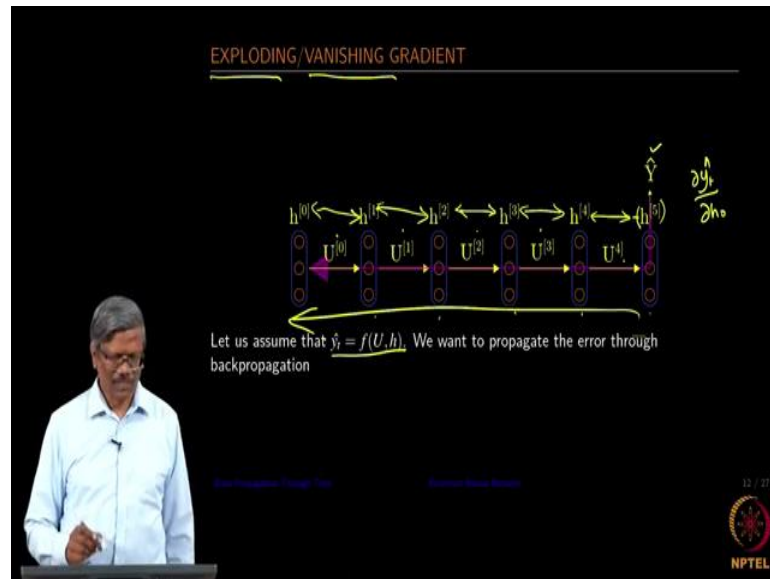


Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture - 59
BPTT - Exploding and Vanishing gradient

(Refer Slide Time: 00:15)



So, we have started with a (Refer Time: 00:17) saying that RNN can solve a lot of problems. Yes, it is a wonderful network, it is able to solve a lot of interesting problems in the time series which the normal neural net could not solve. So, it has come with a lot of promise and when you start using the network for a longer time, you start realizing that there are some problems also in the network and then people when they started implementing the RNN for the various problem they started facing a lot of issues. So, the one is the exploding of the values and then the second one is vanishing of the values.

So, that means, explode meaning the value goes beyond a point and then the computer responds with a not a number. Vanishing means it becomes extremely small very close to 0 but not equal to 0 when that happens no action takes place within the neural network because for you to really move faster in the training, you need a good gradient. If you do not have a good gradient the training stops. So, it will go on and on and on and it will not know what is going to happen, and then people realize that there are 2 inherent problems in the recurrent neural network one is the exploding gradient, the second one is the vanishing gradient.

So, let us see how it occurs right. So, since you are going to be designing the network for various purposes we should understand these two difficulties that we would face in the design of a neural network and that should be addressed so, that we have a very smooth training process ok. So, is for this case let us assume this function ok.

So, we are going to be predicting this value and then there are layers, that compute the values as you move forward and then the error has to be computed using the backpropagation and then the values of h_5 depend on h_4 and then h_3 ; h_4 depends on h_3 and h_3 depends on h_2 ; h_2 depends on h_1 and h_1 depends on h_0 ok. So, this is the arrangement that you saw in the recurrent neural network as well as rights.

(Refer Slide Time: 03:00)

BPTT - UNROLLED RNN

The error for the entire duration of T for all the vocabulary is the sum of all the error across the layers

$$E(\theta) = -\frac{1}{T} \sum_{j=1}^T \sum_{i=1}^{|V|} y_{i,j} \log(y'_{i,j}) \quad (12)$$

This term (\cdot) is known as the perplexity. Lower the value of $2^{E(\theta)}$ better is the confidence of the network in predicting the next word

NPTEL

In the middle layer right this is what we saw; they are connected. So, what happens when these things are connected in this fashion in the time series? So, we are going to be finding the error with respect to let us say you h_0 right; that means, I have to find the derivative for this, derivative for this, derivative for this, derivative for this, derivative for this and so, on right in this direction correct.

(Refer Slide Time: 03:49)

EXPLODING/VANISHING GRADIENT

$$\frac{\partial E^{[5]}}{\partial h^{[0]}} = \frac{\partial E^{[5]}}{\partial h^{[5]}} \times \frac{\partial h^{[5]}}{\partial h^{[4]}} \times \frac{\partial h^{[4]}}{\partial h^{[3]}} \times \frac{\partial h^{[3]}}{\partial h^{[2]}} \times \frac{\partial h^{[2]}}{\partial h^{[1]}} \times \frac{\partial h^{[1]}}{\partial h^{[0]}} = \frac{\partial E^{[5]}}{\partial h^{[5]}} \prod_{t=1}^{5} \frac{\partial h^{[t]}}{\partial h^{[t-1]}} \quad (15)$$

Generalizing

$$\frac{\partial E^{[\tau]}}{\partial h^{[0]}} = \frac{\partial E^{[\tau]}}{\partial h^{[\tau]}} \prod_{t=1}^{\tau} \frac{\partial h^{[t]}}{\partial h^{[t-1]}} \quad \text{where } \tau \text{ represents depth of the layers} \quad (16)$$

14 / 27
NPTEL

So, how do we find this? Again now we go to the chain rule and then try to find out how you E by dou h naught can be found outright.

So, we start with dou E by dou h right to update the weight that connects the Y and this and then you h 5 since dou h 5 depends on this. So, we have this partial derivative and then we have to get the partial derivative of dou h 4 with respect to dou h 3 and then dou h 3 with respect to this and so, on right so; that means, dou E 5 by dou h naught is what we want to achieve and we have used the chain rule to split this into various partial derivatives and then if you look at this you can just take this out and then rest you can put it in the product form right; dou e 5 by dou h 5 this is the first one that you can take it out and then where t equal to 1 to 5, dou h t by dou h t minus 1 is what we have received ok.

And then if you consider for the entire state for example, tau is going to be the total number of states that you are going to have in the neural network, then that many derivatives you have to make and that is represented by the generalized equation of this time ok. So, we have a generalized equation that represents the error with respect to the first one.

(Refer Slide Time: 05:39)

EXPLODING/VANISHING GRADIENT

$$\frac{\partial E^{(t)}}{\partial h^{(0)}} = \frac{\partial E^{(t)}}{\partial h^{(t)}} \prod_{\tau=1}^{t-1} \frac{\partial h^{(\tau)}}{\partial h^{(\tau-1)}} \quad (17)$$

$$h^{(\tau)} = \sigma(WX^{(\tau)} + Uh^{(\tau-1)}) \quad (18)$$

$$\frac{\partial h^{(\tau)}}{\partial h^{(\tau-1)}} = \text{diag}(\sigma'(WX^{(\tau-1)} + Uh^{(\tau-1)}))U \quad (19)$$

where σ' computes element-wise the derivative of σ

$$\frac{\partial h^{(\tau)}}{\partial h^{(\tau-1)}} \text{ is a Jacobian } \quad (20)$$

$$\therefore \frac{\partial E^{(t)}}{\partial h^{(0)}} = \frac{\partial E^{(t)}}{\partial h^{(t)}} U^{\otimes t} \prod_{\tau=1}^{t-1} \text{diag}(\sigma'(WX^{(\tau-1)} + Uh^{(\tau-1)})) \quad (20)$$

U gets very small when the depth increases¹

¹Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013 - <http://proceedings.mlr.press/v28/pascanu13.pdf>

So, what is the problem in this let us see here ok? So, we know that this h going back to what where we started right. So, we know that h t is equal to the linear com the sum of the linear combinations of WX t and U h that is you have the weight connecting the previous state and the current state, you know the dot product of those and then sum it with the linear combination of the weight connecting the hidden unit and the input unit and then you take this sigma of that ok.

Let us not worry about whether it is a hyperbolic tangent or sigma, let us called it as is when you take the derivative of this that is dou h with respect to dou h t minus 1 state, we get our diagonal; why we get diagonals? So, it is going to be like this, only these elements will be available these would be 0. So, when you take the derivative of the second term there is no second term and this with respect to t; correct.

So, these values would become 0 here, the same fashion ok. So, so you get a diagonal value and then when you look at the values here, this dou h t by dou h t minus 1 state it is nothing but the Jacobian matrix consisting of the derivatives of this and then the diagonal elements will have some values and rest would be 0 ok. So, that is why we write this as diagonal values and then you take the U out of that because it is going to be U into h t minus 1 when you do it with respect to this, this goes out and when you represent it with respect to the error that we computed.

So, we have to write it as $\frac{\partial E}{\partial h}$ and then every time when you do the backpropagation this value gets multiplied ok.

So, if we have t states or τ states it is going to be $\tau - 1$. This is $U^{\tau - 1}$ ok. So, that many times you have to multiply. So, you need to make sure that the value of U is well behaved otherwise you are going to have trouble because this is where the trouble starts for us ok.

So, when the U gets very small when we actually increase the depth in when you unroll it and it becomes more than 100 when you do the backpropagation, the value gets smaller and smaller every time right and then there are when you do this you are also multiplying this so, any number of times. So, the value of this would become extremely small for you to do any updation ok. And when U gets very small we are getting into that trouble of vanishing gradient ok.

(Refer Slide Time: 09:40)

EXPLODING/VANISHING GRADIENT

Consider the following sentence:
Raj entered CoffeeDay to meet his partner Dru. Raj said "Hi Dru. In the next few hours they discussed their start-up and devised a plan to develop a product on knowledge management. After a the long discussion and fruitful discussion, Raj said goodbye to his _____ (47th word)

The target word is *partner*. If the long distance gradient (the gap between $U^{(7)}$ and $U^{(S)}$ is large), then the target word is lost in the gradient as it would be too small to contribute

The decay in the gradient value is proportional to the depth of the network. The deeper the net network, the the chance of getting a smaller value of the gradient towards the fag end of the backpropagation. If the some of the are in the range of $[(0.01, 0.5), (0.03, 0.01)]$, then the the derivative would vanish to zero - $0.01^{47} = 1.0e-94$ and $0.5^{47} = 7.1054274e-15$

NPTEL

Let us take some small example and then see how it works ok. So, I have considered this sentence here to find out how the values really explode ok. So, let us take this sentence Raj entered CoffeeDay to meet his partner Dru. Raj said; Hi Dru. In the next few hours and so, on and then finally, the last sentences Raj said goodbye to his we need to find out or predict what that missing what is right. The missing word is a partner which is in the 7th location. If you are going to be using a word-based prediction, this is our 7th word

and we need to predict the 47th word right. So, unless you understand the whole sentence remember this you will not be able to write a partner here.

So, here from this 7th to 47th there are we have 40 Us coming in to place right. So, when the value of U for example, when you assume that the U gets into the range of no I am just giving a small example like this ok. I have taken 47 here it could be 40 but it does not matter the value is really low there too. The value of 0.1 for example, one of the elements of U is 0.01 and then since it has seen so, many states before it is going to be U power 47 correct. So, when you take that value it is 1.0 into e power minus 94 ok.

If when the value is 0.01 when the value is 0.5, it is going to be 7.10 into e power minus 15. See how small it becomes right. So, this is the problem it becomes very very small for you to really remember anything that is what we say that the network forgets whatever it has learnt earlier right. Initially we said that RNN and can go up to any sequence length, now we are finding that there is a problem with this it is not able to remember the long sequence. The value vanishes when the word that you want to predict is formed somewhere very far away in the same sentence ok.

So, this is the vanishing gradient problem for you ok. So, this is a very small example instead of really proving it mathematically that it vanishes I have taken this example to show you this ok. Sometimes the value can become large greater than 1.

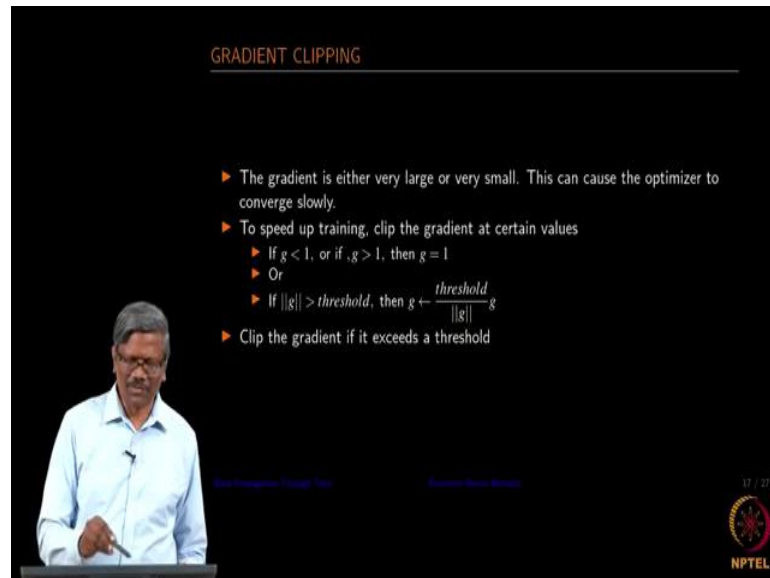
Remember this the values of uU V and W are totally unbounded. We do not have any boundary around those, for example, if you look at the activation functions of st and then Z t, we always have the values between 0 and 1 right; here it is going to be minus 1 and plus 1. So, whereas, in the case of those U, V they are not bounded by this condition or they are not conditioned by these values.

So, it is possible that the values of U becoming smaller during the backpropagation and then when it becomes extremely small, the long-term remembrance is going to be gone or the long term dependency is gone. So, it only remembers short term values. There is nothing that the RNN forgets it is only because of the gradient that we are moving from the last time state to the first time state it getting smaller and smaller ok.

So, that is the problem. It is not the problem with what we are doing in terms of the training, it is because the values that are not properly conditioned and because we do not

really take care of the value of U, V, and W in such a fashion; the gradient either explodes or vanishes.

(Refer Slide Time: 14:37)



The slide is titled "GRADIENT CLIPPING" in orange text at the top. Below the title, there are four bullet points in orange text:

- ▶ The gradient is either very large or very small. This can cause the optimizer to converge slowly.
- ▶ To speed up training, clip the gradient at certain values
 - ▶ If $g < 1$, or if $g > 1$, then $g = 1$
 - ▶ Or
 - ▶ If $\|g\| > threshold$, then $g \leftarrow \frac{threshold}{\|g\|} g$
- ▶ Clip the gradient if it exceeds a threshold

In the bottom left corner, there is a small video inset showing a man in a light blue shirt speaking. In the bottom right corner, there is a red circular logo with the text "NPTEL" below it and "17 / 21" above it.

When the value increases beyond 1 it is easy to control, we can actually check and then bring it down to a value that is manageable right. The problem comes only when the value becomes extremely small ok. So, there are mechanisms available to clip the gradient when it exceeds certain values during the training itself. So, this is a very easy problem to solve when the gradient goes beyond a point vanishing gradient is our bigger problem.