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

COMP 5212
Machine Learning
Lecture 13

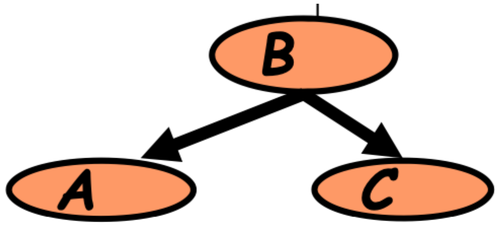
Variational AutoEncoder

Junxian He
Mar 26, 2026

Recap: Local Structures & Independence

- Common parent
  - Fixing B decouples A and C

    "given the level of gene B, the levels of A and C are independent"

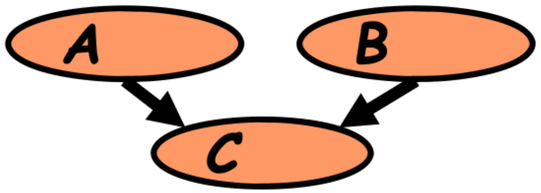


- Cascade
  - Knowing B decouples A and C

    "given the level of gene B, the level gene A provides no extra prediction value for the level of gene C"

- V-structure
  - Knowing C couples A and B because A can "explain away" B w.r.t. C

    "If A correlates to C, then chance for B to also correlate to B will decrease"

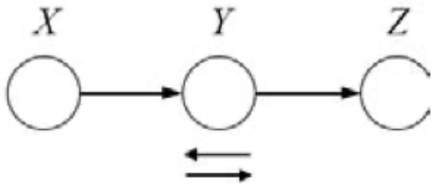


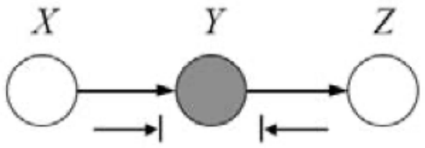
The language is compact, the concepts are rich!

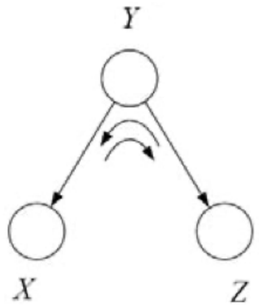
Recap: Global Markov Properties of DAGs

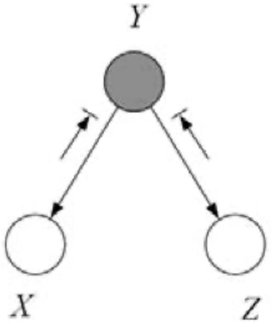
How to determine two variables are conditionally independent given another variable?

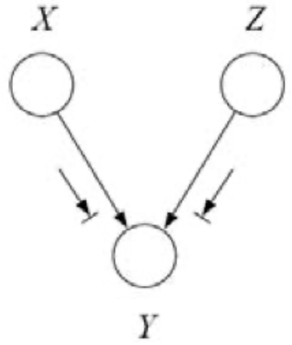
X is **d-separated** (directed-separated) from Z given Y if we can't send a ball from any node in X to any node in Z using the "*Bayes-ball*" algorithm illustrated bellow (and plus some boundary conditions):

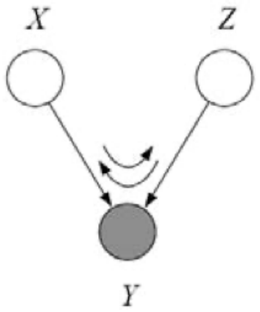




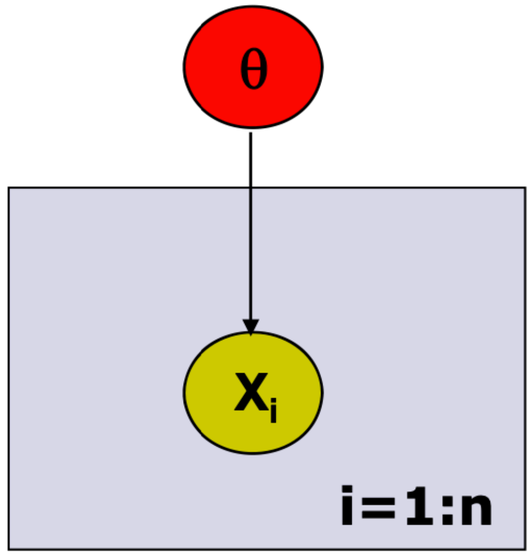








“Plate” Notation



Model parameters

Data = $\{x_1,\ldots x_n\}$

variables within a plate are replicated in a conditionally independent manner

Example: Gaussian Model

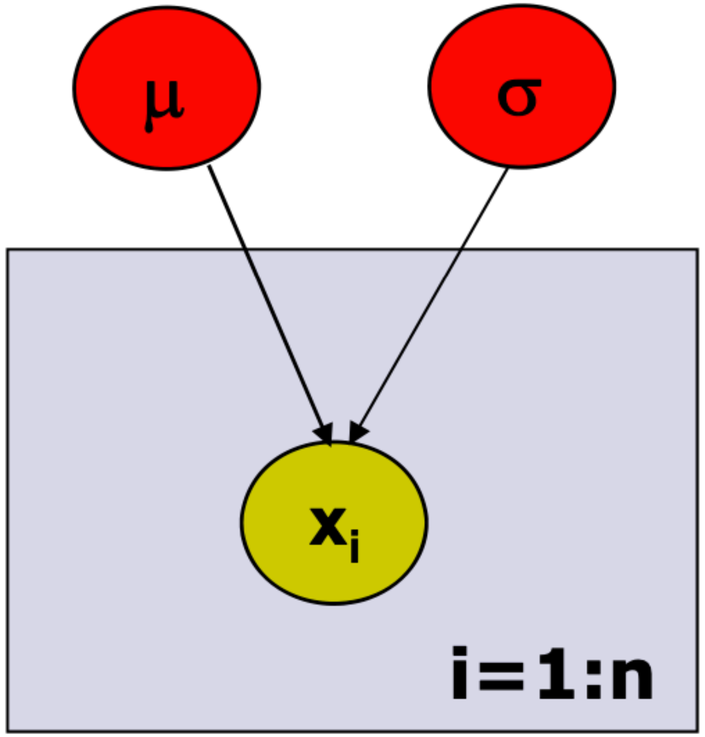
Generative model:

$p(x_1,\ldots x_n \mid \mu, \sigma) = P\ p(x_i \mid \mu, \sigma)$

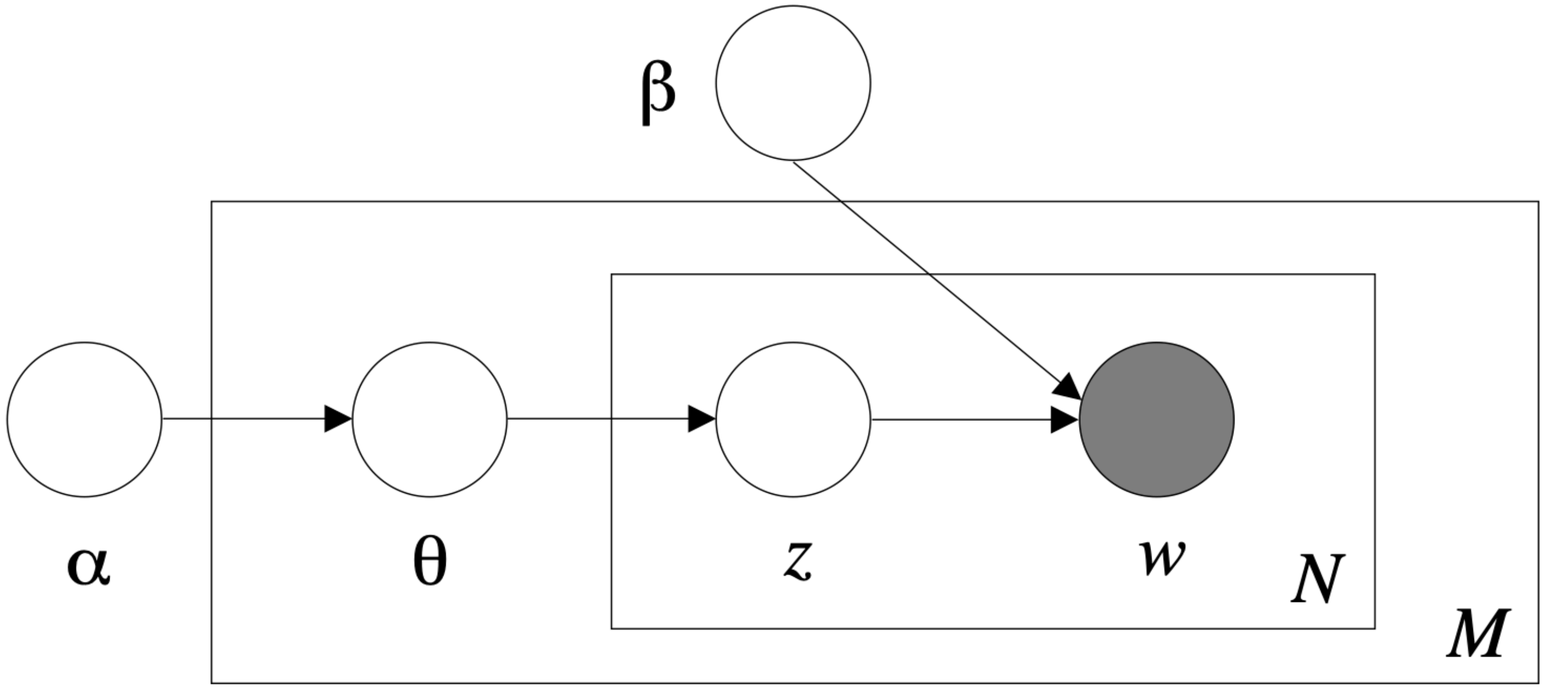
$= p(\text{data} \mid \text{parameters})$

$= p(D \mid \theta)$

where $\theta = \{\mu, \sigma\}$



Observed Variable and Latent Variable Notations



We typically use gray variables to denote observed variables

Gaussian Mixture Model / Gaussian Discriminative Analysis in PGMs

Inference and Learning

Query a node (random variable) in the graph

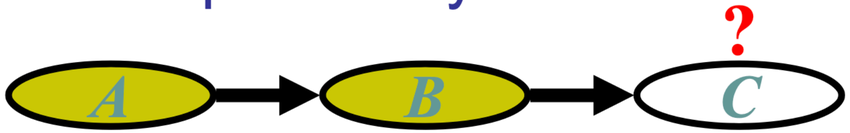
- Task 1: How do we answer **queries** about P?
  - We use **inference** as a name for the process of computing answers to such queries

- Task 2: How do we estimate a **plausible model** M from data D?

i. We use **learning** as a name for the process of obtaining point estimate of M.

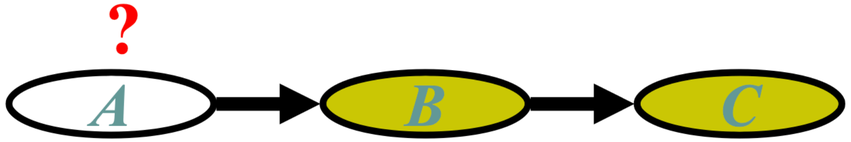
Examples

- **Prediction**: what is the probability of an outcome given the starting condition



  - the query node is a descendent of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms



  - the query node an ancestor of the evidence

In practice, the observed variable is often the data that is on the leaf nodes

How to Learn the Parameters

When θ is the parameter and does not have prior —> MLE

$$p(x, z; \theta)$$

How to do MLE on Latent Variable Models?

Expectation Maximization!

The E-step computes the posterior distribution $p(z|x)$

This process is referred to as inference

Approaches to Inference

- Exact inference algorithms
  - The elimination algorithm
  - Belief propagation
  - The junction tree algorithms (but will not cover in detail here)

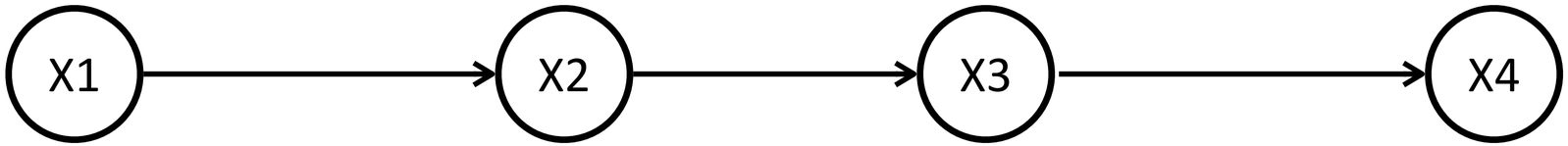
- Approximate inference techniques
  - Variational algorithms
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods

Variational Autoencoders

Elimination Algorithm/ Marginalization

$$P(h) = \sum_{g} \sum_{f} \sum_{e} \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a,b,c,d,e,f,g,h)$$

a naïve summation needs to enumerate over an exponential number of terms



What if the random variables follow this chain structure?

Auto-Encoding Variational Bayes

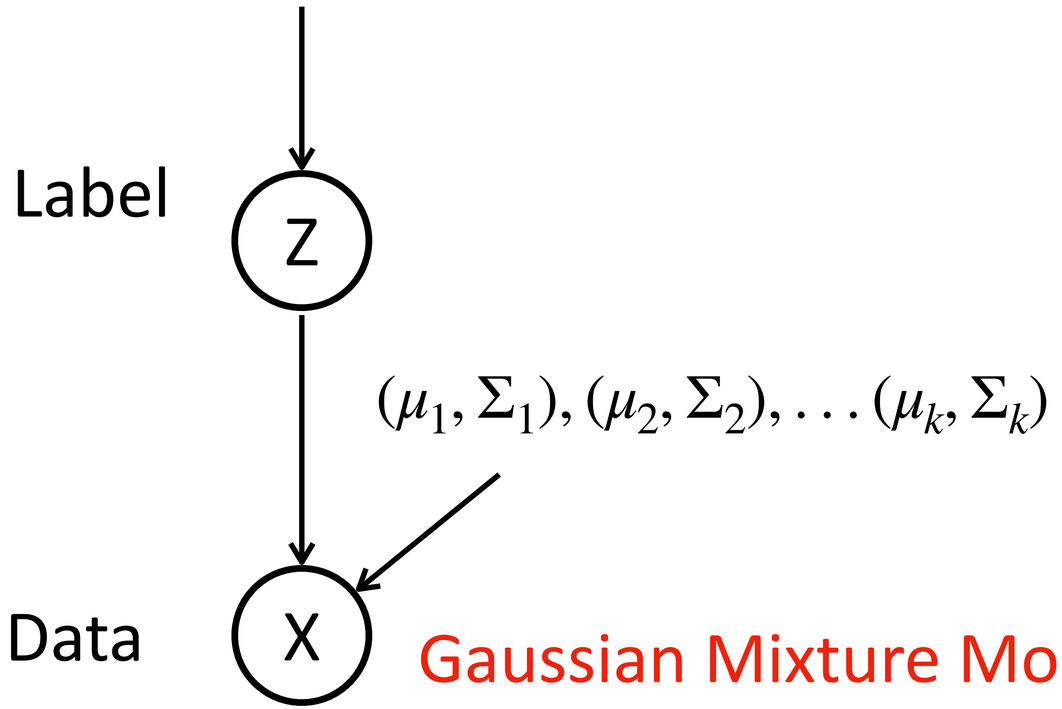
Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Variational Autoencoders

VAE is a Generative Model

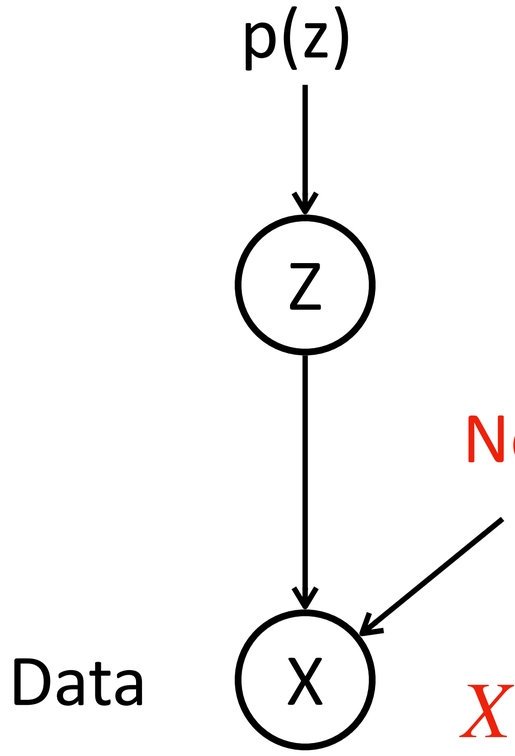
p(z): multinomial , k classes(e.g. uniform)



Gaussian Mixture Model (GMM)

The VAE Model

p(z) is a normal distribution in most cases

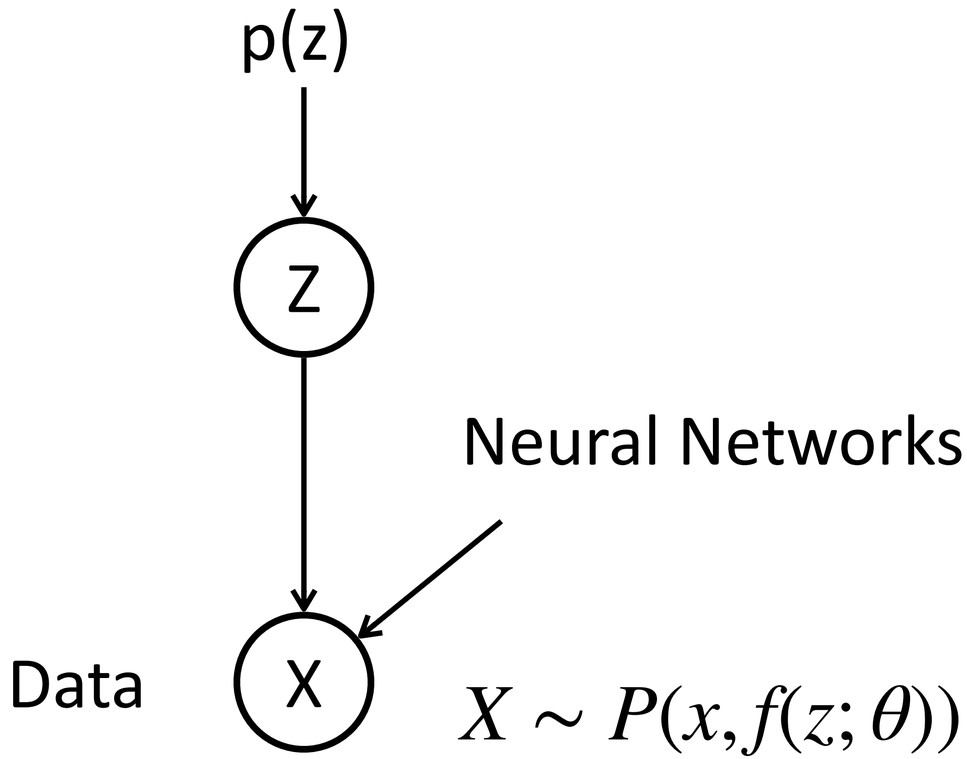


Neural Networks

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

f is a neural network taking Z as input

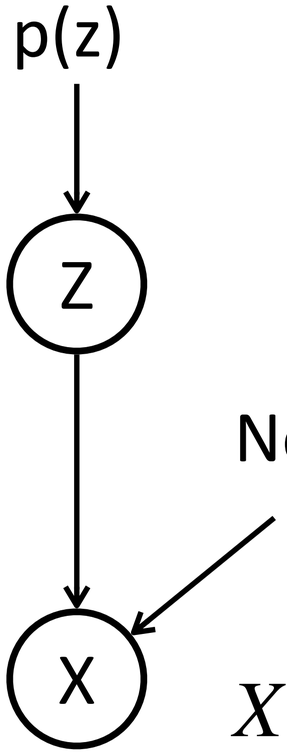
Training



How to train the model? Can we do MLE?

Intractable P(X), EM algorithm?

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

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

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)

$q(z|x;\phi)$ 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?

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:

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

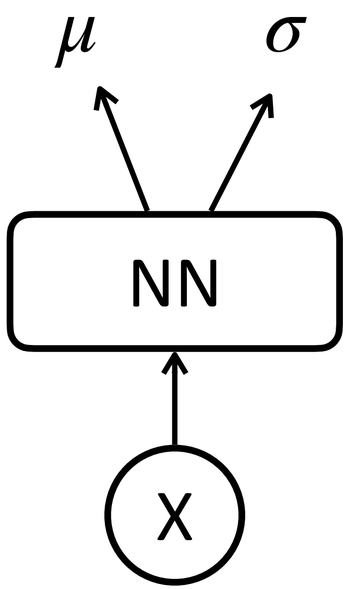
Same objective, different parameters to optimize

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

A Common Choice for $q(z \mid x; \phi)$

$$q(z \mid x; \phi) = N(\mu, \sigma^2)$$

$$\mu, \sigma = g(x; \phi)$$



Inference model/network

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:

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

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:

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

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):

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

Reparameterization Trick

VAE is a class of models

What kind of $q(z \mid x; \phi)$ allows for such a reparameterization trick?

1. Tractable inverse CDF. In this case, let $\boldsymbol{\epsilon} \sim \mathcal{U}(\mathbf{0}, \mathbf{I})$, and let $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ be the inverse CDF of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Examples: Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel and Erlang distributions.
2. Analogous to the Gaussian example, for any "location-scale" family of distributions we can choose the standard distribution (with location $= 0$, scale $= 1$) as the auxiliary variable $\boldsymbol{\epsilon}$, and let $g(.) =$ location $+$ scale $\cdot\, \boldsymbol{\epsilon}$. Examples: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular and Gaussian distributions.
3. Composition: It is often possible to express random variables as different transformations of auxiliary variables. Examples: Log-Normal (exponentiation of normally distributed variable), Gamma (a sum over exponentially distributed variables), Dirichlet (weighted sum of Gamma variates), Beta, Chi-Squared, and F distributions.

Kingma et al. Auto-Encoding Variational Bayes

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:

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

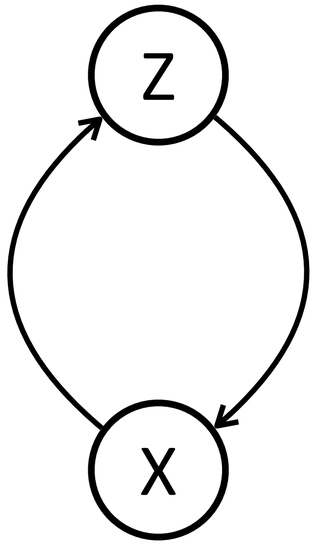
Autoencoder

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



This is why it is called variational “autoencoder”

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:

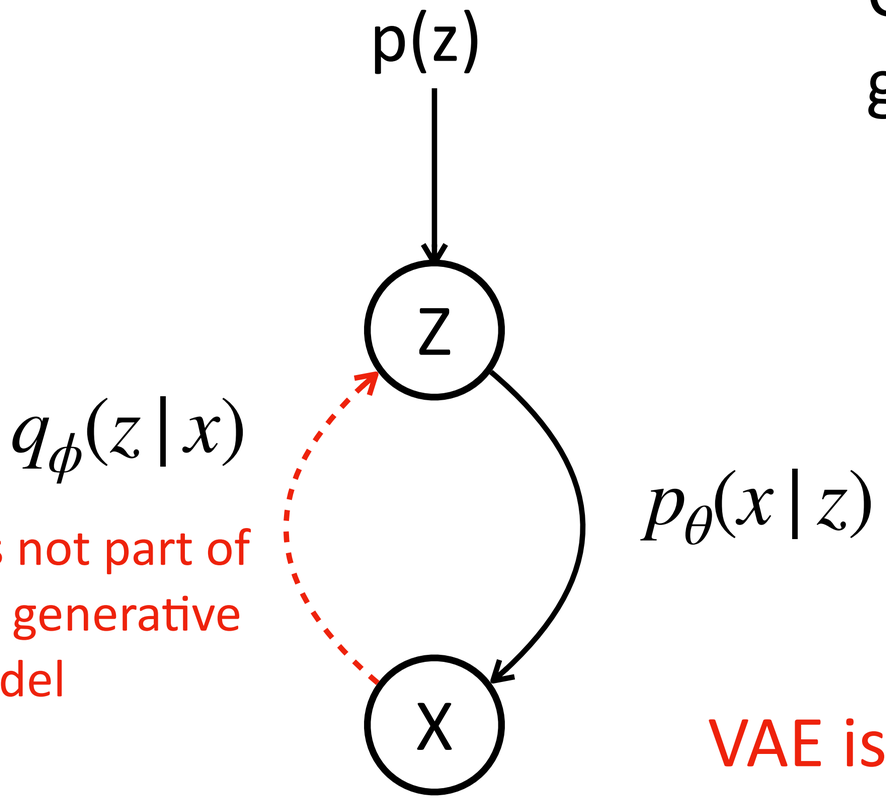
$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

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

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



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 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

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

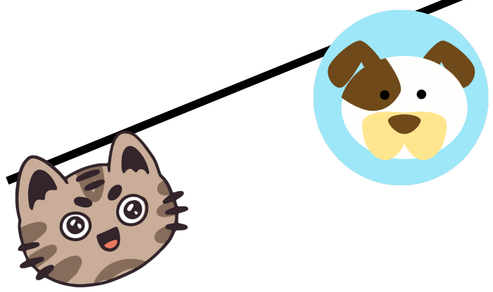
Recap: EM is Hill Climbing

$\log p(x;\theta)$

Only related to θ, no z

ELBO

Larger

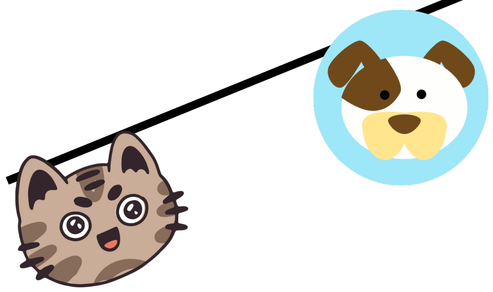


Recap: EM is Hill Climbing

$\log p(x;\theta)$

ELBO

Larger



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

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

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

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

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

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:

Reconstruction Loss - λ * KL regularizer

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

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

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

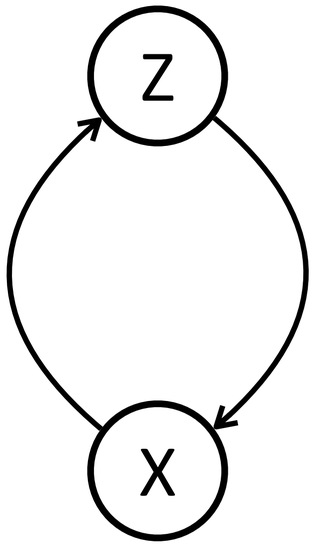
Junxian He, Daniel Spokoyny, Graham Neubig
Carnegie Mellon University
{junxianh,dspokoyn,gneubig}@cs.cmu.edu

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

AutoEncoders

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

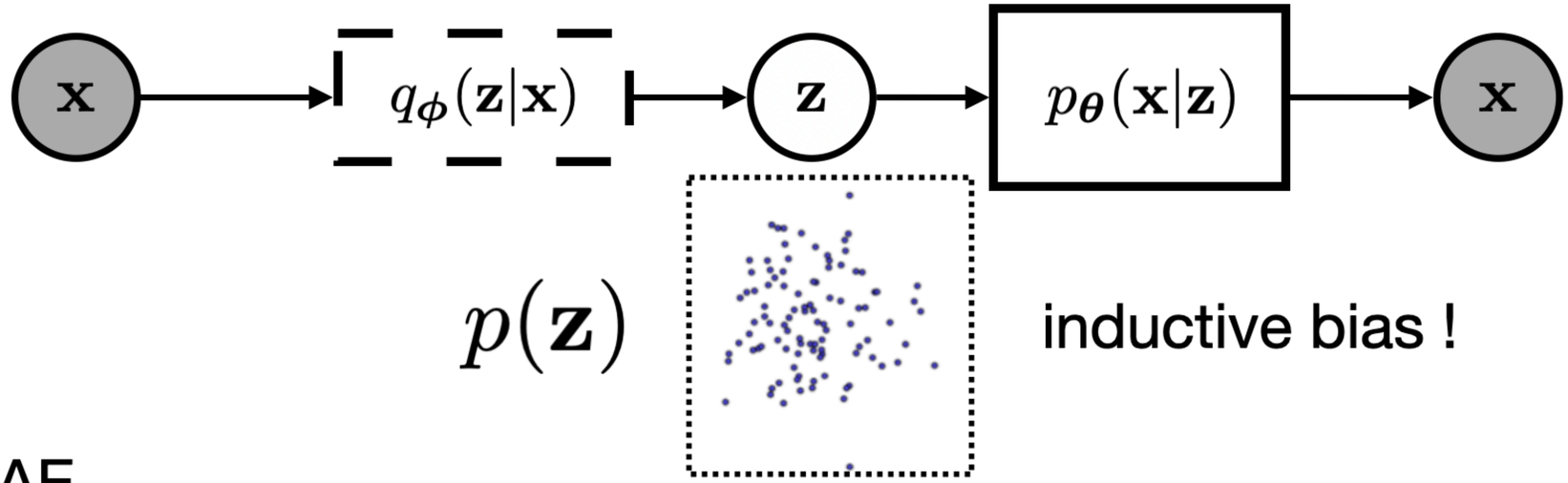
AE: $\log p_\theta(x \mid q(x))$



1. Can we generate X samples from an autoencoder?
2. Can we approximate p(x) given x with an autoencoder?
3. What is the difference between the representation space from AE and VAE?

VAE v.s. AE

VAE



AE