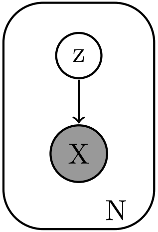
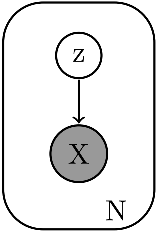
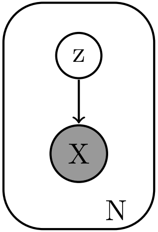
Module 21.3: Variational autoencoders: (The graphical model perspective)



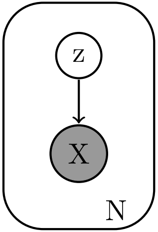
- Here we can think of z and X as random variables



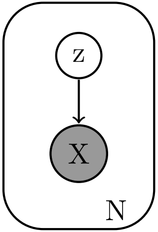
- Here we can think of z and X as random variables
- We are then interested in the joint probability distribution $P(X, z)$ which factorizes as $P(X, z) = P(z)P(X|z)$



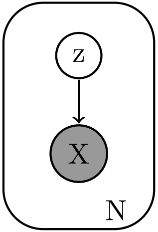
- Here we can think of z and X as random variables
- We are then interested in the joint probability distribution $P(X, z)$ which factorizes as $P(X, z) = P(z)P(X|z)$
- This factorization is natural because we can imagine that the latent variables are fixed first and then the visible variables are drawn based on the latent variables



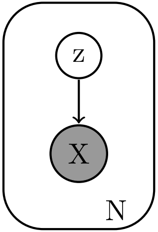
- Here we can think of z and X as random variables
- We are then interested in the joint probability distribution $P(X, z)$ which factorizes as $P(X, z) = P(z)P(X|z)$
- This factorization is natural because we can imagine that the latent variables are fixed first and then the visible variables are drawn 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



- Here we can think of z and X as random variables
- 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
- And of course, unlike RBMs, this is a directed graphical model

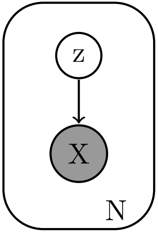


- Now at inference time, we are given an X (observed variable) and we are interested in finding the most likely assignments of latent variables z which would have resulted in this observation



- 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

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



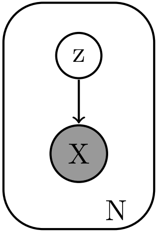
- Now at inference time, we are given an X (observed variable) and we are interested in finding the most likely assignments of latent variables z which would have resulted in this observation
- Mathematically, we want to find

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

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

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

$$= \int \int ... \int P(X|z_1, z_2, ..., z_n)P(z_1, z_2, ..., z_n)dz_1, ...dz_n$$



- 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

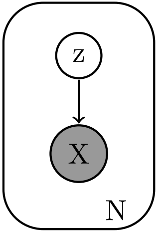
$$P(z|X) = \frac{P(X|z)P(z)}{P(X)}$$

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

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

$$= \int \int ... \int P(X|z_1, z_2, ..., z_n)P(z_1, z_2, ..., z_n)dz_1, ...dz_n$$

- In RBMs, we had a similar integral which we approximated using Gibbs Sampling



- 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

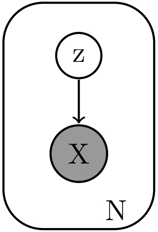
$$P(z|X) = \frac{P(X|z)P(z)}{P(X)}$$

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

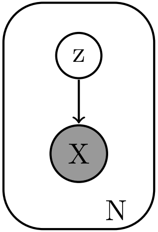
$$P(X) = \int P(X|z)P(z)dz$$

$$= \int\int ... \int P(X|z_1, z_2, ..., z_n)P(z_1, z_2, ..., z_n)dz_1, ...dz_n$$

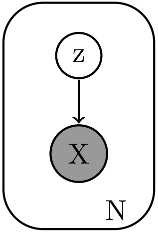
- In RBMs, we had a similar integral which we approximated using Gibbs Sampling
- VAEs, on the other hand, cast this into an optimization problem and learn the parameters of the optimization problem



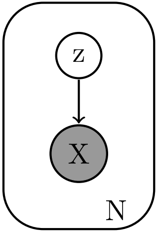
- Specifically, in VAEs, we assume that instead of $P(z|X)$ which is intractable, the posterior distribution is given by $Q_\theta(z|X)$



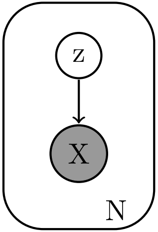
- Specifically, in VAEs, we assume that instead of $P(z|X)$ which is intractable, the posterior distribution is given by $Q_\theta(z|X)$
- Further, we assume that $Q_\theta(z|X)$ is a Gaussian whose parameters are determined by a neural network $\mu, \Sigma = g_\theta(X)$



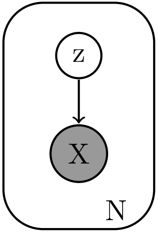
- Specifically, in VAEs, we assume that instead of $P(z|X)$ which is intractable, the posterior distribution is given by $Q_\theta(z|X)$
- Further, we assume that $Q_\theta(z|X)$ is a Gaussian whose parameters are determined by a neural network $\mu, \Sigma = g_\theta(X)$
- The parameters of the distribution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



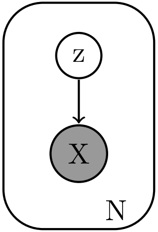
- Specifically, in VAEs, we assume that instead of $P(z|X)$ which is intractable, the posterior distribution is given by $Q_\theta(z|X)$
- Further, we assume that $Q_\theta(z|X)$ is a Gaussian whose parameters are determined by a neural network $\mu, \Sigma = g_\theta(X)$
- The parameters of the distribution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

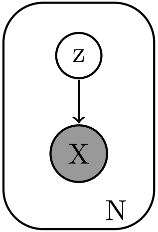


- But what is the objective function for this neural network
- Well we want the proposed distribution $Q_\theta(z|X)$ to be as close to the true distribution



- But what is the objective function for this neural network
- Well we want the proposed distribution $Q_\theta(z|X)$ to be as close to the true distribution
- We can capture this using the following objective function

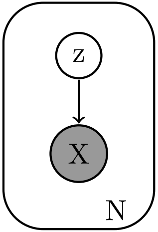
$$minimize \;\; KL(Q_\theta(z|X)||P(z|X))$$



- But what is the objective function for this neural network
- Well we want the proposed distribution $Q_\theta(z|X)$ to be as close to the true distribution
- 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 ?



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

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)]$$

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X)\log Q_\theta(z|X)dz - \int Q_\theta(z|X)\log P(z|X)dz$$

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X) \log Q_\theta(z|X) dz - \int Q_\theta(z|X) \log P(z|X) dz$$
$$= \mathbb{E}_{z \sim Q_\theta(z|X)}[\log Q_\theta(z|X) - \log P(z|X)]$$

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X)\log Q_\theta(z|X)dz - \int Q_\theta(z|X)\log P(z|X)dz$$
$$= \mathbb{E}_{z\sim Q_\theta(z|X)}[\log Q_\theta(z|X) - \log P(z|X)]$$

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X)\log Q_\theta(z|X)dz - \int Q_\theta(z|X)\log P(z|X)dz$$
$$= \mathbb{E}_{z\sim Q_\theta(z|X)}[\log Q_\theta(z|X) - \log P(z|X)]$$

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$
- Substituting $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$, we get

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X)\log Q_\theta(z|X)dz - \int Q_\theta(z|X)\log P(z|X)dz$$
$$= \mathbb{E}_{z\sim Q_\theta(z|X)}[\log Q_\theta(z|X) - \log P(z|X)]$$

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$
- Substituting $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$, we get

$$D[Q_\theta(z|X)||P(z|X)] = \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(X|z) - \log P(z) + \log P(X)]$$

- Let us expand the KL divergence term

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

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$
- Substituting $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$, we get

$$\begin{aligned} D[Q_\theta(z|X)||P(z|X)] &= \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(X|z) - \log P(z) + \log P(X)] \\ &= \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(z)] - \mathbb{E}_Q[\log P(X|z)] + \log P(X) \end{aligned}$$

- Let us expand the KL divergence term

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

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$
- Substituting $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$, we get

$$
\begin{aligned}
D[Q_\theta(z|X)||P(z|X)] &= \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(X|z) - \log P(z) + \log P(X)] \\
&= \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(z)] - \mathbb{E}_Q[\log P(X|z)] + \log P(X) \\
&= D[Q_\theta(z|X)||p(z)] - \mathbb{E}_Q[\log P(X|z)] + \log P(X)
\end{aligned}
$$

- Let us expand the KL divergence term

$$D[Q_\theta(z|X)||P(z|X)] = \int Q_\theta(z|X)\log Q_\theta(z|X)dz - \int Q_\theta(z|X)\log P(z|X)dz$$
$$= \mathbb{E}_{z\sim Q_\theta(z|X)}[\log Q_\theta(z|X) - \log P(z|X)]$$

- For shorthand we will use $\mathbb{E}_Q = \mathbb{E}_{z\sim Q_\theta(z|X)}$
- Substituting $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$, we get

$$D[Q_\theta(z|X)||P(z|X)] = \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(X|z) - \log P(z) + \log P(X)]$$
$$= \mathbb{E}_Q[\log Q_\theta(z|X) - \log P(z)] - \mathbb{E}_Q[\log P(X|z)] + \log P(X)$$
$$= D[Q_\theta(z|X)||p(z)] - \mathbb{E}_Q[\log P(X|z)] + \log P(X)$$

$$\therefore \log p(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$
- Since KL divergence (the red term) is always $>= 0$ we can say that

$$\mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] <= \log P(X)$$

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$
- Since KL divergence (the red term) is always $>= 0$ we can say that

$$\mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] <= \log P(X)$$

- The quantity on the LHS is thus a lower bound for the quantity that we want to maximize and is knows as the Evidence lower bound (ELBO)

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$
- Since KL divergence (the red term) is always $>= 0$ we can say that

$$\mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] <= \log P(X)$$

- The quantity on the LHS is thus a lower bound for the quantity that we want to maximize and is knows as the Evidence lower bound (ELBO)
- Maximizing this lower bound is the same as maximizing $\log P(X)$ and hence our equivalent objective now becomes

$$maximize \;\; \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$
- Since KL divergence (the red term) is always $>= 0$ we can say that

$$\mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] <= \log P(X)$$

- The quantity on the LHS is thus a lower bound for the quantity that we want to maximize and is knows as the Evidence lower bound (ELBO)
- Maximizing this lower bound is the same as maximizing $\log P(X)$ and hence our equivalent objective now becomes

$$maximize \;\; \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- And, this method of learning parameters of probability distributions associated with graphical models using optimization (by maximizing ELBO) is called variational inference

- So, we have

$$\log P(X) = \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] + D[Q_\theta(z|X)||P(z|X)]$$

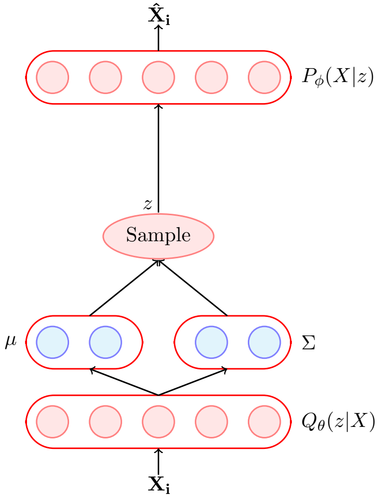
- Recall that we are interested in maximizing the log likelihood of the data *i.e.* $P(X)$
- Since KL divergence (the red term) is always $>= 0$ we can say that

$$\mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)] <= \log P(X)$$

- The quantity on the LHS is thus a lower bound for the quantity that we want to maximize and is knows as the Evidence lower bound (ELBO)
- Maximizing this lower bound is the same as maximizing $\log P(X)$ and hence our equivalent objective now becomes

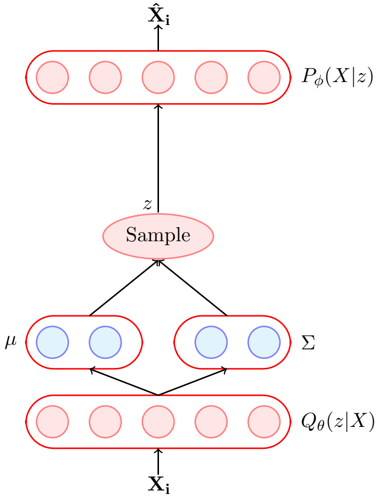
$$maximize \;\; \mathbb{E}_Q[\log P(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- And, this method of learning parameters of probability distributions associated with graphical models using optimization (by maximizing ELBO) is called variational inference
- Why is this any easier? It is easy because of certain assumptions that we make as discussed on the next slide



- First we will just reintroduce the parameters in the equation to make things explicit

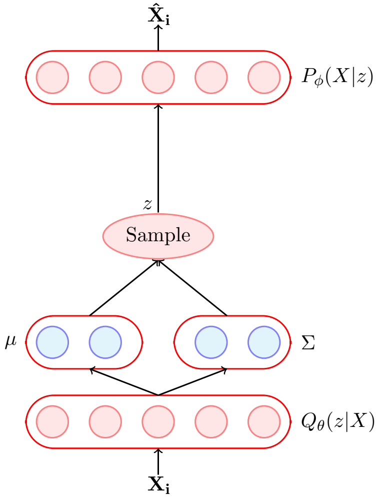
$$maximize \;\; \mathbb{E}_Q[\log P_\phi(X|z)] - D[Q_\theta(z|X)||P(z)]$$



- First we will just reintroduce the parameters in the equation to make things explicit

$$maximize \;\; \mathbb{E}_Q[\log P_\phi(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- At training time, we are interested in learning the parameters θ which maximize the above for every training example ($x_i \in \{x_i\}_{i=1}^N$)

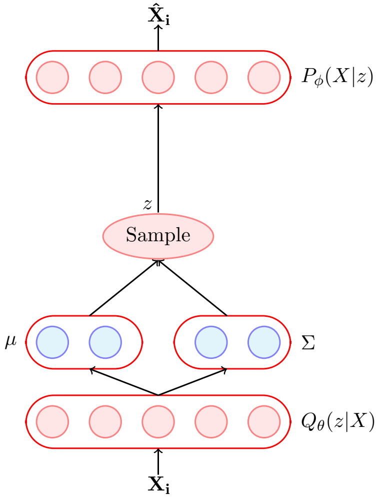


- First we will just reintroduce the parameters in the equation to make things explicit

$$maximize \;\; \mathbb{E}_Q[\log P_\phi(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- At training time, we are interested in learning the parameters θ which maximize the above for every training example $(x_i \in \{x_i\}_{i=1}^N)$
- So our total objective function is

$$\underset{\theta}{maximize} \sum_{i=1}^{N} \mathbb{E}_Q[\log P_\phi(X = x_i|z)] - D[Q_\theta(z|X = x_i)||P(z)]$$



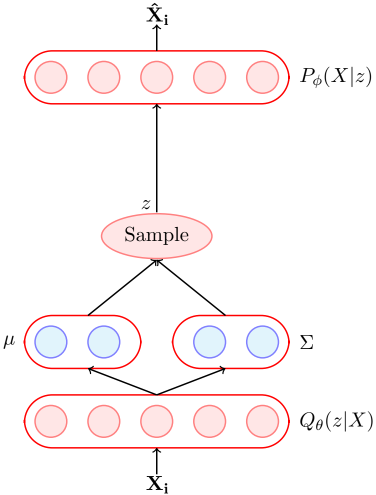
- First we will just reintroduce the parameters in the equation to make things explicit

$$maximize \ \ \mathbb{E}_Q[\log P_\phi(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- At training time, we are interested in learning the parameters θ which maximize the above for every training example ($x_i \in \{x_i\}_{i=1}^N$)
- So our total objective function is

$$\underset{\theta}{maximize} \sum_{i=1}^{N} \mathbb{E}_Q[\log P_\phi(X = x_i|z)] \\ - D[Q_\theta(z|X = x_i)||P(z)]$$

- We will shorthand $P(X = x_i)$ as $P(x_i)$



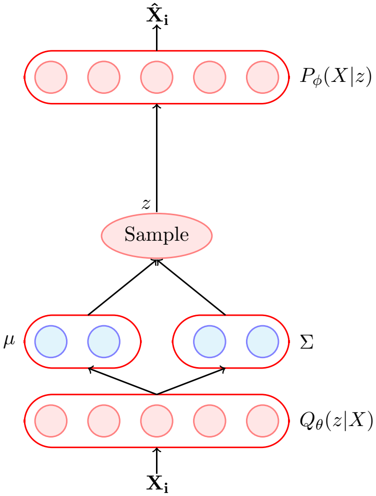
- First we will just reintroduce the parameters in the equation to make things explicit

$$maximize \;\; \mathbb{E}_Q[\log P_\phi(X|z)] - D[Q_\theta(z|X)||P(z)]$$

- At training time, we are interested in learning the parameters θ which maximize the above for every training example ($x_i \in \{x_i\}_{i=1}^N$)
- So our total objective function is

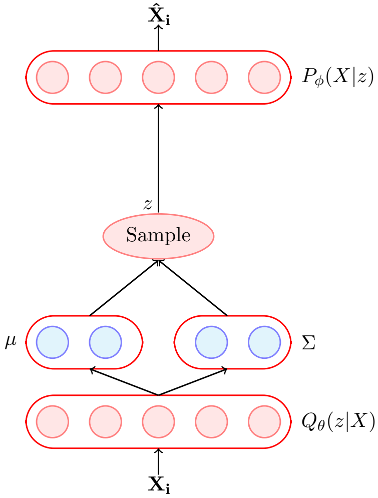
$$\underset{\theta}{maximize} \sum_{i=1}^{N} \mathbb{E}_Q[\log P_\phi(X = x_i|z)] \\ - D[Q_\theta(z|X = x_i)||P(z)]$$

- We will shorthand $P(X = x_i)$ as $P(x_i)$
- However, we will assume that we are using stochastic gradient descent so we need to deal with only one of the terms in the summation corresponding to the current training example



- So our objective function w.r.t. one example is

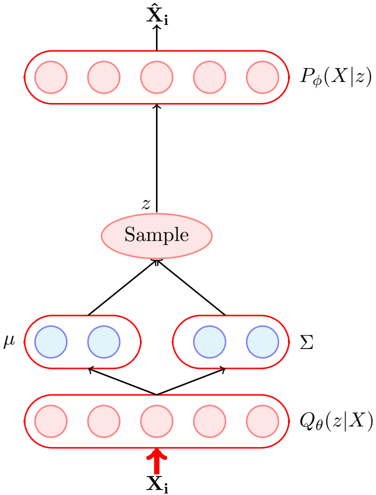
$$\underset{\theta}{maximize} \ \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$



- So our objective function w.r.t. one example is

$$\underset{\theta}{maximize} \ \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

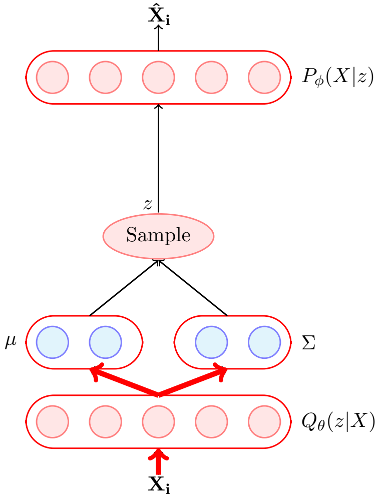
- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$



- So our objective function w.r.t. one example is

$$\underset{\theta}{maximize} \;\; \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

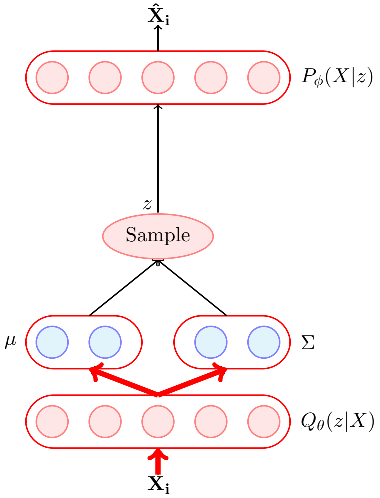
- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$



- So our objective function w.r.t. one example is

$$\underset{\theta}{maximize}\ \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

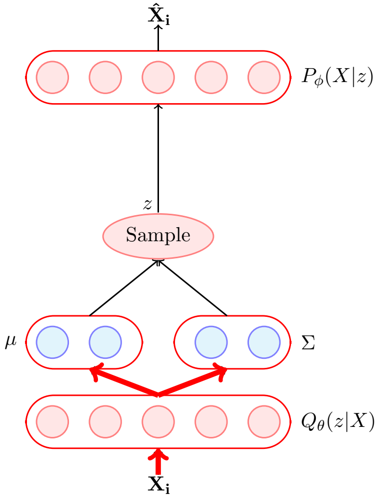
- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$



- So our objective function w.r.t. one example is

$$\underset{\theta}{maximize} \ \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$
- The second term in the above objective function is the difference between two normal distribution $\mathcal{N}(\mu(X), \Sigma(X))$ and $\mathcal{N}(0, I)$



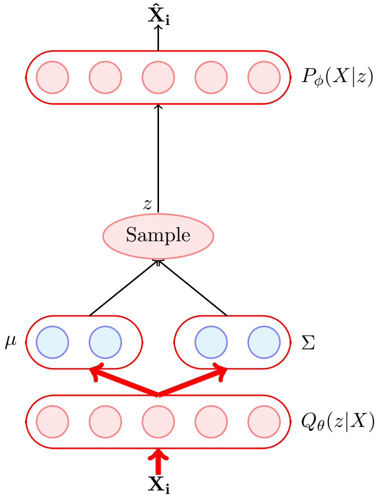
- So our objective function w.r.t. one example is

$$\underset{\theta}{maximize} \;\; \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$
- The second term in the above objective function is the difference between two normal distribution $\mathcal{N}(\mu(X), \Sigma(X))$ and $\mathcal{N}(0, I)$
- With some simple trickery you can show that this term reduces to the following expression (Seep proof here)

$$D[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)]$$
$$= \frac{1}{2}(tr(\Sigma(X)) + (\mu(X))^T[\mu(X)) - k - \log det(\Sigma(X))]$$

where k is the dimensionality of the latent variables



- So our objective function w.r.t. one example is

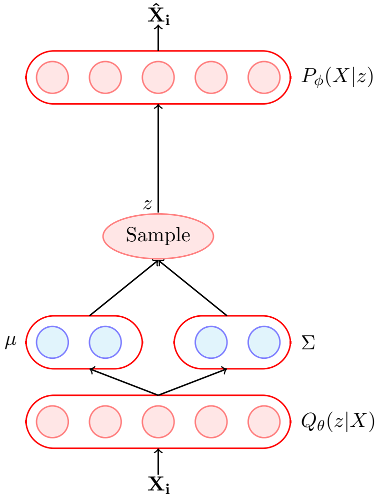
$$\underset{\theta}{maximize} \;\; \mathbb{E}_Q[\log P_\phi(x_i|z)] - D[Q_\theta(z|x_i)||P(z)]$$

- Now, first we will do a forward prop through the encoder using X_i and compute $\mu(X)$ and $\Sigma(X)$
- The second term in the above objective function is the difference between two normal distribution $\mathcal{N}(\mu(X), \Sigma(X))$ and $\mathcal{N}(0, I)$
- With some simple trickery you can show that this term reduces to the following expression (Seep proof here)

$$D[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)]$$
$$= \frac{1}{2}(tr(\Sigma(X)) + (\mu(X))^T[\mu(X)) - k - \log det(\Sigma(X))]$$

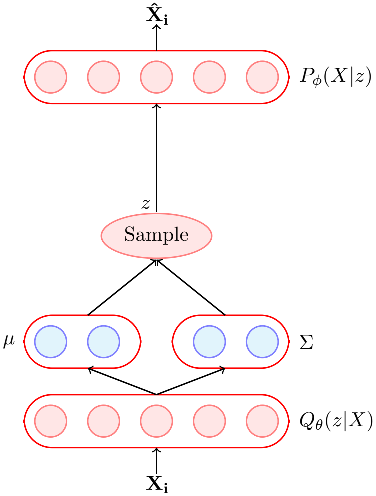
where k is the dimensionality of the latent variables

- This term can be computed easily because we have already computed $\mu(X)$ and $\Sigma(X)$ in the forward pass



- Now let us look at the other term in the objective function

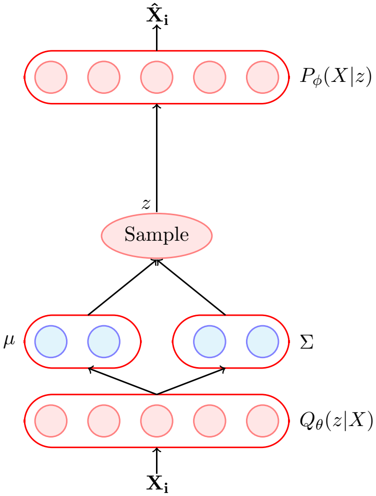
$$\sum_{i=1}^{n} \mathbb{E}_Q[\log P_\phi(X|z)]$$



- Now let us look at the other term in the objective function

$$\sum_{i=1}^{n} \mathbb{E}_Q[\log P_\phi(X|z)]$$

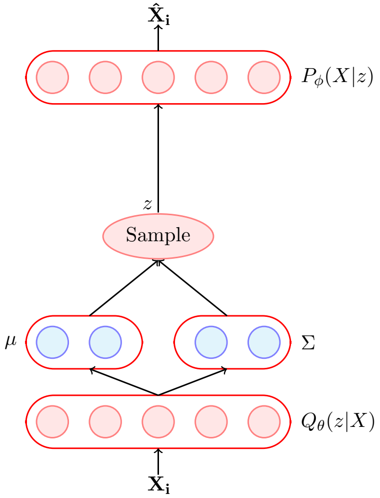
- This is again an expectation and hence intractable (integral over z)



- Now let us look at the other term in the objective function

$$\sum_{i=1}^{n} \mathbb{E}_Q[\log P_\phi(X|z)]$$

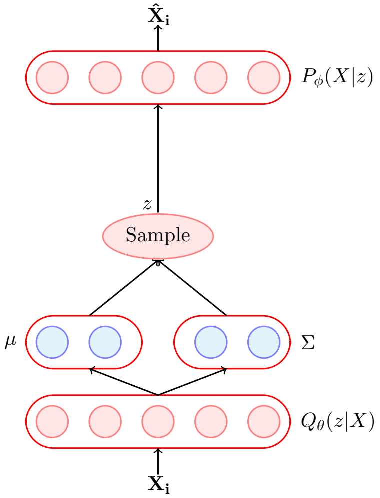
- This is again an expectation and hence intractable (integral over z)
- In VAEs, we approximate this with a single z sampled from $\mathcal{N}(\mu(X), \Sigma(X))$



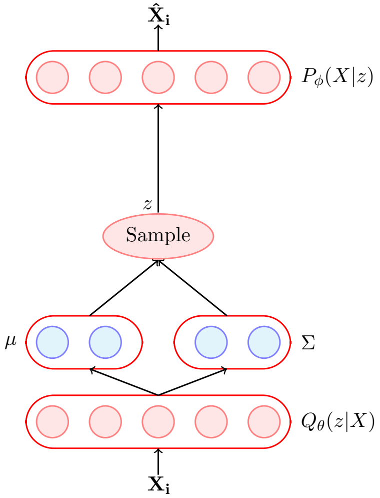
- Now let us look at the other term in the objective function

$$\sum_{i=1}^{n} \mathbb{E}_Q[\log P_\phi(X|z)]$$

- This is again an expectation and hence intractable (integral over z)
- In VAEs, we approximate this with a single z sampled from $\mathcal{N}(\mu(X), \Sigma(X))$
- Hence this term is also easy to compute (of course it is a nasty approximation but we will live with it!)

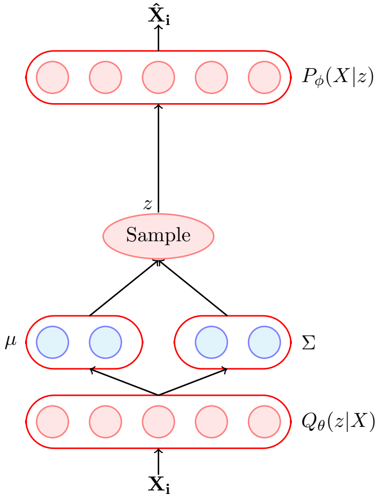


- Further, as usual, we need to assume some parametric form for $P(X|z)$



- Further, as usual, we need to assume some parametric form for $P(X|z)$
- For example, if we assume that $P(X|z)$ is a Gaussian with mean $\mu(z)$ and variance I then

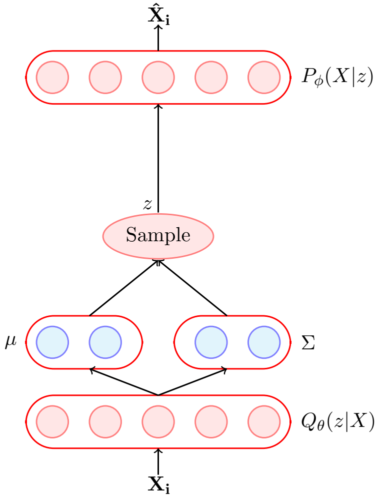
$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - \mu(z)||^2$$



- Further, as usual, we need to assume some parametric form for $P(X|z)$
- For example, if we assume that $P(X|z)$ is a Gaussian with mean $\mu(z)$ and variance I then

$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - \mu(z)||^2$$

- $\mu(z)$ in turn is a function of the parameters of the decoder and can be written as $f_\phi(z)$

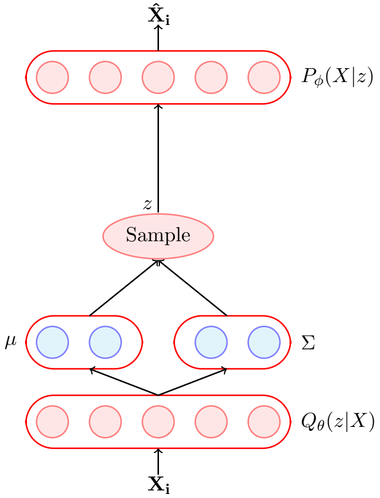


- Further, as usual, we need to assume some parametric form for $P(X|z)$
- For example, if we assume that $P(X|z)$ is a Gaussian with mean $\mu(z)$ and variance I then

$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - \mu(z)||^2$$

- $\mu(z)$ in turn is a function of the parameters of the decoder and can be written as $f_\phi(z)$

$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - f_\phi(z)||^2$$



- Further, as usual, we need to assume some parametric form for $P(X|z)$
- For example, if we assume that $P(X|z)$ is a Gaussian with mean $\mu(z)$ and variance I then

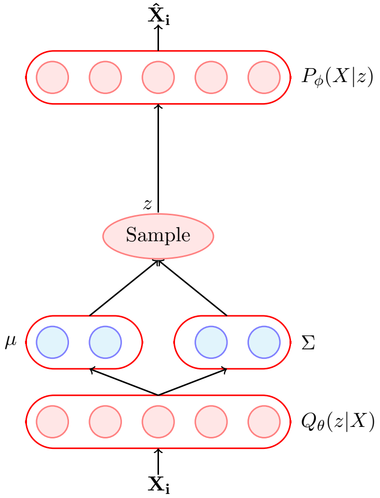
$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - \mu(z)||^2$$

- $\mu(z)$ in turn is a function of the parameters of the decoder and can be written as $f_\phi(z)$

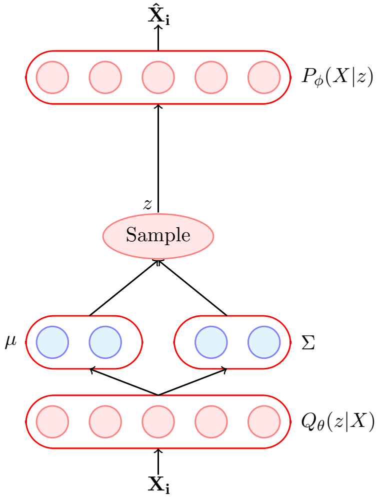
$$\log P(X = X_i|z) = C - \frac{1}{2}||X_i - f_\phi(z)||^2$$

- Our effective objective function thus becomes

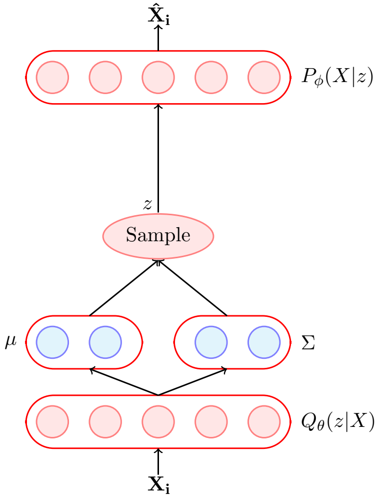
$$\underset{\theta,\phi}{minimize} \sum_{n=1}^{N} \Big[\frac{1}{2}(tr(\Sigma(X_i)) + (\mu(X_i))^T[\mu(X_i)) - k - \log det(\Sigma(X_i))] + ||X_i - f_\phi(z)||^2\Big]$$



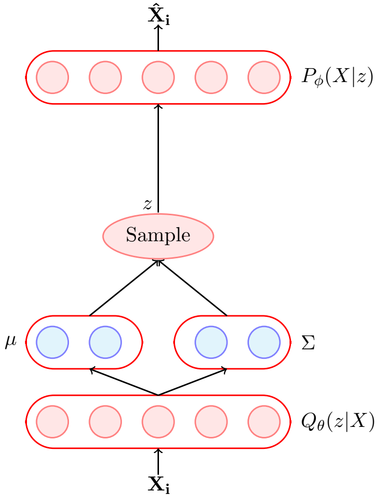
- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation



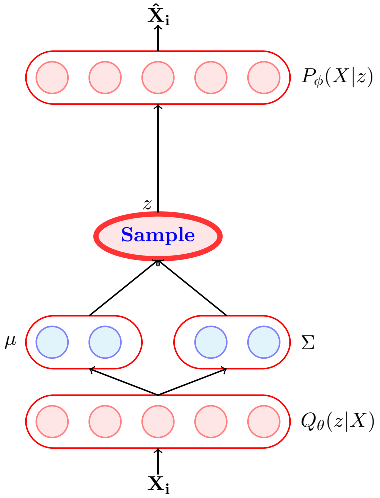
- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation
- However, there is a catch !



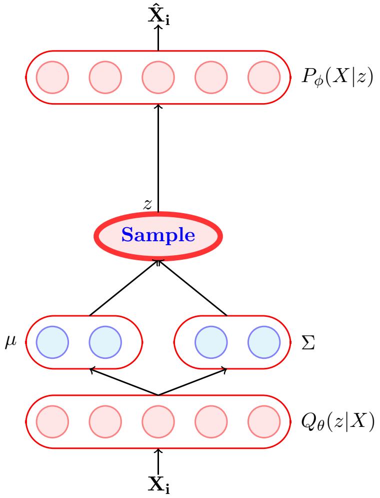
- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation
- However, there is a catch !
- The network is not end to end differentiable because the output $f_\phi(z)$ is not an end to end differentiable function of the input X



- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation
- However, there is a catch !
- The network is not end to end differentiable because the output $f_\phi(z)$ is not an end to end differentiable function of the input X
- Why?

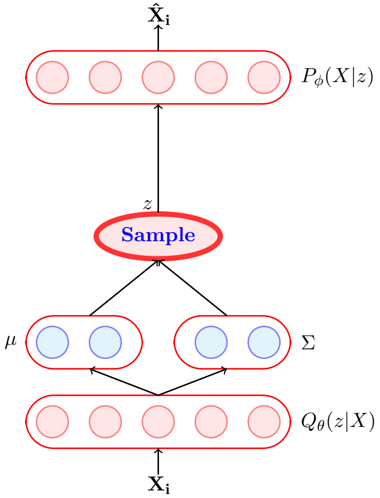


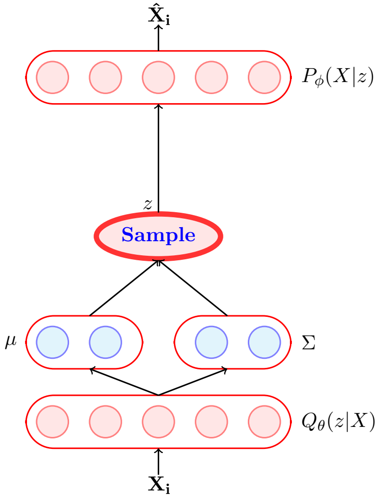
- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation
- However, there is a catch !
- The network is not end to end differentiable because the output $f_\phi(z)$ is not an end to end differentiable function of the input X
- Why? because after passing X through the network we simply compute $\mu(X)$ and $\Sigma(X)$ and then sample a z to be fed to the decoder



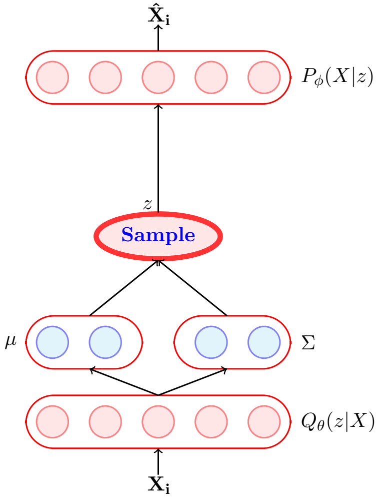
- The above loss can be easily computed and we can update the parameters θ of the encoder and ϕ of decoder using backpropagation
- However, there is a catch !
- The network is not end to end differentiable because the output $f_\phi(z)$ is not an end to end differentiable function of the input X
- Why? because after passing X through the network we simply compute $\mu(X)$ and $\Sigma(X)$ and then sample a z to be fed to the decoder
- This makes the entire process non-deterministic and hence $f_\phi(z)$ is not a continuous function of the input X

- VAEs use a neat trick to get around this problem



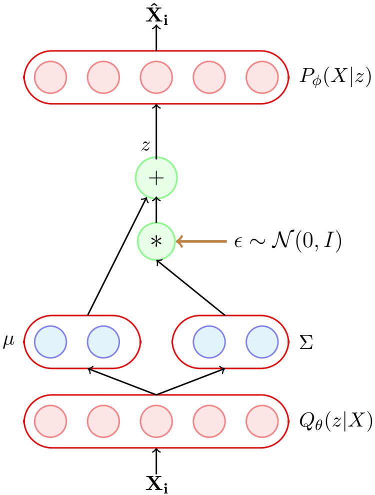


- VAEs use a neat trick to get around this problem
- This is known as the reparameterization trick wherein we move the process of sampling to an input layer



- VAEs use a neat trick to get around this problem
- This is known as the reparameterization trick wherein we move the process of sampling to an input layer
- For 1 dimensional case, given μ and σ we can sample from $\mathcal{N}(\mu, \sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0, 1)$, and then computing

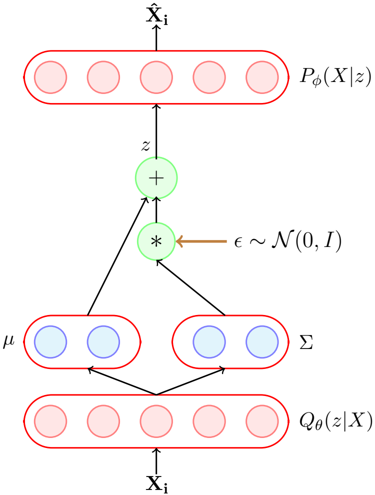
$$z = \mu + \sigma * \epsilon$$



- VAEs use a neat trick to get around this problem
- This is known as the reparameterization trick wherein we move the process of sampling to an input layer
- For 1 dimensional case, given μ and σ we can sample from $\mathcal{N}(\mu, \sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0, 1)$, and then computing

$$z = \mu + \sigma * \epsilon$$

- The adjacent figure shows the difference between the original network and the reparamterized network



- VAEs use a neat trick to get around this problem
- This is known as the reparameterization trick wherein we move the process of sampling to an input layer
- For 1 dimensional case, given μ and σ we can sample from $\mathcal{N}(\mu, \sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0, 1)$, and then computing

$$z = \mu + \sigma * \epsilon$$

- The adjacent figure shows the difference between the original network and the reparamterized network
- The randomness in $f_\phi(z)$ is now associated with ϵ and not X or the parameters of the model

- With that we are done with the process of training VAEs

- **Data:** $\{X_i\}_{i=1}^N$
- **Model:** $\hat{X} = f_\phi(\mu(X) + \Sigma(X) * \epsilon)$
- **Parameters:** θ, ϕ
- **Algorithm:** Gradient descent
- **Objective:**

$$\sum_{n=1}^{N} \left[\frac{1}{2}(tr(\Sigma(X_i)) + (\mu(X_i))^T [\mu(X_i)) - k - \log det(\Sigma(X_i))] + ||X_i - f_\phi(z)||^2\right]$$

- With that we are done with the process of training VAEs
- Specifically, we have described the data, model, parameters, objective function and learning algorithm

- **Data:** $\{X_i\}_{i=1}^N$
- **Model:** $\hat{X} = f_\phi(\mu(X) + \Sigma(X) * \epsilon)$
- **Parameters:** θ, ϕ
- **Algorithm:** Gradient descent
- **Objective:**

$$\sum_{n=1}^{N} \Big[\frac{1}{2}(tr(\Sigma(X_i)) + (\mu(X_i))^T [\mu(X_i))$$
$$- k - \log det(\Sigma(X_i))] + ||X_i - f_\phi(z)||^2 \Big]$$

- With that we are done with the process of training VAEs
- Specifically, we have described the data, model, parameters, objective function and learning algorithm
- Now what happens at test time? We need to consider both *abstraction* and *generation*

- **Data:** $\{X_i\}_{i=1}^N$
- **Model:** $\hat{X} = f_\phi(\mu(X) + \Sigma(X) * \epsilon)$
- **Parameters:** θ, ϕ
- **Algorithm:** Gradient descent
- **Objective:**

$$\sum_{n=1}^{N} \Big[\frac{1}{2}(tr(\Sigma(X_i)) + (\mu(X_i))^T [\mu(X_i))$$
$$- k - \log det(\Sigma(X_i))] + ||X_i - f_\phi(z)||^2 \Big]$$

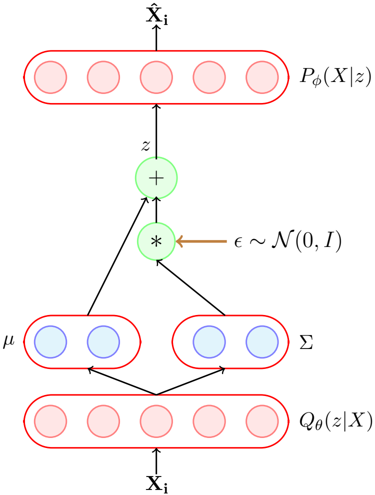
- With that we are done with the process of training VAEs
- Specifically, we have described the data, model, parameters, objective function and learning algorithm
- Now what happens at test time? We need to consider both *abstraction* and *generation*
- In other words we are interested in computing a z given a X as well as in generating a X given a z

- **Data:** $\{X_i\}_{i=1}^{N}$
- **Model:** $\hat{X} = f_\phi(\mu(X) + \Sigma(X) * \epsilon)$
- **Parameters:** θ, ϕ
- **Algorithm:** Gradient descent
- **Objective:**

$$\sum_{n=1}^{N} \Big[\frac{1}{2}(tr(\Sigma(X_i)) + (\mu(X_i))^T [\mu(X_i)) - k - \log det(\Sigma(X_i))] + ||X_i - f_\phi(z)||^2 \Big]$$

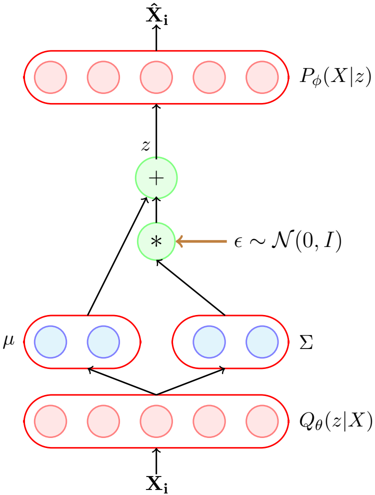
- With that we are done with the process of training VAEs
- Specifically, we have described the data, model, parameters, objective function and learning algorithm
- Now what happens at test time? We need to consider both *abstraction* and *generation*
- In other words we are interested in computing a z given a X as well as in generating a X given a z
- Let us look at each of these goals

Abstraction



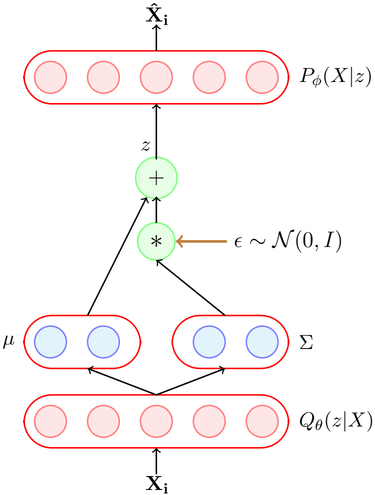
- After the model parameters are learned we feed a X to the encoder

Abstraction

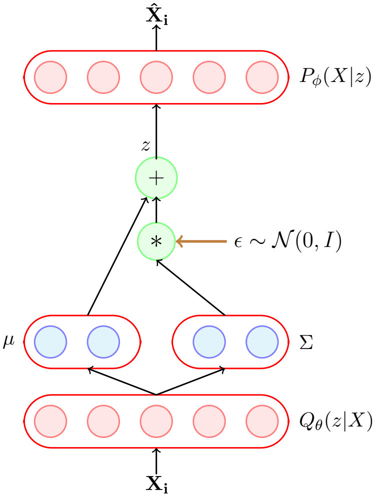


- After the model parameters are learned we feed a X to the encoder
- By doing a forward pass using the learned parameters of the model we compute $\mu(X)$ and $\Sigma(X)$

Abstraction



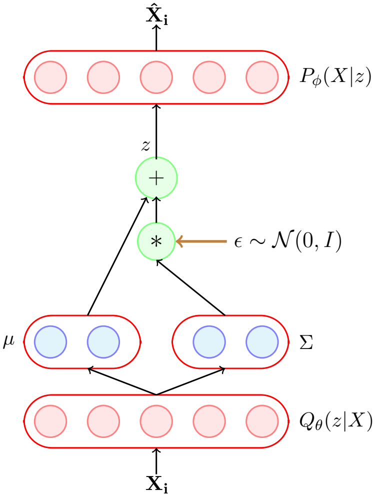
- After the model parameters are learned we feed a X to the encoder
- By doing a forward pass using the learned parameters of the model we compute $\mu(X)$ and $\Sigma(X)$
- We then sample a z from the distribution $\mu(X)$ and $\Sigma(X)$ or using the same reparameterization trick



Abstraction

- After the model parameters are learned we feed a X to the encoder
- By doing a forward pass using the learned parameters of the model we compute $\mu(X)$ and $\Sigma(X)$
- We then sample a z from the distribution $\mu(X)$ and $\Sigma(X)$ or using the same reparameterization trick
- In other words, once we have obtained $\mu(X)$ and $\Sigma(X)$, we first sample $\epsilon \sim \mathcal{N}(\mu(X), \Sigma(X))$ and then compute z

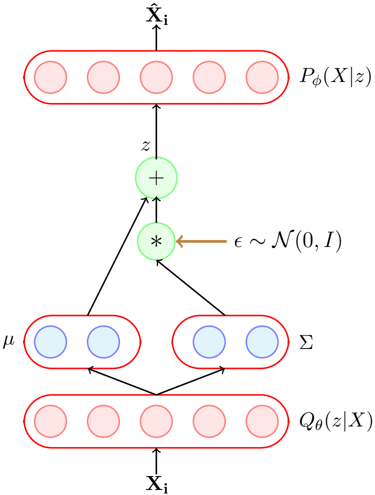
$$z = \mu + \sigma * \epsilon$$



Generation

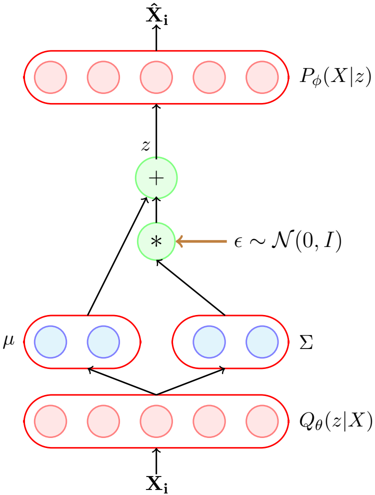
- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder

Generation



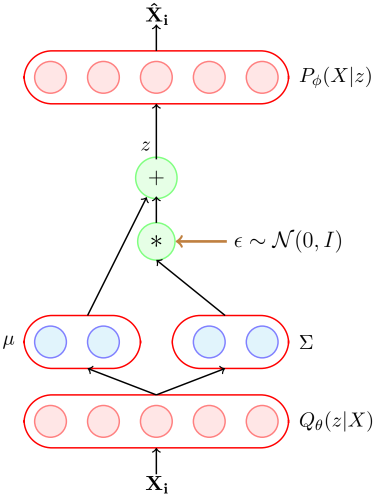
- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$

Generation



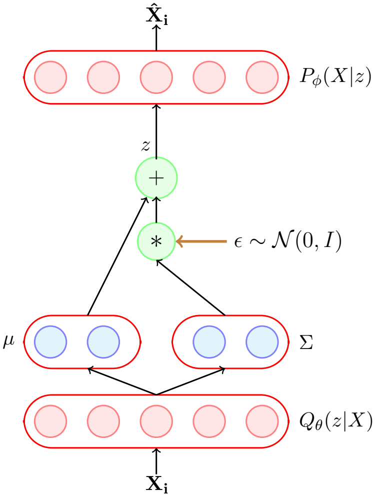
- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$
- Why would this work ?

Generation



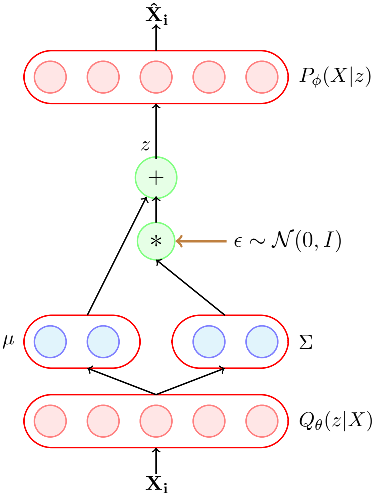
- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$
- Why would this work ?
- Well, we had trained the model to minimize $D(Q_\theta(z|X)||p(z))$ where $p(z)$ was $\mathcal{N}(0, I)$

Generation



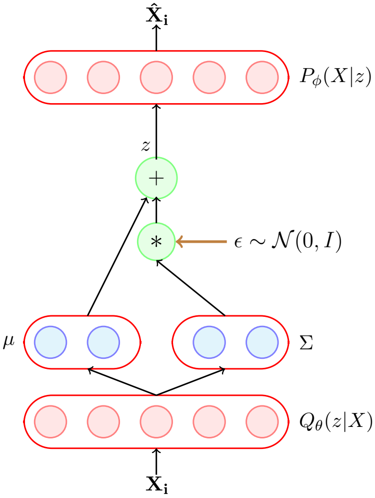
- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$
- Why would this work ?
- Well, we had trained the model to minimize $D(Q_\theta(z|X)||p(z))$ where $p(z)$ was $\mathcal{N}(0, I)$
- If the model is trained well then $Q_\theta(z|X)$ should also become $\mathcal{N}(0, I)$

Generation



- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$
- Why would this work ?
- Well, we had trained the model to minimize $D(Q_\theta(z|X)||p(z))$ where $p(z)$ was $\mathcal{N}(0, I)$
- If the model is trained well then $Q_\theta(z|X)$ should also become $\mathcal{N}(0, I)$
- Hence, if we feed $z \sim \mathcal{N}(0, I)$, it is almost as if we are feeding a $z \sim Q_\theta(z|X)$ and the decoder was indeed trained to produce a good $f_\phi(z)$ from such a z

Generation



- After the model parameters are learned we remove the encoder and feed a $z \sim \mathcal{N}(0, I)$ to the decoder
- The decoder will then predict $f_\phi(z)$ and we can draw an $X \sim \mathcal{N}(f_\phi(z), I)$
- Why would this work ?
- Well, we had trained the model to minimize $D(Q_\theta(z|X)||p(z))$ where $p(z)$ was $\mathcal{N}(0, I)$
- If the model is trained well then $Q_\theta(z|X)$ should also become $\mathcal{N}(0, I)$
- Hence, if we feed $z \sim \mathcal{N}(0, I)$, it is almost as if we are feeding a $z \sim Q_\theta(z|X)$ and the decoder was indeed trained to produce a good $f_\phi(z)$ from such a z
- Hence this will work !