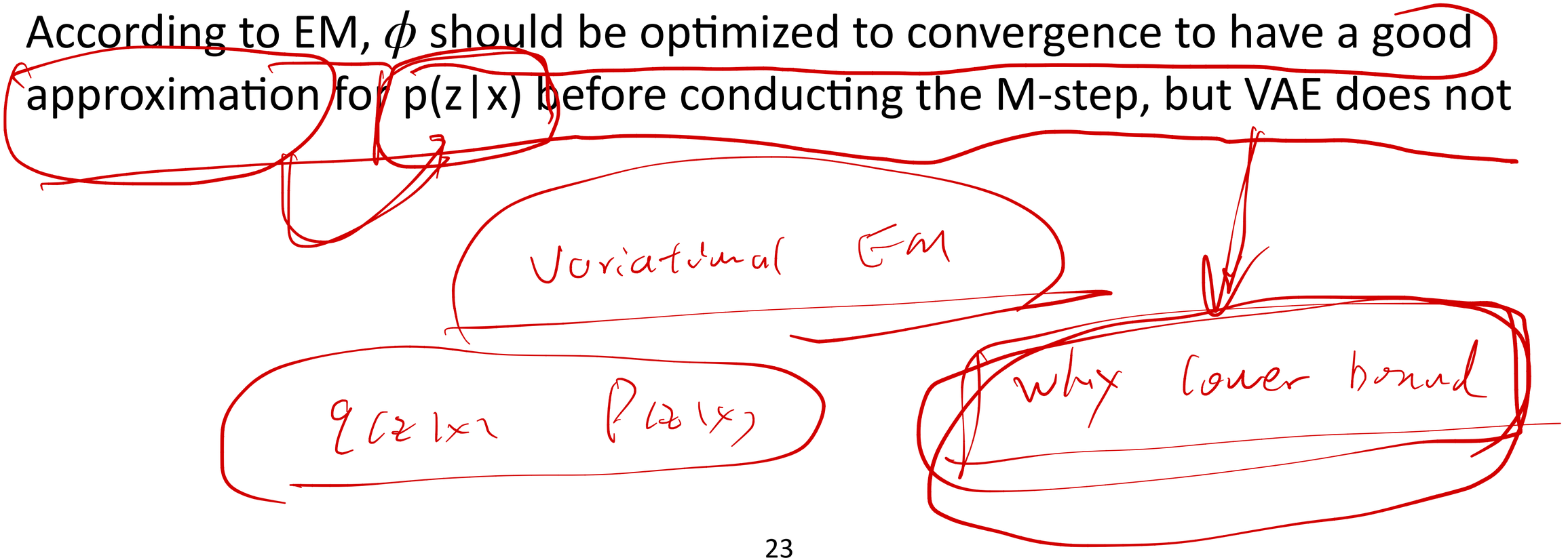
This is not a lower-bound of log p(x) anymore and it breaks MLE, but what is wrong with MLE?

Is VAE training still Hill Climbing?

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, ϕ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not



Is VAE training still Hill Climbing?

E-Step:

$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

According to EM, ϕ should be optimized to convergence to have a good approximation for p(z|x) before conducting the M-step, but VAE does not

Can we make it closer to EM to have good guarantees?

VAE training that is Closer to EM

VAE training that is Closer to EM

At every iteration, perform multiple gradient updates of ϕ (E-step) before performing one step of θ (M-step)

VAE training that is Closer to EM

At every iteration, perform multiple gradient updates of ϕ (E-step) before performing one step of θ (M-step)

Published as a conference paper at ICLR 2019

LAGGING INFERENCE NETWORKS AND POSTERIOR COLLAPSE IN VARIATIONAL AUTOENCODERS

Junxian He, Daniel Spokoyny, Graham Neubig
Carnegie Mellon University
{junxianh,dspokoyn,gneubig}@cs.cmu.edu

Taylor Berg-Kirkpatrick
University of California San Diego
tberg@eng.ucsd.edu