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

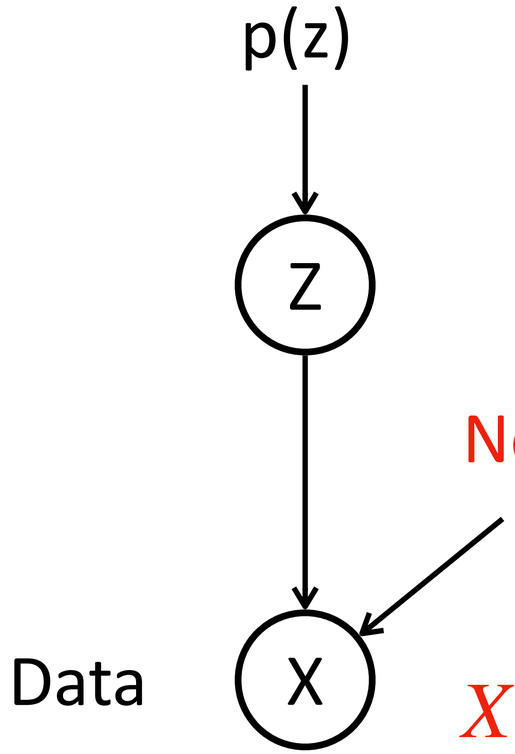
COMP 5212
Machine Learning
Lecture 14

Variational Autoencoders, Hidden Markov Models

Junxian He
Apr 2, 2026

Recap: The VAE Model

p(z) is a normal distribution in most cases

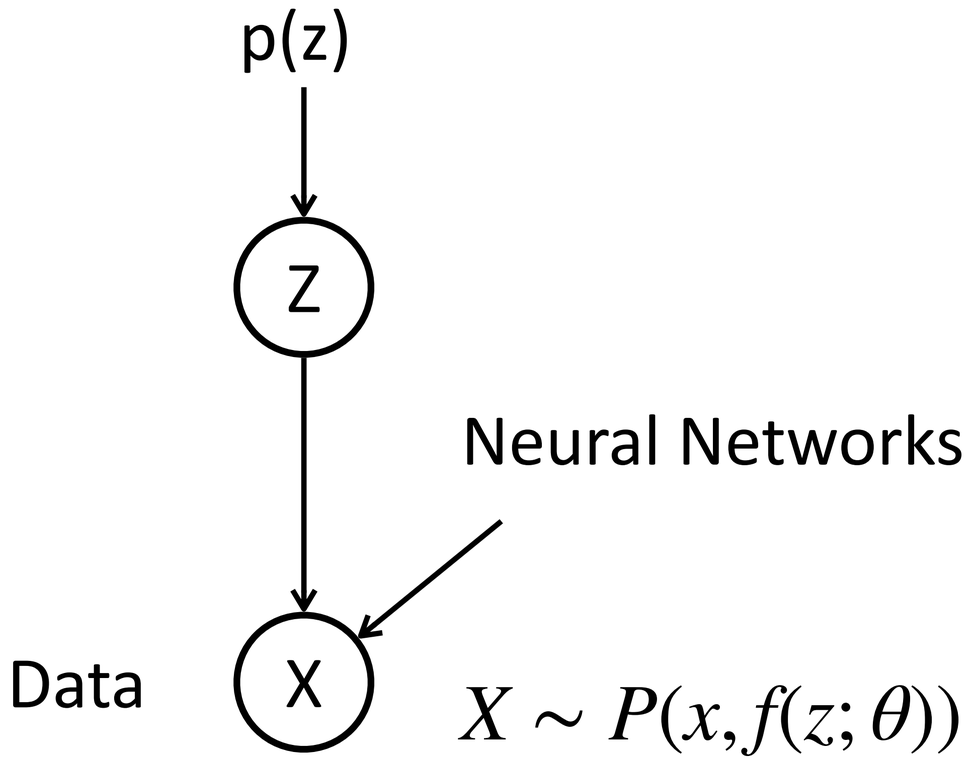


Neural Networks

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

f is a neural network taking Z as input

Training

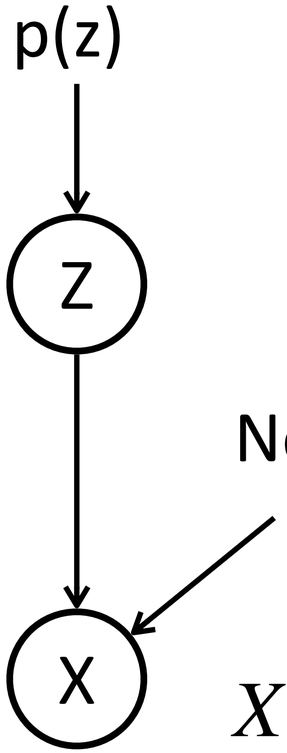


$$X \sim P(x, f(z;\theta))$$

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

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

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:

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

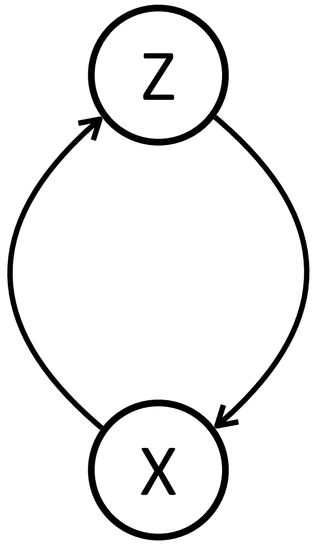
Autoencoder

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:

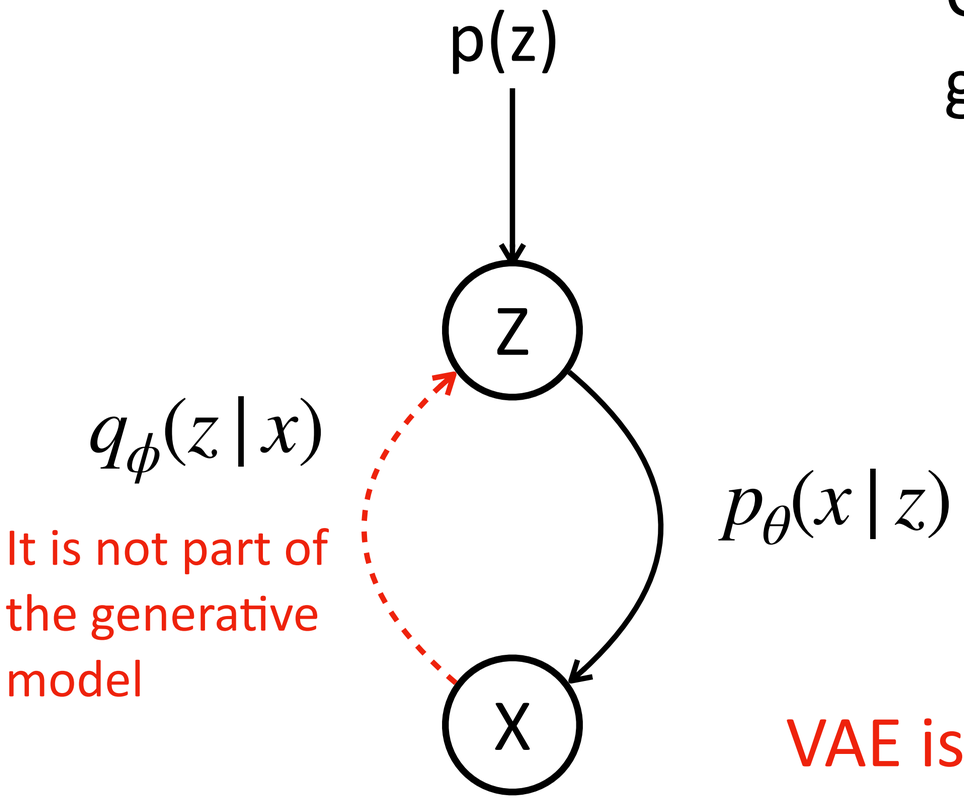
$$\text{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

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

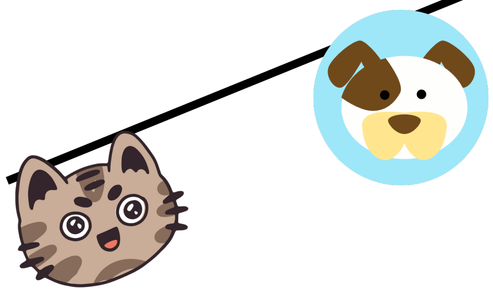
Recap: EM is Hill Climbing

$\log p(x;\theta)$

Only related to θ, no z

ELBO

Larger

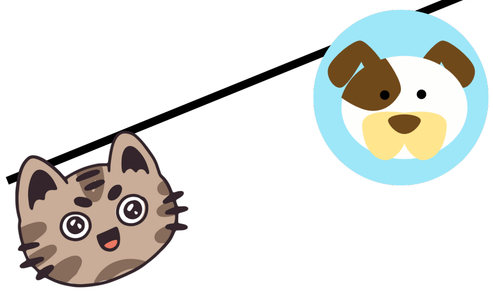


Recap: EM is Hill Climbing

$\log p(x;\theta)$

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

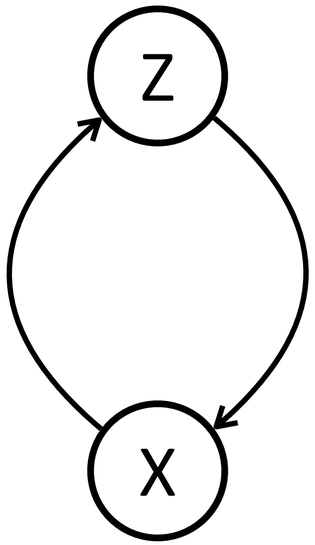
Junxian He, Daniel Spokoyny, Graham Neubig
Carnegie Mellon University
{junxianh,dspokoyn,gneubig}@cs.cmu.edu

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

AutoEncoders

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

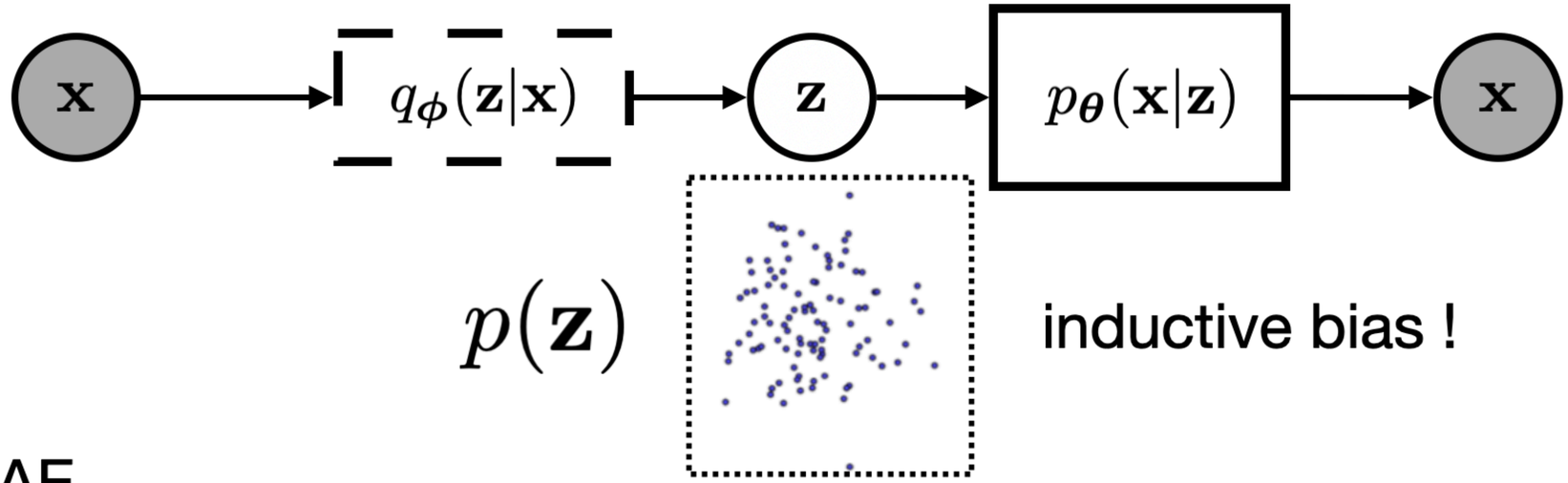
AE: $$\log p_\theta(x \mid q(x))$$



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?

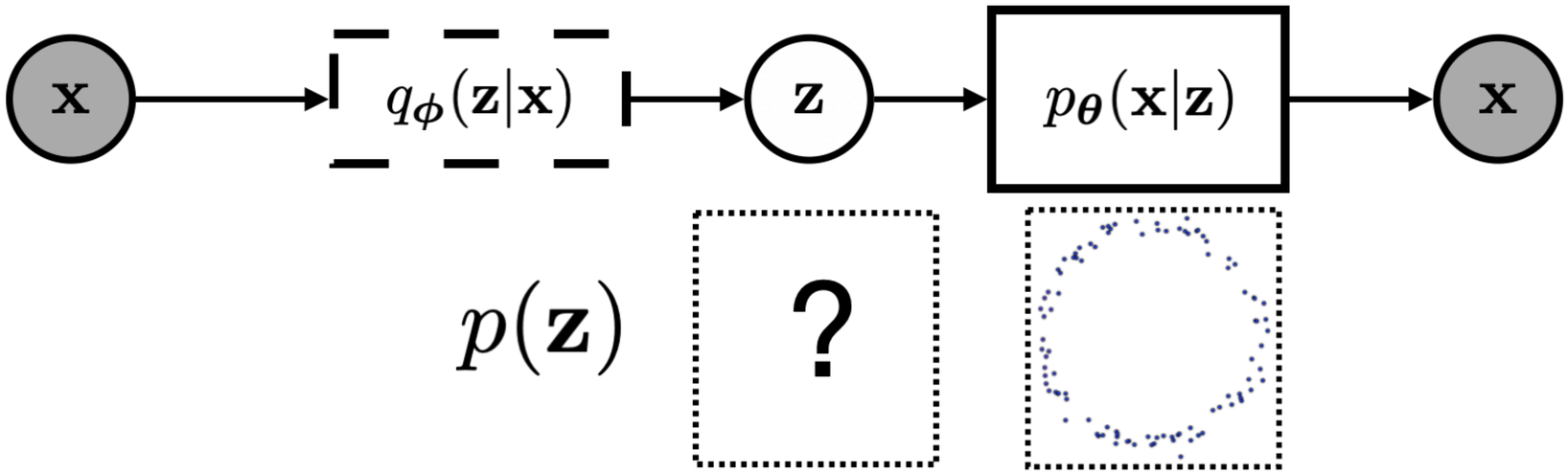
VAE v.s. AE

VAE



inductive bias !

AE



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

Hidden Markov Models

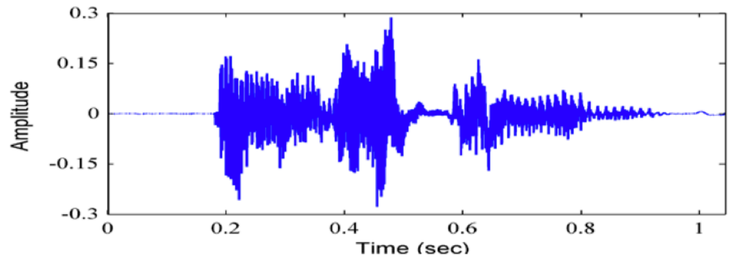
i.i.d to sequential data

- So far we assumed independent, identically distributed data

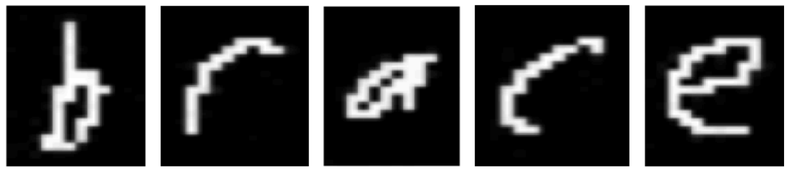
$$\{X_i\}_{i=1}^{n} \overset{iid}{\sim} p(\mathbf{X})$$

- Sequential (non i.i.d.) data
  - Time-series data

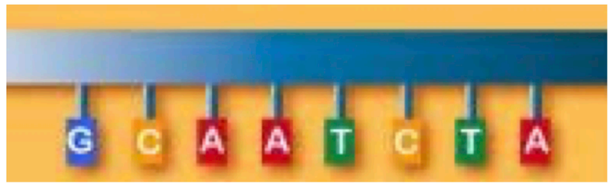
    E.g. Speech



  - Characters in a sentence



  - Base pairs along a DNA strand



(Sequential data is still i.i.d on the sequence level)

Markov Models

- Joint distribution of *n* arbitrary random variables

$$
\begin{aligned}
p(\mathbf{X}) &= p(X_1, X_2, \ldots, X_n) \\
&= p(X_1)p(X_2|X_1)p(X_3|X_2, X_1)\ldots p(X_n|X_{n-1}, ..., X_1) \\
&= \prod_{i=1}^{n} p(X_n|X_{n-1}, \ldots, X_1)
\end{aligned}
$$

Chain rule

- Markov Assumption (mth order)

$$
p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1}, \ldots, X_{n-m})
$$

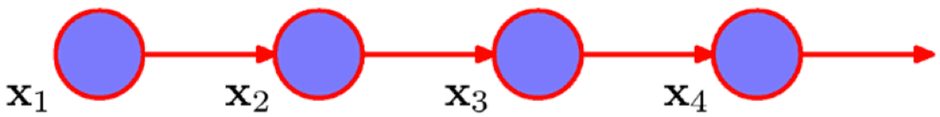
Current observation only depends on past m observations

Markov Models

- Markov Assumption

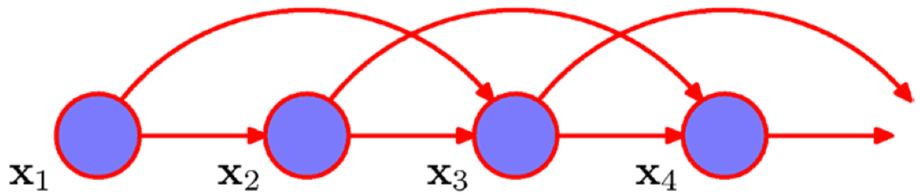
1st order

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1})$$



2nd order

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1}, X_{n-2})$$



Markov Models

Homogeneous/stationary Markov model (probabilities don't depend on n)

parameters in stationary model
K-ary variables

- Markov Assumption

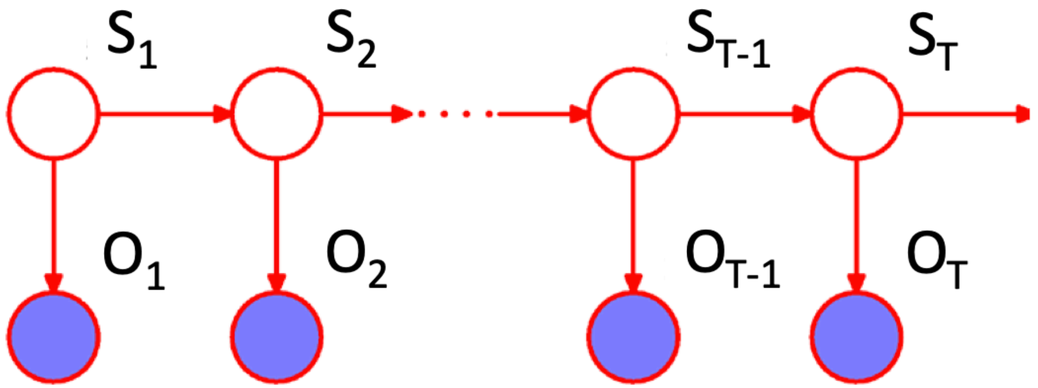
1st order $p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1})$ $O(K^2)$

mth order $p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1},\ldots,X_{n-m})$ $O(K^{m+1})$

n-1th order $p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n|X_{n-1},\ldots,X_1)$ $O(K^n)$

≡ no assumptions – complete (but directed) graph

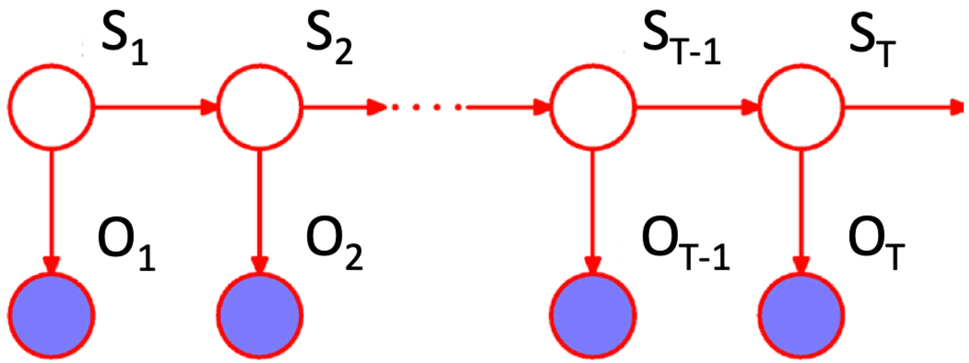
Hidden Markov Models



Observation space $O_t \in \{y_1, y_2, \ldots, y_K\}$

Hidden states $S_t \in \{1, \ldots, I\}$

Hidden Markov Models



$$p(S_1, \ldots, S_T, O_1, \ldots, O_T) = \prod_{t=1}^{T} p(O_t|S_t) \prod_{t=1}^{T} p(S_t|S_{t-1})$$

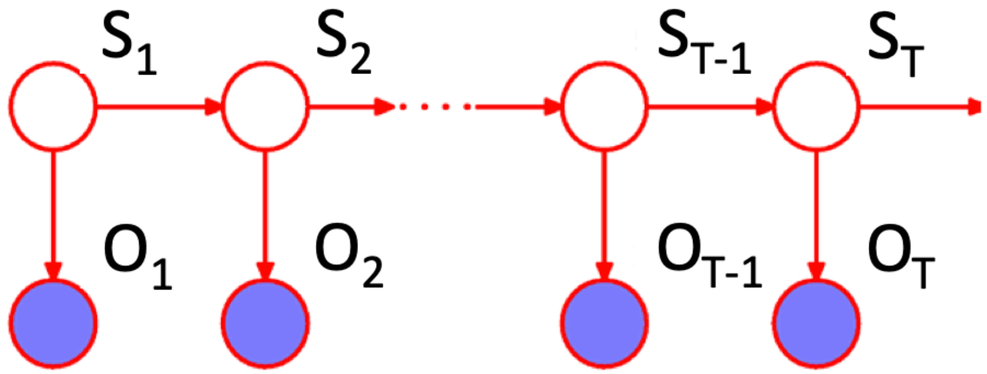
1st order Markov assumption on hidden states $\{S_t\}$ t = 1, …, T

(can be extended to higher order).

Is O_T and O_2 independent?

Hidden Markov Models

- Parameters – stationary/homogeneous markov model (independent of time t)



Initial probabilities

$p(S_1 = i) = \pi_i$

Transition probabilities

$p(S_t = j \mid S_{t-1} = i) = p_{ij}$

Emission probabilities

$p(O_t = y \mid S_t = i) = q_i^y$

$$p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = p(S_1) \prod_{t=2}^{T} p(S_t|S_{t-1}) \prod_{t=1}^{T} p(O_t|S_t)$$

HMM Example

- The Dishonest Casino

A casino has two dices:

Fair dice

P(1) = P(2) = P(3) = P(5) = P(6) = 1/6

Loaded dice

P(1) = P(2) = P(3) = P(5) = 1/10
P(6) = ½

Casino player switches back-&-forth between fair and loaded die with 5% probability

HMM Example

GIVEN: A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

Question

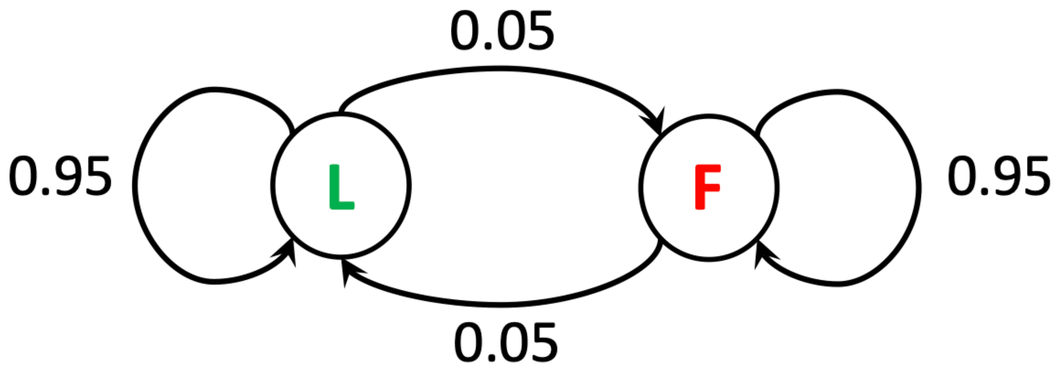
1. How likely is the sequence given our model?
   This is the evaluation problem in HMMs

2. What portion of the sequence was generated with the fair die, and what portion with the loaded die
   This is the decoding question in HMMs

3.How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?
   This is the learning question in HMMs

State Space Representation

- Switch between F and L with 5% probability



- **HMM Parameters**

Initial probs	$P(S_1 = L) = 0.5 = P(S_1 = F)$
Transition probs	$P(S_t = L/F \mid S_{t-1} = L/F) = 0.95$
	$P(S_t = F/L \mid S_{t-1} = L/F) = 0.05$
Emission probabilities	$P(O_t = y \mid S_t = F) = 1/6 \quad y = 1,2,3,4,5,6$
	$P(O_t = y \mid S_t = L) = 1/10 \quad y = 1,2,3,4,5$
	$= 1/2 \quad y = 6$

Three Main Problems in HMMs

- Evaluation – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

  find $p(\{O_t\}_{t=1}^T|\theta)$ prob of observed sequence

- Decoding – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

  find $\arg\max_{s_1,\ldots,s_T} p(\{S_t\}_{t=1}^T|\{O_t\}_{t=1}^T,\theta)$ most probable

  sequence of hidden states

- Learning – Given HMM with unknown parameters and $\{O_t\}_{t=1}^T$ observation sequence

  find $\arg\max_{\theta} p(\{O_t\}_{t=1}^T|\theta)$ parameters that maximize

  likelihood of observed data

HMM Algorithms

- Evaluation – What is the probability of the observed sequence? Forward Algorithm

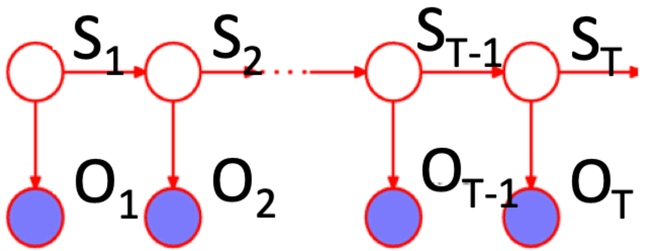
- Decoding – What is the probability that the third roll was loaded given the observed sequence? Forward-Backward Algorithm

  – What is the most likely die sequence given the observed sequence? Viterbi Algorithm

- Learning – Under what parameterization is the observed sequence most probable? Baum-Welch Algorithm (EM)

Evaluation Problem

- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$ & observation sequence $\{O_t\}_{t=1}^T$



find probability of observed sequence

$$p(\{O_t\}_{t=1}^T) = \sum_{S_1,\ldots,S_T} p(\{O_t\}_{t=1}^T, \{S_t\}_{t=1}^T)$$

$$= \sum_{S_1,\ldots,S_T} p(S_1) \prod_{t=2}^T p(S_t|S_{t-1}) \prod_{t=1}^T p(O_t|S_t)$$

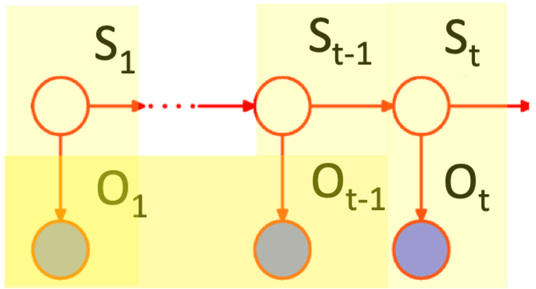
requires summing over all possible hidden state values at all times – K^T exponential # terms!

Forward Probability

$$p(\{O_t\}_{t=1}^T) = \sum_k p(\{O_t\}_{t=1}^T, S_T = k) = \sum_k \alpha_T^k$$

Compute forward probability α_t^k recursively over t

$$\alpha_t^k := p(O_1, \ldots, O_t, S_t = k)$$



Introduce S_{t-1}

$\vdots$

Chain rule

Markov assumption

$$= p(O_t|S_t = k) \sum_i \alpha_{t-1}^i p(S_t = k|S_{t-1} = i)$$

Forward Algorithm

Can compute α_t^k for all k, t using dynamic programming:

- Initialize: $\alpha_1^k = p(O_1 | S_1 = k)\, p(S_1 = k)$ for all k
- Iterate: for t = 2, …, T

$$\alpha_t^k = p(O_t | S_t = k) \sum_i \alpha_{t-1}^i \, p(S_t = k | S_{t-1} = i) \quad \text{for all k}$$

- Termination: $p(\{O_t\}_{t=1}^T) = \sum_k \alpha_T^k$

Can we do in the backward direction?

Decoding Problem 1

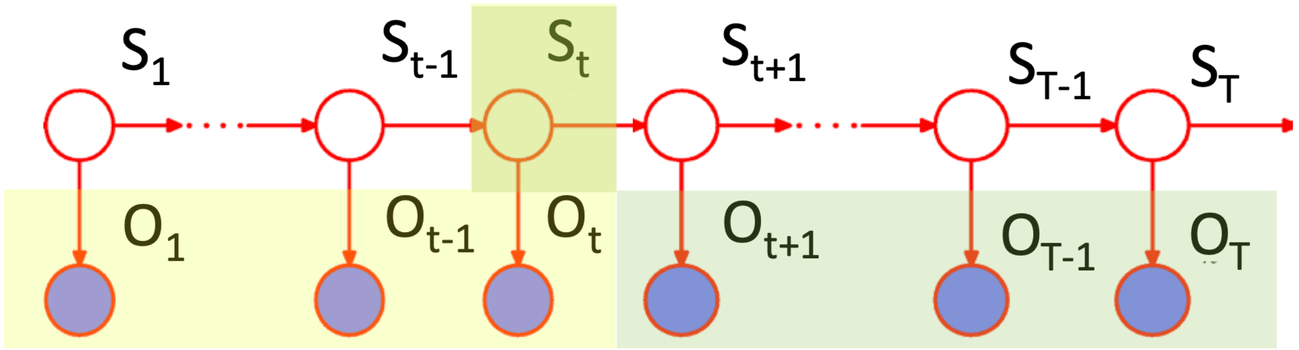
- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$ & observation sequence $\{O_t\}_{t=1}^T$

  find probability that hidden state at time t was k $p(S_t = k|\{O_t\}_{t=1}^T)$

$$
\begin{aligned}
p(S_t = k, \{O_t\}_{t=1}^T) &= p(O_1, \ldots, O_t, S_t = k, O_{t+1}, \ldots, O_T) \\
&= p(O_1, \ldots, O_t, S_t = k) p(O_{t+1}, \ldots, O_T | S_t = k)
\end{aligned}
$$

Compute recursively

α_t^k β_t^k



Forward-Backward Algorithm

Can compute β_t^k for all k, t using dynamic programming:

- Initialize: $\beta_T^k = 1$ for all k

- Iterate: for t = T-1, …, 1

$$\beta_t^k = \sum_i p(S_{t+1} = i | S_t = k) p(O_{t+1} | S_{t+1} = i) \beta_{t+1}^i \quad \text{for all k}$$

- Termination: $p(S_t = k, \{O_t\}_{t=1}^T) = \alpha_t^k \beta_t^k$

$$p(S_t = k | \{O_t\}_{t=1}^T) = \frac{p(S_t = k, \{O_t\}_{t=1}^T)}{p(\{O_t\}_{t=1}^T)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_t^i \beta_t^i}$$

Most Likely State vs. Most Likely Sequence

- Most likely state assignment at time t

$$\arg\max_k p(S_t = k|\{O_t\}_{t=1}^T) = \arg\max_k \alpha_t^k \beta_t^k$$

E.g. Which die was most likely used by the casino in the third roll given the observed sequence?

- Most likely assignment of state sequence

$$\arg\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T|\{O_t\}_{t=1}^T)$$

Are the solutions the same?

Decoding Problem 2

- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$ & observation sequence $\{O_t\}_{t=1}^T$

  find most likely assignment of state sequence

$$\arg \max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T) = \arg \max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T)$$

$$= \arg \max_k \underbrace{\max_{\{S_t\}_{t=1}^{T-1}} p(S_T = k, \{S_t\}_{t=1}^{T-1}, \{O_t\}_{t=1}^T)}_{V_T^k}$$

Compute recursively

V_T^k - probability of most likely sequence of states ending at state $S_T = k$

Viterbi Decoding

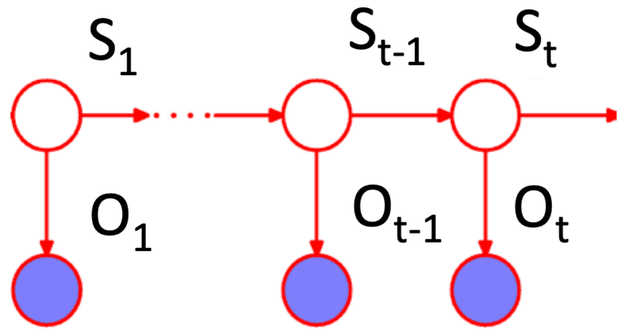
$$\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = \max_k V_T^k$$

Compute probability V_t^k recursively over t

$$V_t^k := \max_{S_1,\ldots,S_{t-1}} p(S_t = k, S_1, \ldots, S_{t-1}, O_1, \ldots, O_t)$$

. Bayes rule

. Markov assumption



$$= p(O_t|S_t = k) \max_i p(S_t = k|S_{t-1} = i) V_{t-1}^i$$

Viterbi Algorithm

Can compute V_t^k for all k, t using dynamic programming:

- Initialize: $V_1^k = p(O_1|S_1=k)p(S_1 = k)$ for all k

- Iterate: for t = 2, …, T

$$V_t^k = p(O_t|S_t = k) \max_i p(S_t = k|S_{t-1} = i)V_{t-1}^i \quad \text{for all k}$$

- Termination: $\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = \max_k V_T^k$

Traceback:

$$S_T^* = \arg\max_k V_T^k$$

$$S_{t-1}^* = \arg\max_i p(S_t^*|S_{t-1} = i)V_{t-1}^i$$

Can we do in the backward direction?

Computational Complexity

- What is the running time for Forward, Backward, Viterbi?

$$\alpha_t^k = q_k^{O_t} \sum_i \alpha_{t-1}^i \; p_{i,k}$$

$$\beta_t^k = \sum_i p_{k,i} \; q_i^{O_{t+1}} \; \beta_{t+1}^i$$

$$V_t^k = q_k^{O_t} \; \max_i p_{i,k} \; V_{t-1}^i$$

$O(K^2T)$ linear in T instead of $O(K^T)$ exponential in T!

Learning with EM

- Start with random initialization of parameters
- **E-step** – Fix parameters, find expected state assignments

$$\gamma_i(t) = p(S_t = i|O,\theta) = \frac{\alpha_t^i \beta_t^i}{\sum_j \alpha_t^j \beta_t^j} \qquad \mathbf{O} = \{O_t\}_{t=1}^T$$

Forward-Backward algorithm

$$\xi_{ij}(t) = p(S_{t-1}= i, S_t = j|O,\theta)$$

$$= \frac{p(S_{t-1} = i|O,\theta)p(S_t = j, O_t, \ldots, O_T|S_{t-1} = i,\theta)}{p(O_t, \ldots, O_T|S_{t-1} = i,\theta)}$$

$$= \frac{\gamma_i(t-1)\ p_{ij}\ q_j^{O_t}\ \beta_t^j}{\beta_{t-1}^i}$$

You will derive the EM in your HW

Thank You!
Q & A