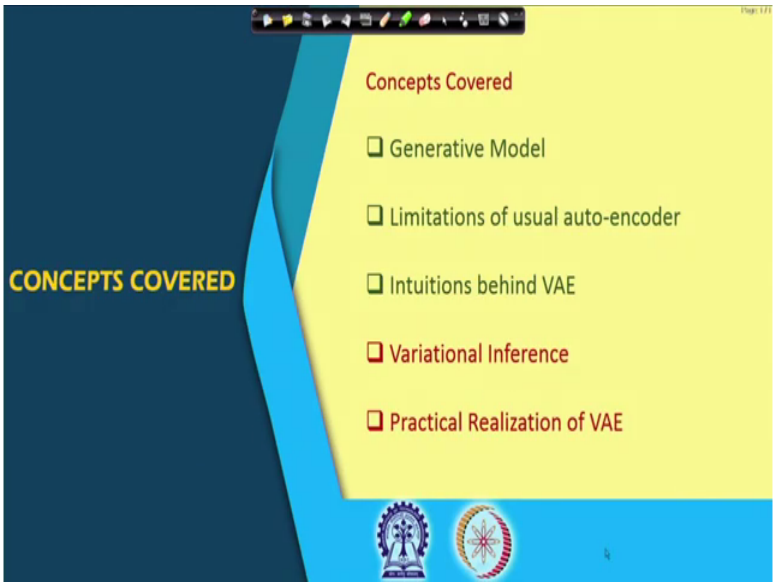
Deep Learning

Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 59
Variational Autoencoder – III

Hello, welcome back to the NPTEL online certification course on Deep Learning. So, for last few classes, we are talking about the generative model and particular model that we are discussing is a Variational Autoencoder.

(Refer Slide Time: 00:41)



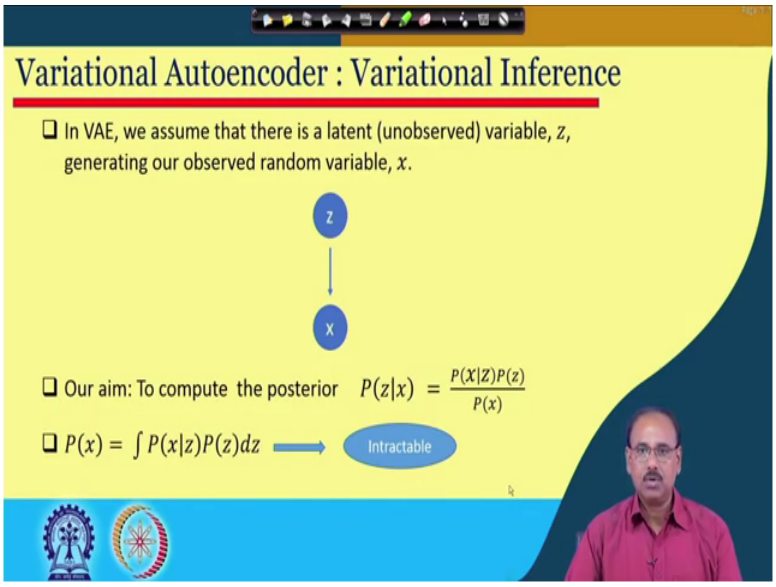
CONCEPTS COVERED

Concepts Covered

- ❑ Generative Model
- ❑ Limitations of usual auto-encoder
- ❑ Intuitions behind VAE
- ❑ Variational Inference
- ❑ Practical Realization of VAE

So, we have started our discussion on the variational inference which is one of the theoretical modeling technique of the variational autoencoder.

(Refer Slide Time: 00:57)



Variational Autoencoder : Variational Inference

- In VAE, we assume that there is a latent (unobserved) variable, z, generating our observed random variable, x.

$$\begin{tikzcd} z \arrow[d] \\ x \end{tikzcd}$$

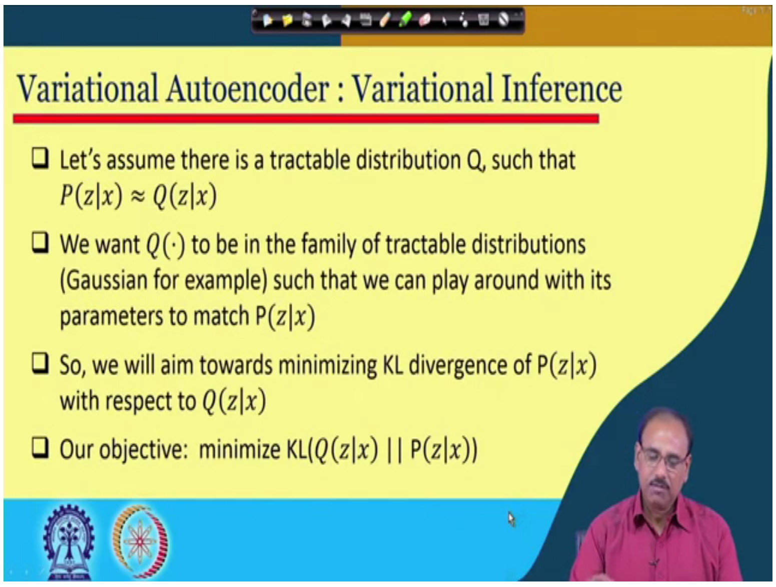
- Our aim: To compute the posterior $P(z|x) = \frac{P(X|Z)P(z)}{P(x)}$
- $P(x) = \int P(x|z)P(z)dz \longrightarrow$ Intractable

So, what we have discussed in our previous classes that z is a latent vector and when this latent vector z is fed as input to your decoder or the generator, the generator is expected to reconstruct your signal or the data x. Now, for x's of different categories z's will have different distributions. So, what I need to compute is a posterior probability that is P offset given x which is nothing, but P of x given Z into P of z upon P of x, where P of x is integral of P of x given z, P z d z.

Now, here we have seen that the first problem that we faced is that this integral is intractable. So, because this integral is intractable we cannot easily compute P of z given x. So, P of z given x also becomes intractable. And the reason we said that why this is intractable is z is a multi-dimensional vector. The dimensionality of vector z can be 10, it can be 15, it can be 100, it can be 500 and so on.

And because it is a multi-dimensional vector when I take this integral P x given z, P z d z, this integral also have to be taken, integration has to be performed over multiple dimensions which is not an easy task. So, in order to solve this problem we have mentioned that there are two different ways; one is you go for Monte Carlo solution simulation and the other approach is of variational inference approach. So, the technique that we are discussing is what is known as variational inference approach.

(Refer Slide Time: 02:47)



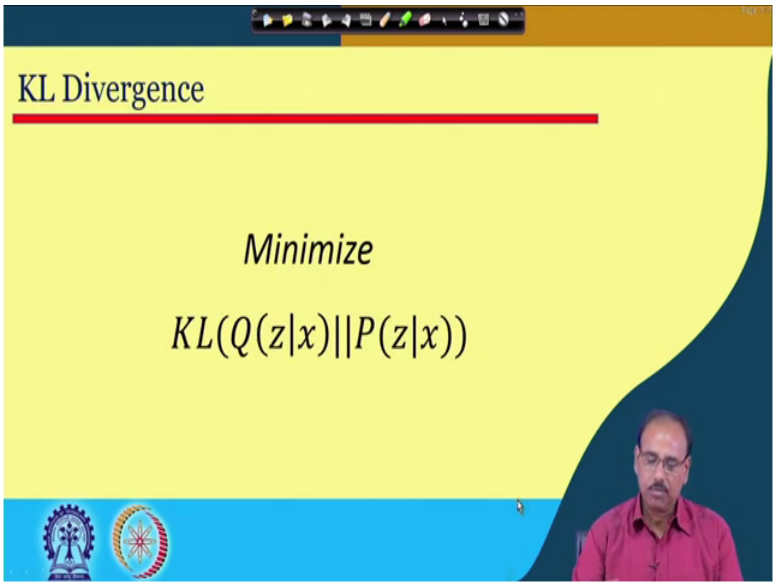
Variational Autoencoder : Variational Inference

- Let's assume there is a tractable distribution Q, such that $P(z|x) \approx Q(z|x)$
- We want $Q(\cdot)$ to be in the family of tractable distributions (Gaussian for example) such that we can play around with its parameters to match $P(z|x)$
- So, we will aim towards minimizing KL divergence of $P(z|x)$ with respect to $Q(z|x)$
- Our objective: minimize $KL(Q(z|x) \,||\, P(z|x))$

So, in this variational inferencing technique what we have assumed that even though P of z given x is intractable, but we assume there is a tractable distribution Q, this Q may be Gaussian for example, and we want P of z given x to be similar to Q of z given x. So, if I want to do that then basically what you have to do is you have to play with the parameters of Q which are nothing, but mean and standard deviation if it is a Gaussian distribution; so, we have to play with these parameters of this distribution Q, so that Q matches closely with the probability P of z given x.

Or effectively what we want to do is we want to find minimize the KL divergence between P z given x and Q z given x. So, the KL divergence of P z given x with respect to Q z given x. So, our objective is to minimize KL divergence of P z given x and Q z given x. So, this is the objective with which we started.

(Refer Slide Time: 03:55)



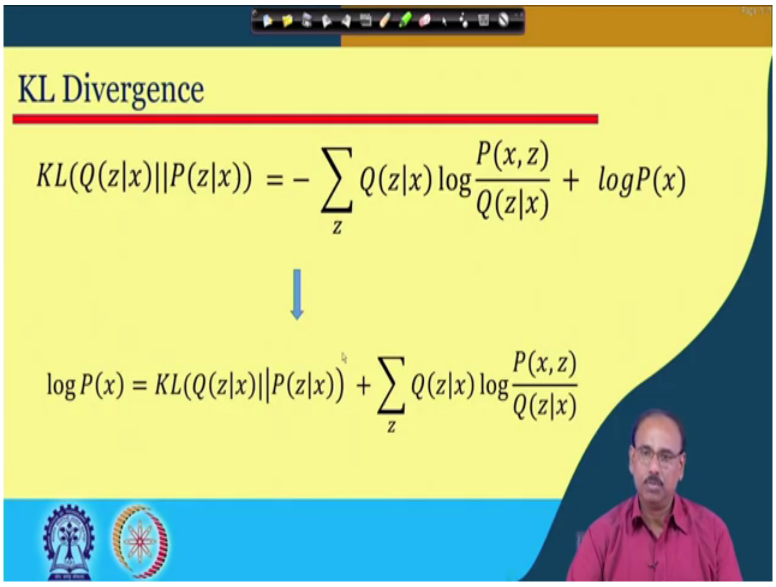
KL Divergence

Minimize

$$KL(Q(z|x)||P(z|x))$$

So, this is a minimization problem minimization of the KL divergence of Q z given x and P z given x.

(Refer Slide Time: 04:06)



KL Divergence

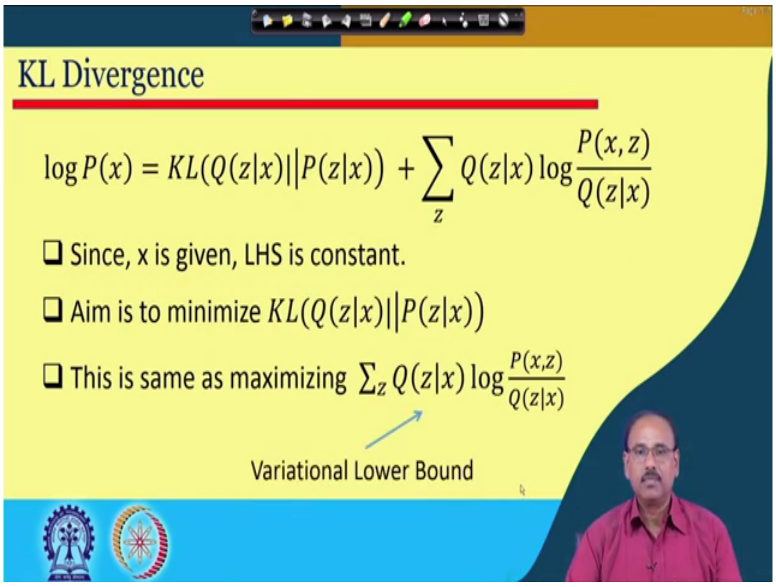
$$KL(Q(z|x)||P(z|x)) = -\sum_{z} Q(z|x)\log\frac{P(x,z)}{Q(z|x)} + logP(x)$$

$$\downarrow$$

$$\log P(x) = KL(Q(z|x)||P(z|x)) + \sum_{z} Q(z|x)\log\frac{P(x,z)}{Q(z|x)}$$

Then, through a series of derivations in the previous lecture we have seen that the KL divergence of Q z given x P z given x is equal to the sum of minus Q z given x log of the joint probability P x given P x z often Q z given x plus log of P x. I can rewrite this expression as log of P x is equal to KL divergence Q P plus sum of Q z given x, log of P x z upon Q z given x, where the summation is to convert z. And then we have seen, so this is what we have derived after a series of derivations starting from minimization of KL divergence Q P.

(Refer Slide Time: 05:02)



KL Divergence

$$\log P(x) = KL(Q(z|x)||P(z|x)) + \sum_z Q(z|x) \log \frac{P(x,z)}{Q(z|x)}$$

- ❑ Since, x is given, LHS is constant.
- ❑ Aim is to minimize $KL(Q(z|x)||P(z|x))$
- ❑ This is same as maximizing $\sum_z Q(z|x) \log \frac{P(x,z)}{Q(z|x)}$

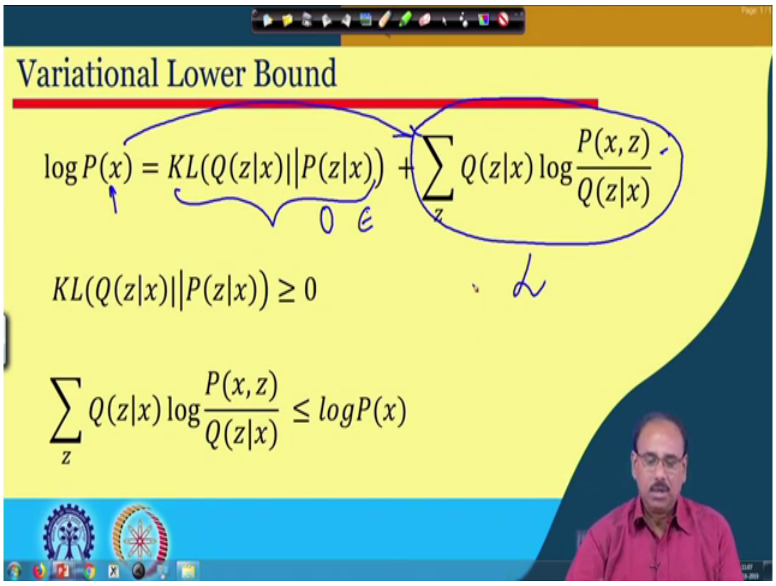
Variational Lower Bound

Now, here we have seen that because x is your training data and which is given. So, the left hand side which is log of P of x that becomes constant. And we started with our aim to minimize the KL divergence Q P between Q P. So, you find that this, right hand side because the left hand side is constant log of P of x is constant, so, right hand side the KL divergence of Q given P plus sum of Q log P upon Q, this is also constant.

So, as this, right hand side is constant and we want to minimize the KL divergence, so if I want to minimize the KL divergence that becomes equivalent to maximizing sum of Q z given x log of P x z upon Q z given x when you take the summation over z.

So, this minimization of KL divergence between Q and P is identical becomes same as maximizing this quantity sum of Q z given x log of P x z upon Q z given x and this is the term which is known as variation on lower bound. Now, why this variational Lower bound or what is this variational lower bound?

(Refer Slide Time: 06:22)



Variational Lower Bound

$$\log P(x) = \underbrace{KL(Q(z|x)||P(z|x))}_{0 \ \epsilon} + \sum_z Q(z|x) \log \frac{P(x,z)}{Q(z|x)}$$

$\mathcal{L}$

$$KL(Q(z|x)||P(z|x)) \geq 0$$

$$\sum_z Q(z|x) \log \frac{P(x,z)}{Q(z|x)} \leq \log P(x)$$

You find that as we have just derived that log of P of x is equal to KL divergence Q P plus norm of Q log of P upon Q summation over z and we know that KL divergence is always greater than or equal to 0. So, we have seen before that KL divergence is not symmetric.

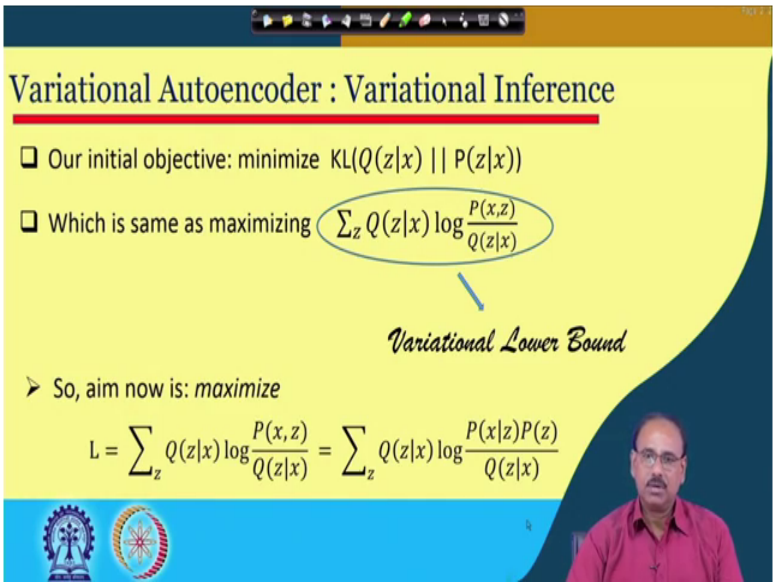
So, if I want to find out KL divergence Q P and KL divergence P Q they will not be same in general, but the KL divergence is always greater than or equal to 0 and it will be equal to 0 when Q equal to P; obviously, because log of P by Q is when P and Q are same, then log of P by Q is log of 1 which is equal to 0. So, when P and Q are same then KL divergence is 0, but if P and Q are different then KL divergence will be greater than 0. So, KL divergence is always greater than 0.

So, again analyzing this, right hand side you find that we have seen that our P x was intractable, and here you find in this expression, in this case when I consider this particular expression, this is constant, this is always greater than 0 and if this KL divergence is equal to 0, then log of P of x is same as this particular term. Let me call this term as L.

So, log of P of x will be equal to L which is sum of Q z given x into log of P x z upon Q x z given x. But if this is greater than 0 if this is positive say some value epsilon then your log of P of x will be epsilon plus L; that means, L will be P minus epsilon. So, that clearly says that this L which is sum of Q z given x into log of P x z upon Q z given x, this quantity is always less than or equal to log of P x, right.

So, we know that P of x is intactable as a result Q log of P of x is also intactable, but I know what is its lower bound. And what I want is I want to maximize this P of x, but which is intactable, so I can derive, I cannot directly do it. But I have a lower bound. So, because I have a lower bound if I maximize this lower bound and I know log of P of x is always greater than this lower bound greater than or equal to this lower bound. So, if I maximize the lower bound then effectively I am maximizing log of P of x or maximizing P of x. So, that is how this concept of lower bound comes.

(Refer Slide Time: 09:22)



Variational Autoencoder : Variational Inference

- Our initial objective: minimize KL($Q(z|x)$ || P($z|x$))
- Which is same as maximizing $\sum_z Q(z|x)\log\frac{P(x,z)}{Q(z|x)}$

Variational Lower Bound

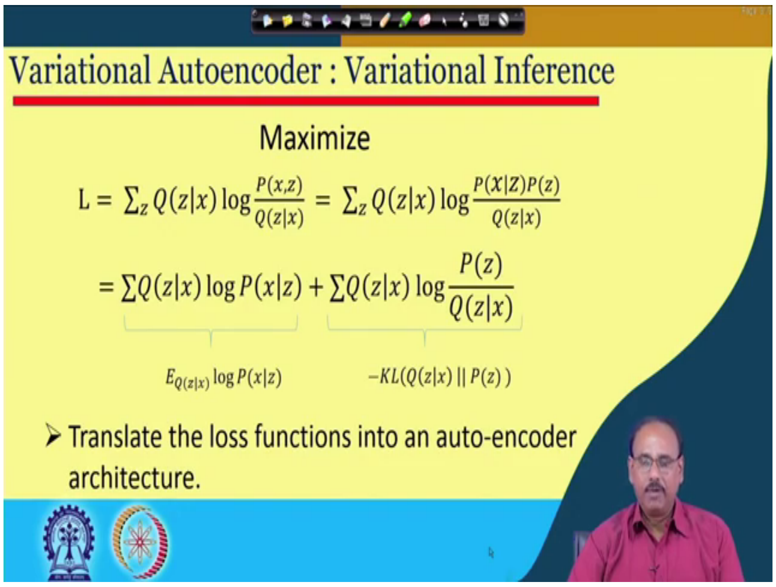
- So, aim now is: *maximize*

$$L = \sum_z Q(z|x)\log\frac{P(x,z)}{Q(z|x)} = \sum_z Q(z|x)\log\frac{P(x|z)P(z)}{Q(z|x)}$$

So, effectively the problem has boiled down that we started with or in to minimize the KL divergence P Q and which has boiled it down to maximization of the lower bound where lower bound is Q z given x, log of P x z upon Q given Q z given x. So, this minimization problem has been converted to the maximization of this variational lower bound term.

So, our aim is now that we want to maximize this variational lower bound which is sum of Q of z given x into log of P x z upon Q of z given x which will be same as log of Q z given x sorry Q z given x into log of P x z into P z upon Q z given x summation over z. They are equivalent because P x z the joint probability is nothing, but P x given z into P z, right. So, I can always write that variational lower bound L is equal to Q z given x, log of P x given z into P z upon Q z given x take the summation over z. And this is the term that we want to maximize. So, effectively we want to maximize this.

(Refer Slide Time: 10:45)



Variational Autoencoder : Variational Inference

Maximize

$$L = \sum_z Q(z|x) \log \frac{P(x,z)}{Q(z|x)} = \sum_z Q(z|x) \log \frac{P(x|Z)P(z)}{Q(z|x)}$$

$$= \underbrace{\sum Q(z|x) \log P(x|z)}_{E_{Q(z|x)} \log P(x|z)} + \underbrace{\sum Q(z|x) \log \frac{P(z)}{Q(z|x)}}_{-KL(Q(z|x) \,||\, P(z))}$$

- Translate the loss functions into an auto-encoder architecture.

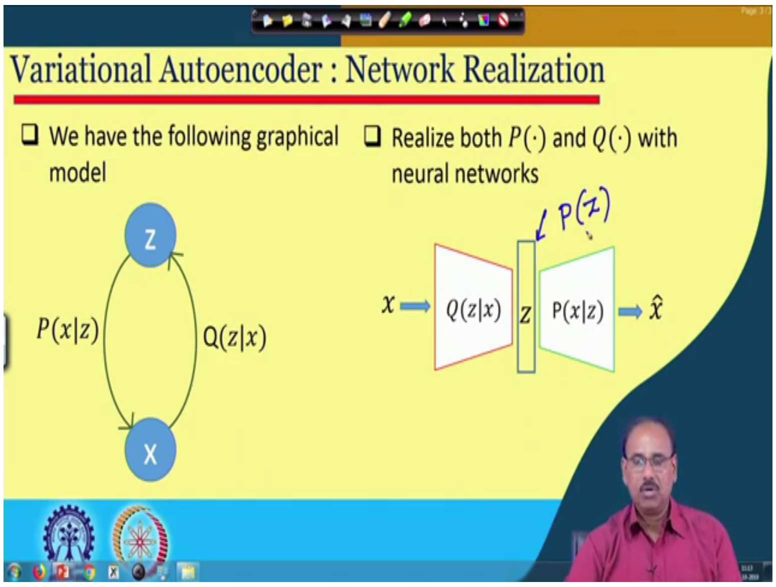
Again, let us see that this term that sum of Q z given x into log of P x given z into P z upon Q z given x this can be written as sum of Q z given x into log of P x given z plus summation of Q z given x into log of P z upon Q z given x. And if you look at the second term over here you find that second term is again another KL divergence, right.

So, what we have is if you further simplify this, you find that the first component of this expression is an expectation of log of P x given z, where the expectation is with respect to Q z given x and the, right hand side is negative of the KL divergence between P Q z given x upon P of z and this entire expression I want to maximize, right.

So, if I want to maximize this entire expression; that means, effectively what I have to do is I have to maximize P of x given z and I have to minimize the KL divergence Q of z given x P z, right. So, now, let us see that how this maximization problem can be translated to an autoencoder architecture.

So, here you find that I have a few terms, one is Q z given x which tells you that given your training data x the probability of your latent variable or the latent vector. Similarly, P of x given z tells you that given a latent vector z to the decoder or to the generator the probability of generating the data x, so that is P of x given z. And I have another term which is P of z that is the probability distribution of the latent variable or the latent vector z itself. So, if I translate this maximization term in the form of an autoencoder I will have something like this.

(Refer Slide Time: 13:17)



Variational Autoencoder : Network Realization

- We have the following graphical model
- Realize both $P(\cdot)$ and $Q(\cdot)$ with neural networks

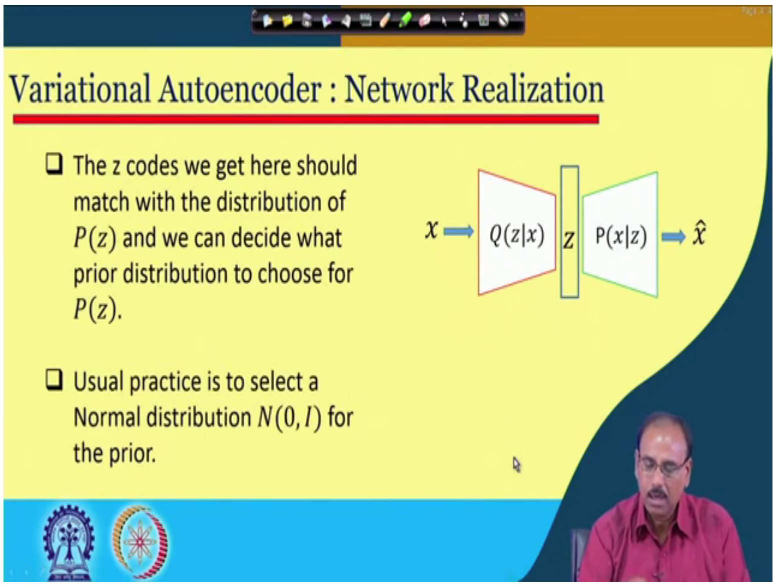
So, this is what was our graphical model that given z you can generate x following P of x given z that is the generator part and given x I can get the latent variable z following Q of z given x which is my encoder part. So, this is what is the graphical model and the corresponding network, neural network can be done like this that both P and Q can be implemented using neural networks.

And the neural network will be something like this, that Q z given x now is modeled by the encoder network and P of x given z is modeled by the decoder or the generator network and in between I have z which is the variable or the latent vector which is generated by Q z given x.

Now, if I simply go by this you find that it becomes an autoencoder model. But what makes it variational autoencoder not just autoencoder is that that along with z I now have our distribution which is P of z which was not there in traditional autoencoder, in variational autoencoder I have this P of z term.

So, the encoder in this case it does not give me a deterministic latent vector z, but the encoder gives me a distribution of the latent vector z and what we effectively do is during generation or during decoding that you sample a vector from this latent distribution, distribution of the latent variables and that sample is fed to the decoder or the generator and the generator generates the corresponding data which is x, right.

(Refer Slide Time: 15:21)



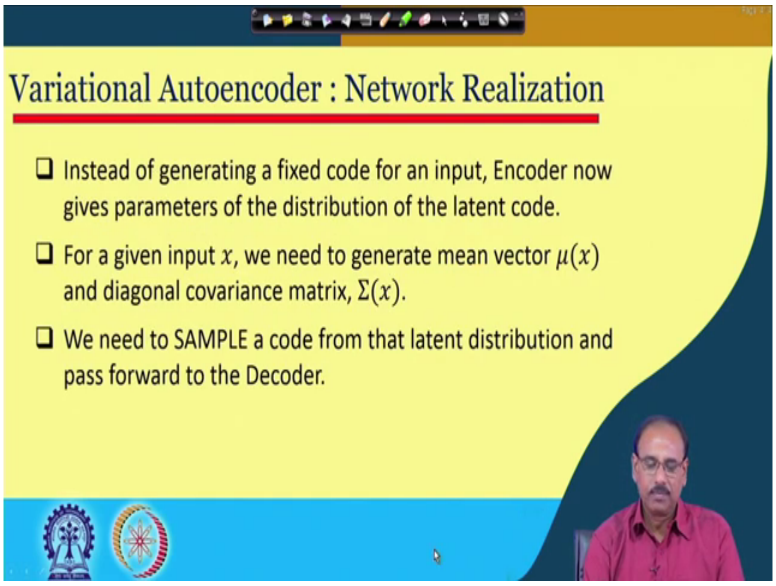
Variational Autoencoder : Network Realization

- The z codes we get here should match with the distribution of $P(z)$ and we can decide what prior distribution to choose for $P(z)$.
- Usual practice is to select a Normal distribution $N(0, I)$ for the prior.

So, this is what is our neural network for realization of that maximization problem. So, this z code that we get now as we have just said it has to match with the distribution P z. And what we will do is we will decide a prior for this P z that is prior distribution which will be chosen for this distribution P z. And usually this P z is chosen to be a normal distribution with zero mean and unit variance and we assume that the covariance matrix is a diagonal matrix; that means, it is assumed that the components of the latent vector they are independent of each other.

So, as a result the off diagonal elements in the covariance matrix will be equal to 0 and because every component will have unit variance. So, all the diagonal elements will be equal to 1 or in other words your covariance matrix is an unit matrix. So, this normal distribution that is assumed for P z is N of 0 I, and this is the prior that will impose on P z.

(Refer Slide Time: 16:32)

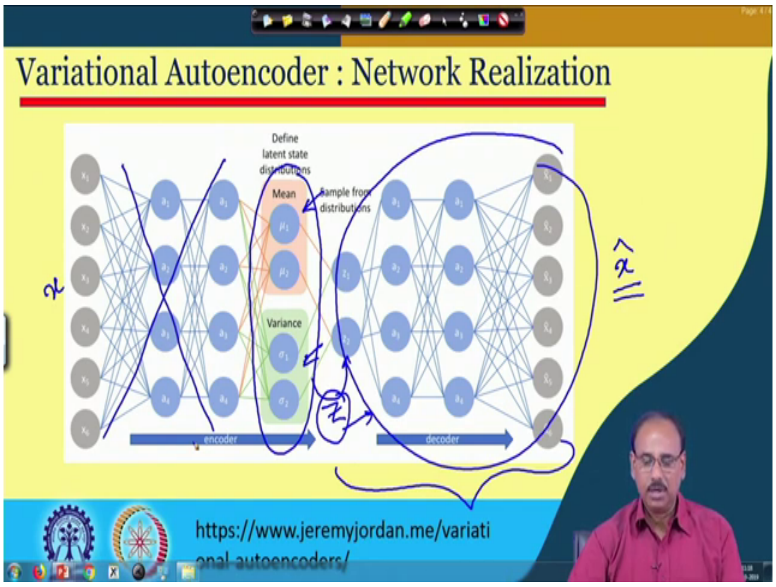


Variational Autoencoder : Network Realization

- Instead of generating a fixed code for an input, Encoder now gives parameters of the distribution of the latent code.
- For a given input x, we need to generate mean vector $\mu(x)$ and diagonal covariance matrix, $\Sigma(x)$.
- We need to SAMPLE a code from that latent distribution and pass forward to the Decoder.

So, what finally our variational autoencoder gives is that instead of generating a fixed code for an input to the decoder, the encoder now gives you a parameter of the distribution of the latent code. So, for a given input x the encoder generates a mean vector mu x and it generates a diagonal covariance matrix sigma x and what we need is we need to sample a vector z from this distribution and that vector z which is now my latent vector has to be passed to the decoder or the generator and the generator will generate the data accordingly.

(Refer Slide Time: 17:22)



Variational Autoencoder : Network Realization

https://www.jeremyjordan.me/variational-autoencoders/

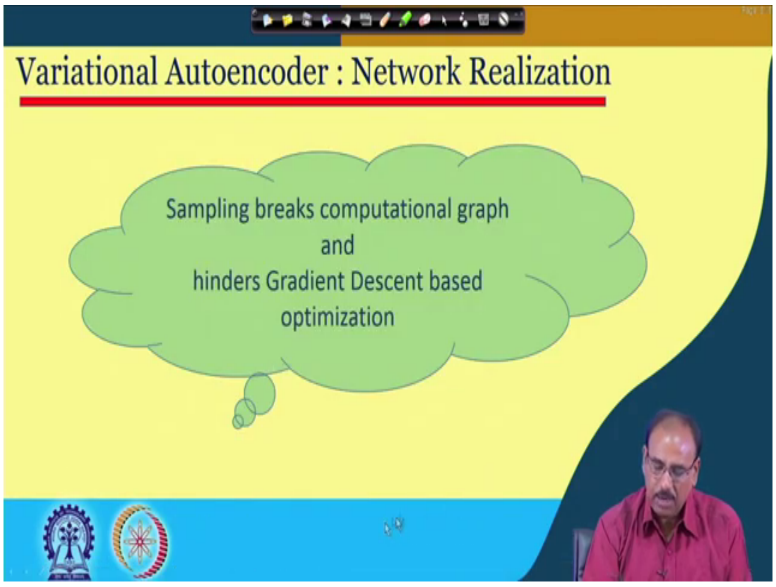
So, this is what is our variational autoencoder and in explicit form in the architecture of a neural network it looks like this. So, here you are feeding the data x to the input of the encoder; the encoder gives you the distribution of the latent variable z. So, it gives you the mean as well as the variance, then you sample a z from this distribution and feed this z to your generator or the decoder network and the decoder network gives you the reconstructed output x hat.

Now, find that when we talked about the traditional autoencoder we said that the decoder in a traditional autoencoder was not of interest. In traditional autoencoder our interest was the encoder part because we wanted to encode the input data or the input data has to be represented in a compressed form which is useful for classification or for understanding.

In case of variational autoencoder, our aim is exactly opposite. After your network is trained we are no more interested in this encoder part, because the encoder has already given you the distribution parameters mu and sigma. And once I have this I am only interested in the decoder or the generator part because what I want is that because this mean and variance are the parameters of the distribution of the latent variable is already known I want to sample our latent vector z from that distribution, that sample is fed to the decoder and the decoder has to give me the output x hat.

So, this encoder part is not of interest. We are not interested in the encoder part of the variational autoencoder or in a generative model. But as we have seen in case of traditional autoencoder where after training you simply discard the decoder. Similarly, in this case after the training is over or once you learn your distribution of the latent variables you can discard the encoder part because I do not need it anymore for the generation or the decoding purpose. So, that is what your autoencoder is.

(Refer Slide Time: 19:47)



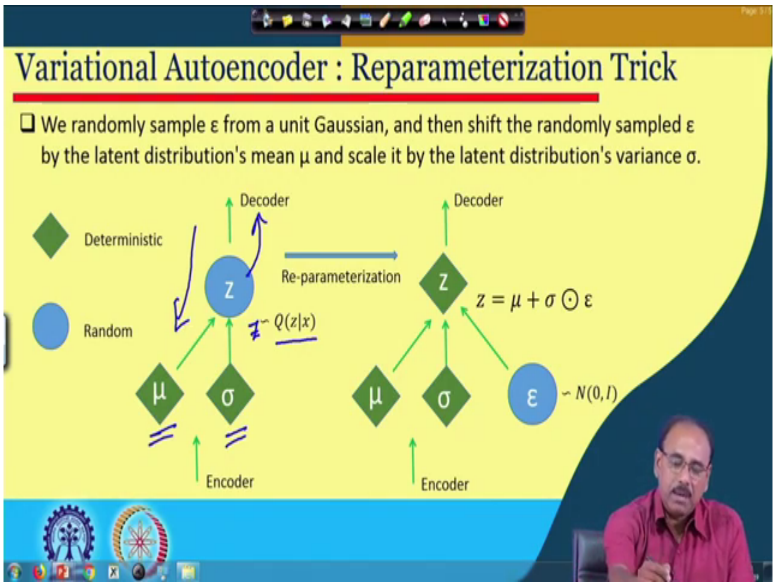
Variational Autoencoder : Network Realization

Sampling breaks computational graph
and
hinders Gradient Descent based
optimization

Now, again we have certain problems that you find that in this particular operation we had an operation of sampling involved, because the encoder has given of the distribution parameters and from that distribution have to sample a z to be fed to the decoder for generation of the data.

Now, this sampling breaks the computational graph and it hinders the gradient descent based optimization algorithm because to a sampling operator I cannot pass that gradient. So, that is the problem. So, how do you solve this problem?

(Refer Slide Time: 20:24)



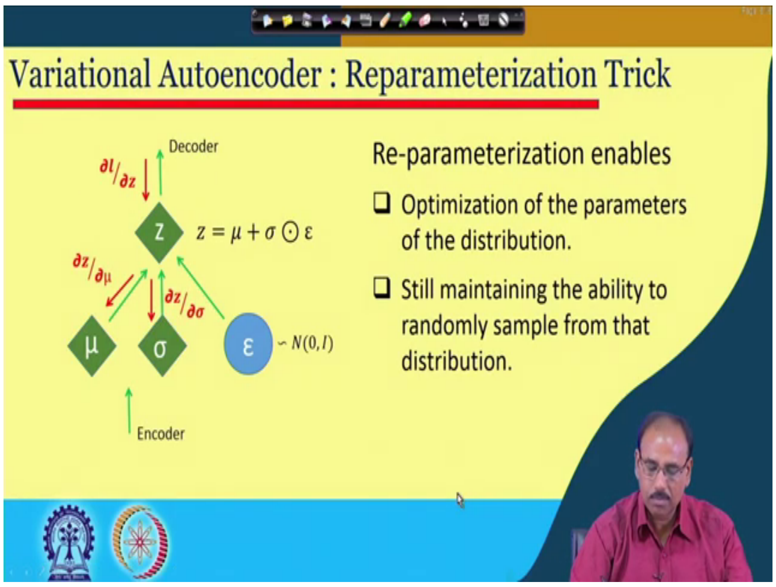
Variational Autoencoder : Reparameterization Trick

- We randomly sample ε from a unit Gaussian, and then shift the randomly sampled ε by the latent distribution's mean μ and scale it by the latent distribution's variance σ.

So, effectively what we have is if again if we put in the form of a graph, so this is what I have. This is the graph what the encoder has given me is the mu and sigma, those are the parameters of the distribution and then what you do is you sample a z from this distribution Q z given x, for which I am already know the parameters mu and sigma, and this sample z is fed to the decoder for decoding or generation of the data x.

And because I have a sampling process involved my back propagation learning cannot work anymore because through a sampling operator the sampling operator does not support back propagation. So, what you do is you use a trick which is known as reparameterization trick. So, in this reparameterization trick what you do is, you dissociate the sampling operator, the sampling operation from the network.

(Refer Slide Time: 21:25)



Variational Autoencoder : Reparameterization Trick

Re-parameterization enables

- ❑ Optimization of the parameters of the distribution.
- ❑ Still maintaining the ability to randomly sample from that distribution.

Decoder
$\partial l/\partial z$
$z = \mu + \sigma \odot \varepsilon$
$\partial z/\partial \mu$
$\partial z/\partial \sigma$
μ
σ
ε
$\sim N(0, I)$
Encoder

So, what you do is you sample from a normal distribution with zero mean and unit variance and after you get a sample you reparameterize it. So, suppose I sample a vector say epsilon a variable epsilon, then I reparameterize it in the form z is equal to mu plus sigma times epsilon, where mu and sigma are the parameters which are given by your encoder network. And once you do this you find as has been shown over here, this is my network part and this is the sampling operator.

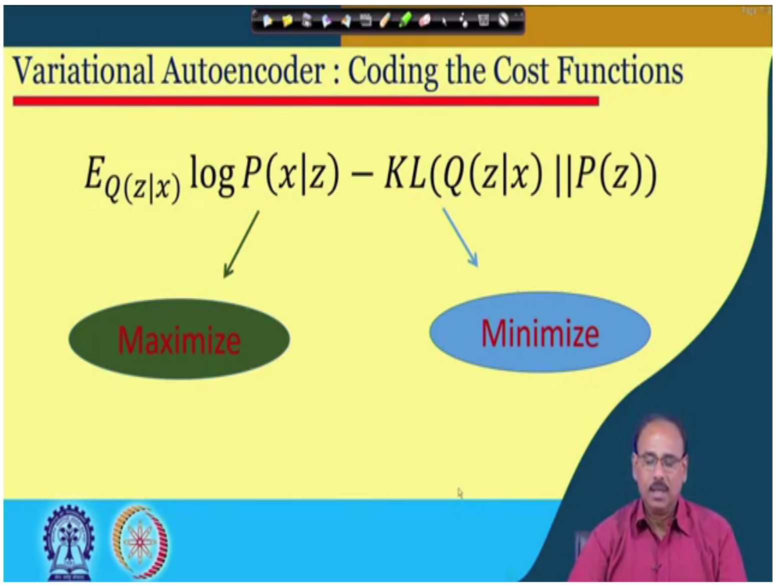
The sampling operator is now dissociated from the network. What we are doing is we are sampling epsilon from a normal distribution with zero mean unit variance and that sample epsilon is now converted to z, matched to z through this parameters mu and sigma, and once it is done now it is fed to your network. So, this reparameterized vector is now fed to the decoder of the generator and the generator gives the output, ok.

So, as the sampling process has been now kept out of the network. So, as a result I stop the problem or I avoid the problem which we faced earlier that the sampling process or the sampling operation does not support back propagation. So, now, with this I can easily go for back propagation as shown over here, that as I have the latent variable z I can take the derivative of my final output with respect to z or the error loss function with respect to z.

Of course, there is a chain rule involved in it because final loss function may not be directly visible to z, so as a result to compute del L del z, where L is your loss function I may have to apply chain rule and then I can compute del z del mu, I can also compute del z, del sigma as has been shown over here. So, now your back propagation can simply follow this path as well as this path, where the sampling process does not take place or the sampling process does not hinder the back propagation operation.

So, because of this reparameterization it enables us the optimization of the parameters of the distribution following back propagation, while it still maintains the ability to randomly sample data or sample a latent vector from that distribution because we have kept the sampling process out, right.

(Refer Slide Time: 24:58)



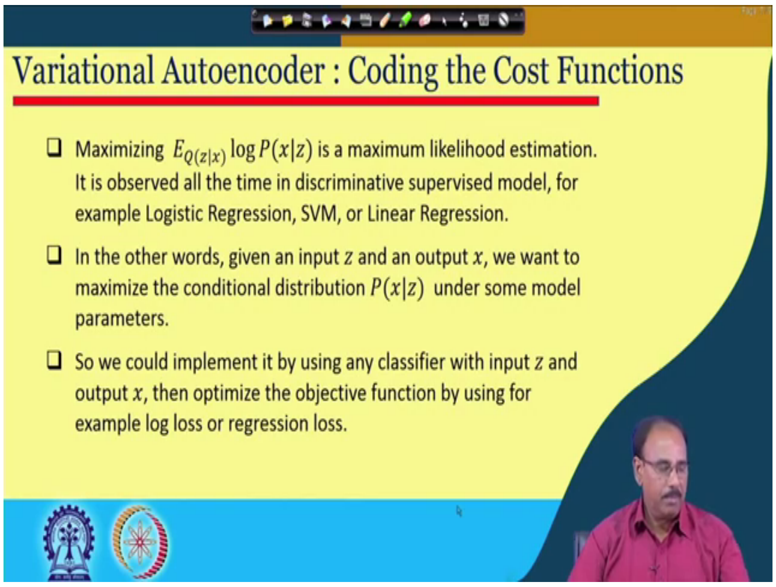
Variational Autoencoder : Coding the Cost Functions

$$E_{Q(z|x)} \log P(x|z) - KL(Q(z|x)\,||P(z))$$

So, let us try to see that how this cost function can actually be coded. So, you remember that the cost function that we wanted to maximize is Q of z given, log of P of x given z expectation value of that or the expectation value is with respect to Q z given x minus KL divergence of Q z given x P z. And this is the lost function or the cost function that we wanted to maximize.

And if I want to do that what I have to do is, I have to maximize the expectation value of log of P x given z and I have to minimize the KL divergence Q z given x P z. So, I have a maximization operation involved, I have a minimization operation involved.

(Refer Slide Time: 25:57)



Variational Autoencoder : Coding the Cost Functions

- Maximizing $E_{Q(z|x)} \log P(x|z)$ is a maximum likelihood estimation. It is observed all the time in discriminative supervised model, for example Logistic Regression, SVM, or Linear Regression.
- In the other words, given an input z and an output x, we want to maximize the conditional distribution $P(x|z)$ under some model parameters.
- So we could implement it by using any classifier with input z and output x, then optimize the objective function by using for example log loss or regression loss.

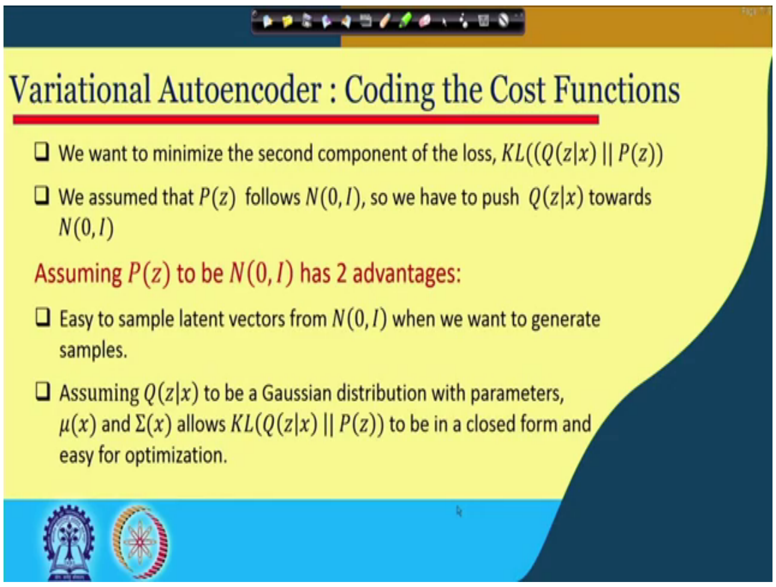
So, if you look at this maximization part that is maximization of the expectation value of log of P of x given z with respect to Q z given x, this is a maximum likelihood estimation and you find that my decoder is just a neural network. So, once I have a z my output x hat that is deterministic, there is no non-determinism involved in it. So, this P of x given z is equivalent to P of x given x hat, where x hat is your reconstructed value or the generated data.

And this effectively means, that I this effectively tells you that you have to reduce your reconstruction error between input vector x or input data x and the generated or the reconstructed data x hat. And this is similar to what you do in case of a discriminative supervised model. For example, logistic regression, SVM or linear regression or all that.

So, in other words given an input z and an output x, we want to maximize the conditional distribution P of x given z under some model parameters and we have to play with the model parameters you have to modify them for model parameters. So, that this distribution P of x given z can be maximized or P of x given x hat can be maximized or the reconstruction error between x and x hat can be minimized.

And this can be implemented using any classifier with input z and output x, then optimize the object objective function by using for example say, log loss or regression loss. So, that is how the maximization part over this can be implemented.

(Refer Slide Time: 27:51)



Variational Autoencoder : Coding the Cost Functions

- ❑ We want to minimize the second component of the loss, $KL((Q(z|x)\ ||\ P(z))$
- ❑ We assumed that $P(z)$ follows $N(0,I)$, so we have to push $Q(z|x)$ towards $N(0,I)$

Assuming $P(z)$ to be $N(0,I)$ has 2 advantages:

- ❑ Easy to sample latent vectors from $N(0,I)$ when we want to generate samples.
- ❑ Assuming $Q(z|x)$ to be a Gaussian distribution with parameters, $\mu(x)$ and $\Sigma(x)$ allows $KL(Q(z|x)\ ||\ P(z))$ to be in a closed form and easy for optimization.

And for minimization of the KL divergence, what you do is you assume that P z follows a normal distribution with zero mean and unit variance, so that we can play around with the parameters of Q z given x towards this normal distribution or the parameters of Q z given x can be pushed towards this normal distribution. And that is how the scale divergence has to be minimized.

So, assuming this P z, the distribution P z to have a normal distribution that has two advantages. Firstly, because it is a normal distribution with zero mean and unit variance, I can easily sample a vector epsilon from this distribution, right and then we are going for reparameterize, reparameterizing and assuming this Q z given x to be Gaussian distribution with parameters mu x and sigma x that allows our KL divergence between Q z x, P z to be enclosed form and that becomes easy for optimization.

So, I will stop this lecture here. In our next lecture, we will discuss from this point and we will also briefly talk about another generative model which is known as generative adversarial network.

Thank you.