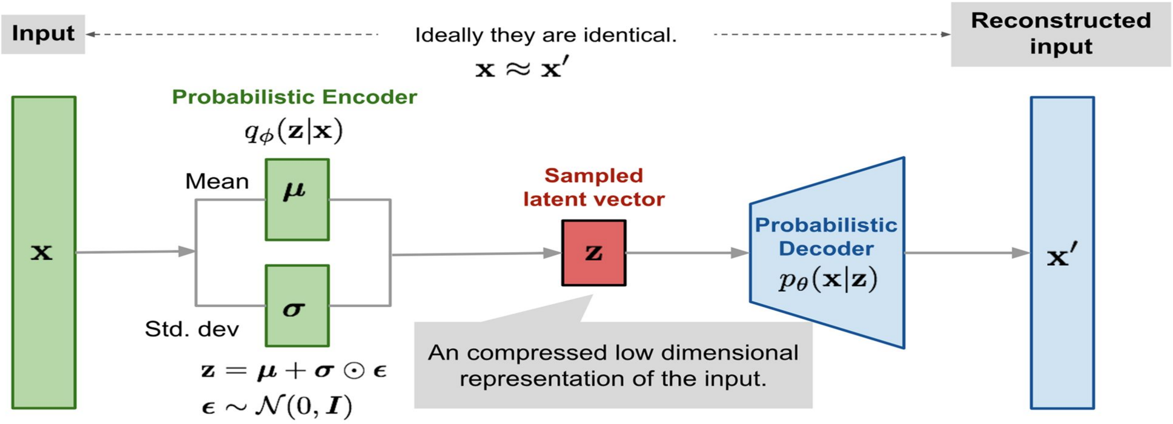
VAE Motivation

Instead of mapping the input into a *fixed* vector, we want to map it into a distribution.
Blog to read: https://www.jeremyjordan.me/variational-autoencoders/
Tutorial - What is a variational autoencoder? – Jaan Altosaar


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{x}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

We will now look at variational autoencoders which have the same structure as autoencoders but they learn a distribution over the hidden variables

Variational Autoencoders: The Neural Network Perspective

VAE - Goal

The goal of VAE is to find a distribution $Q\theta(z|x)$ *of some latent varaibles , which we can sample from* Z ~ $Q_{\varphi}(z|x)$, to generate new sampes x` from $P_{\varphi}(x|z)$

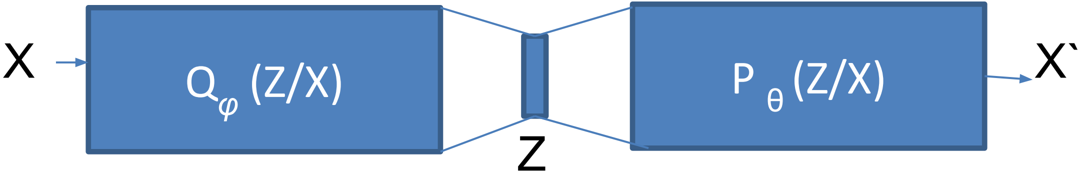


Figure: Abstraction

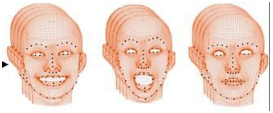
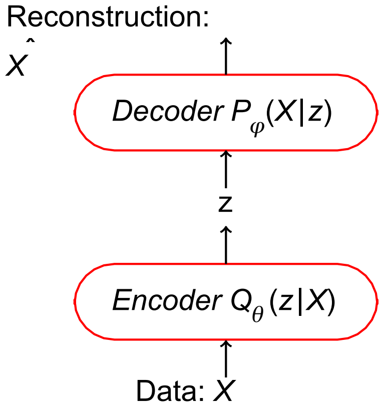


Figure: Generation

- 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)



Reconstruction: $\hat{X}$

$Decoder\ P_\varphi(X|z)$

z

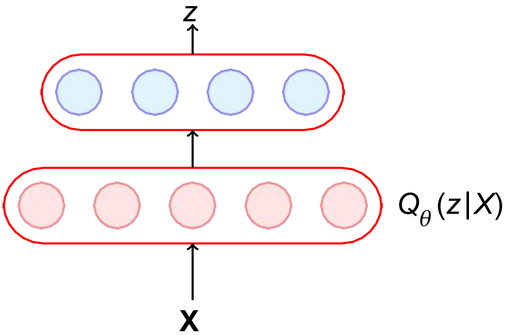
$Encoder\ Q_\theta(z|X)$

Data: X

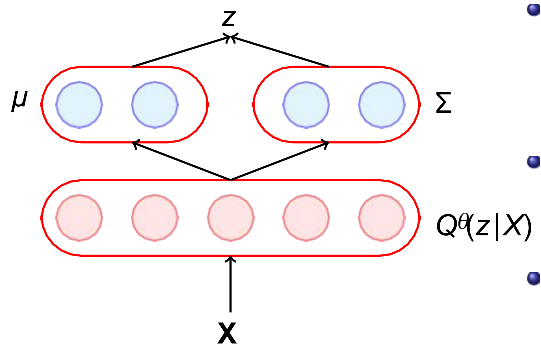
θ: the parameters of the encoder neural network

φ: the parameters of the decoder neural network

- 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



- **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!

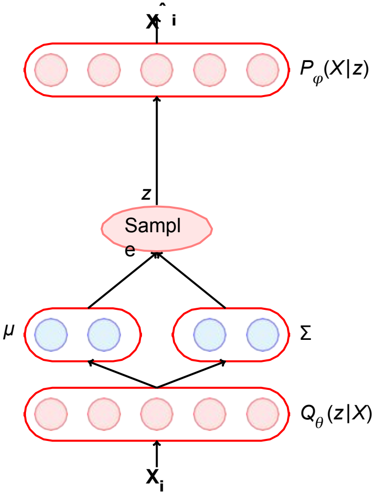


$X \in \mathrm{R}^n$, $\mu \in \mathrm{R}^m$ and $\Sigma \in \mathrm{R}^{m \times m}$

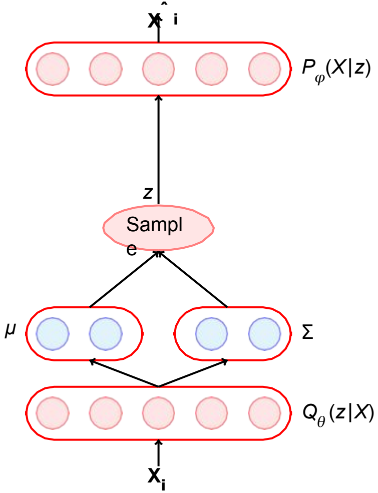
- **Encoder:** What do we mean when we say we want to learn a distribution? We mean that we want to learn the parameters of the distribution
- But what are the parameters of $Q(z|X)$?

Well it depends on our modeling assumption!

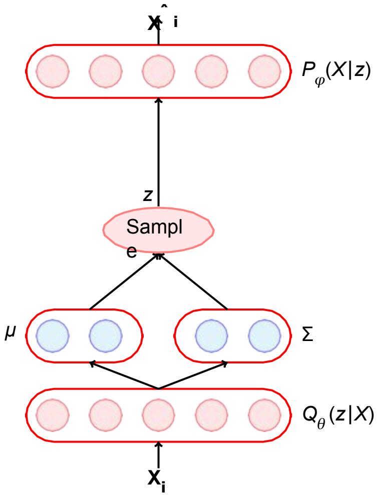
- In VAEs we assume that the latent variables come from a standard normal distribution N (0, I) and the job of the encoder is to then predict the parameters of this distribution



- Now what about the decoder?
- The job of the decoder is to predict a probab- ility 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 R (*i.e.*, $X \in$ R^{784}) then what would be a suitable family for $P(X|z)$?



- 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 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 $f_\varphi(z)$

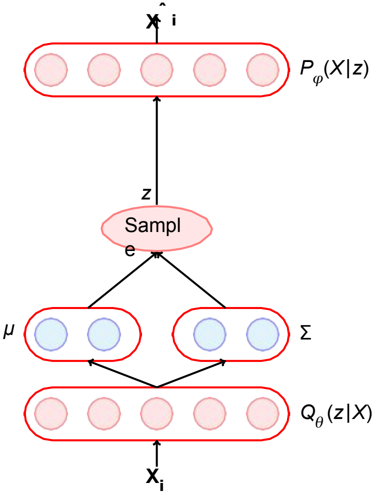


- What would be the objective function of the decoder ?
- For any given training sample x_i it should maximize $P(x_i)$ given by

$$P(x_i) = P(z)P(x_i|z)dz$$

$$= -\mathrm{E}_{z\sim Q\theta(z|xi)}[\log P_\varphi(x_i|z)]$$

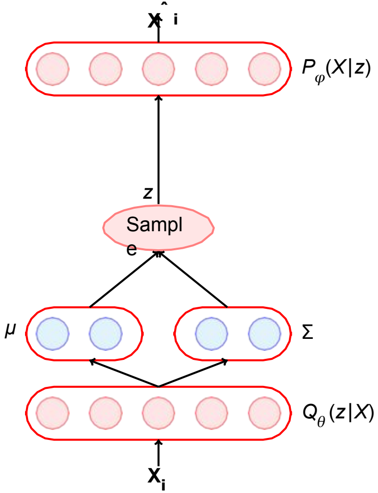
- (As usual we take log for numerical stability)



- 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 $N(0, I)$ and we want $Q(z|X)$ to be as close to $P(z)$ as possible

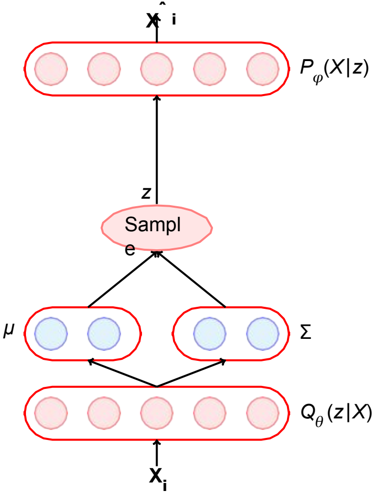


- This is the loss function for one data point ($l_i(\theta)$) and we will just sum over all the data points to get the total loss $\mathcal{L}(\theta)$

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

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

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



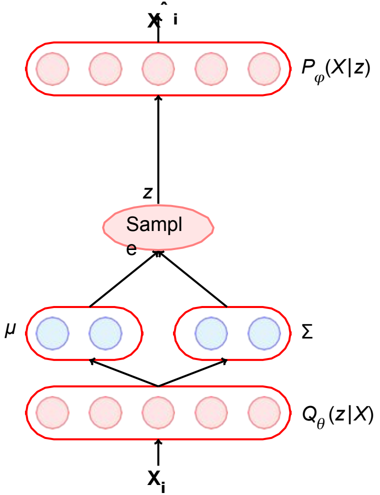
- KL divergence captures the difference (or distance) between 2 distributions

- 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 $\mathcal{L}(\theta)$

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

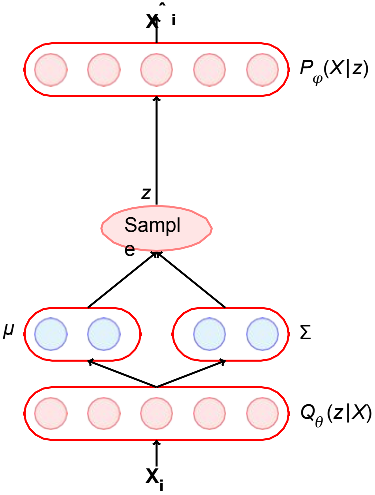
- In addition, we also want a constraint on the distribution over the latent variables
- Specifically, we had assumed $P(z)$ to be $N(0, I)$ and we want $Q(z|X)$ to be as close to $P(z)$ as possible
- Thus, we will modify the loss function such that

$$l_i(\theta, \varphi) = -E_{z\sim Q_\theta(z|x_i)}[\log P_\varphi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$



$$l_i(\theta, \varphi) = -E_{z \sim Q_\theta(z|x_i)}[\log P_\varphi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

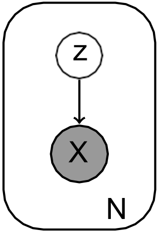
- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space
- In other words, in the absence of the regularizer the encoder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



$$l_i(\theta, \varphi) = -E_{z \sim Q_\theta(z|x_i)}[\log P_\varphi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

- 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

Module 21.3: Variational autoencoders: (The graphical model perspective)



The inference problem is to compute the conditional Distribution of latent variables given observations

Modelling p(z/x) using Q(z/x) where q(z/x) has a simple Gaussian distribution

- Now at inference time, we are given an X (observed variable) and we are interested in finding the most likely assignments of latent variables z which would have resulted in this observation
- Mathematically, we want to find

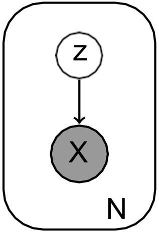
$$P(z|X) = \frac{P(X|z)P(z)}{P(X)}$$

- This is hard to compute because the LHS contains $P(X)$ which is intractable require exponential time to compute

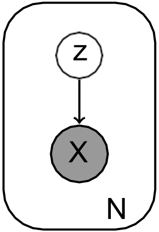
$$P(X) = P(X|z)P(z)dz$$

$$= \hat{}\ \hat{}\ \ldots\ \hat{}\ P(X|z_1, z_2, \ldots, z_n)P(z_1, z_2, \ldots, z_n)dz_1, \ldots dz_n$$

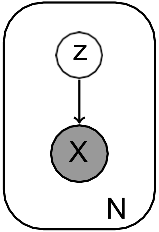
- The alternative is to approximate p(z/x) by another distribution q(z/x) which have tractable solutions.
- This is a inference problem solved as an optimaztaion problem



- Here we can think of z and X as random vari- ables
- We are then interested in the joint probability distribution $P(X, z)$ which factorizes as $P(X, z) = P(z)P(X|z)$
- This factorization is natural because we can imagine that the latent variables are fixed first and then the visible variables are drawn based on the latent variables
- For example, if we want to draw a digit we could first fix the latent variables: *the digit, size, angle, thickness, position and so on* and then draw a digit which corresponds to these latent variables



- Specifically, in VAEs, we assume that instead of $P(z|X)$ which is intractable, the posterior distribution is given by $Q_\theta(z|X)$
- Further, we assume that $Q_\theta(z|X)$ is a Gaussian whose parameters are determined by a neural network $\mu, \Sigma = g_\theta(X)$
- The parameters of the distribution are thus determined by the parameters θ of a neural network
- Our job then is to learn the parameters of this neural network



- But what is the objective function for this neural network
- Well we want the proposed distribution Q_θ $(z|X)$ to be as close to the true distribu- tion

- We can capture this using the following objective function

$$minimize\ KL(Q_\theta(z|X)||P(z|X))$$

- What are the parameters of the objective function ? (they are the parameters of the neural network - we will return back to this again)

KL betⁿ $P(Z/x)$ & $q(z/x)$ / $q(Z/x)$ is a Gaussian distribution

$$D_{KL}(q_\phi(Z/x) \,||\, P_\theta(Z/x))$$

$$= \sum q_\phi(z/x) \log\left(\frac{q_\phi(z/x)}{P_\theta(z/x)}\right)$$

$$= E_{z \sim q_\phi(z/x)}\left[\log \frac{q_\phi(z/x)}{P_\theta(z/x)}\right]$$

$$= E_{z \sim q_\phi(z/x)}\left[\log q_\phi(z/x) - \log P_\theta(z/x)\right]$$

∵ $z = z \sim q_\phi(z/x)$

$$= E_z\left[\log q_\phi(z/x) - \log \frac{P_\theta(x/z) P_\theta(z)}{P_\theta(x)}\right]$$

$$= E_z\left[\log q_\phi(z/x) - \log P_\theta(x/z) - P_\theta(z) + \log P_\theta(x)\right]$$

∵ $P_\theta(z)$ does not involve z

$$= E_z\left[\log q_\phi(z/x) - \log P_\theta(x/z) - \log P_\theta(z)\right]$$

$D_{KL}[q_\phi(z/x) \,||\, P_\theta(z/x)] - \log P_\theta(x) =$ ———— " ———— .

Rearranging the eq^n we obtain $\quad \because Z = q\phi_{\theta}(z|x)$

$$\log P_\theta(x) - DL[q_\phi(z|x) \,||\, P_\theta(z|x)]$$

$$= E_z[\log(P_\theta(x/z)] - E_z[\log q_\phi(z|x) - \log P_\theta(z)]$$

$$= \underbrace{E_z[\log P_\theta(x|z)]}_{\text{likehood}} - \underbrace{D_{KL}[q_\phi(z|x) \,||\, P_\theta(z)]}_{\text{relarned distribution } q \text{ is similar to } p.}$$

Loss = − obj f^n.

$$L(\theta,\phi) = -E_{z\sim q_\phi(z|x)}[\log P_\theta(x/z)] + D_{KL}[q_\phi(z|x) \,||\, P_\theta(z)]$$

Loss Function of VAE

$$Li(\theta,\phi)=-Ez\sim q\theta(z\mid xi)[\mathrm{log}p\phi(xi\mid z)]+\mathrm{KL}(q\theta(z\mid xi)\mid\mid p(z))$$

Loss = Reconstruction loss + Regularization

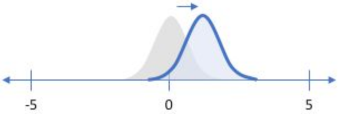
- **Without Reconstruction loss**: Learned distribution deviate form the prior

- **Without Regularization** :Network cheat by learning narrow distribution With small variance this distribution is effectively representing single value

- **With both term in loss function** : the two distribution will be attracted due to first term i.e. Reconstruction loss Sufficient variance is ensured using KL Divergence .

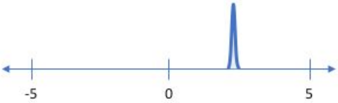
Loss function understanding

Penalizing reconstruction loss encourages the distribution to describe the input



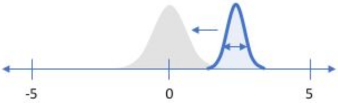
Our distribution deviates from the prior to describe some characteristic of the data

Without regularization, our network can “cheat” by learning narrow distributions



With a small enough variance, this distribution is effectively only representing a single value

Penalizing KL divergence acts as a regularizing force



Attract distribution to have zero mean

Ensure sufficient variance to yield a smooth latent space