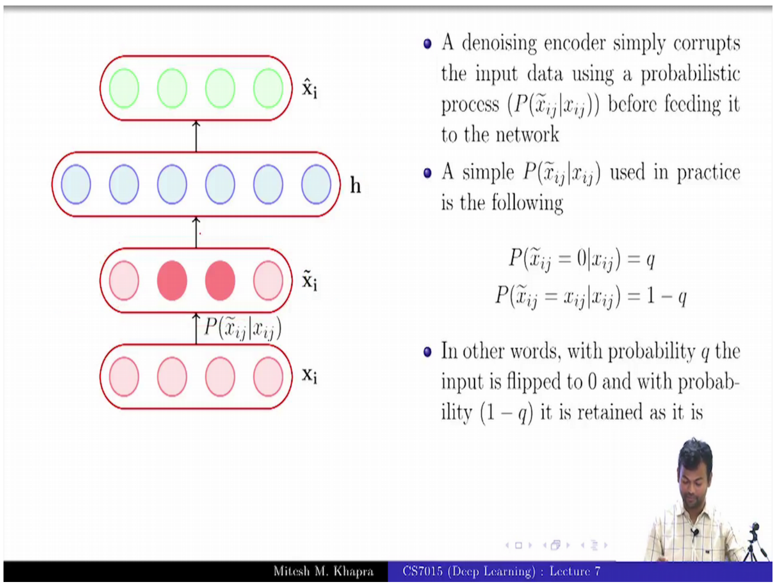
Deep Learning
Prof. Mitesh M Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 7.4
Lecture – 07
Denoising Autoencoders

Right so now, we will do something known as Denoising Autoencoders.

(Refer Slide Time: 00:17)



$\hat{x}_i$
h
$\tilde{x}_i$
$P(\tilde{x}_{ij}|x_{ij})$
x_i
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 7

- A denoising encoder simply corrupts the input data using a probabilistic process $(P(\tilde{x}_{ij}|x_{ij}))$ before feeding it to the network
- A simple $P(\tilde{x}_{ij}|x_{ij})$ used in practice is the following

$$P(\tilde{x}_{ij} = 0|x_{ij}) = q$$
$$P(\tilde{x}_{ij} = x_{ij}|x_{ij}) = 1 - q$$

- In other words, with probability q the input is flipped to 0 and with probability $(1 - q)$ it is retained as it is

So, the idea behind the denoising autoencoder is very simple, what you do is you have your original x i, ok. Now for the minute for a minute just consider the discussion when your x i's are binary inputs ok. So, each of these red guys can be between can be 0 or 1. Now what I do is, before feeding it this input to my autoencoder; the box is the autoencoder what I do is I do a corruption ok.

So, the corruption is as follows with probability q I will set x ij; that means, one of these guys to 0, right? And with probability 1 minus q, I will keep it as it is ok. So, with some probability q, I am actually corrupting the data otherwise I am retaining the data as it is ok. And then feeding that data to the autoencoder, why would this work? Binary input case as I said just assume that the inputs are binding.

We will also see the other case, why would this work? What was our problem earlier? that was completely able to reconstruct the training data right, but at test time I hide issues. Now what I have done to the training data? Corrupted it just think for a minute what will happen; now I want someone to ask me a question, in return oh that is the corruption that I am choosing or you could flip it is what you are saying. Yeah if it is 0 change it to 1 so that is also fine. That is the question I was expecting, what is the loss function now; what is the loss function? X hat my i minus x tilde i or x hat i minus x i which choice makes sense.

Student: First tilde.

First let us the case take the case when I do x tilde i what happens in that case from this networks perspective it is still learning to memorize the training data right it just this is what it thinks as the training data, and just trying to learn that transformation right. So, it is not really helping my case, do you understand that? I just corrupted the training data that is fine, but from the networks point of view; it still gets away by memorizing this data, and that is not what I want. So, what should I do? Can anyone tell me the; I mean can everyone tell me the answers?

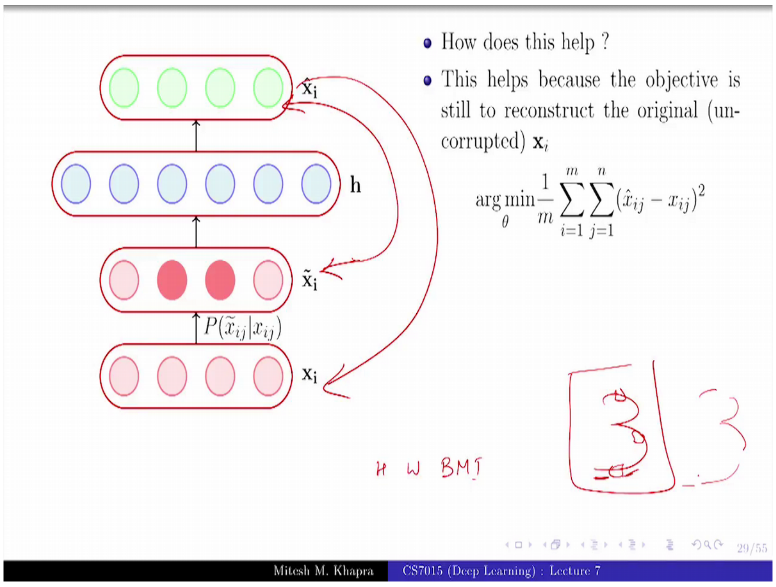
Student: Minimize.

Minimize the error between.

Student: An x i.

An x i, how many if you understand; why that should help? All of you gave the answer, but only few of your raised your hands why so, hard to deal with this inconsistency.

(Refer Slide Time: 03:04)


$\hat{x}_i$
h
$\tilde{x}_i$
$P(\tilde{x}_{ij}|x_{ij})$
x_i
H W BMI
29/55
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 7

- How does this help ?
- This helps because the objective is still to reconstruct the original (uncorrupted) $\mathbf{x}_i$

$$\arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\hat{x}_{ij} - x_{ij})^2$$

The cells because I am still going to minimize my original objective function ok; now can the network get away by copying the input to the output? So, input remember the input to the network is this and what I am trying to minimize this, if I just copy x tilde i to the output, will my objective function be minimized; no right? So, it does not have incentive to copy now. So, what will it have to rely on? Say a reasonable probably 20 percent is the standard right. So, even if I reconstruct I will not get 0 error I will at least get some 20 percentage ok.

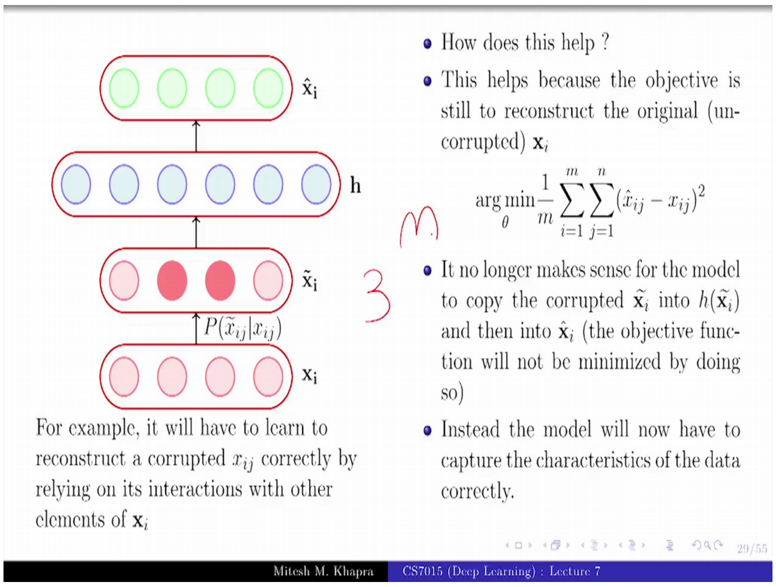
So, let us let me give you an example and then let me know if you can figure out what happens. This example will contradict something else that we have done before, but just play along. Suppose my input features were height weight and BMI and we all know that BMI depends on height and weight I hope all of us know.

Now, can you think what is happening? I am corrupting one of these inputs. And I still want everything to be reconstructed back. So, what will the network now have to rely on? It will have a now rely on this relations between these inputs also. So, again if I take my example of digit 3, I have corrupted some of these pixels right, but I still want to be able to reconstruct 3. So, it will have to be smart enough to learn that if I have seen this and I have seen this then it has to be something in between which gives me a 3, do you get the intuition right?

So now I am making it is job harder. So, that it is robust to changes at test time; that means, a test time if my digit looked something like this, it should still be able to predict it as a 3 or it will still be able to learn the same representation as 3 do you get the intuition, right? So, that is what I am trying to do I am trying to somehow bring in the corruptions that I would expect a test case and trying to make the model more robust.

It can no longer get away by memorizing the training data because I am not feeding it the correct training data, it has to do something smarter than that, everyone gets this. I will come back to your question everyone gets this please raise your hands yes, yes this is all under regularization no this is regularization no so at that case I have already made that overfitting can happen in an over complete as well as under complete autoencoder, everyone gets that right? I show that example where it could happen in both the cases. So, my figure maybe over compete, but it can just happen in any of these is that fine.

(Refer Slide Time: 05:44)



For example, it will have to learn to reconstruct a corrupted x_{ij} correctly by relying on its interactions with other elements of $\mathbf{x}_i$

- How does this help ?
- This helps because the objective is still to reconstruct the original (uncorrupted) $\mathbf{x}_i$

$$\arg\min_{\theta} \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{x}_{ij} - x_{ij})^2$$

- It no longer makes sense for the model to copy the corrupted $\tilde{\mathbf{x}}_i$ into $h(\tilde{\mathbf{x}}_i)$ and then into $\hat{\mathbf{x}}_i$ (the objective function will not be minimized by doing so)
- Instead the model will now have to capture the characteristics of the data correctly.

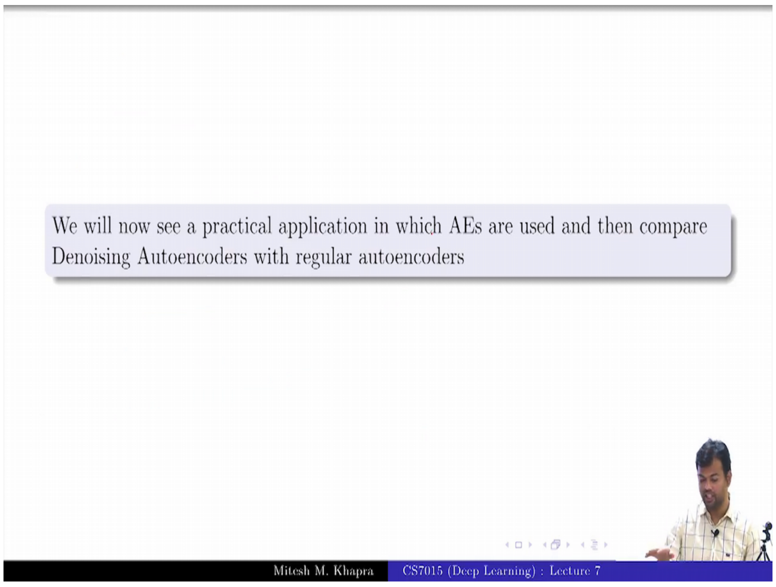
29/55

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

It no longer makes sense for the network to just start copying the input data.

Different kinds of noises means yeah. So, let me try to answer that right. So, what probably you are trying to say is that all my input images were 3 vertically written. I added some noise and managed it, but now at test time suddenly you show me a 3 of this kind, like that will not work also. That is what were your question was a different types mean different values of the noise 20 percent 25 percent and so on.

(Refer Slide Time: 06:16)


We will now see a practical application in which AEs are used and then compare Denoising Autoencoders with regular autoencoders

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

So, we will first see practical application in which autoencoders are used and then compare it to denoising at autoencoders. So, this the next few slides for those of you may care is also a small answer to the difference between machine learning and dp ok.

(Refer Slide Time: 06:36)

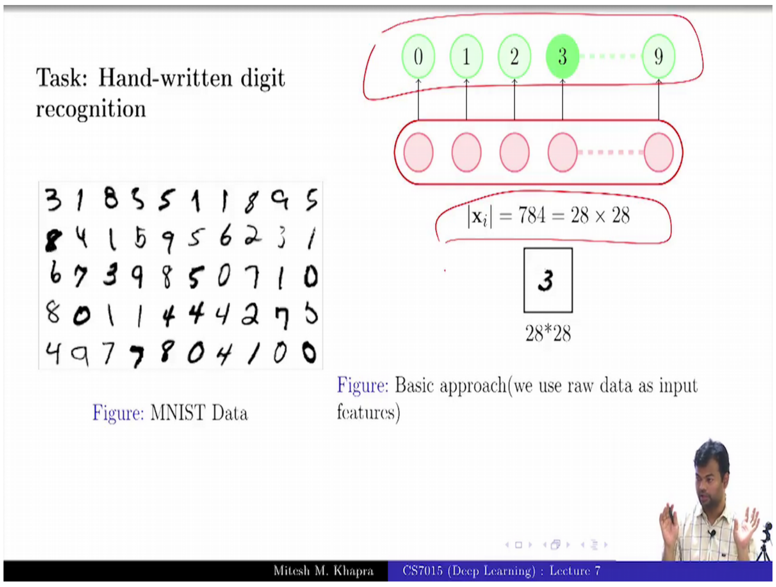

Task: Hand-written digit recognition

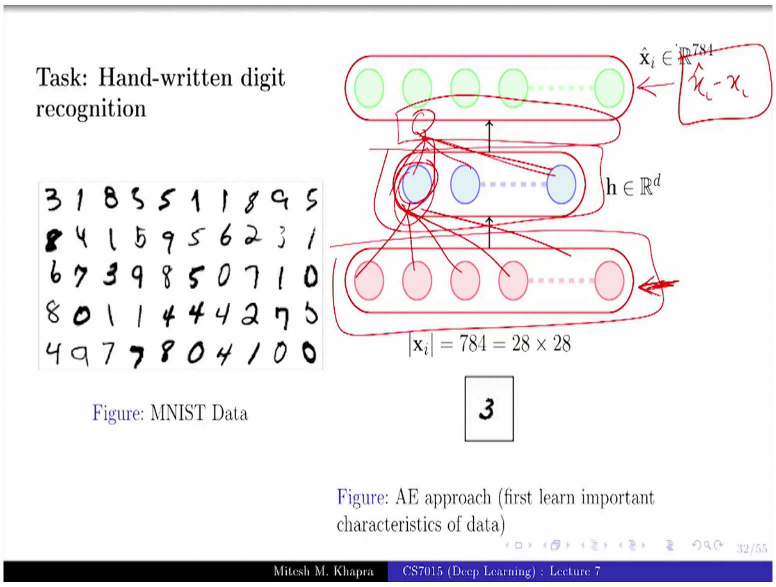
Figure: MNIST Data

Figure: Basic approach(we use raw data as input features)

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

So, suppose you are given this task, which is handwritten digit recognition I see everyone paying attention. Now I should say this before every slide. So, this is the task handwritten digit recognition you are given some data, where you want to classify the digits into one of these 10 glasses. The traditional machine learning approach to this is we just construct a feature vector this is a 28 cross 28 image. So, I guess 28 cross 28 pixels which is 784. I treat this as a feature vector and feed it to any of my machine learning algorithms say SVM or multi class SVM or logistic regression or any of these right and do a classification based on them. This is what you would have done in your machine learning course if I had given you this assignment right ok.

(Refer Slide Time: 07:25)



$\hat{x}_i \in \mathbb{R}^{784}$
$\hat{x}_i - x_i$
$h \in \mathbb{R}^d$
$|x_i| = 784 = 28 \times 28$
3

Task: Hand-written digit recognition

Figure: MNIST Data

Figure: AE approach (first learn important characteristics of data)

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

32/55

Now, the autoencoder approach or in general the deep learning approach would be you take this data which is the original feature representation that you had. There is no engineering feature engineering happening here right, ideally I want to have features of the form that if pixel 25 comma 30 was black and if pixel 30 comma 20 was also black then probably I am drawing a curve somewhere. So, it could be one of these curvy digits and not one or any of these 7 or any of these things right. So, you want to do some feature engineering.

So, typically in machine learning what you do is; you start with these 784 features you observe a few things and you have these handcrafted features added on top of these right. So, you will add some more features to the data. Now the deep learning approach is that you let you also learn the features on their own. So, how did we learn these features? We took this original input we passed it through the an autoencoder which captured some of these relevant characteristics.

The differences we do not really know what these relevant characteristics are; that means, you and I cannot read them and make sense of them. I cannot say that this pixel is actually capturing the interaction, oh sorry this neuron is actually capturing the interaction between my 700 pixel and 710 pixel, I cannot do that. I could have handcrafted those features. If I believe that all my data is around the center, I could have handcrafted some features which say that capture the interactions between those that is what you do in machine learning.

Here you are trying to learn the features also on their own right, what would happen if I add one more layer to this autoencoder? I would learn even more complex interactions between these features. So, this neuron is actually learning interactions between all the input neurons. I add one more layer here again this neuron will learn all the interactions between these abstract representations, right? So, I could learn more and more abstract representations of the input. So, I am not doing feature engineering I am just throwing data at the network and I am assuming that it will learn better and better representations, is it fine ok?

Now, I am doing this in the autoencoder setup where actually I am trying to optimize the objective function of minimizing this loss and of course, the squared of this loss is just fine. So, first what I will do is; I am not happy with my original 784 dimensions. So, I train a autoencoder to learn some k dimensions which are good. I know these are good because they are able to reconstruct the data perfectly to a certain extent right of course, because you add regularization it may not be perfect, but it captures the essence of the data, you get that.

So, I have better dash representations, now feature representations right my original feature representation was 784 I have come up with some better representations. Now what will I do; was my task to learn feature representations, what was it? Classification right, so what will I do now is I will I have learned this much from the autoencoder.

(Refer Slide Time: 10:18)

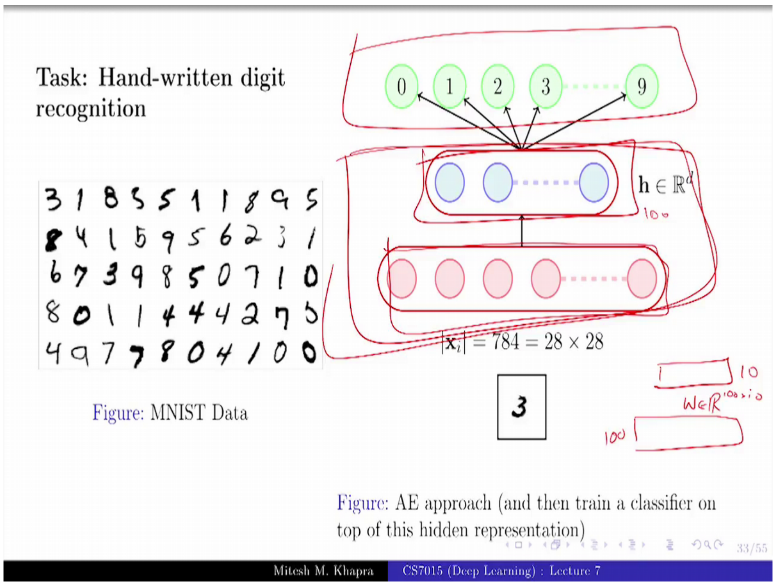

Task: Hand-written digit recognition

Figure: MNIST Data

Figure: AE approach (and then train a classifier on top of this hidden representation)

0
1
2
3
9
$h \in \mathbb{R}^d$
100
$|x_i| = 784 = 28 \times 28$
3
10
$W \in \mathbb{R}^{100 \times 10}$
100
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 7
33/55

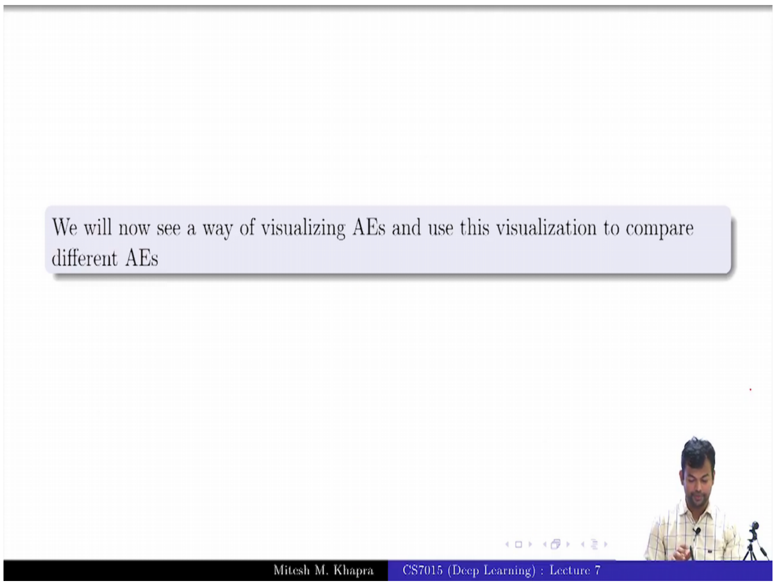
I will throw away the last layer I do not care about the last layer, what I care at the last layer is a classification problem, right? So, I will construct a new neural network, where the first 2 layers of the network are the same as what I learned from the autoencoder and on top of that I will add an output layer, and now I will try to train this network, how many of you get what is happening here? Those of you do not get it can you ask me some questions let me just try to answer on my own it is like playing chess with yourself.

So, this is my original input 784 dimensions, what I have learned with autoencoders is a smarter representation of this data ok. Now one simple solution that I have is I have this 100 dimensional data suppose this is the representation. So, for all the training examples instead of using that 784-dimension data and feeding it to a multi class SVM, what I can do is; I can first compute this 100 dimensional representation and feed that to a multi class SVM, is that fine and you see that should work better in practice because I have reduced the dimensions, I have reduced the dimensions smartly.

And now I can train this network is this fine. All I am saying is instead of a multi class SVM. I could also have a neural network right; I could feed that representation to a neural network. So, what would that neural network look like; 100 what are the parameters here? W belonging to 100 cross 10 how many if get it now ok. So, this is what I could have done.

So, I have learned a better feature representation and now I am using that representation to learn my classifier. If I do this in an end to end manner; that means, my feature representation is also came out of a neural network. And my classifier is also a neural network then I have a complete end to end solution for this you get this, fine.

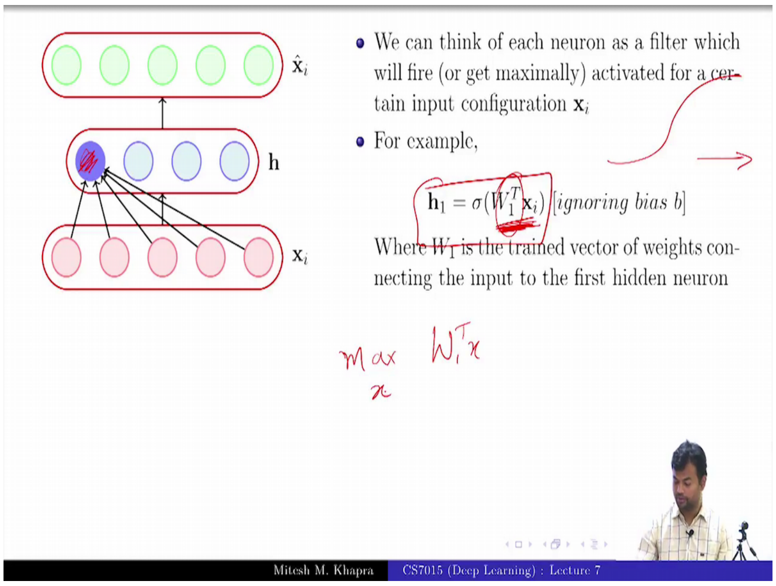
(Refer Slide Time: 12:14)



We will now see a way of visualizing AEs and use this visualization to compare different AEs

Now, we will see a way of visualizing this and then we will make some observations from the visualizations. So, first let me tell you what the visualizations is.

(Refer Slide Time: 12:23)



- We can think of each neuron as a filter which will fire (or get maximally) activated for a certain input configuration x_i
- For example,

$$h_1 = \sigma(W_1^T x_i) \; [ignoring\ bias\ b]$$

Where W_1 is the trained vector of weights connecting the input to the first hidden neuron

$$\max_{x} W_1^T x$$

So, I am returning to the autoencoder setup. So, I had this input and I had this h dimensional or k dimensional hidden layer. Now I can think of each of these neurons as something which gets activated for a particular type of input, is that fine; what do I mean by activated it is output would be?

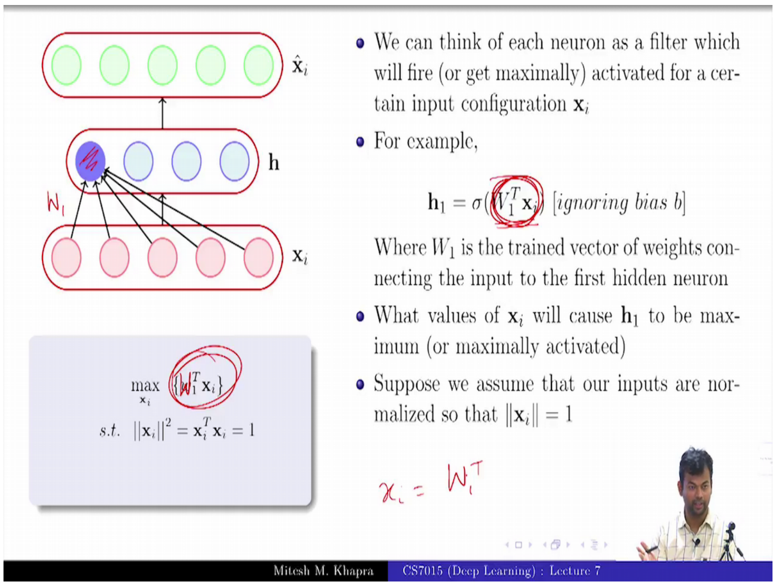
Student: 1.

Remember this is a logistic neurons that we are talking about or even tans neurons the output would be 1. So, it is the maximum output that you could gain fine. Now so for example, h 1 is equal to sigmoid of this when would this fire when where W 1 transpose x i is very high right when you are in that regime where the sigmoid flattens right, this regime ok; when it is very high.

So, I want to be able to maximize W 1 transpose x i, do you get this? I want to be able to maximize this I want to find my W 1 transpose is fixed now because I have trained the autoencoder. I have got these weights this is all post mortem, right? I have trained the autoencoder, I have got these weights. Now I want to find an input which will cause this particular neuron to fire ok.

So, what is my max what is my optimization problem maximize just help me out, maximize W 1 transpose x let me just call it x, and the optimization is with respect to x right because I want to find the x which maximizes this quantity my training is done, I do not no longer care about changing Ws my training has been done I am interested in finding xs which will maximally fire this ok.

(Refer Slide Time: 14:08)



$$\max_{x_i} \{W_1^T x_i\}$$
$$s.t. \; ||x_i||^2 = x_i^T x_i = 1$$

- We can think of each neuron as a filter which will fire (or get maximally) activated for a certain input configuration x_i
- For example,

$$h_1 = \sigma(W_1^T x_i) \; [ignoring \; bias \; b]$$

Where W_1 is the trained vector of weights connecting the input to the first hidden neuron
- What values of x_i will cause h_1 to be maximum (or maximally activated)
- Suppose we assume that our inputs are normalized so that $||x_i|| = 1$

$x_i = W_1^T$

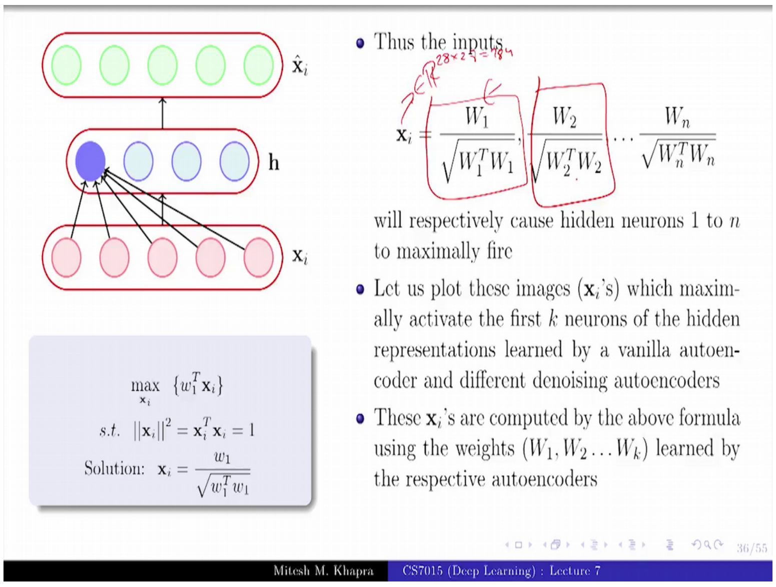
So, and I am going to assume that all my inputs are normalized this just makes some analysis easier and remember that normalization is always ok, you always do that. So, this is the optimization problem that I am interested in solving. What is the solution to this; how many if you can solve this? No I want to find the x i.

Student: (Refer Time: 14:29).

Now, I have trained the autoencoder, now I have known all these the one I am considering one column of the matrix W 1. I want to see what is the input that I should give. So, that I am sure that this neuron will get activated and I know that this neuron will get activated if I maximize this quantity right.

So, I want to maximize that quantity and find an x such that it will get maximized. I was just hoping that no one brings in eigenvectors W 1 is a column it is not a matrix just try to work it out. What is this? This is a dash between W and transpose and x i dot product, when would the dot product be maximized? When they are both in the same direction right; that means, you know the direction is going to be x i is equal to and what did I want the norm to be; how do you get it fine.

(Refer Slide Time: 15:26)



$$\max_{\mathbf{x}_i} \{w_1^T \mathbf{x}_i\}$$

$$s.t. \quad ||\mathbf{x}_i||^2 = \mathbf{x}_i^T \mathbf{x}_i = 1$$

$$\text{Solution:} \quad \mathbf{x}_i = \frac{w_1}{\sqrt{w_1^T w_1}}$$

- Thus the inputs

$$\mathbf{x}_i \overset{\in \mathbb{R}^{28\times 28 = 784}}{=} \boxed{\frac{W_1}{\sqrt{W_1^T W_1}}}, \boxed{\frac{W_2}{\sqrt{W_2^T W_2}}} \dots \frac{W_n}{\sqrt{W_n^T W_n}}$$

  will respectively cause hidden neurons 1 to n to maximally fire
- Let us plot these images ($\mathbf{x}_i$'s) which maximally activate the first k neurons of the hidden representations learned by a vanilla autoencoder and different denoising autoencoders
- These $\mathbf{x}_i$'s are computed by the above formula using the weights ($W_1, W_2 \dots W_k$) learned by the respective autoencoders

36/55

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

So, the solution is going to be this is fine W 1 by the norm of W 1. So, just remember that this quantity is going to get maximized when the dot product is maximized the dot product is maximized when both x i and W 1 transpose are in the same direction right so; that means, x i should be in the same direction as W 1 and I also wanted this constraint that x i should be the norm of x i should be 1. So, I am just dividing W 1 by the norm of W 1 this is clear to everyone how many of you get this.

So, I know now what is the input I should feed to the network so, that one of these neurons fires. Now what I am going to do is I am going to plot the xis which maximize each of these neurons I am going to consider some 100 neurons in the hidden layer. And I am trying to find out the input image which is going to maximize or which is going to cause each of these neurons to fire, do you get what I am trying to do? Even though you do not get why I am doing it, but do you get what I am trying to do.

So, what am I going to do is; this is a vector, right? So, I am just going to try to plot this as an image of the appropriate dimension.

(Refer Slide Time: 16:41)

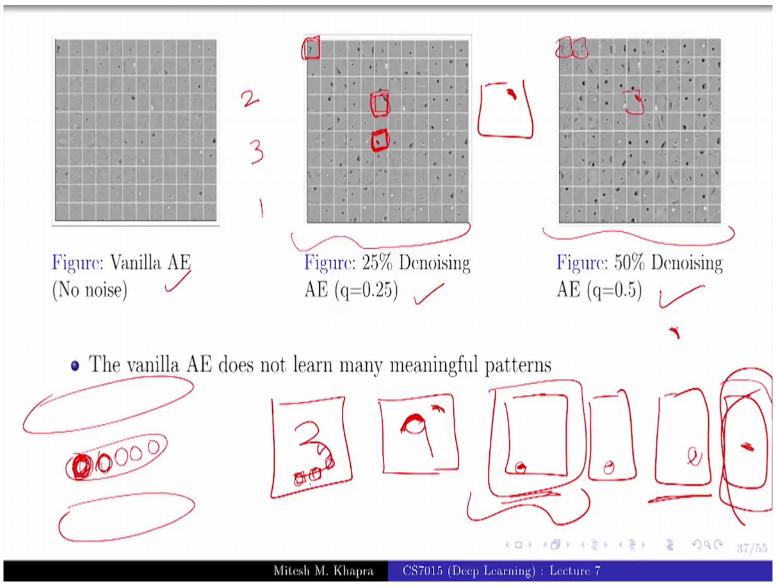


Figure: Vanilla AE (No noise)

Figure: 25% Denoising AE (q=0.25)

Figure: 50% Denoising AE (q=0.5)

- The vanilla AE does not learn many meaningful patterns

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

37/55

And this is what I get with a vanilla autoencoder there was no noise this is what I get and this is for the MNIST digit data set right. So, my data is 2 3 1 and so on digits, this is what happens when I get 25 percent nice, and this is what happens when I get 50 percent, what do you understand from these figures? Remember that each of this is the figure which caused one particular neuron to fire is that clear each of these is a trigger which caused one neuron to fire.

One image yeah one box corresponds to one column yeah. So, it is just that the dimension of the column is again 28 cross 28. So, I am just plotting it as a 28 by 28 image. So, I will just let me just clarify that is I think that is what I said yet. So, what is the dimension of this? In fact, you just know this right this dimension of this is 28 cross 28, right?

So, I can just take that vector and again plotted as a 28 cross 28 image. So, what I mean is this is 784 right. So, x i is a 784 dimensional vector I am just taking it as a 28 cross 28 image and plotting it because my inputs were actually images. So, I am just plotting those images fine. So, at least you see, what I am doing here and what I am telling you is that each of these boxes that you see corresponds to one of these images. So, I had images x 1 x 2 up to x k such that; each of these caused the k th neuron to fire. Now what are you seeing here I mean what how do you make sense of what you are seeing and remember in the mnist sorry. So, let us try to forget all this neural network and everything and let us just try to see yes the weights would be.

Student: More distinct.

No why do you say the weights are more distinct yeah, but on average you would be still reducing it right ok. So, let me just explain what is happening then we can come back to this. So now, we have this set up we had some input. We had a certain number of neurons here and then we had the output ok, this is what our neural network was trying to do.

Student: (Refer Time: 18:53).

Now, let us take this task of recognizing a digit. Now how do I actually recognize a digit if I want to distinguish between a 9 and a 3, I would try to see if there is a curve in these positions and it is not there in this, hence this is a 3 this is a 9 that is something roughly like that right. So, in other words I am now I have given delegated so; that means, what I do is I think of 3, as a combination of you get the idea as a combination of these images with these strokes right. So, this is actually this stroke this is actually this stroke, this is roughly this stroke and so on you get the idea.

So, I think of 3 as a combination of many of these strokes, right? Now what I would like is if this guy could detect one of these strokes, right? The other guy could detect one of these other strokes right. Now you see that some of these strokes are shared across digits for example, all these strokes here look at the digit 9 these strokes gives common to 3 and 9 both right, but some strokes would be missing for 3 some strokes would be missing for 9 and you would have extra strokes in both of these. So now, each of these neurons could actually recognize these strokes; then a combination of the information that each of these neurons is capturing could help me decide whether it is a 3 or a 9, how many of you get that intuition?

Student: (Refer Time: 20:23).

So, I would like each of these neurons to detect certain strokes ok; that means, I would like this neuron the first neuron to fire for an input like this, where there is a stroke at the bottom. I would like some other neuron to fire for a different input whether there is stroke here. Now can you relate this to what you are seeing in the picture, in the second and third picture; this neuron is firing for inputs which would have a stroke at the corner right and you see different neurons are firing 4 different strokes. So, each neuron is trying to capture something relevant. And together now I could combine them to get the final output, how many of you see this; how many of you do not get this?

So, to ask questions otherwise I cannot really help it, how many of you want me to go over this again? Which part yeah so let me just repeat what each of these boxes is right. So, each of these boxes is the image which causes the kth neuron to fire right. So, remember I decide I came up with this that this is the input which causes the second neuron to fire, what was the dimension of this input? 28 cross 28.

So, I am just plotting that 28 cross 28 input right, and I am realizing that this input seems to be something which has a dark spot here right? So now, just related to the analogy that I am trying to give at the bottom that this neuron fires for inputs which have a stroke here, that is that is capturing and there are other neurons which are trying to fire for other strokes.

And I would want these neurons to capture different strokes. So, that together they captured all the information in the image and helped me decide that a combination of these strokes gives me a 9 a combination of these other strokes gives me a 3 is that clear now you also.

Student: Yes, sir.

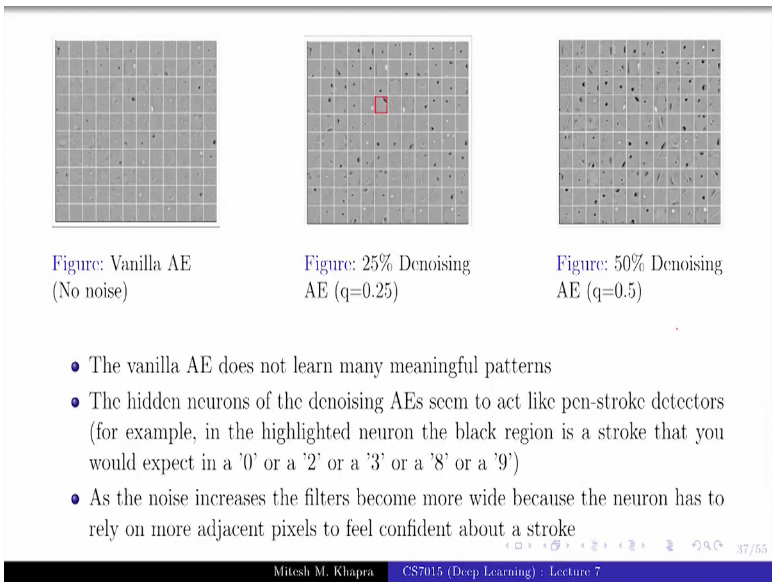
So, yeah. So now, the thing is this right the again the same thing you could learn to reconstruct the output, but you may not capture the important characteristics in the input, right? So now, as you keep making it is job harder it has to rely on capturing these important characteristics in the input, right? And actually if you look at the difference between the second figure and the third figure right let us look at the same guy here.

So, you see that this is actually thicker and wider, the stroke that you see here is thicker and wider. So now, it is actually relying on more neighborhood information to fire it is not firing just for this stroke, but it is fighting for a larger stroke it is also requires more neighborhood information because you are corrupting the pitch.

So, it has to rely on information from the other guys, the same example that I gave for height weight and body mass index right the same thing holds here. I have corrupted a lot of inputs. So now, it will fire only if it gets a lot of information from the neighboring inputs also is that fine. And I now coming back to your question, yeah I do realize now what you are saying that the weights are actually becoming larger yeah, it makes it more robust, but again.

So, regularization just does not always mean that your weights have to be small right that is one way of constraining or regularizing, but this is another way of regularizing where you are making it more robust, but it does not necessarily need to lead to the same solution where your smaller weights does that make sense, it is; for most of you any please raise your hands if this group?

(Refer Slide Time: 23:59)



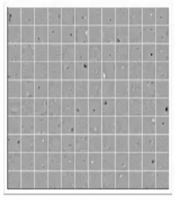


Figure: Vanilla AE (No noise)

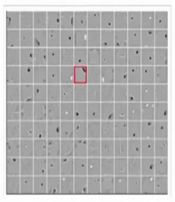


Figure: 25% Denoising AE (q=0.25)

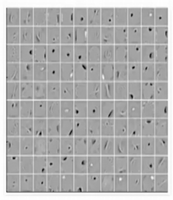


Figure: 50% Denoising AE (q=0.5)

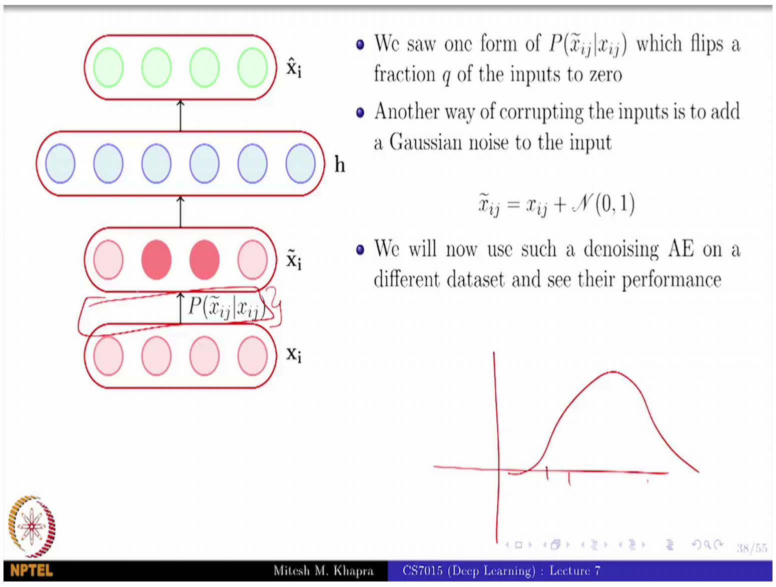
- The vanilla AE does not learn many meaningful patterns
- The hidden neurons of the denoising AEs seem to act like pen-stroke detectors (for example, in the highlighted neuron the black region is a stroke that you would expect in a '0' or a '2' or a '3' or a '8' or a '9')
- As the noise increases the filters become more wide because the neuron has to rely on more adjacent pixels to feel confident about a stroke

37/55

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

And this is same thing that I have written here ok.

(Refer Slide Time: 24:02)



- We saw one form of $P(\tilde{x}_{ij}|x_{ij})$ which flips a fraction q of the inputs to zero
- Another way of corrupting the inputs is to add a Gaussian noise to the input

$$\tilde{x}_{ij} = x_{ij} + \mathcal{N}(0, 1)$$

- We will now use such a denoising AE on a different dataset and see their performance

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7 38/55

Now, we saw one form of this function ok, which was just flip the input if the output is just corrupt the input right, you could also add a Gaussian noise. So, you could take the input add a Gaussian noise to it with 0 mean and then again try to reconstruct the original input back is that fine. So, you could just use different noise functions to do this. So, we will now see such a denoising autoencoder where we have actually added a Gaussian noise, instead of the 0 1 noise or the corruption that we were doing, right?

(Refer Slide Time: 24:39)

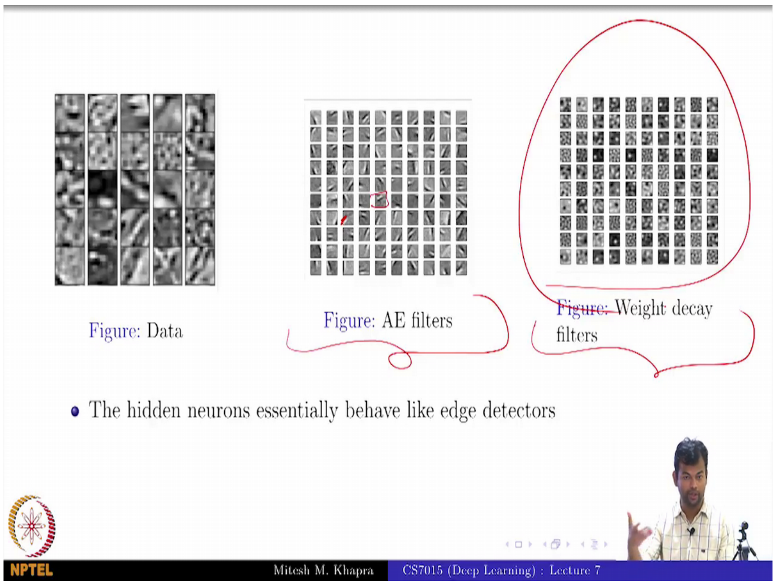


Figure: Data

Figure: AE filters

Figure: Weight decay filters

- The hidden neurons essentially behave like edge detectors

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 7

Yeah so the purpose of this particular example that I am giving is to compare an autoencoder which is regularized by adding this Gaussian noise, with an autoencoder which is regularize by using weight decaying right the l 2 regularization. So, l 2 regularization is also known as weight decaying because you kind of decay the weights right you force the weights to be small.

So, what they showed is that with Denoising Autoencoder using a Gaussian noise, you actually learn something known as edge detectors right. So, you see all of these are trying to detect edge, again the same thing is happening I am plotting the images which will maximally cause a particular neuron to fire and it looks like all these neurons fire for different edge patterns in your original data.

So now they are capturing all the edges in the data, and the combination of these edges should tell you what your final class is. And this seems to work much better than the weight decay filter which is not really capturing any regular pattern ok. So, this is just an empirical evidence that an autoencoder with a Gaussian noise seems to do better than autoencoder with the l 2 regularization ok.