Lecture 20.2
Variational Autoencoders: The Neural Network Perspective

Okay.

Refer Slide Time :(0: 14)

Module 20.2 Variational Autoencoders: The Neural Network Perspective

So, as I was saying that, there are two different perspectives for Variational Autoencoders. One is the neural network perspective, we just view it as a Neural Network and just list on the operations, do the standard thing. What do we do with neural networks? We write down what the model equations are. How do you get the output as a function of the input? What's your objective function? And what's your learning algorithm? If you do that, it's just the neural network perspective. The other perspective of course is the graphical model perspective; you think this, think of these things in terms of random variables, dependencies between random variables and then certain things on top of that. Right? So, we are going to look at both these perspectives. So, we'll start with the neural network perspective first and then go on to the graphical model perspective. Okay.

Refer Slide Time :(0: 54)

Figure: Abstraction

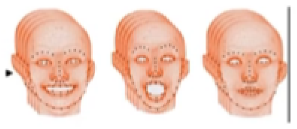
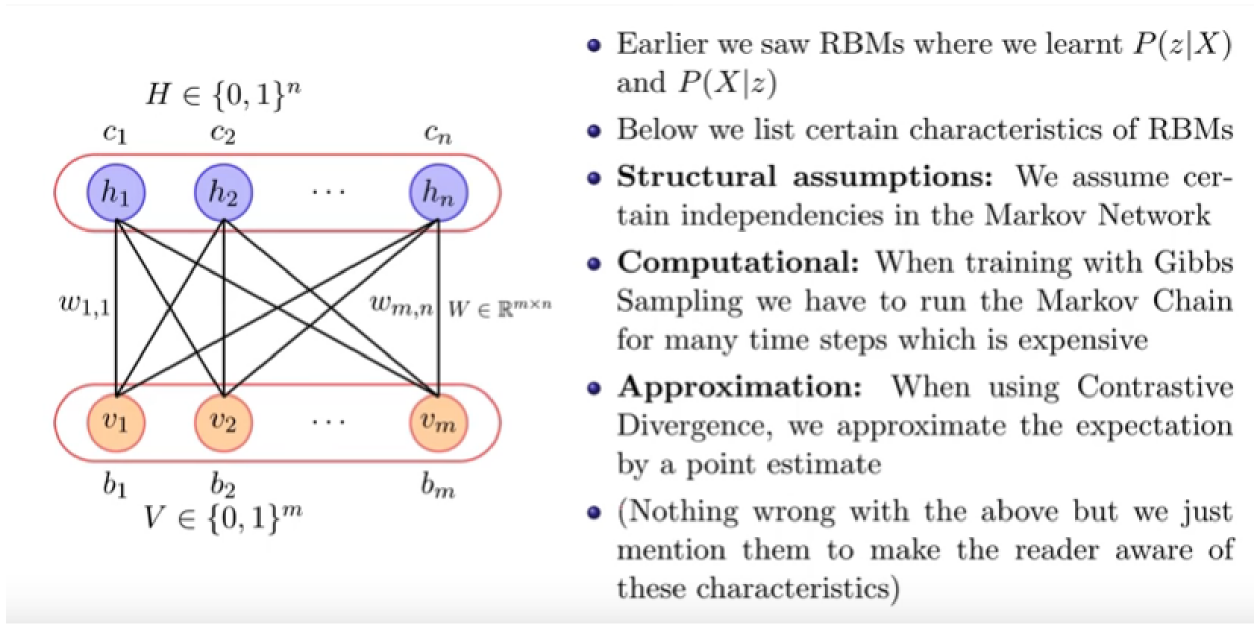


Figure: Generation

- 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)

So, once again, our setup is that, they are given some images. So, they are given some N images and we can think of images, as a collection of random variables, each image has 1, 0 to 4 pixels. So, our X is of dimension n in general or in this case of images that we have been taking 1 0 to 4. Right? So, that's your training data. And for each, X or for each of these images, the dimensions correspond to the pixels in the image. Now, what we are interested in, is in learning an abstraction that it given as X, well interested in learning a hidden representation z. So, here what I've shown you is actually a two-dimensional hidden representation? So, assume you just have z1 and z2 as two dimensions and what I have done is, for all the inputs that I had, in my training data. Right? For all the images that I had in my training data, I just computed this two-dimensional representation for them and just plotted them. And this are probably all the images of beaches, these are all the images of mountains; say these are images of buildings and so on. Right? So, you can see that, things which are similar, they'll have a very similar or clustered hidden representation and they would all lie, very close to each other. Right? So, that's what an abstraction means. Right? You just take this, high dimensional image and then try to learn a low dimension representation for that and in that representation space, things which are similar. But, were not so, obvious in the high dimensional space, come together in the low dimensional space. Right? That's the whole point of abstraction and the other thing that we are interested in doing is, generation. That if I give you, a hidden representation, you should be able to generate data back from it. Right? So, suppose I'd learn hidden representations for faces. Now if I give you a hidden representation, you should be able to give me faces back. Okay? So, these are the two things that we are interested in and in probabilistic terms, what we want is, we want a distribution over the latent variables, given the observed variables and a distribution over the observed variables, given the latent variables. Right? This is the drawback or the limitation that we had in Autoencoder and this is what we want to, fix in Variational Autoencoders. Okay? Fine.

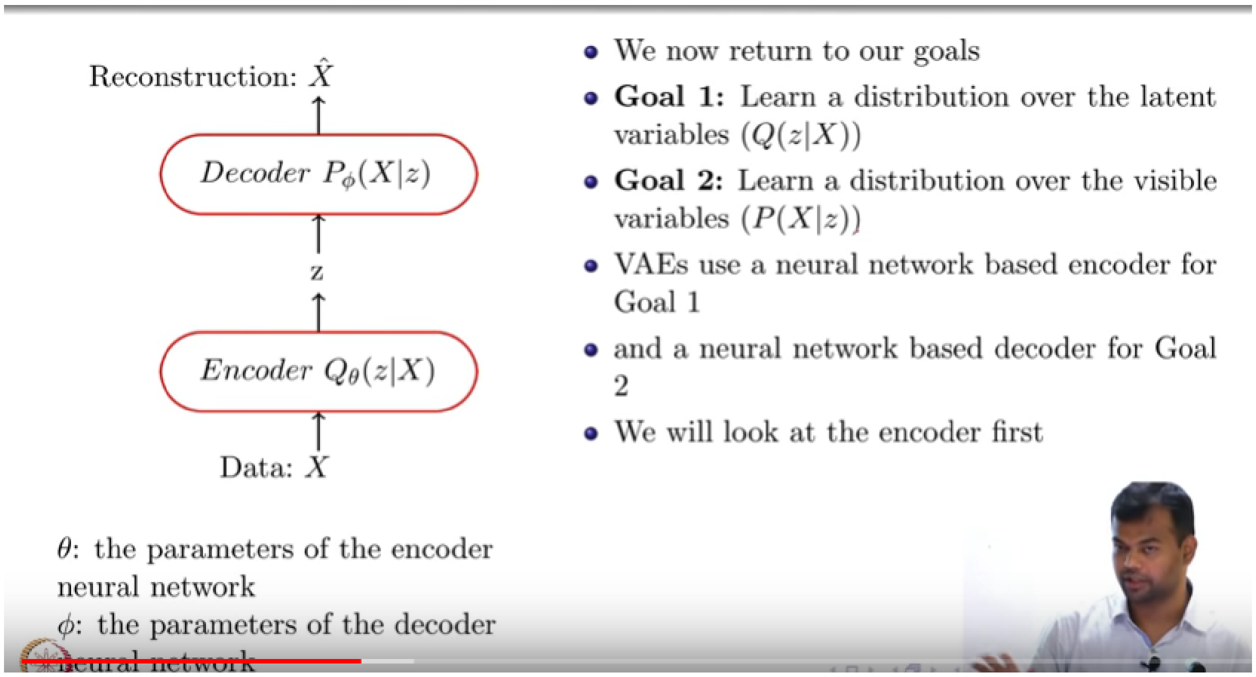
Refer Slide Time :(3: 00)



- 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)

So, you know what's a big deal? Right? I just told you that, I'm going to give you a neural network perspective. But, you have already seen a neural network, which can do this. Right? You have already seen RBMs, which does exactly the same thing, it's a neural network, it's a stochastic neutral Network. It gives you both the distributions P of Z given X and P of X given Z. So what? Why I mean, this model just ends here. Right? I've given you a neural network which can, are there to your wish list. So, what is? Why do you want to do something different? So, the answer actually, comes from literature and graphical models in general. So, the idea is that, all algorithms or models that you see, in all graphical models are generating models that you see, they always have, a lot of assumptions. These assumptions are either in the form of independencies or they are in the form of approximations that you do, while making certain computations or they are expensive, because you need to do something very expensive, like a Markov chain and so on. Right? So, I'll just list on, some characteristics I'll not call them limitations, because almost all the models that we'll see, starting with various not encoders, to auto regressive models and later on Ganz, they all have their own set of limitations. So, I not call them, 'Limitations' I'll just highlight some characteristics of RBMs and then later on we'll try to compare them, to very stern auto-encoders, which I'll probably do in the last lecture, once I finish all the deep generative models that I want to cover. So, here are the characteristics, first thing that we do is this, was a Markov network, where we had made, certain structural assumptions, what do I mean by that? What were the assumptions structural assumptions that we had made? The hidden variables do not depend on each other, the visible variables do not depend on each other and the only, so we had assumed this bipartite graph like water, it is visible in the figure. Right? So, that's a, structural assumption that we mean, it's not clear, why we made that assumption or what should that be the necessary thing or could I have been different or whatever it. Let's one assumption that we made, the other was in terms of computational. Right? So, when we are doing RBMs training, using Gibbs sampling, remember we were doing stochastic gradient descent and at every, step of stochastic gradient descent, what did we have to do? We had to run this Markov chain and you always got away by saying that, just run the Markov chain for a large number of steps. In practice that large number of steps, can actually be very, very large, in theory of course it is as n tends to infinity. Right? So, all those guarantees hold, only asymptotically that means only if you run the Gibbs change for an infinite number of time steps. Right? But, in practice also, you need to learn it for, a large number of steps. Right? And that leads to some computational limitations. Okay? The third was, approximation and this is what we had done when we were doing constructive divergence? Where we had done this nasty approximation that this, entire summation, which should have actually contained to raise to n terms, we just approximated by a point estimate of one single term, you remember that, hidden contrastive divergence, we just took one negative sample and approximated the second summation by, that sound. Right? So, these are some of the things that we did, in RBMs, you might do, a similar set or a different set of assumptions and approximations in the case of VS. Right? The point is there's nothing wrong, with any of these I'm not saying that, this is why RBMs are bad, it says that we need to be aware of these and such or similar, limitations or characteristics we'll see for all the, different deeps generative models that we are going to talk about. Right? So, variation auto-encoders might have something different, since it's good to have these multiple flavors, of these models, each one comes with their own set of properties and whatever suits you best for your application, you could rely that, that's why we're doing multiple of these. So, starting with RBMs. And now, looking at various nutrients. Okay? So, that's why the noodle perspective does not end here, this is just one possible neural network based solution for learning this joint distribution. But, there are others possible and we are going to look at those also.

Refer Slide Time :(7: 09)

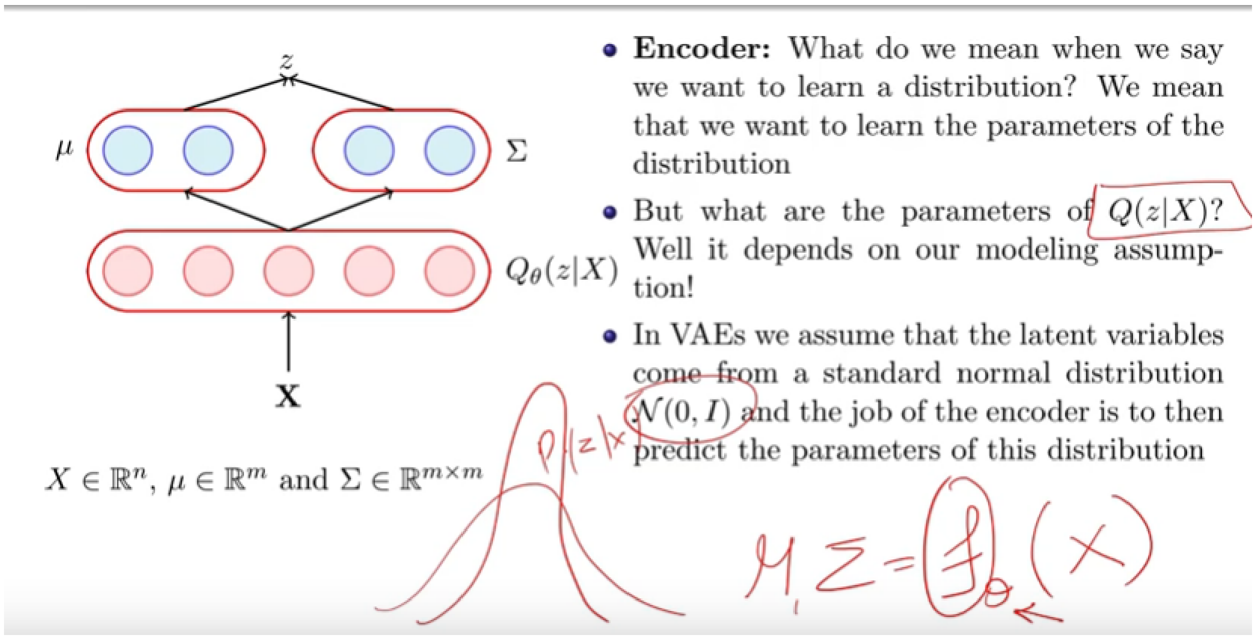


- We now return to our goals
- **Goal 1:** Learn a distribution over the latent variables ($Q(z|X)$)
- **Goal 2:** Learn a distribution over the visible variables ($P(X|z)$)
- VAEs use a neural network based encoder for Goal 1
- and a neural network based decoder for Goal 2
- We will look at the encoder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

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

So, with that background let's return to our goals. So, that first goal was: to learn this distribution key of Z given X, the second goal was: to learn the distribution X given Z, VAEs, you use a neural network based encoder for goal 1 and a neural network based decoder for goal two. Now, based on this initial recipe that I gave you, does do these two statements make sense, what do I mean by these two statements? That are you use a neural network based encoder to learn Q of Z given X and a neural network based decoder to learn P of K X given Z, what does that mean? What am I going to do? No, just go back to the recipe that I gave you at the beginning of the lecture. I'm going to represent P of x given Z, what are the parameters of this distribution? I don't know, because I've not told you which family it belongs to that, I could assume it's a Gaussian, I could be assume it's a Bernoulli, I could assume multinomial, whatever I want? But, what this high-level recipe tells me is that? I could choose any family that family would come with its own set of parameters, in fact let me give you the family, the family is going to be Gaussian, because for VAEs, everything that we look at, will make Gaussian assumptions, at all these distributions that we care about, follow a Gaussian distribution. Now, if I'm going to make that assumption, what are the parameters of a Gaussian distribution? Mu and Sigma. Now, what do I mean that I'm going to learn this using a neural network? I'm going to express, mu and Sigma, as a neural network function, which is this complex function and it has some parameters theta and so, my objective function is going to be with respect to theta. Right? Do you get that? So, can you relate this to the original recipe that we had seen, of course the details are still missing. But, I want you to keep the high-level picture in mind. How many of you get this please raise your hands? Okay? Good so, now we look at the encoder first and as I said. Right? So, it will take me almost the entire 35 slides, to get to the full story, I will deliberately skip some, you not probably realize it. But, I will deliberately not give you everything, when I'm talking about the neural network perspective, things will become clear only when I talk about the graphical model perspective. Right? So, if you feel there are some things missing, just wait till the end of the lecture and see if, everything becomes clear. Right? But, as of now, I am NOT Thole, much so, they shouldn't be anything missing. Right?

Refer Slide Time :(9: 36)

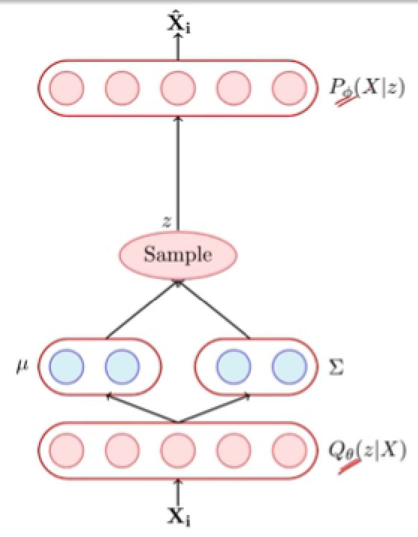


$X \in \mathbb{R}^n$, $\mu \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m\times m}$

- **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 $\mathcal{N}(0, I)$ and the job of the encoder is to then predict the parameters of this distribution

So, let's go ahead with that. So, again the same question, what do we mean when you want to when we say we want to learn a distribution? We mean that, you want to learn the parameters of the distribution. Right? But, what are the parameters of Q z, given X, V I said I want to learn this distribution that’s very, clear and when I told you that, you also told me rightly that, what I’m actually saying is, I want to learn the parameters of this distribution. But, what are the parameters of the distribution? You know, that's the parameters, of the parameters. Right? The weights of the neural network, are the parameters of the parameters of the distribution. But, what are the parameters of the distribution? So, I have not told you that. Right? So, now I’m going to introduce that, it depends on our modeling assumption. Right? What kind of family do we assume for P of Z given X, in variation auto-encoders, we are not going to deal with arbitrary families. Right? So, we are going to assume, a per ton certain family for the distribution and then try to learn the parameters of that distribution. Okay? Do you get the difference? That one is I tell you this is a joint distribution, I don’t care, what kind of family it belongs to, it could just be any arbitrary distribution. Right? So, it could be a distribution of the following form. Right? So, if this, is my X 1 dimensional, then it could be a distribution of this form. Right? But, what we are going to assume is that? The latent variables, actually come from a normal distribution. Right? And we are also going to assume that the normal distribution is 0 comma I. Okay? So, now what's the job of the encoder? It has to take as input an X and what should it output? What should it output? The parameters of this normal distribution. Right? Which is mu and Sigma? Okay? I just leave it at that and I will come back to it again. Okay? So, now the encoded part is clear, because I have assumed: that the distribution Q, is actually going to be a normal distribution, which has parameters mu and Sigma. So, this mu and Sigma, I'm going to express it as, f of theta, of X. Right? Where f is, what is f? On neural network. Right? And theta are the parameters of the neural network, it takes as input X and gives me the distribution over Z or rather it gives me the parameters. Right? Now, the advantage of working with the family of distribution is that, the only thing I need to give you is the parameters of the distribution, once I give that, you can sample things from that distribution. Right? So, once I tell you that this is what my normal distribution looks like and this almost looks like a standard. But, if it was not a standard distribution, if it was something like this, once I tell you what mu and Sigma is, now you can generate samples from this. Right? So, you can generate Z, from this distribution, if I tell you that, this is what? Z given X looks like. Do you get that? Okay? So, that's the advantage of working with a family, rather than working with arbitrary distributions. Okay? Right?

Refer Slide Time :(12: 37)

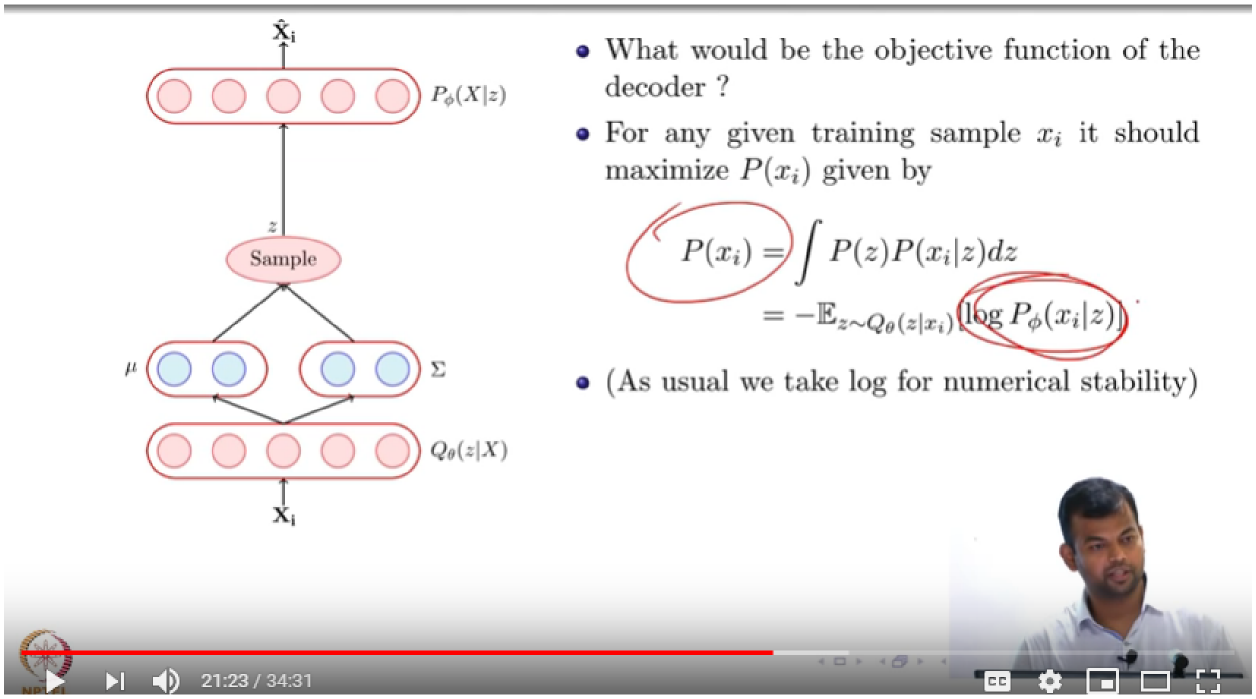


- Now what about the decoder?
- The job of the decoder is to predict a probability distribution over $X : P(X|z)$
- Once again we will assume a certain form for this distribution
- For example, if we want to predict 28 x 28 pixels and each pixel belongs to $\mathbb{R}$ (*i.e.*, $X \in \mathbb{R}^{784}$)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

Now, what about the decoder? So, what is the decoder going to do? So, this is what the encoder has done. Right? It has taken an X, it has mapped this through a sufficiently complex function of parameters theta and it has now, tried to produced mu and Sigma. Okay? What is mu and Sigma? Together parameters of the distribution. Now, what is the input to the decoder? Is it, mu and Sigma, what does the decoder do? It takes a hidden representation and tries to reconstruct the input. Okay? But, now there is some misunderstanding between the encoder and a decoder. Right? And an Autoencoder, they had perfect understanding, the encoder produces the hidden representation, the decoder takes the hen representations and tries to reconstruct something. But, now the encoder has in apartheid, what it is saying is I not producer hidden representation, I'm just going to produce mu and Sigma. But, what was the decoder do with mu and Sigma? It cannot work with me when Sigma, it needs to work with us Z. So, what will the decoder do now? So, what we have to do? Sample from this distribution. So, the encoder has told you what the MU and Sigma is, once you know, what the MU and Sigma is? You can sample from that distribution, everyone gets that, please our hands if you get that. Okay? Now, when I say sample, the obvious statement is I'm going to make is it’s not deterministic anymore. Right? So, if I'm going to say it as many times it takes me, to see it, but, this on the this is your z axis and this is the distribution which I've assumed, in the case of, very in the case of normal auto-encoders, you could produce only one of these values. Right? You just had a fixed Z. Now, I have given you this mu and Sigma and I could sample any of these Z’s of course the probability of sampling would be proportional, to the probability distribution. Right? So, the values which are closer to the mean, are going to be more likely. Okay? But, it’s still going to be, they're still going to give me some randomness distribution. Okay? So, I'm going to sample from this distribution. Okay? Now, once I have sampled from it, the job of the decoder, is to predict a distribution over X's. So, now I have given you Z as the input and now, I am interested in learning a probability distribution over X's. Okay? That's what the decoder is supposed to do, once again we’ll assume a certain form for this distribution, because by now, it should be obvious that, you can't really work with arbitrary distributions or rather the other thing is obvious that, if you work with the family of distributions, life becomes much more easier. Right? So, we are going to, work with images, say for example and you could think of images as28 cos 28 inputs. So, they belong to this R raised, to 784 space. So, what would be a suitable family for such a distribution of real numbers. Again I could assume Gaussian, there could be many other families. But, I'm going to stick to Gaussian again. Right? So, again I'm, going to assume that, it's a Gaussian distribution and further I'm going to assume unit variance, what does that mean? Can you relate it to the assumption that been made for RBMs, how many forget that? Please raise your hands if you get that? What does it mean that the covariance matrix is identity or unit variance, all the off-diagonal elements are 0 that means all, the random variables I J, where I is not equal to J R independent of each other, given what does, what is the distribution that we are talking about? X given Z. So, this is same as saying that, given the latent variables, we are assuming that the visible variables are independent of each other, this is very similar to the assumption that we had made in RBMs, with a different way of arriving at the same assumption. Right? Do you get that? Please make that connection, again is fine is that. Okay.

So, now you assumed it's a Gaussian and further we have made life even more simple that we have assumed it's a unit variance. So, what do we need to predict now? Only the mean of the distribution. Right? So, we just need to produce the mean of the distribution and what are we going to do again, following a recipe? What's the input to the decoder? Z we are going to predict mu as a, function of come on, Z parameterize by, some parameters and we'll call it, 'Phi'. Okay? So, the encoder parameters are theta and the decoder parameters are Phi and although, it looks different we have actually followed the same recipe for the encoder and the decoder. Right? We assumed a form for the first distribution, express the parameters of that distribution using a neural network with parameters theta, we assumed a form for the second distribution, express the parameters of that distribution using a neural network with parameters Phi and now, our final objective function whatever it is, I have not defined it yet, is going to be with respect to, MU and Sigma. Everyone, everyone, everyone theta in Phi. Right? Is going to be with respect to theta in Phi, is that clear? Anyone who's, not cleared with this? Please either hands, if it this is clear. Okay? See a very small, set of students. So, people who did not raise hands please ask me specific questions, four comma two, yeah! You know, the one, the one before you. Okay? So, you don't want to raise your hand, even the lazy ones, please raise your hands again if you have understood this, everyone please raise your hands. Okay? Now it's fine, you guys are just lazy you know, how much does it take to raise your hands. Okay? So, I assume this is clear, we have followed the same recipe, it's the same as what I started the lecture with. Right? Okay? Now, I very conveniently told you that, the optimization is with respect to theta and Phi. But, I am not even giving you the objective function. Right? So, we need to first figure out the objective function and then ask, those nasty questions that we had in RBMs, whether the objective function is tractable or not, because we came up with a nice objective function in the case of RBMs. But, what was the problem there? Computing that required of an exponential computation. Right? Because you had this expectation, over to raise to, n plus M, terms like the summation was over to raise to n plus M terms. Okay?

Refer slide time :(19:07)



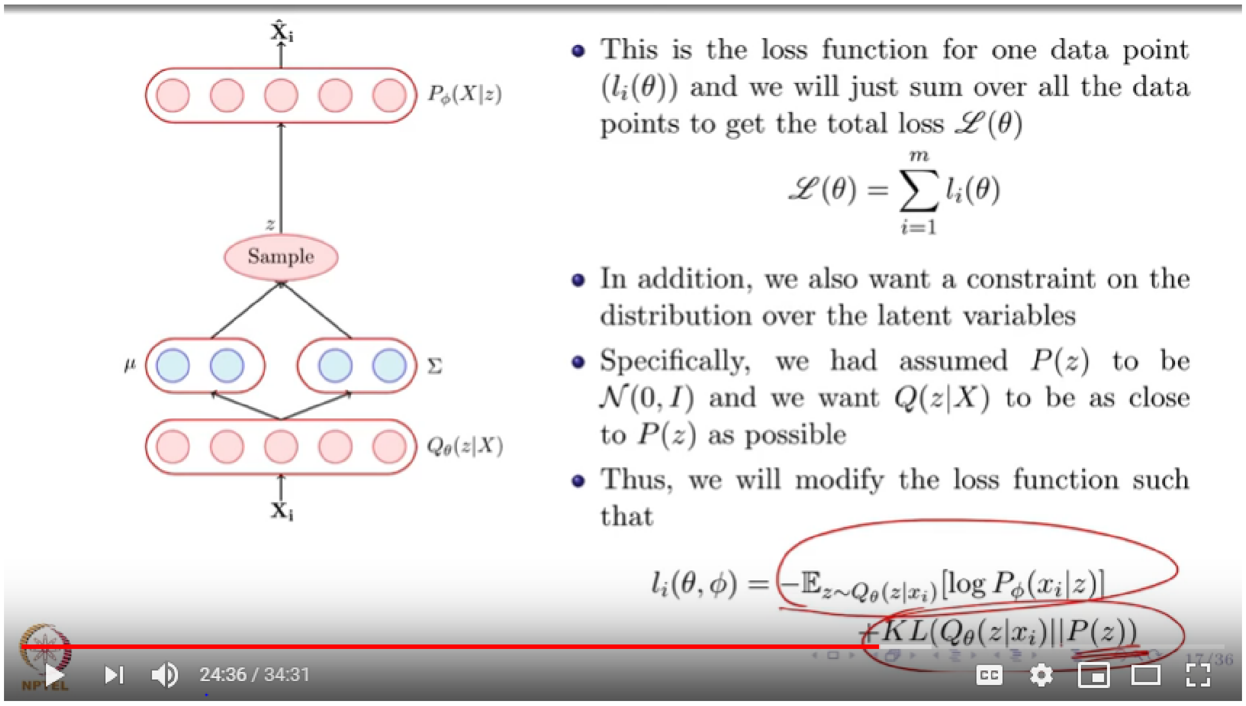
- What would be the objective function of the decoder ?
- For any given training sample x_i it should maximize $P(x_i)$ given by

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

- (As usual we take log for numerical stability)

So, what would be the objective function of the decoder? What should we maximize? What is the decoder triangular was the decoder time we'll learn a probability distribution, P of X given Z. What did I give you as the super input, X. So, what should the decoder better do, maximize the remember the recipe, maximize the log likelihood of the, input that I gave you. Right? So, what's the objective function maximize, log of P of X. Right? For all the N training examples that, everyone agrees with that. Okay? Now it should, now for this one given example, it should just maximize P of X I and now just a shorthand notation. Right? So, when I say P of X, I it means P of X taking on the value X Y. Right? Okay? Now this of course can be written as, the following integral which is very easy to, compute how many integrals are there actually. So, it's actually when I say integral with respect to Z, Z itself is a vector. Right? So, this is integral with Z 1, Z 2, Z 3, up to Z M, read the number of dimensions that Z it has. Right? So, this is again, a very nasty integral and this is actually again the same kind of expectation that we were dealing, with in the case of RPM. Right? Not the same expectation but the same kind of expectation, where you have this term inside and this integral is nothing, but the expectation of that term, with respect to. Okay? So, that's the problem that we have you again are left with this, was in wrong class. Okay? And as usual, we take the log for numerical statement. Right? So, we have again comeback to the same, kind of story that we want to maximize the log likelihood, but it's very hard to compute, and so on. Right? But I'm just going to keep it at that for now and we'll come back to this question, I just gave you the intuition that is going to be hard, but I'm just going to write it, as it is that this was, my objective function or rather this, was my objective function, with respect to one training sample.

Refer slide time :(21:28)



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

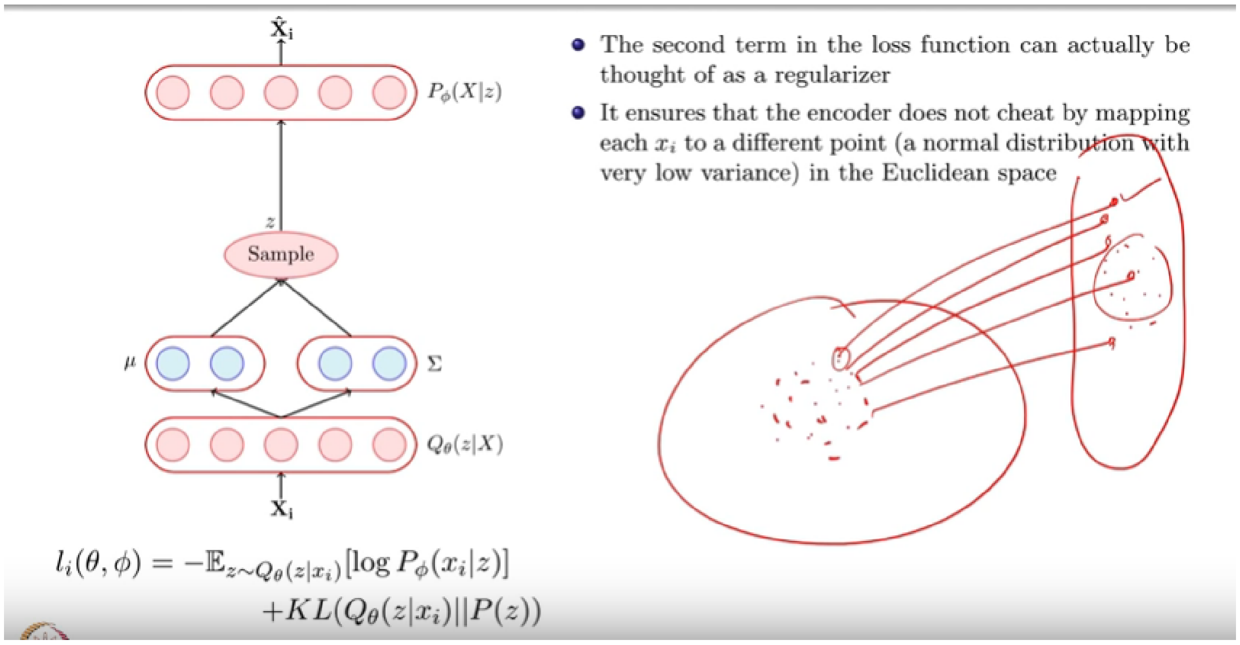
$$\mathscr{L}(\theta) = \sum_{i=1}^{m} l_i(\theta)$$

- 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

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

And if I want to consider all the training samples in my data, it's just going to be the sum overall the training samples. Right? Okay? And is that all, what is the other thing that, we had remember we wanted to learn two distributions, does this objective function capture both, the distributions, it only captures P of X given Z. Right? What about the other guy, do you get the question. Right? All I'm asking you to do is, maximize this quantity, but this is just the circle quantity, which is just P of X I given Z that was one of the two distributions that I was worried about, the other distribution, was Q of Z given, X is there anything in the objective function which takes care of Q, what is the constraint that I wanted on Q ?What did I want cute be, what did I start off with, I assume that Z is, Gaussian with, zero mean and unit really based, on that can you give me a constraint. So, that I can add it to the objective function, the first term is clear is just the log likelihood of the data, written in that fancy expectation form. Okay? But it's just P of X that's all it is. Okay? The second term has to do something with Q I, want Q to be as close to a, normal distribution, standard normal distribution, how can I convert this English sentence into a equation? I want to but we don't like to, do square errors when we are dealing with probability distributions, I have a distribution Q; I want it to be as close to a normal distribution. So, I'm talking about the distance between two distributions. So, what should I try to minimize, the KL divergence between the two distribution .Right? So, now what I'm going to do is, to take care of the other guy in that I care about which is Q, I'm going to add this constraint that and I'm just so, I was talking about maximizing. So, that's the same as minimizing the negative of the log likelihood. Right? And minimizing the KL divergence, between the two distributions. Right? Where I have assumed that the second guy, P of Z is a normal distribution, does this story make sense. Okay? So we have seen the entire neural network, you've seen what the encoder is going to compute, we have seen what the decoder is going to compute, now we have come up with an objective function, which has two terms, the first term takes care of the objective of the decoder, which is to maximize the log likelihood of the data, it better construct samples, which look like your data. And the second term tries to take care of the other intention which we had, is learn the distribution over the Z, but try to make it as close to the normal distribution, is it fine .Okay? Now of course the question comes, whether this loss function, is differentiable is it tractable, can I easily optimize it, those questions I am not touching at this point, we'll come back. Okay? So,

Refer slide time :(24:40)



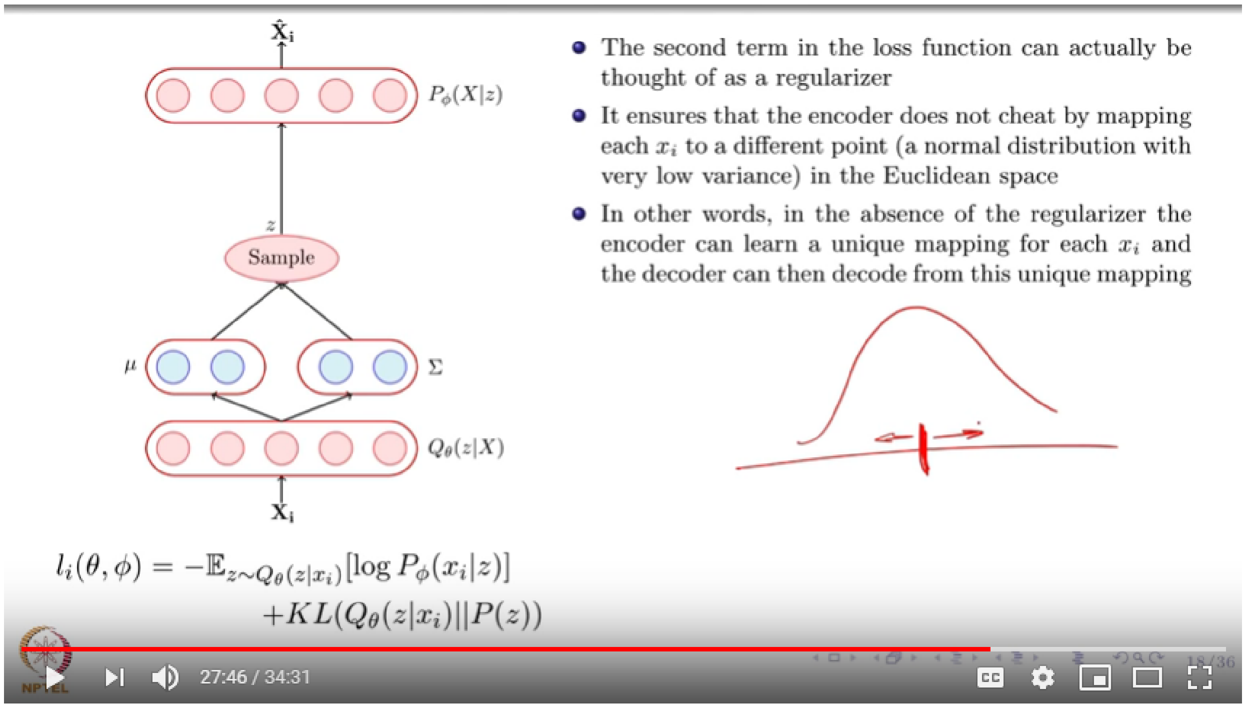
- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space

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

and so, the second term actually, let’s focus on the second term again. So, let’s assume this is an objective function and we are not worried about computational aspects, as of now, whether I can compute these terms or not I am not worried about, even I though that know that there's a nasty integral sitting over here. Okay? Now the second term actually acts as a regularizer. So, remember now this loss function looks, eerily similar to, remember this, from way back five years back or ten years back or maybe a two months back, remember this form that this is the loss function over the training data. And you add some regularizer, on top of that. So, now can you think of these two terms, as the first term, which corresponds to the likelihood of the training data. And the second term, actually corresponds to some called sign some kind of a regularize. Right? So, what it does it does is that ensures that the encoder does not cheat, by mapping every X that I feed to it, to some unique Z. Okay?

So, I have some X's, we actually come from some distribution, it may be normal it may be something else, but it's some distribution. Right? The X's because these are all images of Beaches these are all images of mountains. So, it might be a multi modal distribution. One mode corresponding to beaches and other mode corresponding to mountains and so on. But there's still some distribution, now if I don't add any constraint, what it could do is, what the encoder could do is in the z space, it could just not each of these guys to a unique guy. Right? And then the decoder now it's easier for the decoder, it's just a lookup that whenever, I see this person I just need to reconstruct this guy. Right? So, it could do this kind of a cheating, where it does not really try to capture, any latent representations in the data, it does not really try to make sure all the mountains are close together or all the oceans, I flows together. But it just plies to do some mapping and then recovers from there, but now if I put a constraint on that, that these Z's also need to come from a normal distribution, then it acts as a regular I it does not allow it to you learn these unique mappings. So, these mappings, you could think of as zero variance mapping Z, each of the Z has a very unique identity, given the X. So, that's the same as saying that there is no variance in the Z, you give me an X I'll just give you a fixed Z. Okay? So, this regular is this second term acts as a regularizer.

Refer slide time :(27:12)

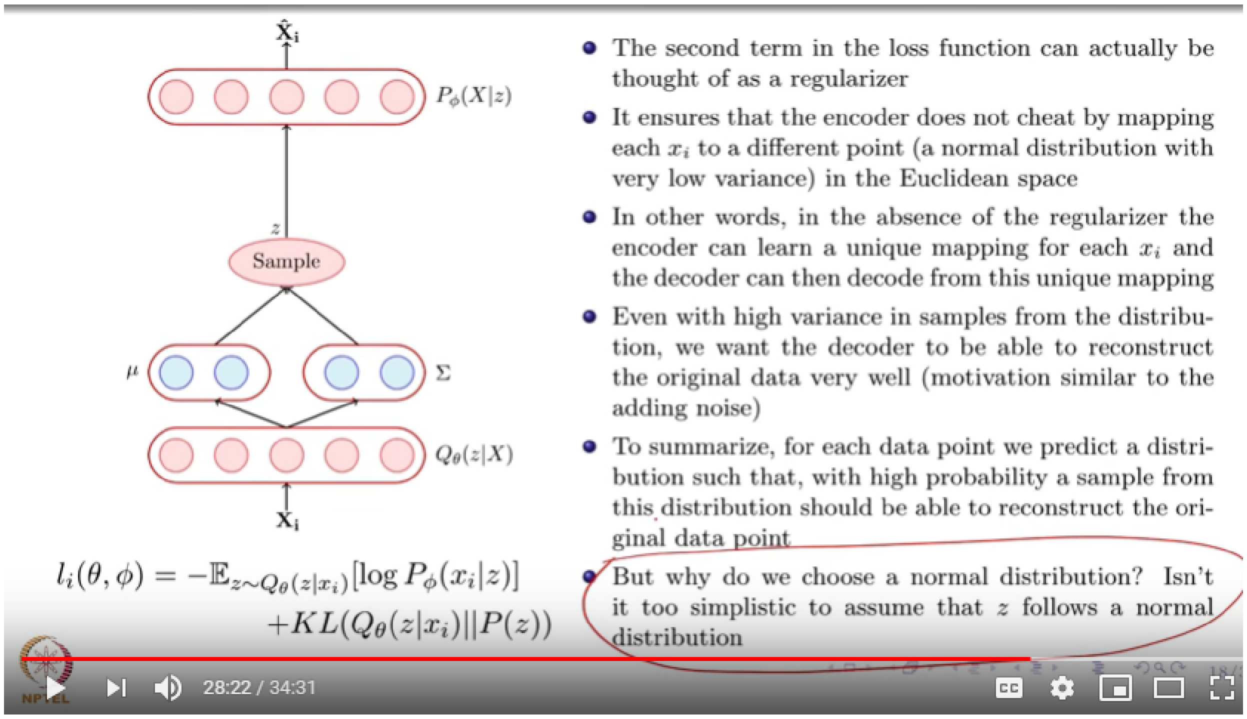


- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space
- In other words, in the absence of the regularizer the encoder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

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

And it ensures that you don't learn these unique mapping. And the other thing that it ensures is that, I am learning a distribution over Z. Okay? And now if in the case of an autoencoder, I was given a single Z and the job of the decoder was to reconstruct from there, but now since I am sampling from this distribution, I could even sample a noisy Z, which comes from this distribution. And the decoder now has to be robust to this noise, even if I give it a Z, which comes from this distribution, it's noisy it should still be able to, reconstruct the input. Okay?

Refer slide time :(27:48)

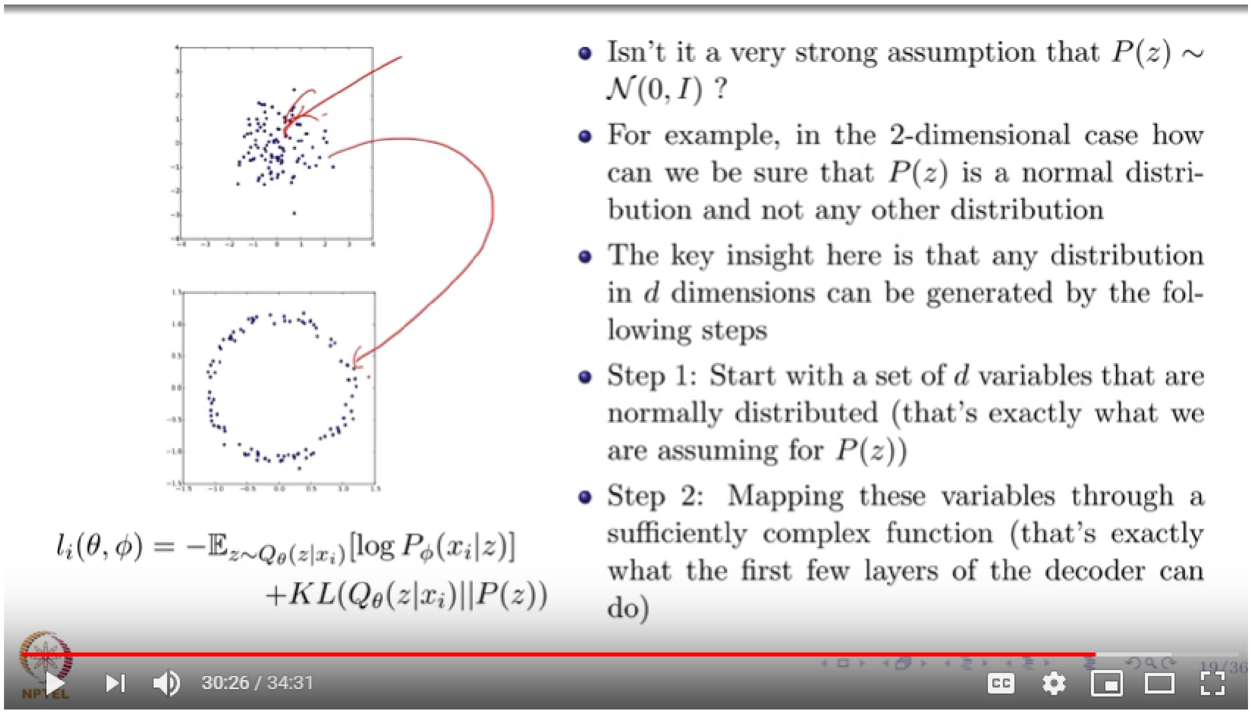


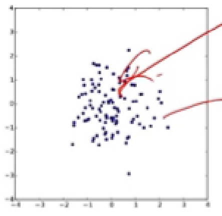
- The second term in the loss function can actually be thought of as a regularizer
- It ensures that the encoder does not cheat by mapping each x_i to a different point (a normal distribution with very low variance) in the Euclidean space
- In other words, in the absence of the regularizer the encoder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

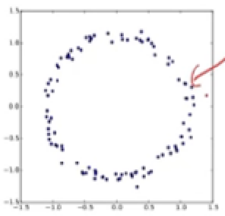
$$l_i(\theta,\phi) = -\mathbb{E}_{z\sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

So, that's how you should view these two terms that are there in the objective function, but now this critical question. Right? Which is there on the last bullet. Okay? I'll do it so. why do we choose a normal distribution? So, what we a reassuming is that these latent variables, actually follow a normal distribution. Why a normal distribution? And what's the first thing that comes to your mind like it's, off all the possible distributions, I have first restricted the family, I have said only Gaussian distributions. Within Gaussian distributions, of all the means and variances as possible, I just said that this is a standard normal distribution, doesn't that look too restrictive.

Refer slide time :(28:28)





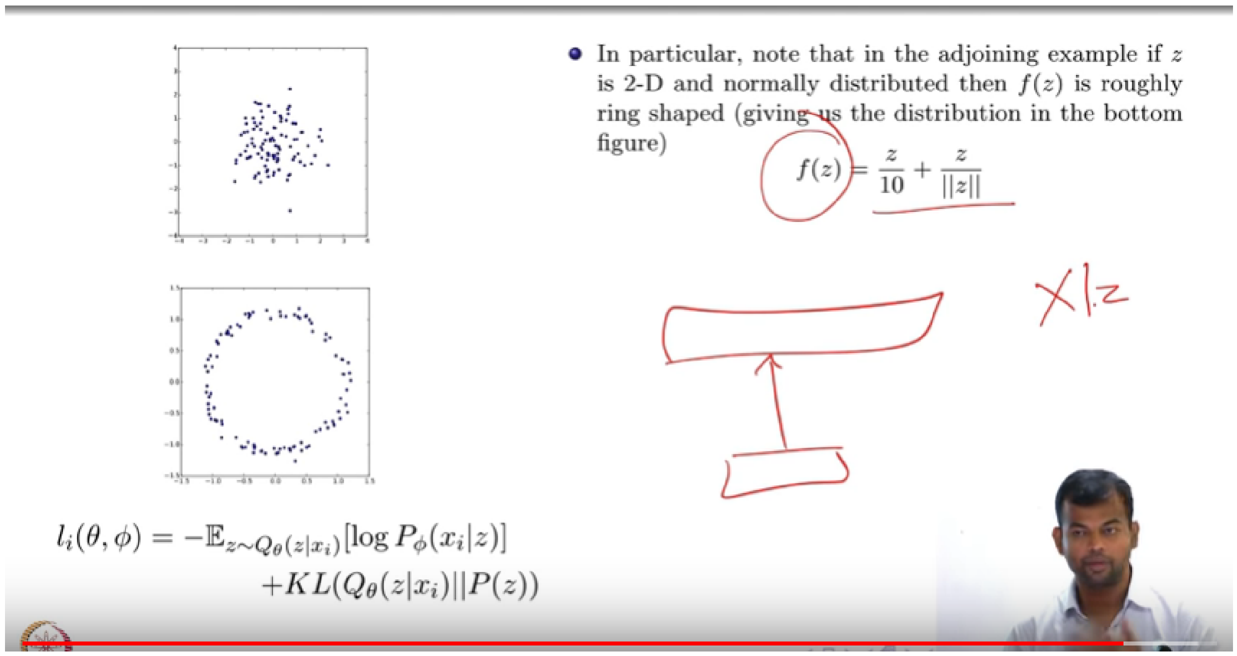


$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim Q_\theta(z|x_i)}[\log P_\phi(x_i|z)] + KL(Q_\theta(z|x_i)||P(z))$$

- 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)

Isn't it a very strong assumption that Z belongs to a normal distribution. And since the figure is only already there, the top figure is actually trying to show you data which comes from a normal distribution. So, you can think of it as a bell from the top view. So, you're looking at the area under the bell. Right? So, this is all the things are there's high density, under the bell or the circle under the bell. So, why did we assume something like that what if these eggs were actually distributed like what I've shown in the second figure, no but we are not using that. Right? What we are doing is we are just assuming that Z's are, come from a normal distribution. But what if Z comes from such a distribution, the second figure and that's where you need to connect everything to a neural network, which is the theorem that he's referring to so, again he related to that, I guess I will try to complete that story. Right? So, so in the two dimensional case. Right? Why should it be a normal distribution like the figure one, it could be any arbitrary distribution like the figure two, but a key insight here, comes from the following observation that and this is this is nothing to do with neural networks, there is just normal probability and statistics that you can start with a set of D variables, which are distributed normally. Right? In our toy example D is equal to two. So, I could start with a variable, which is distributed two variables, which are distributed normally. And from this I could map these variables, through a sufficiently complex function and create any arbitrary distribution. What does that mean? So, I've drawn Z, from a normal distribution. So, that means I'm going to get Z's which come from here. Now what I know is that, I could take such as it and you have to just take my word on this and maybe go back and read some of your probability and statistics notes. But just for now just take my word for that and I'll actually explain, it in the case of this toy example, you could take a z drawn from here. Right? And pass it through a reasonably complex function. So, that you get start getting values, which look like as if they had come from this other arbitrary distribution that you care about. Okay? So, let me give you an example, in this particular case. Right?

Refer slide time :(30:31)



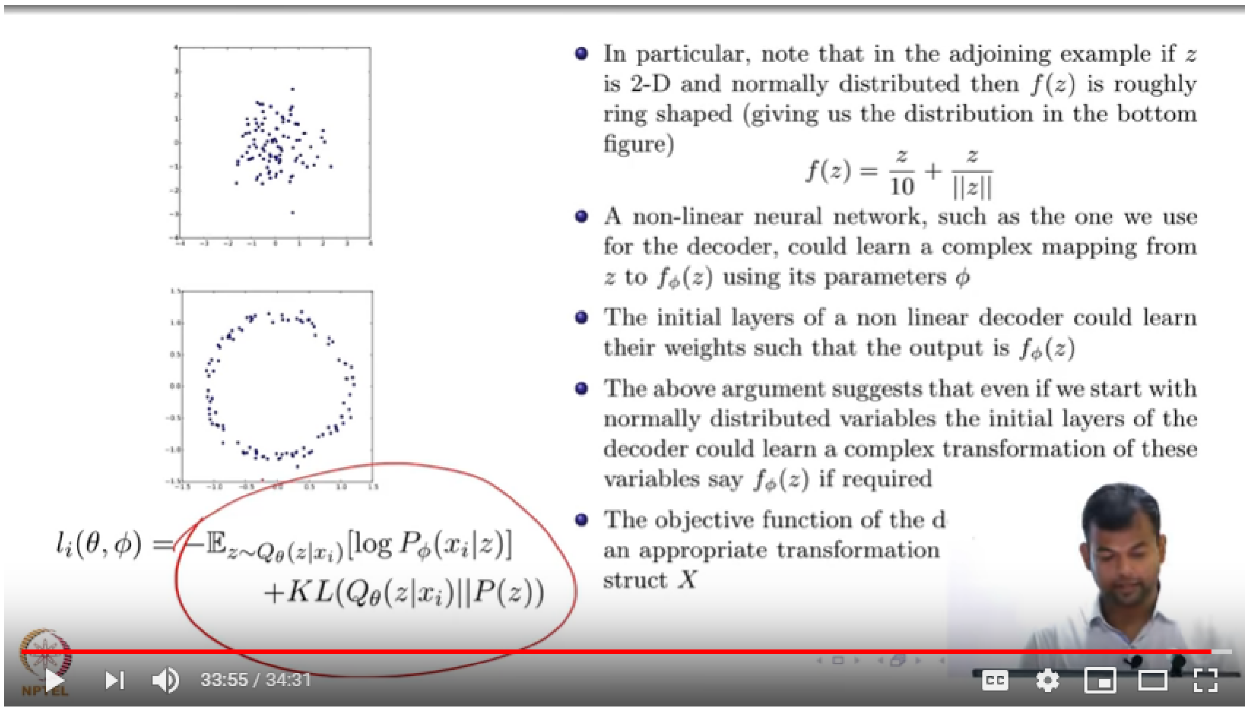
- In particular, note that in the adjoining example if z is 2-D and normally distributed then $f(z)$ is roughly ring shaped (giving us the distribution in the bottom figure)

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

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

For example, this complex function that I am talking about, is this function. So, try this, you draw samples, from a normal distribution, map it to this function and then plot the values, you will get back the circle that you see in the second distribution. So, what I'm asking you is go back and draw say around thousand10,000 samples from a normal distribution, compute the following function, for that variable that sample, which you have drawn and plot it and you will see that you'll get this second thing. So, what I have done is I have taken as Z, I have passed it through a sufficiently complex function and then I know that I can go to any arbitrary, distribution that I want. Now what is a sufficiently complex function that you know of, you relative, hence decoder is a. So, what does that mean can you tie these things together, I could make this assumption that it's a normally distributed latent variable, if it is not the case, then the initial layers of the decoder could, learn this complex mapping. So, that I get to whatever distribution for Z, which is actually the real distribution for Z, such that when I now compute X given Zit will maximize the probability that, I care about how many forget this statement? Please raise your hands Okay? Good.

Refer slide time :(31:53)



- 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 d an appropriate transformation struct X

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

So, that's exactly what we do. Right? So, if you can pass these Z's, drawn from a normal distribution, through a sufficiently complex function and that function could have its own parameters, which are Phi and the function that I'm integrating a neural network and we know based on the new universal approximation theorem that a neural network under certain conditions of course, can learn these arbitrary functions, of the input your input is Z, which is normally distributed, the actual distribution that you care about is some complex function of this normal distribution. But a neural network the initial layers could learn this and then finally when you get, the P of X given Z, it would have come from a Z, which was the actual latent distribution and not the initial normal distribution that you had started with. Right? Does that make sense. Okay? So, what it means is that if the decoder really needs to learn its parameters Phi, such that it transforms the Z from the normal distribution to an arbitrary distribution. And only then the objective function would be maximized, then the decoder will do that. Okay?
The decoder can't do that, because its job is to maximize the objective function and to maximize the log likelihood if you, need to learn this transformation for Z, the initial layers of the decoder can't do that. Okay? That's why it's fine, even if we start with a normal distribution. Is everyone clear about this? Please raise your hands if you are. Okay, good. So, that's where we'll end the neural network, perspective of variation autoencoders and I'm ending, on this sad note, that I haven't actually told you, whether this loss function is tractable, if it is tractable, how are we going to deal with it. I am not really told you about that. Okay? So, we will come back to that later on, but for now, this loss function makes perfect sense, from the new relative perspective, it's very similar to everything that we have done before, that you want to maximize the log, likelihood and that's just this loss function just follows from that. And the additional constraint was that, you want the latent variable distribution, to be close to the normal distribution. Okay? So, that's where we end the neural network perspective and now we will go to the graphical model perspective.