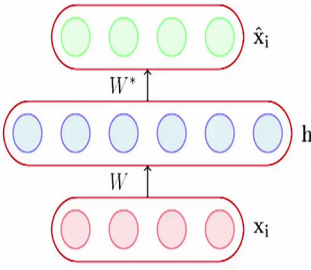


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 7.3
Lecture - 07
Regularization in autoencoders (Motivation)

Then we will go to the next module, where we will talk about Regularization in auto encoders, and we will talk about a Motivation for doing that.

(Refer Slide Time: 00:22)



- While poor generalization could happen even in undercomplete autoencoders it is an even more serious problem for overcomplete auto encoders
- Here, (as stated earlier) the model can simply learn to copy x_i to h and then h to \hat{x}_i
- To avoid poor generalization, we need to introduce regularization

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

So, poor generalization so, why do we need a regularization? People have done the machine learning course or any equivalent course, why do we need regularization? To avoid.

Student: Or enable.

Or enable generalization, right? Now in the case of an over complete auto encoder what is likely? Overfitting is likely, why is it so? What does what do you mean when you see generalization actually, when you talk in terms of training time test time and so on?

So, generalization is essentially that your are training. So, remember that at training time you are trying to solve an optimization problem, guyse you are looking only at the training data. So, it is quite likely that you will drive the error to 0 for the training data;

that means, you have learnt perfectly everything for the training data, right? But now it is also possible that when I give you a new test instance which you had not seen during training; that means, you had not seen instance while doing the optimization; that means, this instance did not contribute to your loss function.

Then it is very lightly that when I gave this instance, then you would get a non 0 loss or a loss much higher than what you get for your training data. Does that make sense? That is what over fitting is and it leads to less generalization. Your model should have generalize to unseen data, but it cannot do this one typical situation, where over or where generalization happens is, if you have a dash number of parameters. Now what did I ask actually?

Student: Generalization.

No, ok, if a case where a over fitting would happen is when you have a dash number of parameters.

Student: Large number of.

Large number of parameters, right; now do you see why I am saying this? What is there on the slide? An over complete auto encoder, what would it have?

Student: A large number.

A large number of parameters. So, what could it do?

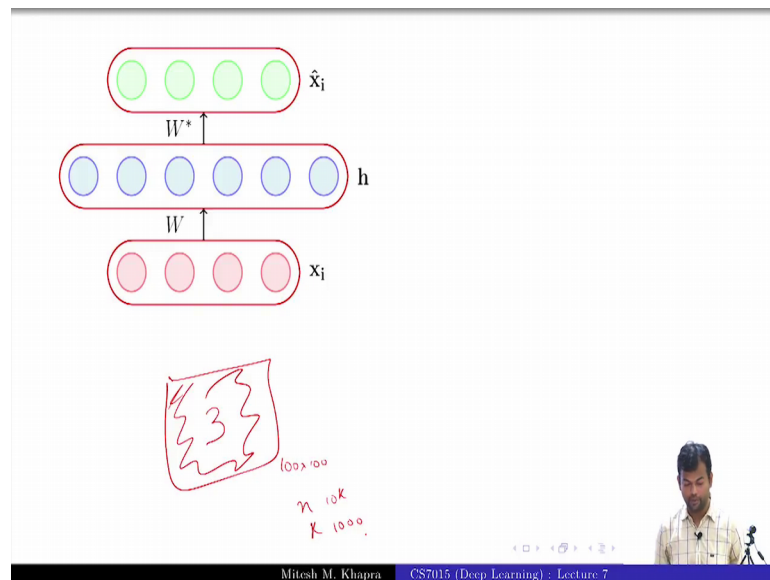
Student: Overfitting.

Over fitting, what do we do to avoid over fitting?

Student: Regularization.

Regularization, ok so that is why we need regularization. I have still no told you why do we need an over complete auto encoder, ok. Still that is an random variable I still need to decide. But can this happen in an under complete auto encoder also it can right, because under complete auto encoder just says that your k is less than n , it does not say how much less it is. So, it is it is still have and depending on a data that you are trying to model, it could still have a large number of parameters.

(Refer Slide Time: 02:58)



So, for example, let us take an example for the under complete case, suppose you are doing image classification where you have a digit 3 at the center of the image, ok. And a lot of these are white spaces. So, what is the dimension? And suppose this is a 100 cross 100 image. What is the dimension of this image input? How many if you cannot multiply 100 into 100?

Student: 10.

10 k right of this a lot of data is not important, right? So, my n is 10 k and at least by this thing that I have drawn it looks like probably only 20 percent of that is what actually captures the digit, but now if I choose k to be equal to 1000. It might still be large for this application. So, I am using an under complete auto encoder, but it could still be a situation that my under complete is still having a large number of parameters, all of get this intuition?

It is a very weird example, but still really do you get the intuition you could have a very high dimensional input, and you might think you are shrinking it a lot. But there is so much redundancy in your input that even that shrinking still leads to a large number of parameters and you could still over fit. Every gets the idea? Therefore, even for an under complete auto encoder, you could still need over a regularization, is that ok?

So, fine so, that was the motivation, since the over complete case of course, the model can simply learn to copy, we have seen that, and that is why we need to introduce generalization, fine. Now what is the simplest, sorry, we need to introduce a regularization, what is the simplest regularization technique that you know? That is not the simplest L_2 regularization.

And you see why I say that is the simplest, we can take the derivative for those of you do not get it do not worry we will get to it. Or if you do not get to it do not worry.

(Refer Slide Time: 04:41)

• The simplest solution is to add a L_2 -regularization term to the objective function

$$\min_{\theta, w, w^*, b, c} \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n L(\theta) + \lambda \|\theta\|^2$$

$\theta = (w_1, w_2, \dots)$

$$\frac{\partial L(\theta)}{\partial w_1}$$

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

So, the simplest solution is to add the L_2 regularization to the objective function. So, this was my objective function, I wanted to minimize the squared error loss. I have added a term to this. What does this term do? What does it doing? First of all, tell me what is this quantity, theta is a.

Student: All.

All the parameters that you have; and I am assuming that they have just put it into large vector I am taking the L_2 norm of that vector. So, even you though you have those matrices, and just flattening them all out and putting them into a large vector called theta, right. So, what is happening here? I am not allowing my weights to shrink or grow, grow, because if my weights are very large what would happen?

Student: Grow, grow.

This quantity would grow. So, then I cannot really minimize this minimize this as effectively as I want, right? Why this makes sense? How many of you why this makes sense? So, I am now why am I not preventing the weights to go to 0? Ok, so, we will see this in more detail in the next lecture. This is again a basic lecture on bias variance and regularization and so on. So, we will try to arrive at a more reasonable answer for this. For now, just see that I am putting some constraints and the weights.

So, effectively and I am doing gradient descent, I am not allowing the weights to take very large values, I am trying to restrict them to a certain area. So, I am not allowing it to explore the entire w comma b plane, but trying to restrict it to smaller values of w comma v how many of you get this intuitive explanation.

So, in other words what I am trying to do is, that I am not giving it in a freedom so that it can completely drive the error on the training data to 0. And my hope is that if I do not do this, if I do not allow it to completely memorize a training data, then it should generalize well on the test data. Is that intuitive fine? Now I have changed the loss function again. I have the square, I have told you how to do it for squared error loss, for the cross entropy loss and so on, but now I have changed the a loss function again. So, again I need to teach you back propagation, no what will change now? Again I need to derive with respect to the last layer.

What is the minimalistic change that is going to happen now? Just tell me, this θ is actually w_1 w_2 and so on, right. Just assume all the parameters, just flattened out into a vector, fine? And now tell me what is $\frac{dL}{d\theta}$ by $\frac{dL}{dw_1}$ going to be or let us simplify things. Let us call this L θ and let us call this ω θ . Let us call this L dash θ , and then your L θ is the combination of these 2 terms. So, this derivative is going to be a sum of 2 derivatives, out of that one you already know, what is the second?

Student: 2 times lambda.

Two times lambda w_1 . So, it is a very simply change to your gradient descent update rule. How many of you see that? Whatever update you will had just add minus 2 lambda w_1 to that, ok, should have been 2 lambda w , but of course, you do a half here, so it is fine, is it?.

(Refer Slide Time: 07:44)

The diagram shows an autoencoder architecture. At the bottom is the input layer x_i (red circles). Above it is the hidden layer h (blue circles). At the top is the reconstructed output \hat{x}_i (green circles). Weights W connect the input layer to the hidden layer, and weights W^* connect the hidden layer to the output layer.

- The simplest solution is to add a L_2 -regularization term to the objective function

$$\min_{\theta, w, w^*, b, c} \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n (\hat{x}_{ij} - x_{ij})^2 + \lambda \|\theta\|^2$$

- This is very easy to implement and just adds a term λW to the gradient $\frac{\partial \mathcal{L}(\theta)}{\partial W}$ (and similarly for other parameters)

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

(Refer Slide Time: 07:53)

The diagram is similar to the previous one, but with handwritten annotations. A red arrow points from the weight matrix W to W^* , indicating they are tied. Below the diagram, handwritten notes show a matrix W with dimensions $n \times k$ and its transpose W^T with dimensions $k \times n$.

- Another trick is to tie the weights of the encoder and decoder i.e., $W^* = W^T$

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

Another trick which is typically used at least in the context of auto encoders is to tie the weights of the encoder and the decoder. How does that help? What does tying the weights mean? Now I appreciate what you are trying to say. So, one we have doing this is just say W star is equal to W transpose. You will enforce that, you actually have only one matrix W , and here you are using W transpose. Mathematically does that make sense? All your operations go through, because this is going to be n cross k , and this is going to be k cross n .

So, whatever effectively done, I have reduce the number of parameters in my network, right? I am enforcing, I am forcing this upon the network that I am not going to give you 2 sets of weights, you just learn the ws in a way such that when you use W transpose you should be able to reconstruct this. How many of you get this? Not many, ok, please ask me doubts if you do not, there is nothing very.

Student: Why is it W transpose?

Why is it W transpose? Because otherwise.

Student: (Refer Time: 08:52) Claim that.

How can you claim that that would work? Because you have no linearity's in between right no W inverse would not work, what is the simplest thing to do? Why would you want to compute an inverse? That is an interesting question how would you implement this? How would you, if there are multiple paths from a weight to the output, how do you compute the gradient? Sum it across all those paths, what is happening here? How many paths to there exist from the weight to the output? One is this direct path and then the other is another this path also. So, you just sum it across these 2 paths. Do you get that? How many of you do not get that? How many of you do not get that.

So, if this was W star you did not have a problem? You could just have computed $\text{dout} \cdot 1$ by $\text{dout} \cdot W$ star and $\text{dout} \cdot 1$ by $\text{dout} \cdot W$. Now think of it as this, right that you have this, this is one path W , W to the output, ok. And now the gradient is just going to be sum across these 2 parts, one path is the single path and the other path is the double path. So, it is just going to be a sum across these 2 paths, oh, no, no. So, you just have one matrix W which are going to update. You do not have 2 matrices, you just have one matrix W , at 1 place you are using W , the other place you are using W transpose. But just look at it element wise right, do not try to look at it in the terms of matrices.

So, you have n cross k elements here w_{11} to w_{nk} , you have to computing the partial derivative with respect to each of these, and every time they are considering all possible paths to the output, and that value is getting updated, right? And at 1 place you are using a particular arrangement of these W 's at the other place you are using a different arrangement of those W 's, that so, it will just remain the same? Is that ok?

Student: No.

No, this is for regularization, right. So, we are reducing the number of parameters by half.

Student: (Refer Time: 10:51).

Yes.

Student: (Refer Time: 10:52).

No, that I mean that also has, but that is not the objective, we are trying to do regularization. How many of you have lost at this point? Please ask me if you have questions, really I do not mind answering. But if you just give me blank spaces, I cannot read them. So, this is used at quite a few places where you tie some weights right so that; so, effectively you are saying that learn it in such a way that it works at both the places. And you are reducing the number of parameters. So, weight tying is something which is very commonly used for regularization in the context of neural networks.

So, that is where we will end the motivation part, and it is too very simple ways of doing regularization. One is the standard known trick which is to use l-2 regularization, and the other one was something special that we saw which was tying the weights you all have a lot of doubts about tying the weights. Do not let that affect the rest of the lecture just throw it away. It is if you do not understand it. May be go back and think about it and then we discuss again if you want, ok.

So, do not, it is a very small thing, do not let it affect your rest of the lecture.