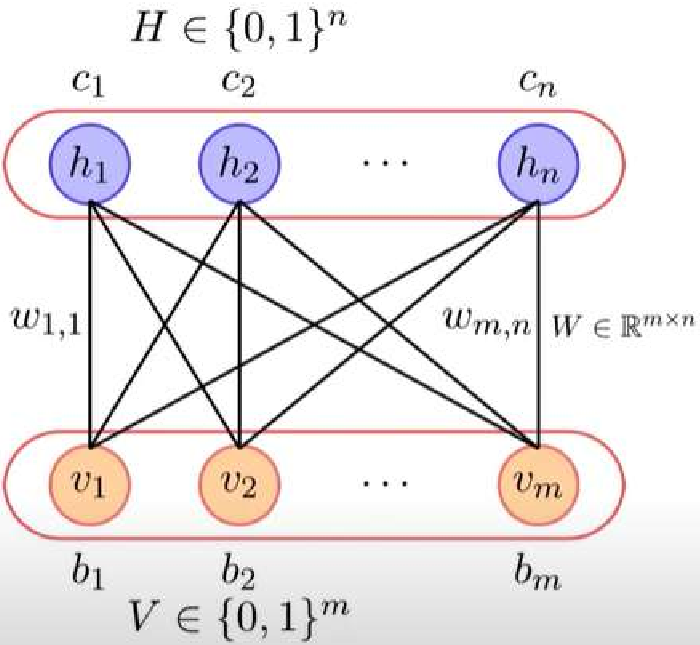
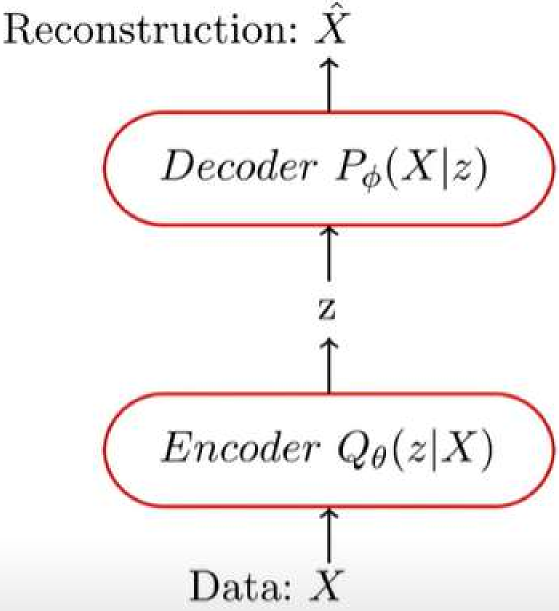
Figure: Abstraction

Figure: Generation

- Let $\{X = x_i\}_{i=1}^{N}$ be the training data
- We can think of X as a random variable in R^n
- For example, X could be an image and the dimensions of X correspond to pixels of the image
- We are interested in learning an abstraction (i.e., given an X find the hidden representation z)
- We are also interested in generation (*i.e.*, given a hidden representation generate an X)
- In probabilistic terms we are interested in $P(z|X)$ and $P(X|z)$ (to be consistent with the literation on VAEs we will use z instead of H and X instead of V)

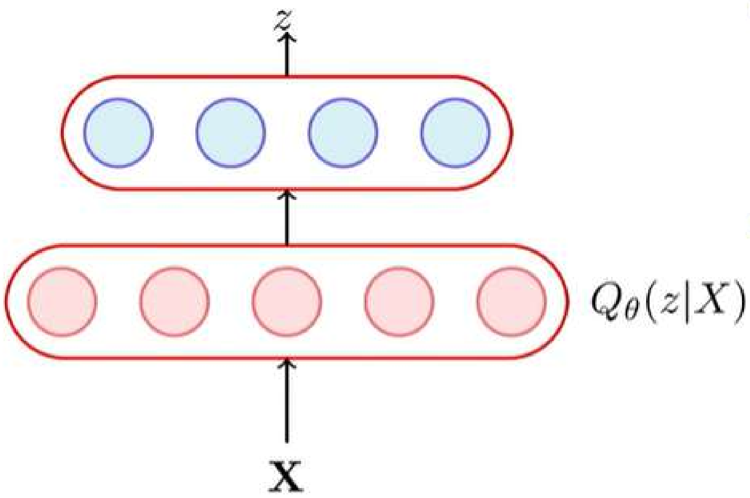


- Earlier we saw RBMs where we learnt $P(z|X)$ and $P(X|z)$
- Below we list certain characteristics of RBMs
- **Structural assumptions:** We assume certain independencies in the Markov Network
- **Computational:** When training with Gibbs Sampling we have to run the Markov Chain for many time steps which is expensive
- **Approximation:** When using Contrastive Divergence, we approximate the expectation by a point estimate
- (Nothing wrong with the above but we just mention them to make the reader aware of these characteristics)

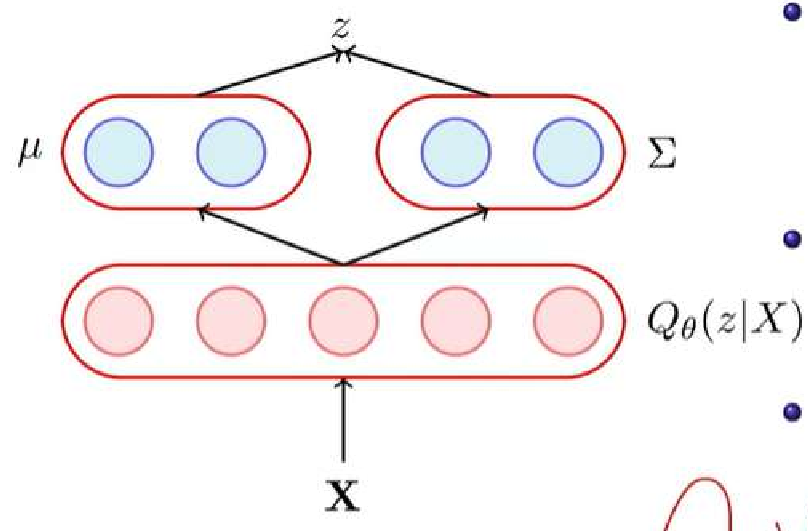


- We now return to our goals
- **Goal 1:** Learn a distribution over the latent variables ($Q(z|X)$)
- **Goal 2:** Learn a distribution over the visible variables ($P(X|z)$)
- VAEs use a neural network based encoder for Goal 1
- and a neural network based decoder for Goal 2
- We will look at the encoder first

θ: the parameters of the encoder neural network
ϕ: the parameters of the decoder neural network

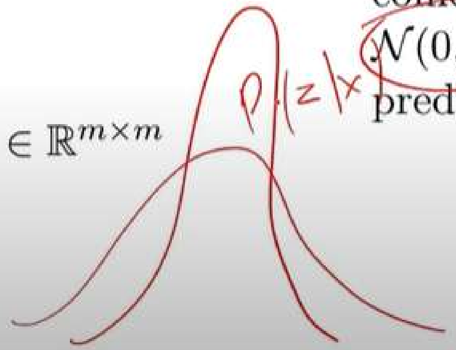


- **Encoder:** What do we mean when we say we want to learn a distribution? We mean that we want to learn the parameters of the distribution
- But what are the parameters of $Q(z|X)$? Well it depends on our modeling assumption!

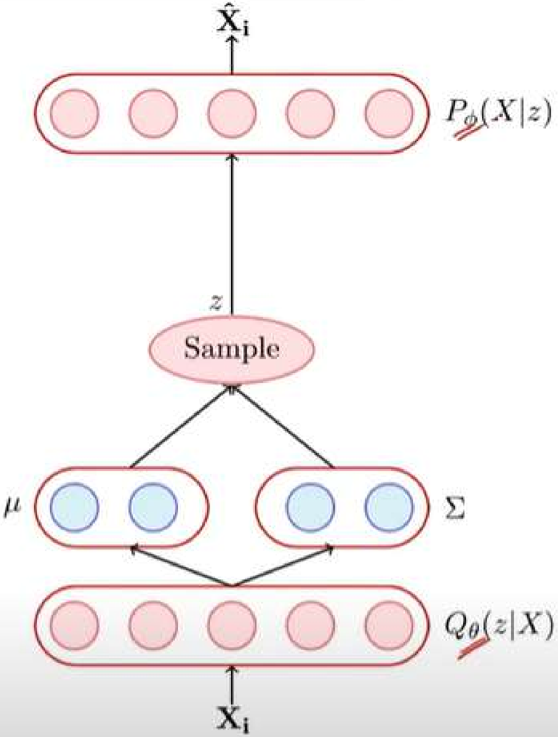


- **Encoder:** What do we mean when we say we want to learn a distribution? We mean that we want to learn the parameters of the distribution
- But what are the parameters of $Q(z|X)$? Well it depends on our modeling assumption!
- In VAEs we assume that the latent variables come from a standard normal distribution $\mathcal{N}(0, I)$ and the job of the encoder is to then predict the parameters of this distribution

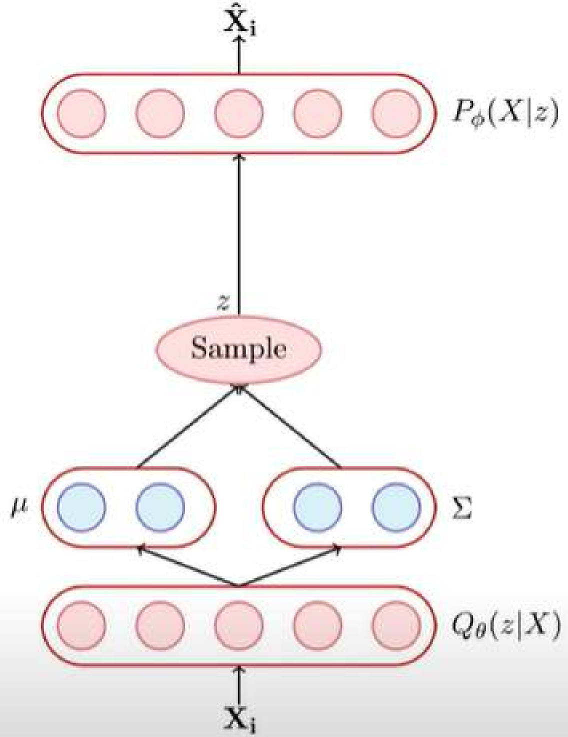
$X \in \mathbb{R}^n$, $\mu \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m\times m}$



$$\mu, \Sigma = f_\theta(x)$$



- Now what about the decoder?
- The job of the decoder is to predict a probability distribution over $X : P(X|z)$
- Once again we will assume a certain form for this distribution
- For example, if we want to predict 28 x 28 pixels and each pixel belongs to $\mathbb{R}$ (*i.e.*, $X \in \mathbb{R}^{784}$) then what would be a suitable family for $P(X|z)$?
- We could assume that $P(X|z)$ is a Gaussian distribution with unit variance
- The job of the decoder f would then be to predict the mean of this distribution as $f_\phi(z)$



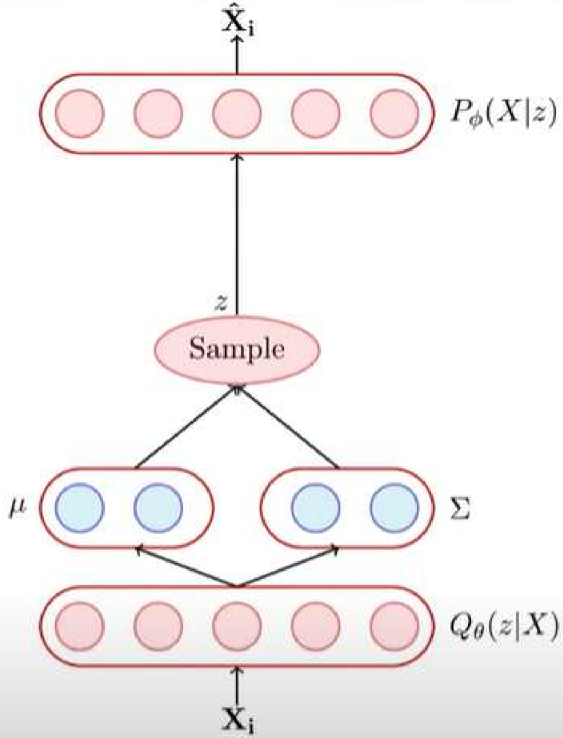
- What would be the objective function of the decoder ?
- For any given training sample x_i it should maximize $P(x_i)$ given by

$$
\begin{aligned}
P(x_i) &= \int P(z)P(x_i|z)dz \\
&= -\mathbb{E}_{z\sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)]
\end{aligned}
$$

- (As usual we take log for numerical stability)

21:07 / 34:31

Deep Learning Part - II (CS7015): Lec 20.2 Variational Autoencoders: The Neural Network Perspective



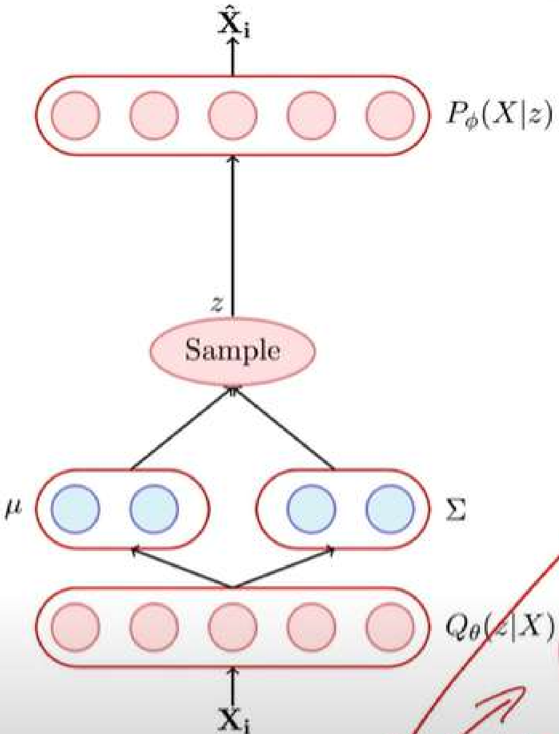
- KL divergence captures the difference (or distance) between 2 distributions

- This is the loss function for one data point ($l_i(\theta)$) and we will just sum over all the data points to get the total loss $\mathscr{L}(\theta)$

$$\mathscr{L}(\theta) = \sum_{i=1}^{m} l_i(\theta)$$

- In addition, we also want a constraint on the distribution over the latent variables
- Specifically, we had assumed $P(z)$ to be $\mathcal{N}(0, I)$ and we want $Q(z|X)$ to be as close to $P(z)$ as possible
- Thus, we will modify the loss function such that

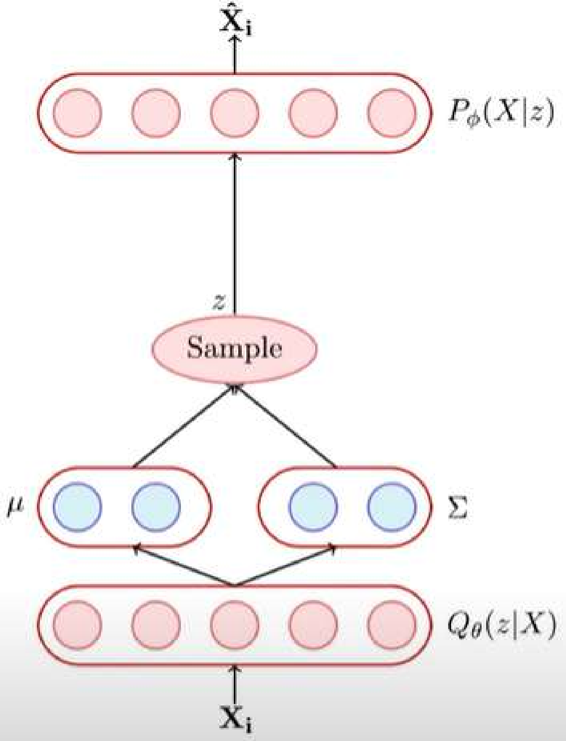
$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] \\ + KL(Q_\theta(z|x_i)||P(z))$$



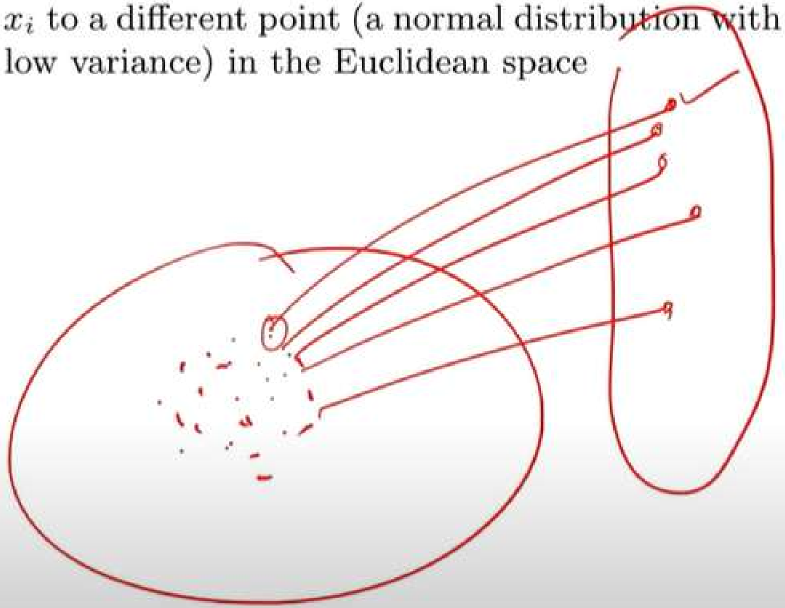
- The second term in the loss function can actually be thought of as a regularizer

$$\mathcal{L}(\theta) + \Omega(\theta)$$

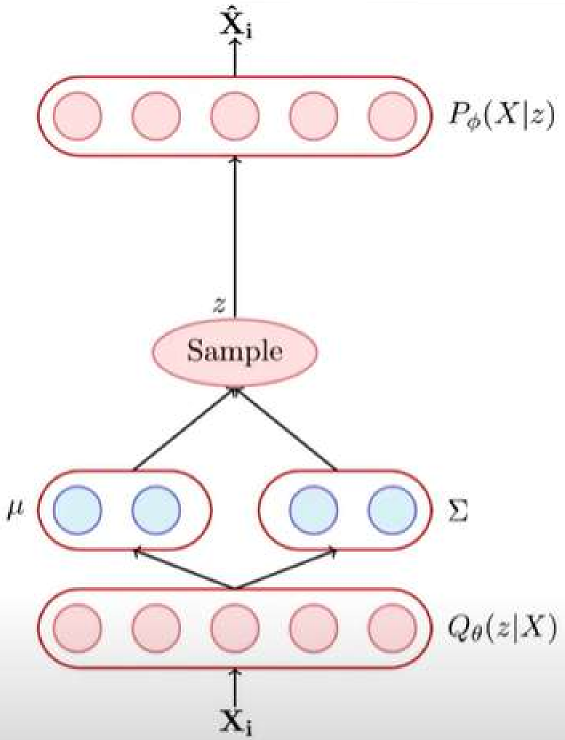
$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$



- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space

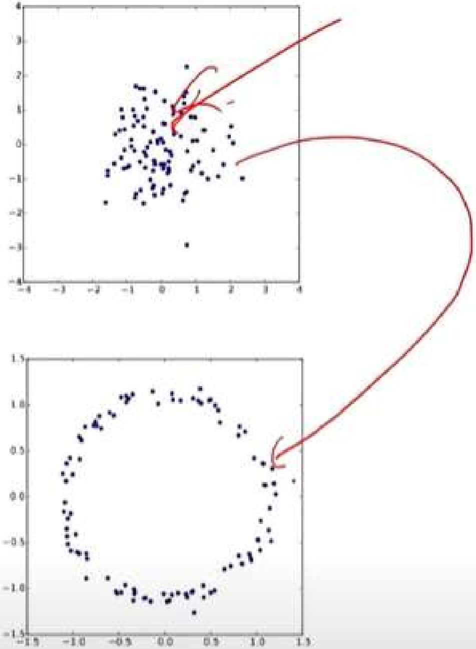


$$
l_i(\theta, \phi) = -\mathbb{E}_{z \sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))
$$



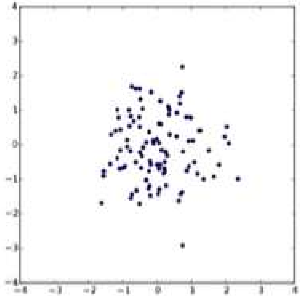
$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

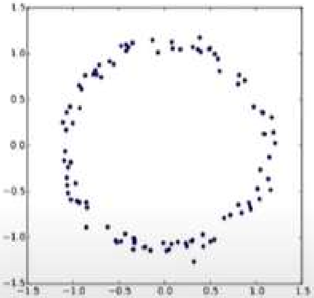
- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space
- In other words, in the absence of the regularizer the encoder can learn a unique mapping for each x_i and the decoder can then decode from this unique mapping
- Even with high variance in samples from the distribution, we want the decoder to be able to reconstruct the original data very well (motivation similar to the adding noise)
- To summarize, for each data point we predict a distribution such that, with high probability a sample from this distribution should be able to reconstruct the original data point
- But why do we choose a normal distribution? Isn't it too simplistic to assume that z follows a normal distribution



$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

- Isn't it a very strong assumption that $P(z) \sim \mathcal{N}(0, I)$?
- For example, in the 2-dimensional case how can we be sure that $P(z)$ is a normal distribution and not any other distribution
- The key insight here is that any distribution in d dimensions can be generated by the following steps
- Step 1: Start with a set of d variables that are normally distributed (that's exactly what we are assuming for $P(z)$)
- Step 2: Mapping these variables through a sufficiently complex function (that's exactly what the first few layers of the decoder can do)





$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

- In particular, note that in the adjoining example if z is 2-D and normally distributed then $f(z)$ is roughly ring shaped (giving us the distribution in the bottom figure)

$$f(z) = \frac{z}{10} + \frac{z}{||z||}$$

- A non-linear neural network, such as the one we use for the decoder, could learn a complex mapping from z to $f_\phi(z)$ using its parameters ϕ
- The initial layers of a non linear decoder could learn their weights such that the output is $f_\phi(z)$
- The above argument suggests that even if we start with normally distributed variables the initial layers of the decoder could learn a complex transformation of these variables say $f_\phi(z)$ if required
- The objective function of the decoder will ensure that an appropriate transformation of z is learnt to reconstruct X