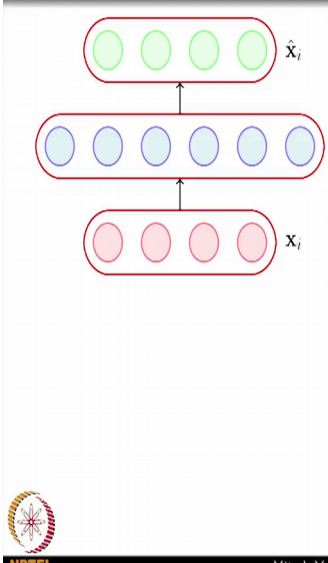


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 7.5
Lecture – 07
Sparse Autoencoders

So, in this module we will talk about Sparse Autoencoders.

(Refer Slide Time: 00:17)



The diagram illustrates a sparse autoencoder neural network. It consists of three layers of neurons, each enclosed in a red rounded rectangle. The bottom layer, labeled x_i , has four pink neurons. The middle layer, labeled h , has five blue neurons. The top layer, labeled \hat{x}_i , has four green neurons. Arrows indicate the flow of information from the input layer to the hidden layer, and from the hidden layer to the output layer. To the right of the diagram is a list of three bullet points explaining the properties of a hidden neuron with sigmoid activation.

- A hidden neuron with sigmoid activation will have values between 0 and 1
- We say that the neuron is activated when its output is close to 1 and not activated when its output is close to 0.
- A sparse autoencoder tries to ensure the neuron is inactive most of the times.

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

Just some concepts before we jump into the actual way of doing this. So, hidden neuron with sigmoid activation will have values between 0 to 1 and you say that the neuron is activated when this output is close to 1 and it is not activated when its output is close to 0 ok. Now, a sparse encoder tries to ensure the neuron is inactive most of the times, what is that mean?

Student: Close.

It is close to 0 for.

Student: Most of the.

Most of the.

Student: (Refer Time: 00:47).

Inputs right. So, I am passing a lot of inputs to it, it will try to ensure that it is close to 0 for most of the inputs. So, in other words what does it trying you ensure? I am looking for the word average. The average activation of a neuron is close to 0 does that make sense is that fine ok.

(Refer Slide Time: 01:06)

• If the neuron l is sparse (i.e. mostly inactive) then $\hat{\rho}_l \rightarrow 0$

• A sparse autoencoder uses a sparsity parameter ρ (typically very close to 0, say, 0.005) and tries to enforce the constraint $\hat{\rho}_l = \rho$

• One way of ensuring this is to add the following term to the objective function

The average value of the activation of a neuron l is given by

$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^m h(x_i)_l$$

$$\Omega(\theta) = \sum_{l=1}^k \rho \log \frac{\rho}{\hat{\rho}_l} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_l}$$

$\mathcal{L}(\theta) = \mathcal{L}'(\theta) + \Omega(\theta)$

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

So, this is on, what you see on the left hand side, this is how you would compute the average activation of a given neuron, you have all the m examples you see what the activation was for each of these and take the average right.

Now, if the neuron is sparse then the average activation would be close to 0 is that fine, this is all just different ways of saying the same thing. Now, a sparse encoder uses a sparsity parameter say ρ and it is very close to 0 say 0.005.

And it tries to enforce the constraint that on average the activation of any neuron in the hidden layer should be equal to ρ , which is again close to 0. Now, can you think of a, this is all fine in plain English right, you understand what we are trying to do. First of all tell me why does this makes sense? What is it that you are trying to ensure? Over fitting happens because there is lot of dash.

Student: Parameters.

Parameters slightly abstract it out.

Student: Memorization.

Lot of?

Student: Memorization.

Memorization ok, lot of freedom right, I mean the weights have a lot of freedom to move where ever they want to do, whatever they want to do such that they can just drive the training error to 0. What have we done to that freedom now?

Student: We are restrict. (Refer Time: 02:22).

We are restricting them. So, any kind of regularization always tries to restrict this freedom that the parameters or the network have in general right, and there are different ways of restricting this freedom. You see that this is one of those ways right; you are trying to ensure that on average the neuron should not fire. So, it is clear that this some kind of regularization, any one has a doubt with that? No.

Now, the second question is taking slightly more on this right, it is I can just move ahead and I have convince you that this is regularization. But can you think of bit more and see what is actually being tried to achieve here, what are we trying to do? How many of you get that or at least could here that first of all, only the second row ok. So, yeah how many of you can think about this, like what is it trying to achieve?

Student: (Refer Time: 03:13).

Right. So, on average neuron is going to be inactive, that means, where ever it is active it is really going to capture some relevant information, right. So, it is going to be active whenever it is active it is going to adhere to certain patterns. So, we are ensuring that each of these neurons are just a very few patterns and it has discriminative power in that sense, do you get that?

So, now if; that means, if I show it a 3, if I show it a 2, if I show it a 1 every time if the neuron fires, when there is no discriminative power in that. But now, if I ensure that the neuron fires only a few times it will try to fire for meaning full patterns. So, it will try to fire for a curve or a curve in the between as you have it in the case of 3 right, you have this cusp in the between, in the middle. So, it will fire for some kinds of pattern.

So, that is what the hope is, it is not just like adding some math and adding some regularization, but at least there is some intuition behind that, how many of you get that intuition? Ok good. And now can tell me a way of putting this, everything English is fine, intuition is fine, but how do convert this to a mathematically equation?

You want to ensure that $\hat{\rho}$ is equal to ρ , there will of course, be different ways of doing this, the way these guys do it by adding this term to the loss function. So, remember your loss function is always going to be $L(\theta)$ plus $\omega(\theta)$ right. Where $\omega(\theta)$ does the regularization and $L(\theta)$ is your regular loss, which would be the squared error loss or the cross entropy loss or whatever loss you are dealing with right.

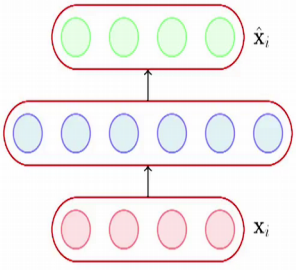
So, remember this term is always there, but the reason I do not bring it up so often is because we have already dealt with it. We know how to compute the gradients, we know how to do the back propagation and all that. And now since your final loss is just a sum of these two terms, I know how to deal with this and I know that gradients are additive so I just need to deal with the second term. That is why I am only focusing on $\omega(\theta)$, $L(\theta)$ has been dealt with, is that fine ok.

Now, this is what $\omega(\theta)$ is, why does this make sense? When would this take its minimum value when ρ is equal to?

Student: Watt.

Watt, everyone sees that how many of you sees that? Please raise your hands ok, fine let us plot it and check actually, right.

(Refer Slide Time: 05:23)



- If the neuron l is sparse (i.e. mostly inactive) then $\hat{\rho}_l \rightarrow 0$
- A sparse autoencoder uses a sparsity parameter ρ (typically very close to 0, say, 0.005) and tries to enforce the constraint $\hat{\rho}_l = \rho$
- One way of ensuring this is to add the following term to the objective function

$$\Omega(\theta) = \sum_{l=1}^k \rho \log \frac{\rho}{\hat{\rho}_l} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_l}$$

- When will this term reach its minimum value and what is the minimum value? Let us plot it and check.

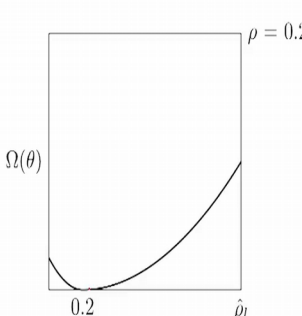
The average value of the activation of a neuron l is given by

$$\hat{\rho}_l = \frac{1}{m} \sum_{i=1}^m h(\mathbf{x}_i)_l$$

42/55

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

(Refer Slide Time: 05:27)



- The function will reach its minimum value(s) when $\hat{\rho}_l = \rho$.

43/55

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

So, this is how that function looks like. So, I have plotted the function which I have written here for a of course, a single k right and my ρ that I have taken is 0.2 and if I plot that function for different values of ρ hat l , it will reach the value 0 only when ρ hat l is equal to ρ . So, again go back and plot this and check and it is actually clear from the equations itself that it will be minimized only when ρ hat l is equal to ρ , right everyone gets this? Fine.

So; that means, this is a genuine, I mean this is a reasonable thing to do, we would think of other ways of doing that and I am sure you can, but this is also a reasonable way of doing this fine ok.

(Refer Slide Time: 06:08)

$$\Omega(\theta) = \sum_{i=1}^k \rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i}$$

Can be re-written as

$$-\Omega(\theta) = \sum_{i=1}^k \rho \log \rho - \rho \log \hat{\rho}_i + (1-\rho) \log(1-\rho) - (1-\rho) \log(1-\hat{\rho}_i)$$

By Chain rule

$$\frac{\partial \Omega(\theta)}{\partial W} = \frac{\partial \Omega(\theta)}{\partial \hat{\rho}_i} * \frac{\partial \hat{\rho}_i}{\partial W}$$

$$\hat{\rho}_i \rightarrow h_i \rightarrow W$$

- Now,
- $\hat{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \Omega(\theta)$
- $\mathcal{L}(\theta)$ is the squared error loss or cross entropy loss and $\Omega(\theta)$ is the sparsity constraint.
- We already know how to calculate $\frac{\partial \mathcal{L}(\theta)}{\partial W}$
- Let us see how to calculate $\frac{\partial \Omega(\theta)}{\partial W}$.

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

So, now our last function is as I said it is going to be a combination of two values, \mathcal{L} theta is a normal squared error loss that we have been dealing with and Ω theta is this sparsity constraint that you have added ok.

Now, you already know how to calculate the first term, what are we interested in now? So, you see that this pattern will keep repeating right so now, you can do whatever you want your loss function. You have this generic frame of called the back propagation algorithm and you know that a last part of that back propagation algorithm is going to remain the same, right. Only thing you are changing is the output layer or the loss function.

Just need to compute something there and the rest of it will remain the same how many of you get this general idea? And also appreciate it right. That is why this is a very powerful frame right, you can just make minor tweaks at the top and you are rest of the code has to remain the same.

So, you can actually go back and try out these regularization terms in (Refer Time: 06:58) assignments right, if you really want to see what happens. So, now, this is what

omega theta is and now what I am going to do? It can be rewritten as this, that is obvious just expanding out the law of function ok.

And by chain rule this is what I get, now unfortunately the rest of the slide there is an error the tsp is note this I can kind of overlooked this ah, but I will just convey the idea, right. So, you would want to do something of this sort, everyone agrees with that, remember what is rho hat? It depends on, sorry rho hat l it depends on.

Student: (Refer Time: 07:36).

h of l it is the average activation of the lth neuron and this depends on some of the weights. So, that is why this chain rule makes sense and now how to compute this? There is an error on this slide but you have done enough gradients in the class for me to have confidence that you can do it on your own, everyone is confident that they can work it out on your own? Ok.

(Refer Slide Time: 07:58)

$$\Omega(\theta) = \sum_{i=1}^k \rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i}$$

Can be re-written as

$$-\Omega(\theta) = \sum_{i=1}^k \rho \log \rho - \rho \log \hat{\rho}_i + (1-\rho) \log(1-\rho) - (1-\rho) \log(1-\hat{\rho}_i)$$

By Chain rule

$$\frac{\partial \Omega(\theta)}{\partial W} = \frac{\partial \Omega(\theta)}{\partial \hat{\rho}} * \frac{\partial \hat{\rho}}{\partial W}$$

$$\frac{\partial \Omega(\theta)}{\partial \hat{\rho}} = -\frac{\rho}{\hat{\rho}} + \frac{(1-\rho)}{1-\hat{\rho}}$$

For each neuron $l \in 1 \dots k$ in hidden layer, we have

$$\frac{\partial \hat{\rho}_l}{\partial W} = \mathbf{x}_i (g'(W^T \mathbf{x}_i + \mathbf{b}))^T$$

Finally,

$$\frac{\partial \mathcal{L}(\theta)}{\partial W} = \frac{\partial \mathcal{L}(\theta)}{\partial W} + \frac{\partial \Omega(\theta)}{\partial W}$$

and we know how to calculate both terms on R.H.S)

- Now,
- $\mathcal{L}(\theta)$ is the squared error loss or cross entropy loss and $\Omega(\theta)$ is the sparsity constraint.
- We already know how to calculate $\frac{\partial \mathcal{L}(\theta)}{\partial W}$
- Let us see how to calculate $\frac{\partial \Omega(\theta)}{\partial W}$.

NPTEL | MITESH M. KHAPRA | CS7015 (Deep Learning) : Lecture 7

So, I will skip this, we will fix these errors there are some summation and other terms missing here and the second part is actually correct which has been derived on the next slide.

(Refer Slide Time: 08:05)



Derivation

$$\frac{\partial \hat{\rho}}{\partial W} = \left[\frac{\partial \hat{\rho}_1}{\partial W} \quad \frac{\partial \hat{\rho}_2}{\partial W} \quad \dots \quad \frac{\partial \hat{\rho}_k}{\partial W} \right]$$

For each element in the above equation we can calculate $\frac{\partial \hat{\rho}_l}{\partial W}$ (which is the partial derivative of a scalar w.r.t. a matrix = matrix). For a single element of a matrix W_{jl} :-

$$\begin{aligned} \frac{\partial \hat{\rho}_l}{\partial W_{jl}} &= \frac{\partial \left[\frac{1}{m} \sum_{i=1}^m g(W_{:,l}^T \mathbf{x}_i + b_l) \right]}{\partial W_{jl}} \\ &= \frac{1}{m} \sum_{i=1}^m \frac{\partial \left[g(W_{:,l}^T \mathbf{x}_i + b_l) \right]}{\partial W_{jl}} \\ &= \frac{1}{m} \sum_{i=1}^m g'(W_{:,l}^T \mathbf{x}_i + b_l) x_{ij} \end{aligned}$$

So in matrix notation we can write it as :

$$\frac{\partial \hat{\rho}_l}{\partial W} = \mathbf{x}_l (g'(W^T \mathbf{x}_l + \mathbf{b}))^T$$


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

But I would not go over this, this is there are the slides again go back and look at it. How many of you are confident that you can do this on your own? Please raise your hands yeah because we have done enough of this in class right.

So, you can, you should be able to it no if you are not able to do it then I am not doing a good job at teaching you right. So, you should be able to do it now fine and we will fix these errors; so, TA just remind me after the class. So, everyone gets the general idea, you find a loss, you find a constraint, you define it with omega theta, find out the derivative of that with respect to your parameters and just change your gradient descent upgrade tool accordingly right, is that fine? Ok.