WEBVTT

1

00:00:04.540 --> 00:00:15.609

Jisun An: Alright. Thanks for joining finally, the weather is a slightly more warmer, and I'm so glad. So today's passcode is neural net. Please mark your attendance.

2

00:00:16.344 --> 00:00:23.719

Jisun An: I think there are still people coming, so I will show this password later as well.

3

00:00:25.110 --> 00:00:42.301

Jisun An: anyhow. So so I will continue the lecture. So so today we will talk about the neural network. So that's the reason that the passcode is neural net. And the reason that we talk about neural net is because of this one particular characteristic of the

4

00:00:43.571 --> 00:00:50.447

Jisun An: that this required for the language modeling. And we're handling the language which is the the combination feature?

5

00:00:53.345 --> 00:00:54.400

Jisun An: so

6

00:00:54.860 --> 00:01:14.380

Jisun An: so these were 4 of things that was missing from the back of world model. And here, I mean, there are many different like interpretation that you can have about each of these models. But here we are focusing on when we build the Nlp system, and when we build our Nlp system where we have a input

7

00:01:14.540 --> 00:01:18.270

Jisun An: which are the text and the output is right particular, like

8

00:01:18.780 --> 00:01:26.980

Jisun An: classification labels. And then if we are using the input as a bigger word. Vector then what would be the problem? So

9

00:01:27.320 --> 00:01:46.769

Jisun An: once again, I I found that maybe if you are, I mean, there could be different interpretations here. So I just want to empathize once again that we were talking we were looking at from the perspective of building on Np system. And then when we are using the bag of words vectors as an input, then these are the potential

10

00:01:46.900 --> 00:02:02.250

Jisun An: issues that would be missing from these models. And so we talked about the subord models and the word embeddings in the last 2 lectures. And today we will talk about the neural network which will hopefully handle the combination feature combination of the features.

11

00:02:02.900 --> 00:02:22.479

Jisun An: So what does it mean by the combination of features? So if you have a sentence, so this is still an example of movie review sentimental classification. So if you have a sentence like, I don't love this movie, and if you are using backward, or even the word embedding. They will still not be able to handle this.

12

00:02:22.480 --> 00:02:46.400

Jisun An: Once again, if we are building on an Ap. System, we're doing the sentimental classification. And if the input was this, and if we are, we are now represented this sentence, using the the ski gram model that we have built. So each of the words will have a vector. And then now to represent this sentence, maybe we will sum them up, or we can just give a different weight to each of these words.

13

00:02:46.400 --> 00:02:47.310

Jisun An: and then

14

00:02:47.310 --> 00:03:10.599

Jisun An: and then compute, calculate the score, and then the score turning into the decision, and that will turn into the label right? So that is how the Ndp system was working. So even if you are just looking at individual word level. So if you consider tokens, or the word as a feature, then you will not be able to capture the combination feature where don't love basically mean like a bad right? So

15

00:03:10.700 --> 00:03:30.049

Jisun An: and this could be even more complicated, depending on the like the length sentences like, there's nothing I don't love about this movie. And now you probably need more subtle kind of relations or interactions, although nothing don't, and love altogether mean very good.

16

00:03:30.050 --> 00:03:50.979

Jisun An: So this kind of like these features and combination of the feature also mean another thing, and these things cannot be extracted well, or interpreted well by the model. If you are considering each of the word as an independent feature and the solution for this is now the using the neural network as a model, so that

17

00:03:51.310 --> 00:04:07.609

Jisun An: you probably heard of these hidden layers hidden node from the neural network. Right? So those are the nodes that will help to find out how these features are interact with each other, and they will find which combination of the features are important to have a better classification, for example.

18

00:04:08.580 --> 00:04:30.080

Jisun An: So yeah, so this is like visualization of what I just mentioned. So we are really just focusing on the Nfp system, and that Nfp system is work based on you have now each of the word that serve as a feature, and we can represent each of the words in different way. And one of them is debacle. But which is, we are basically using one hub factor.

19

00:04:30.760 --> 00:04:47.509

Jisun An: And you can assume that these weight vectors is now serve as determining how important each of these words, or each of the feature is in in getting the correct kind of label. So once we have this, vector once we have weights, we can get some score. Given this sentence

20

00:04:47.510 --> 00:04:53.770

Jisun An: and given this score, you can have another decision function to assign the label based on that.

21

00:04:53.770 --> 00:05:18.239

Jisun An: And the continuous bag of word is basically instead of one hot coding for the fact representing the word we are now using the dense vectors. And the continuous bag of word is basically another method of your word to back. And we went through this gigram model last week. And it's very similar model. It's slightly different. But the key idea here is, if you also look at the differences between these 2 diagrams.

22

00:05:18.300 --> 00:05:29.639

Jisun An: Here the is a 1 hot encoding. But here is a dense. Vector, so even for the same word. Now, it is represented by this dense. Vector each of the vector element can

23

00:05:29.800 --> 00:05:36.929

Jisun An: tell something about some characteristic of the word relationship or association of these words. So now, maybe the 1st

24

00:05:36.970 --> 00:05:51.340

Jisun An: 1st vector. Element could mean something positive. Second element could be something negative. 3, rd one could be, it's a it's like determinant. Or 4, th one could be 4.th So now, each of the vector element could

25

00:05:51.340 --> 00:06:10.159

Jisun An: represent something. And by giving different rates on different vector elements, it can also represent a word. So the vector now contains more meanings about even for the one word, and that enables that they can look for which words are more similar to each other. So that was the characteristic of this word embedding

26

00:06:10.743 --> 00:06:34.010

Jisun An: so so that was the second step. But then the rest of the rest of the steps are the same. So once you represent each of the tokens or the word or features into the vector we have the weight vector and there's some bias vector which basically like the default value for your score, and you sum them up. Then you will have the score, and then you can based on certain decision function. You can label the sentence with the particular sentiment.

27

00:06:34.930 --> 00:06:58.664

Jisun An: Now the basic idea of the neural network, especially for the prediction task in the Nfp. Now, instead, you only had one weight. Vector if you compare with this one, instead of that, one weight, you now have some more complicated function to extract the combinations of these features, and the rest will be the same. I mean you. So these parts where that

28

00:06:59.240 --> 00:07:11.740

Jisun An: computing this weights is just getting more deeper and more complicated. But after that, if you end up giving a score, and then that score can turn into some probability so that you can.

29

00:07:12.233 --> 00:07:20.459

Jisun An: Depending on those probability you can now label something one over that I will talk more about that last part a bit later. But that is basically the idea.

30

00:07:21.090 --> 00:07:41.140

Jisun An: And in like. So the cocbow is the continuous spec over. Just so that means the inspector and the deep Cbo W means that now we have, like a deeper neural network in computing these scores. On top of that. So now we will talk a little bit more about that intermediate part.

31

00:07:45.630 --> 00:08:11.940

Jisun An: And so after so assuming that we already have this network and also learn the model what our vectors also represents. So now things are getting more, little more interesting. So once again, when you were just using the word 2 vec, like ski gram model, then they will only know about like one. They will only have one particular characteristic of the words like, whether they are forward, whether they are determinant, whether they are positive or negative, etc. Etc. And now.

32

00:08:12.410 --> 00:08:24.590

Jisun An: in the neural net, one of our weights may be able to learn feature combinations. So, for example, a node in the second layer might be feature one and feature, 5 are active, etcetera. So we now they will need to

33

00:08:24.680 --> 00:08:50.249

Jisun An: be able to know how to which 2 functions should be active at the same time, and that may lead to better prediction, and etc. Etc. Etc. So they may be able to capture something like Okay, not. And hate would eventually mean the positive. So these are the something that these vectors will. I mean, after the learning of the models, the vector will represent the combination of these features.

34

00:08:51.040 --> 00:09:17.280

Jisun An: So today, I mean, we will talk once more about neural net. Next lectures as well when we talk about the language modeling. So today, we will cover the basic concepts of the neural lab. So if you're already familiar with the neural net, I think this will be a little boring, but for those who just surfacely know about this. I mean, it is still very high level. I just want to let you know what is the neural net and how it actually works in the Nlp kind of system today.

35

00:09:17.830 --> 00:09:23.254

Jisun An: And even we may not be actually get there. But let's see, so what is the neural net.

36

00:09:25.220 --> 00:09:40.910

Jisun An: so this is like like a basic unit of the neural net and neural network unit is is coming from originally from like the biology, and where the it kind of resembles the how, the the neurons in our brain cells

37

00:09:40.910 --> 00:10:05.669

Jisun An: where the neurons are essentially, they have multiple inputs coming in. And then there are the outputs, and many of the neurons are connected to each other, and then they kind of like delete our signals, and one is activated and the other is less activated. And all together, they kind of help us to function on something in our body. So the neural network and the names of them neurons and the like.

38

00:10:05.670 --> 00:10:13.830

Jisun An: perceptrons and etc. These are all the metaphors that coming from the biology, even though nowadays we don't kind of use them anymore.

39

00:10:14.550 --> 00:10:40.119

Jisun An: And so these are kind of resembles to the, to the neuron that that in our body and in computationally, this is one like the smallest unit in the neural network where basically we have multiple inputs. And then we do weighted sum. And then there are like nonlinear activation function which will reduce in the output. So in one neuron, we will always have an input and inputs and then one output.

40

00:10:40.120 --> 00:11:01.789

Jisun An: But then inside of this one neuron, we will do weighted sum and some kind of non-activated function. So the input layer and then weights, and there are usually the bias and bias. Some, you can consider them as a default value. So if all the weights are 0, then you will still have the bias as a value valid value, so you will have some default value.

41

00:11:02.480 --> 00:11:09.990

Jisun An: and you will do weighted sum. And then for weighted sum, you take nonlinear transformation, and then you have this output value.

42

00:11:11.171 --> 00:11:15.609

Jisun An: So this is like a single kind of unit of the neural network.

43

00:11:16.190 --> 00:11:45.519

Jisun An: and more formally, this would be, and if we are kind of dividing them into like the 2 portion, then you can represent them as these are, and as a vector representation, these are the weighted sum of the X and weight weight W and the input x plus b is the bias. And the second line of the equation is just a vector representation. So you can assume that there are vector of the input vector of the weight. And these are the multiplication of these 2 vectors. And that will be this weighted sum.

44

00:11:45.920 --> 00:11:58.420

Jisun An: And then we apply, like the nonlinear activation function F, and this nonlinear activation function can be anything. But after you're applying them to this weighted sum, then that will turn into the output.

45

00:11:59.290 --> 00:12:00.903

Jisun An: So and

46

00:12:01.900 --> 00:12:11.860

Jisun An: and finally, this nonlinear activation function can be anything. But if it's the sigmoid like the sigma, then this will be like this. So

47

00:12:12.460 --> 00:12:30.350

Jisun An: so I don't know how familiar you are with like logistic regression. But logistic regression is actually very similar to this kind of form. And if you think, think back to some of the example that we dealt with, like like rule-based methods, or

48

00:12:30.510 --> 00:12:32.776

Jisun An: simple, like the CEO, we

49

00:12:33.771 --> 00:12:36.060

Jisun An: the bigger world kind of model.

50

00:12:36.842 --> 00:12:57.277

Jisun An: It's it's very similar to this without the I mean, even with the nonlinear transformation. This is exactly the representing the logistic regression, because we have a few inputs and we have some weighted weight for each of the feature, and then it will have some score. And then for the logistic regression. We want it to

51

00:12:58.210 --> 00:13:18.579

Jisun An: transform that score into some probability like from 0 to one. And the Sigma function is that function that, given any input, it will output to 0 to one which now will represent as our probability. So essentially, I mean, it's a very small, tiny network. But this actually resembles very well with the like registration.

52

00:13:19.605 --> 00:13:31.310

Jisun An: And so in other way, you have multiple different features, and you have the output. And you want to kind of find. The weight essentially, once again means the importance of each of these inputs.

53

00:13:31.310 --> 00:13:56.130

Jisun An: And so this will be like the linear kind of transform linear kind of equation. But the reason that we are adding this nonlinear activation function is because there are many things that cannot be explained. Only the linearly right. So basically, by adding this nonlinear activation function, you may be able to explain the patterns that cannot be

54

00:13:56.130 --> 00:14:14.089

Jisun An: explained by the linear relationship. But so if, like Y equal x squared, that itself is already like not the linear vacation. Right? So by having this nonlinear function, you can like now explore more complicated kind of features of a combination of these features.

55

00:14:16.006 --> 00:14:17.280

Jisun An: So, so.

56

00:14:17.360 --> 00:14:27.060

Jisun An: so this was just one example. So if we are taking the sigmoid as this nonlinear activation function. Then we will have this occasion for the Y, but there are

57

00:14:27.493 --> 00:14:48.569

Jisun An: many other examples of nonlinear activation function, like tenage or the value rectified linear unit. You probably heard this somewhere here and there, and the tenage is now, given an input, it will output from minus one to one. So it it's in in the neural networks.

58

00:14:48.988 --> 00:15:03.220

Jisun An: They tend to work better than the sigmoid, because it has a larger range. But then the value is the most commonly used nonlinear activation function beside the sigmoid or the panage, because

59

00:15:03.580 --> 00:15:17.959

Jisun An: henage or the sigmoid has the crucial problem. So we will discuss more about that propagations and the gradient distance later a little bit more. But if you remember back when we train our model.

60

00:15:18.630 --> 00:15:42.070

Jisun An: so when we basically updated our weight given. And then and then when we update those weights, we determine them based on how much this variable affects to our loss function right, and to determine how much that of that has the effects to our loss function. We were computing the derivatives right? So in 10 each.

61

00:15:42.070 --> 00:16:05.700

Jisun An: if the value is like really getting high, even greater than 5. Then you can see that the Y value reaches to y equals 0, right? So any value x greater than 5 will have y equal near one. And when the Y equal near one is derivatives is like basically 0, meaning that if your derivative is 0, then basically, there will be no updates.

62

00:16:05.960 --> 00:16:31.689

Jisun An: So if you're because of that, if your network is getting really deeper and your values are getting higher, then these nodes will output like near one, meaning that their derivative will be near 0. In other words, those gradients will become like meaningless, so there will be no proper updates going on. And this is the problem called as a vanishing gradients.

63

00:16:32.000 --> 00:16:41.869

Jisun An: but then, at the but, on the other hand, in the value. You can see that as the axis getting increasing, the Y value is also increasing, if the value is greater than 0.

64

00:16:42.440 --> 00:16:53.529

Jisun An: In other words, unlike the 10 h. In the value. If the value is getting increasing, your derivative will be always near one, because it has this.

65

00:16:53.740 --> 00:17:03.899

Jisun An: Yes, so keep increasing. So your gradient will be always alive. So you will have some updates, because your gradient will be like near one

66

00:17:05.174 --> 00:17:16.890

Jisun An: so so that's the some of the reason, why the value is most commonly used, and once again the sigmoid and the 10 inch are likely to have the Spanish ingredient problems.

67

00:17:17.869 --> 00:17:43.849

Jisun An: Right? So then. So these were like the basic units of 1 1 neuron. And now, if you are looking at this feed for neural networks. Then they are this networks where they have an inputs, and then they have like the nodes where these one neuron is connected to the other. And then they are kind of just like

68

00:17:44.143 --> 00:18:08.210

Jisun An: connecting to each other without any cycle. So so that's the reason that it is called as a feed forward. So the key characteristic of the feed forward in that neural network is, it's always going forward. So you will see that the way that the the output of the one node will become the input of the other node. But there will be no the opposite direction that the value is coming back. So you will only see the like.

69

00:18:08.210 --> 00:18:24.159

Jisun An: Basically, it doesn't have any cyclic relationship within the feedforward neural networks. But then, so basically, you can have multiple neurons, and then you can layer them up. And then you just connect them with their inputs and outputs. And that will be your feed forward neural networks.

70

00:18:24.160 --> 00:18:32.309

Jisun An: and it also called as a multi-layer perceptions, or an ep 4 historical region. But yeah, so I will. I may just interchangeably use those terms

71

00:18:33.490 --> 00:18:54.619

Jisun An: anything. So and remember, this node is now, each of them are neurons. So within each of the neuron, they will function. They will do these weighted sum and the applying the non activation nonlinear activation functions, and then having the output. So each node will have the exact, the same structure as you've seen from the neuron.

72

00:18:55.240 --> 00:18:56.420

Jisun An: does it make sense?

73

00:18:58.860 --> 00:19:17.510

Jisun An: Oh, right? So thanks for bringing that up. So so usually in the neural network, they have 3 different types of the node. Where is the usually is the input node, the and then the hidden node, and then the output node. So the X is the input H is the hidden node, and the Y is the output nodes.

74

00:19:20.040 --> 00:19:31.849

Jisun An: yeah, I think through the few slide. I hope that what hidden knows are what? What are they? Hopefully they will get more clearer. But I don't want yet. It's just a hidden note.

75

00:19:31.990 --> 00:19:45.680

Jisun An: So once again, so coming back to this? the notion of the the logistic regression. So these are like a very, very simple neural network, one layer network that.

76

00:19:45.690 --> 00:20:04.819

Jisun An: and that actually represents in logistic regression. And and here the reason that you see one layer so usually the input layer is is not counted as a layer. So, assuming that these X layers are just 0, and you only have this output layer. So we we call this as a 1 layer, one layer network

77

00:20:05.220 --> 00:20:31.810

Jisun An: and the logistic regression. So you will have like multiple features. So once again, think about the sentiment classifier. So you will have a few features. Right? So, whether given a sentence, how many positive words it includes, how many negative words it includes, or whether the No is in the sentence or not whether or not is in the sentence or not. So you will have these different features and these these features will be your input. Vector and then you have waited

78

00:20:32.216 --> 00:20:53.733

Jisun An: weight for each of this feature, and in the output layer you take the sigmoid to turn the score into the probability which is in this case a sigmoid is our nonlinear activation function. So actually, the logistic regression is just one layer neural network that you can you can train

79

00:20:54.500 --> 00:20:58.099

Jisun An: to to have the sentiment classifier

80

00:21:02.350 --> 00:21:09.987

Jisun An: and and if it is a multinomial logistic regression, in other words, if it's not a binary classification. But if it's like

81

00:21:10.420 --> 00:21:34.749

Jisun An: multi-label classification, then basically, the output will be. Now, there will be like N output node. So in this case the sentiment, whether it's a positive or negative, or yes or no. In this case it could be positive, neutral, negative. So if it is multinomial, logistic regression that it will, it will still use this one layer network and fully connected, fit for network.

82

00:21:34.880 --> 00:21:38.050

Jisun An: but with multiple output node.

83

00:21:38.920 --> 00:22:05.050

Jisun An: But in this case, instead of the sigmoid, we would use the softmax function, which is which helps to when you have a N scholar values, then softmax function will transform those value into the probability. And so softmax is, in a way, a generalization of the sigmoid function. So once again, Sigmoid, given a particular score

84

00:22:05.050 --> 00:22:23.329

Jisun An: as an input you will the output will be scaled from 0 to one. So that we will basically change a score to the probability. And the software, Max, we're going to do exactly the same. But for the multiple dimension or multiple kind of inputs. So for example, if you and and I mean

85

00:22:23.410 --> 00:22:28.059

Jisun An: it will take given a K jet value

86

00:22:28.550 --> 00:22:53.579

Jisun An: for a factor jet with the dimensionality K, meaning that you have K values. Then you will basically take the exponent exponent explanation of those values, and then basically finding the proportion of each of those value. So so this is like fairly easy concept so if you have these values, and if you take this optimex function, then it will basically return a probabilities. Given those scores.

87

00:22:56.222 --> 00:23:23.047

Jisun An: And now, if now so, that was like a single layer, which is resemble to the logistic regression. But now we are having 2 layer network with a scholar output. So it's the same. So we have an input layer. And we have now hidden units which will basically tells I mean connects gets the weight weighted some and the

88

00:23:23.920 --> 00:23:25.790

Jisun An: oh.

89

00:23:26.890 --> 00:23:44.301

Jisun An: weighted sum and do some nonlinear transformation in each of these hidden units and those output will be now also the input of the output layer. So these are like the feed board neural networks. And here we say that the hidden units are using the sigmoid as a

90

00:23:44.990 --> 00:23:50.670

Jisun An: the the nonlinear activation. But it could be actually like the value or the tenage or anything else.

91

00:23:51.077 --> 00:24:09.474

Jisun An: So in terms of the vector representation, so we will code as jet as a weighted sum of the inputs output of the hidden units which are the h, and then the the weighted weights of the hidden units which are the U. So the jet will be the

92

00:24:10.210 --> 00:24:23.889

Jisun An: And our final output y will be sigma jet, which will be another sigmoid. So you can see that at each of the layer there will be weighted sum and nonlinear activation. And so you kind of repeat them.

93

00:24:25.290 --> 00:24:51.570

Jisun An: And similarly, if you have now, multiple outputs, you can exactly use the same network. But instead of the sigmoid, you can use the softmax at the last layer, so that so for each. So you can imagine now, here the input would be some sentence which will turning into the vectors values and then rate it and some hidden units. And then the last output layer basically may mean like

94

00:24:51.850 --> 00:25:17.280

Jisun An: positive or the negative. And then for both of these class, they will have some scores. So given this, this network eventually will calculate. Given this input what is their likelihood to be positive, or what is their likely to be negative, and then given the score. Using the soft mix, they will turning into the probability, and given those probability, you will be able to determine whether they are positive or negative, and etc.

95

00:25:19.280 --> 00:25:23.639

Jisun An: Yeah. So I think these are fairly simple. Any question up to here

96

00:25:29.387 --> 00:25:34.859

Jisun An: in this situation it looks like we're going to have, like

97

00:25:35.010 --> 00:25:43.084

Jisun An: the pre-trained embeddings has the input for this. Oh, right? Yes, that will be right coming right after

98

00:25:43.640 --> 00:25:49.518

Jisun An: Oh, yeah. So I mean, it's so, I think.

99

00:25:50.090 --> 00:25:54.036

Jisun An: yes. So here the X now will be

100

00:25:54.650 --> 00:25:57.530

Jisun An: embedding. Yeah, the vector. Embeddings themselves.

101

00:25:57.780 --> 00:26:02.720

Jisun An: And the so to get the amendments. In the 1st place, the Infen.

102

00:26:03.210 --> 00:26:30.750

Jisun An: so that there will be another layer, I mean. So there will not be part of the network, because that is like more like a table lookup. So it is not part of this figure, so you can assume that there's another steps here. So, given a word, you have your table where it's either one hard encoding or the embeddings that you already learned from the word to back. And given that word, you can select that role. And this Xn is already embedded.

103

00:26:30.870 --> 00:26:36.779

Jisun An: So these are presented as a 1 node. But it's not necessarily to be one value. It can be a factor.

104

00:26:36.920 --> 00:27:00.740

Jisun An: Yeah. So, yeah, I mean, so hopefully, I have another figure that will show in more details. But yeah, that's correct. So the input layer, it itself is already. Vector but the x 1 can be also. Vector and in that case now the wet weight matrix became a 2 dimensional vector and the after the hidden unit it became now tensor. So yeah.

105

00:27:00.750 --> 00:27:16.659

Jisun An: so I think just it becomes like far more, harder to visualize. So make it as a simple, and we leave it to you to visualize it. But you are correct. Yeah. So these input x 1 is the representation of a token. So it can be a vector.

106

00:27:18.650 --> 00:27:31.520

Jisun An: all right. Yeah. So I I want to highlight the the fit for network for the sentimental classification. So here, the logistic regression case.

107

00:27:31.840 --> 00:27:53.189

Jisun An: It's a still like one layer network, and it has like the features, and we have radius of and the sigmoid. And then we are turning into a text to a sentimental classifier right? And then in our 2 layer fit for network, it's the same that we have a features, and then words. And we have hidden units, and then the finer output as a sigmoid. But

108

00:27:53.640 --> 00:27:56.420

Jisun An: and and I think maybe I

109

00:27:56.610 --> 00:28:26.200

Jisun An: maybe you got kind of confused by it. But the the key of the neural network is that we don't need to select or manually select this feature. So in the logistic regulation cases, we had to actually select those features right, like either the number of the positive words or a number of the negative words, or whether some, whether the sentence actually included no, or the sentence included. Yes, so these were all manually selected features at one.

110

00:28:26.200 --> 00:28:44.239

Jisun An: But then in the in the neural network. Now this model helps you to learn like which features are. I mean, you don't need to actually select these features, and the model themselves will just learn naturally which features are most important and contributing to the finer features.

111

00:28:45.470 --> 00:28:49.600

Jisun An: So that's the kind of the major major differences. So

112

00:28:52.530 --> 00:28:57.580

Jisun An: in compare these 2 by just adding, one layer of these hidden units

113

00:28:57.740 --> 00:29:27.419

Jisun An: will help to use this nonlinear interaction between features. So so in the previous example. You will. Only so, even though you have this nonlinear activation in the logistic regression that only works to change your score to probability. And it doesn't do anything to determining which feature is more important. But if you have one extra layer there, then, now that hidden layer will start to learn which of these features are

114

00:29:27.420 --> 00:29:44.609

Jisun An: should be activated together to do better classification. So once again, the 1st hidden unit may learn that the 1st feature and the 4th feature should be combined together. I mean, when those 2 are activated at the same time, then that is more likely to be positive.

115

00:29:44.610 --> 00:30:02.809

Jisun An: And maybe the second hidden union may learn. Okay, feature 2 and 8 and 10 altogether need to be activated, then that will contribute. That will result in the sentence to be positive, etc. So that would be the big difference between a simple logistic regression versus like the 2 layer fit for network

116

00:30:03.920 --> 00:30:14.829

Jisun An: and so the real power of the learning comes from the ability to learn features from the data themselves, and once again, even though they called it as f. 1 f. 2 fn. The same. But the

117

00:30:14.940 --> 00:30:39.630

Jisun An: the logistic regression. One we pick the features at manually, and here these can be just like the tokens or the tokens that you have, or all the words that you have, and each of these tokens or words also can be represented as embedding themselves, which, embedding already also has a lot of meanings of all the words right? So if you're using the one hot encoding, it'll be just individual feature.

118

00:30:39.630 --> 00:30:50.479

Jisun An: If you are using these key program models embedding, then that embedding already rich enough to tell you how the relationship among the words, etc. So you already are quite advanced to kind of models.

119

00:30:52.110 --> 00:31:06.568

Jisun An: So so this is the like, so that one was like the very simplified version of the neural network. And this is giving you a little bit more details. How the neural network would be look like in the sentimental classification example. So

120

00:31:07.220 --> 00:31:11.485

Jisun An: so once again, when we firstly saw the

121

00:31:12.100 --> 00:31:17.339

Jisun An: the in the previous example, the the figure started from here. So this one was the X,

122

00:31:17.350 --> 00:31:31.340

Jisun An: the input layer. And this is the hidden and the output layer. So this is the 2 layer network. And this part were kind of the missed out and or altogether just they consider to be the same. So basically, you think that even

123

00:31:31.340 --> 00:31:54.980

Jisun An: so, this is like sentiment, classification, example. So even the word you assume that there are big table where you can for that particular word, you can just extract that factor. So this input layer already has its embedding factors. And then for for each of the embedding factors we have a weight. And then that the output comes through the layer.

124

00:31:54.980 --> 00:32:00.629

Jisun An: And then the output of the input layer goes to the input as a as a input of the output layer

125

00:32:00.630 --> 00:32:20.619

Jisun An: and then finally, and then for this single layer they also had the nonlinear activation function which can be sigmoid or Tennant or the value, but that the output layer y will take the sigmoid so that the score can be turning into the positive. So this will be something looking like like more

126

00:32:24.370 --> 00:32:27.550

Jisun An: the neural network for the Nfp system.

127

00:32:27.890 --> 00:32:42.417

Jisun An: And then here, I mean this, just like really within this example. So here we have assumed the fixed size. Length 3. Right? So we in in this particular visualization, we assume that there are only 3 words.

128

00:32:43.100 --> 00:32:52.960

Jisun An: this model takes the 3 word as an input, but this is kind of unrealistic. So but and also usually we have, like the sentences, like longer than 3 words.

129

00:32:53.260 --> 00:33:01.280

Jisun An: So how would you like change this model to work? What would be the solution? That how how would you do that?

130

00:33:02.570 --> 00:33:08.929

Jisun An: How would you change this model to make a any sentence, be on input to the model. Yes.

131

00:33:09.060 --> 00:33:27.946

Jisun An: yes, that's actually great. Yeah, thank you. So one solution would be we. We took embeddings for each of the word. So given a sentence for each of the words, we take the embedding for each of the words. And we just take this, some average

132

00:33:28.340 --> 00:33:55.299

Jisun An: of those vector and we make them just one big embedding, so which we call as our sentence embedding. So instead of having, like N, input, we can actually have one input node, which will have an dimensional vector which is the average of the word vectors. And we can consider that as an input vector so that would be one solution. So maybe I wasn't really clear about what was the question. So now the every sentences will have a different length.

133

00:33:55.300 --> 00:34:13.599

Jisun An: So what would be our solution to make the model to get that input for diverse kind of sentences. And one of the solutions as mentioned was that so instead of we consider individual words, we consider sentence embedding, which is the average factor of the word embeddings.

134

00:34:13.810 --> 00:34:19.619

Jisun An: So each word will have is the 512, vector.

135

00:34:20.300 --> 00:34:31.340

Jisun An: and then we just you just average them for all the words. Then there will be the sentence embedding. So that would be one possible way to. You can change this model and any any other way.

136

00:34:32.670 --> 00:34:36.670

Jisun An: Are there any other solution that you can think of?

137

00:34:46.380 --> 00:34:47.349

Jisun An: I mean.

138

00:34:47.400 --> 00:35:16.479

Jisun An: this will still work it, only it still works it. Only just consider the 1st 3 words of the sentence right? And you, basically, the test is now changing. Even the 1st 3 words of the sentence predict the sentiments right? But we want to actually consider like entire sentences. So you can basically increase the number of words that you can consider in the model right? But then, if we are increasing let's say that we are increasing the window to 10

139

00:35:16.540 --> 00:35:21.900

Jisun An: meaning that you will consider the the 1st 10 words right? What would be the problem?

140

00:35:25.520 --> 00:35:26.320

Jisun An: He's

141

00:35:31.060 --> 00:35:41.300

Jisun An: yes, exactly. So. Basically, there will be sentences. That is longer than 10 sentences. And maybe the important part could be after the 10th word. So this may be some problem.

142

00:35:45.330 --> 00:35:46.460

Jisun An: even size.

143

00:35:48.580 --> 00:35:58.139

Jisun An: input size. Yeah, so we can increase the input size, right? Yeah, so so what if then, we increase it to like 200? What would be the problem then?

144

00:36:02.020 --> 00:36:18.004

Jisun An: Because it can be too complicated. And also yeah, that's also, yeah, that's that's true. Because our vector, will be now become like, far larger it may not be really necessary, because maybe not all the sentences have like 200 words and

145

00:36:18.510 --> 00:36:43.270

Jisun An: But maybe one of the sentence would have like the 200 words, and then the rest are on average, maybe 100, and then the rest. 100 will be now padded right? So so you probably heard padding somewhere. So padding here is that basically you are now just fixing some windows input size. And then, if the sentence is short, then the remaining the empty part you are just padded with, like the 0 values.

146

00:36:43.330 --> 00:36:51.613

Jisun An: And if sentence is like larger than that context window, then you were just considering that. So I mean, there's a no like solution here. I'm just

147

00:36:52.090 --> 00:37:20.730

Jisun An: testing what that has been just of talking with so usually like the larger context would helpful. But then it will be computationally more cost. So you need to kind of find the balance between, like, what would be the like nice size of this input, or sometimes like just shrinking them into one concise vector, like sentence. Embedding would also work. So both strategies have been tested. I think it just depends on your kind of applications.

148

00:37:23.430 --> 00:37:25.790

Jisun An: Okay, any questions?

149

00:37:32.280 --> 00:37:38.110

Jisun An: Alright. So now we know how the network looks like right?

150

00:37:38.670 --> 00:37:45.180

Jisun An: Right? And then the question is, then, how do we train this model? Is the next question.

151

00:37:45.430 --> 00:37:58.430

Jisun An: and this is the like overview of the question. So, assuming that we have, like 2 layer network with the 2 different inputs outputs once again, the here the X can be

152

00:37:58.660 --> 00:38:16.830

Jisun An: it at the moment. It represents a word or token, but this word or tokens are represented by the embeddings. But you can, you can, I mean, for simplicity, you can think it as a scholar value. But eventually it is the vector and then all the weights that are multiplying by 2. It will be be like n dimension, or

153

00:38:16.830 --> 00:38:32.808

Jisun An: multiply to the end. Dimension. but but anyhow, so how we train. So this train is very similar to the ski Gram model that we talked on Tuesday so. And and I think essentially how the machine learning works has

154

00:38:33.200 --> 00:38:55.289

Jisun An: different model have all the similarities. So I hope that you can kind of get the idea over the last lecture. And this lecture. So the 1st thing that we will do is we prepare. We have this network and given an input we do, and all the weights here W and the U will be initialized with the random values initially. So, so each weight. Now now have all the values.

155

00:38:55.290 --> 00:39:13.139

Jisun An: So given on input, we can do the forward pels. And if we do forward pests, what at the end of the upper layer it will have some kind of probability which we call there's a y head. So y head is usually representing the predicted values in in your system.

156

00:39:13.770 --> 00:39:39.309

Jisun An: and because our last output, either taking Sigmoid or the softmax. In this case there's a softmax. It will be a probability. So, for each of the each output is basically representing a some kind of label, and then we are having the probability given this sentence, the probability of that sentence will be in outputting that label in particular. So our Y hat will be some kind of probability.

157

00:39:40.150 --> 00:39:54.430

Jisun An: But then we actually have, because we are training the model with the training data set. So we assume that we have a training data where sentences and the labels with the positive negative neutral? So we actually have the actual answer, why, right?

158

00:39:54.430 --> 00:40:13.570

Jisun An: And then using this y and y hat value, we kind of define a particular loss function. So we want to measure what is the gap between y and y hat and the the hour and the loss function, is basically yeah, how much gap? You can consider it as a like error.

159

00:40:13.910 --> 00:40:30.640

Jisun An: The the function that computes the error. And then, once we know that error, we can now do the backward pass. Given this loss function, we want to update our parameter W and U to reduce the loss because we define it as a loss.

160

00:40:30.640 --> 00:40:45.220

Jisun An: So whatever value we have in the loss function, we just want to update our parameter toward to that the by changing this parameter we want. Then these loss will get decreasing. So that's the how we will train the model.

161

00:40:51.140 --> 00:41:03.240

Jisun An: So more formally so for every training to pull X and y, we run the forward computation to find out our estimate Y hat. So this will be our predicted probability.

162

00:41:03.350 --> 00:41:21.299

Jisun An: and then we run the backward computation to update the weight. So for every output note, we compute the loss L between true label and the estimate label, and then we update the weight for hidden layer to the out, which was you in our previous example, and

163

00:41:21.370 --> 00:41:33.659

Jisun An: and for every hidden node. We also assess how much blame it deserved for the current answer and for every weight for the input layer to the hidden layer. We also update the weights.

164

00:41:34.060 --> 00:41:37.269

Jisun An: So this will be how the training will kind of go.

165

00:41:39.042 --> 00:42:04.057

Jisun An: Then just one reminder. So in the in the last lecture we talked about the gradient descent. So what we were trying to do is for each of the weight. And we have the loss function. And we basically need to find how much this weight affects this loss function, which can be found from the derivative of the loss function with respect to a individual the weight. So that was the

166

00:42:04.400 --> 00:42:09.829

Jisun An: basically the slope that this weight is influencing to the loss function.

167

00:42:10.500 --> 00:42:29.860

Jisun An: And once we find that derivatives, we move the weight, the opposite direction to these gradients, so that if the slope was negative, then you need to increase the weight value, because because if the the function looking those function looking like this.

168

00:42:29.860 --> 00:42:43.409

Jisun An: And if the given the current X, if this law was negative, then by increasing your rate, the loss will be getting minimizing. So so that was the kind of intuition that we had in the gradient to descent so

169

00:42:43.520 --> 00:43:01.169

Jisun An: given for the current weight, we just finding this, the slope which is the coming from the derivative of the loss function with respect to that weight, and with some learning rate. So we should determine how much we want to update the parameter and given that, using that, we will update this value.

170

00:43:02.190 --> 00:43:30.529

Jisun An: then how can we calculate these derivatives? I don't want to go details here, and this is not the calculus course. And also we're not going to ask you to compute the derivatives. So I will just give you a high level concept. But these derivatives, especially for those compound functions, can be computed using the chain law. So if the target function is fx is the U of Vx. Then the derivatives of F of x

171

00:43:30.580 --> 00:43:38.670

Jisun An: can be computed by derivative of u of x, multiplied by derivative V. In respect to the X

172

00:43:39.049 --> 00:43:45.539

Jisun An: and this can be if you have even more compound function that you can go on and on. So that's like basically the chain rule.

173

00:43:46.390 --> 00:44:02.159

Jisun An: So in this case, if we talk about one neuron, the neuron itself is. And if you think about how the output is computed, it's already compound function, right? So y equal y equal basically weighted sum of x

174

00:44:02.250 --> 00:44:24.329

Jisun An: and the sigma, the function. So it's already like compound function. But if but what we want to know is the derivative derivatives of the L with respect to each of the weight. So we want to know, and that will give you how much, how important the the input is toward to the loss function and

175

00:44:24.570 --> 00:44:35.610

Jisun An: how basically, how I should update this weight to minimize the loss. But because these loss function is a compound function

176

00:44:35.610 --> 00:44:55.100

Jisun An: where the weight is just one small part of this loss function, we to to get the derivative of the L of Wi, we 1st need to get start from the output layer. The derivative of the loss which is the derivative l. Over of

177

00:44:55.100 --> 00:45:11.030

Jisun An: y, multiplied by derivative y in respect to the jet, which is the derivative of the activation, multiply by derivative jet in respect to the wi, which is the derivative of the weight itself.

178

00:45:11.290 --> 00:45:12.890

Jisun An: So I mean.

179

00:45:13.330 --> 00:45:31.580

Jisun An: So once again, I will not ask you to compute all these things, but these are, I mean, you need to conceptually understand this. So going back. So how we compute these values for the compound function, like Fx equal U of Vx

180

00:45:31.580 --> 00:45:52.589

Jisun An: can be computed by chain loop, where basically, you need to find those 2 derivatives to get Ddf over the derivative of the F in respect to the X, so, and then our neuron. Each of the neuron is essentially is a compound function where y equal weighted sum of X

181

00:45:52.610 --> 00:45:58.350

Jisun An: and taken the sigmoid. So by and each of the function can be

182

00:45:59.318 --> 00:46:05.950

Jisun An: decomposed using the chain law. And by using each of these value you will be able to get this

183

00:46:06.859 --> 00:46:22.039

Jisun An: derivative of L in respects to wi, meaning that for each of the weights like, what is the slope that each weight has at the moment given the X given the inputs towards the loss function.

184

00:46:23.970 --> 00:46:52.185

Jisun An: And then the question is now, so how can we find? But then here the problem is that. And this is just one neuron and our bit. 4th networks are combined of the multiple neurons with the multiple layers. Right? So you can see that these are getting really, really complicated. And and the problem is, we want to know how each of the weight and how much it influenced to the loss function. But the loss function is only computed at the end of the network.

185

00:46:52.750 --> 00:47:07.190

Jisun An: so how can we find that gradient for every weight in the network. And that's the when this back propagation is coming in. So the back propagation is on very important concept, and

186

00:47:08.531 --> 00:47:34.669

Jisun An: this is what I just mentioned. So we want to know for training, and for each of the training we will compute the loss, and we will compute how much. Each of the weights are influencing the loss function. And then we will update that update that parameters or the weights. And that's the what this models we're going to do and and those we need to find these derivatives of the loss with with respect to each of the weights.

187

00:47:34.978 --> 00:47:41.769

Jisun An: But then the loss is only computed at the end of the network. And so the solution is this error, back propagation

188

00:47:42.080 --> 00:47:56.050

Jisun An: and back propagation is a special case of the backward differentiation which relies on the computation graph. So the so the to do the back propagation, we need to have this concept, the computation graph

189

00:47:56.471 --> 00:48:17.810

Jisun An: so the computation graph is the is represents the process of computing a mathematical expression. So I will give you a very, very simple example. So let's say, our loss function, even though it's very, very simplified version. So we want to be our loss function. We don't input ABC is C multiplied by a plus 2 b.

190

00:48:18.200 --> 00:48:43.819

Jisun An: Then if we explain this equation into only the multiplication and the summation, then we can split the computation into smaller modules, which is the d, and E and l, and we can define like the different computation. So if we say that D is now 2 multiplied by B, then the first, st the 2 B term really changes it to the d.

191

00:48:43.820 --> 00:49:09.559

Jisun An: and then now we can like we can have, like e like, define a new function e equal a plus d, and then now given that now l. Now can be expressed as a C multiple by E right. So now the L was, which were a bit more. It's the like mixture of the computations can be decomposed into simpler mathematic expressions.

192

00:49:10.520 --> 00:49:31.767

Jisun An: and once you can get them into the composite to simple expressions. You can build this kind of computation draft. So now our L loss function, which was C multiplied by a plus 2 B can be expressed in this way. So here once again,

193

00:49:33.381 --> 00:49:57.159

Jisun An: you can see that we have input ABC and our new function, E will basically I mean new function d is 2 b, so it's a 2 multiply by B, and then the new function, E will be some da, and t, which is the 2 B, and then the our Lc is a ce, so e will be now a plus 2 b, so that will be C multiplied by E,

194

00:49:58.920 --> 00:50:05.664

Jisun An: so this is the so the the cool idea here is that we just

195

00:50:06.430 --> 00:50:23.690

Jisun An: like find the intermediate computation to decompose a long, complex function and make them as a this graph where you can kind of track down individual inputs and how they computing for to this loss function.

196

00:50:24.720 --> 00:50:45.900

Jisun An: So so in this example, if we are doing like the 4 pass. So assuming that we have these 3 inputs, 3, 1 minus 2, 4, A and B and C, then you can now doing the the d. Now our d value will be 2, because it's the 2 multiplied by B, and our E will E will be 5, because it's a d plus

197

00:50:46.411 --> 00:50:55.250

Jisun An: a, and then now our final ce will be minus 10, because it's the 2 minus 2 multiply by 5,

198

00:50:55.340 --> 00:50:56.070

Jisun An: right?

199

00:50:56.300 --> 00:51:06.200

Jisun An: So this will be our forward pest. So once again, even though we are using just a scholar value. This thing also can be now changing to like the vectors. And

200

00:51:07.671 --> 00:51:11.380

Jisun An: yeah. But these are just more simplified version of it.

201

00:51:11.790 --> 00:51:15.140

Jisun An: So then.

202

00:51:15.270 --> 00:51:30.659

Jisun An: Now, this computation graph is useful for this stack propagation, because it it it can help to for individual weights that how much it impacts to the the last function.

203

00:51:31.270 --> 00:51:55.889

Jisun An: And to do so, we need to get the derivatives for each of these intermediate modules. And once you and then, if you use also like the the chain rule, then you will eventually be able to compute these last derivatives by computing all the intermediate derivatives for the intermediate functions.

204

00:51:57.546 --> 00:52:10.739

Jisun An: So so here, these were the our initial function. And then these are de, are the our new kind of function that we needed for adding to our computation graph to get these values.

205

00:52:10.740 --> 00:52:30.180

Jisun An: But what we eventually want to know is the these 3 derivatives. So derivatives L, with respect to A, B and C, and what this eventually will tell us is that how much a small change in each of these value will affect L, so that will be this this value?

206

00:52:31.780 --> 00:52:56.686

Jisun An: So, in other words, if the derivative l of a is 5, then it means that if we increase the A by one, then the loss function L will increase by 5. So that's the what the slope actually means. Right? So we want to find these relationships so that we can determine how much we should update the A or B or C values

207

00:52:57.420 --> 00:53:00.979

Jisun An: and and that's the reason that we are computing these data tips.

208

00:53:02.010 --> 00:53:09.179

Jisun An: So once again, these are now some of the equations that is expanded using the chain rules.

209

00:53:09.670 --> 00:53:20.940

Jisun An: So. And and I put this computational graph bigger, so that because it's easier to follow the chain rule with the computational computational graph.

210

00:53:20.940 --> 00:53:43.219

Jisun An: So if you see now how we can, how the variable a weight, A or input a is relating back to the L, then you can see that it relates with the function E, and then the themselves a themselves. Right. So the derivative L with respect to a can be computed by derivative l. Over e.

211

00:53:43.430 --> 00:54:04.320

Jisun An: multiply by derivative e of a, so this will be, this will be just in will give us the posture for the variable A and for the variable B. Now it'll be a little bit more complicated, because there are 2 intermediary computation that it requires to get this derivatives.

212

00:54:04.320 --> 00:54:22.199

Jisun An: So starting from derivatives l. Over the of e, multiplied by derivatives of e derivative of e, of T multiplied by derivative D of B, so you will need all these 3 components to to get those value.

213

00:54:22.810 --> 00:54:40.179

Jisun An: And because these are relatively simple calculus, and also you can see that many of these derivative will be shared by other equations. So once you compute some of them, you should be able to get more of them, and maybe you can take your time to

214

00:54:40.570 --> 00:54:45.829

Jisun An: to to see how how the actual calculation can be can be done.

215

00:54:49.780 --> 00:54:50.886

Jisun An: So here,

216

00:54:51.640 --> 00:55:14.830

Jisun An: So given that we have all the necessary derivatives required for computing our like finer derivatives. These are the how the backward path kind of will will will be done. So the firstly the the first.st So because it's a backward path, we need to come from right to the left side. So we start from the L,

217

00:55:14.850 --> 00:55:23.746

Jisun An: the 1st thing that we can compute is the derivative L of with respect to E, and that is the

218

00:55:24.390 --> 00:55:47.389

Jisun An: Can you see, my okay, I should be brought the pen here. So if the 1st thing is here that we need to compute this one derivative L over the derivative with respect to E, and this is if we. So these are already computed. The equation. And this value is actually the c, and the C is basically the value C we had here. So this value is minus 2.

219

00:55:47.880 --> 00:56:03.930

Jisun An: So the derivative L with respect to E is the minus 2, and then what we need to compute is the derivative E over derivative of a, which is once again, we already computed this, these values. So this is the value one.

220

00:56:03.930 --> 00:56:25.510

Jisun An: And so now we have these 2 value of the partial L of E and partial E of a, then the partial l of a can be computed by multiplying these 2 values, so the minus 2 multiply by one will be minus 2. So we know that the the partial l of a is minus 2.

221

00:56:25.970 --> 00:56:52.189

Jisun An: So once again, it may take some time, but you can. These are relatively simple examples. So I think you can all follow this and and similarly, and also the easiest one would be just here. The variable weight. C doesn't have any intermediate computation, so you can simply get the value right away. So the derivative partial L of C was computed as just

222

00:56:56.089 --> 00:57:04.030

Jisun An: l of C, okay, just the value. E, so

223

00:57:28.170 --> 00:57:40.850

Jisun An: yeah, so so the the derivative L of C is a simply E, and the E is 5. So this value is 5, and you can also follow this. So we will now have these 3 values. So so these

224

00:57:41.360 --> 00:57:58.479

Jisun An: once again, what we were interested in are the derivatives of A and B and C, these are the 3 value that we were interested in because now, once we calculate these values, now we know what we know is that so by changing a value.

225

00:57:58.934 --> 00:58:13.519

Jisun An: Then it will have a by changing the A by one, then it will change the loss value as minus 2. But we so we want to minimize the loss value eventually. So we want to increase the A value.

226

00:58:13.530 --> 00:58:21.179

Jisun An: And similarly, for the variable B, now, the derivative was minus 4, meaning that it also has.

227

00:58:21.180 --> 00:58:45.490

Jisun An: If we increase the B, then the loss function will be like decreasing, like by the magnitude of the minus 4. But for the weight. C. It means that if we increase the C by one, then the loss will increase by 5. So we want to actually decrease the C, so that the loss function will go lower. So this basically, the derivative will tell you

228

00:58:45.490 --> 00:58:50.190

Jisun An: basically the directions and also the magnitude magnitude and

229

00:58:50.410 --> 00:59:16.289

Jisun An: together with the learning date, you can determine how much derivative value you can take into to update to your parameter. So in the this will happen for every single training example. So with a 1 single example, you do for the past, you compute the loss value, and then using the value you do back, probe, and then compute all the derivatives for different weights, and determining how you want to update each of the weights.

230

00:59:16.480 --> 00:59:26.809

Jisun An: and then you just repeat until certain threshold. So until the loss, the loss value will not basically change. So saturating or

231

00:59:28.184 --> 00:59:40.349

Jisun An: and then, after all this training, then your final model will have updated value of A and B and C, which are basically the your weights in the metric.

232

00:59:43.560 --> 00:59:44.860

Jisun An: Is this clear?

233

00:59:45.970 --> 00:59:47.240

Jisun An: Any questions

234

00:59:50.470 --> 00:59:53.821

Jisun An: so and this will be just

235

00:59:54.620 --> 01:00:24.219

Jisun An: far more harder and complicated in the rear neural network. I mean, that was just a like very simple example. But even now it goes to something in the rear 2 layer network that we just saw it where the hidden layer are using the relu and the up layer, using the sigmoid things will be a bit more complicated. But these are the the computation graph that has been decomposed of those network

236

01:00:24.220 --> 01:00:35.459

Jisun An: and so basically, the orange notes, which are the weights for the input and the hidden layers.

237

01:00:35.890 --> 01:00:39.579

Jisun An: hidden notes. So they are the weights that we want to update.

238

01:00:40.230 --> 01:00:57.110

Jisun An: So given. And then, so basically, you need to kind of need to know how to compute the derivative for each of these computation nodes. And but then the way to train the model will be very similar. So given the x 1 and x 2. In this network

239

01:00:57.110 --> 01:01:19.989

Jisun An: you will do the forward pass, which will give the at the end of the network, it will have the lowest value, and then, using that value, you now do the back prop to compute the derivative. Given that inputs and then update this weights and then you get the new example and you go backward and forward continuously, and then your model will be slowly, slowly trained.

240

01:01:20.190 --> 01:01:33.150

Jisun An: and the finer outputs of the weight will tell you like which features are most important and which combination of the features need to be activated together and etc, which we were discussing in the audio of this lecture.

241

01:01:34.380 --> 01:01:51.889

Jisun An: So these were like the very basic process of this neural networks so like. And you will, I mean from the next lecture you will see now how the language modeling have been using these neural networks. But the basic principles are the same. So you create a kind of model.

242

01:01:51.890 --> 01:02:14.800

Jisun An: It can be like sequential or non sequential but then you also need to create a computation graph that represent the computations. Then you calculate the results. And then you, if you are training, you are doing the back propagation and the update parameter. This will be like the very basic component of neural networks model, I mean training of the neural networks. And how we do work.

243

01:02:17.349 --> 01:02:40.790

Jisun An: So now like, I don't know like, what was your impression when you saw this visualization in the 1st place. But now it's kind of getting hopefully, it's getting more makes sense. So you have these inputs where once again, these are the Vectors, and it can be either one half vector or the like, the Cbo W, or like ski Gram models. And then

244

01:02:41.470 --> 01:02:52.422

Jisun An: you get you take them as a input and then you can have, like multiple hidden layers. So in this example, we actually have, like 2 hidden layers and

245

01:02:54.140 --> 01:02:55.240

Jisun An: and then

246

01:02:55.330 --> 01:03:06.579

Jisun An: and then, together with the bias, you can get this score, which is your output layers. So these are actually a 1, 2, 3, 3 layer network. I don't know whether you can. You can. You can see that.

247

01:03:06.620 --> 01:03:28.749

Jisun An: So the 1st part are the just, the the part where it was hidden in our visualization. So you assume that there are big table where it shows the word embeddings. And then you look up each of the word and get the vector so the first, st the red one will be your input layer, which we usually don't count as a layer, so that will be layer 0, and we have 2 hidden layers.

248

01:03:29.202 --> 01:03:36.817

Jisun An: And then the score layer will be our our output layer. So it's essentially, these are the 3 layer network.

249

01:03:37.450 --> 01:03:40.664

Jisun An: and yeah, so that would be

250

01:03:42.814 --> 01:03:51.510

Jisun An: what? What this neural network is. So yeah, just one more important concept. Oh, any any questions up to here?

251

01:03:52.760 --> 01:03:53.520

Jisun An: Yes.

252

01:03:54.090 --> 01:03:59.759

Jisun An: So see the bios one

253

01:04:00.368 --> 01:04:11.124

Jisun An: so how do we get the bias package? Is there something like we get some explanation like, what is bias? So so these bias are usually like,

254

01:04:13.320 --> 01:04:23.250

Jisun An: serving as a as a default value. So it's just initializing as a randomly, yeah, yeah.

255

01:04:25.300 --> 01:04:30.359

Jisun An: yeah, thanks for the question, any other questions.

256

01:04:30.960 --> 01:04:33.050

Jisun An: Wow, it's far more bright.

257

01:04:34.050 --> 01:04:52.949

Jisun An: Yeah. Just one last important concept. So we talked about gradient decent. And if you think back to the gradient descent equation, it was simply Wt plus one equal Wt minus running rate multiplied by the derivative value. Right?

258

01:04:53.560 --> 01:05:06.680

Jisun An: But then this is like like very, very old algorithm. And people do not use that anymore because it has many limitations, and one of the limitation is that it it's it. If I don't know. If you actually

259

01:05:06.680 --> 01:05:29.879

Jisun An: computing this derivative, I mean, if you think about different examples in the training data, then the derivative value can be like plus and minus, and it can really change rapidly. Then, if you are just throwing those value in the example. Then you will see that there's like very like variance that is changing all the time. But do you really want to update your parameter with that much amount of

260

01:05:29.880 --> 01:05:43.989

Jisun An: changes you probably don't want. So the atom, the the adaptive moment estimation is trying to kind of reserve that issue. So instead of using the derivative value, it is itself it using like the moving average.

261

01:05:43.990 --> 01:05:56.570

Jisun An: I mean, this makes sense right? So you don't want to update your parameter very significantly at very sudden, I mean at. But at the next step you just want to smoothing that using the moving average.

262

01:05:56.750 --> 01:06:02.879

Jisun An: And also so the the 1st moment is basically like the average. So you're you're keep

263

01:06:03.240 --> 01:06:28.140

Jisun An: keeping the direction. But you are using the moving average of that value. And then you are also using the second moment term. So you have the strength, and you want to use them like the square of those values so that you can highlight the strength, the magnitude of it. So if the derivative was like 5, then you scare them. So it became like 25. So you want to use the strength far more, and this will be very helpful in a case when

264

01:06:28.350 --> 01:06:36.764

Jisun An: you may have an example, that is very rarely happen in your training data. And in that case, you want to like

265

01:06:37.670 --> 01:07:01.428

Jisun An: use that opportunity to update your model as much as possible, so that also helps for reserving some of the low frequency kind of words. And also initially, this value tend to be near 0. So they also do some correction in the early in the training. So they adding these tricks here and there. So the final update, instead of using just

266

01:07:02.000 --> 01:07:18.564

Jisun An: So in the last moment, like learning rate multiplied by derivative, they are using the moving average of the derivative and also divided by the root of the the second moment. second momentum value, which will help you to

267

01:07:19.130 --> 01:07:37.559

Jisun An: is helping for this low frequency word to be updated better within the model. So these are the I mean, probably some optimizer that are mostly commonly used when you're training instead of the stochastic gradient descent. So that was the one last thing that that I'd like to talk today.

268

01:07:39.530 --> 01:07:43.090

Jisun An: Yeah, any questions.

269

01:07:47.330 --> 01:07:51.140

Jisun An: All right. So I mean.

270

01:07:51.290 --> 01:08:14.570

Jisun An: we still have a few few minutes. So I I will just give you, just a a bit introduction of the lecture. So, and also we are a bit behind of the schedule. So I may need to update our schedule a little bit. But what we were gonna do next week, and how all these are connected to will be so next next week we will talk about the language modeling, and

271

01:08:14.940 --> 01:08:29.040

Jisun An: everything that we have talked about will be connected back to the language modeling, I hope, and I hope hope you to see that how each of these modules are coming into these models as well. And then so

272

01:08:29.479 --> 01:08:41.516

Jisun An: so we have some word model word embedding a neural network which can be reserving some of the issues. But the last thing that hasn't been handled was the sentence structure. So now

273

01:08:42.660 --> 01:09:02.361

Jisun An: it. It may be better at handling words, and it may be handling better at combination of those features. But then if now we have, like the longer sentence, still, this model may not be able to capture how to interpret these models. And that's where the sequence model is coming in. So

274

01:09:03.609 --> 01:09:10.169

Jisun An: what is the language model? So basically, we want to assign a probability to a sentence.

275

01:09:10.817 --> 01:09:22.510

Jisun An: And anyone anyone knows about language modeling like, what kind of language model have you? Have you heard any any language modeling? Have you heard before?

276

01:09:23.850 --> 01:09:26.409

Jisun An: We also talk one in the last lecture.

277

01:09:31.109 --> 01:09:31.850

Jisun An: Sorry?

278

01:09:33.390 --> 01:09:52.161

Jisun An: Oh, Bert, yes, Bert, Bert is the one of the language model that is trained by one of the techniques of the language modeling. Yeah, so we we talked about the unigram model. So that is the actually the language model. So so Unigram and the Bygram models are the ones that we will start to talk.

279

01:09:52.830 --> 01:09:55.299

Jisun An: and yeah,

280

01:09:56.240 --> 01:10:21.200

Jisun An: the reason so. But the aim of the language modeling is basically, we want to assign a particular probability to a sentence, and the reason that we want to do is because it can be really, if we can correctly assign a probability to any sentence, then it can be useful for many different applications. So if we are doing like the machine translations, then, if we know that the high wind tonight

281

01:10:21.200 --> 01:10:44.779

Jisun An: is highly probable than the large winter tonight. Then we basically know that what would be the natural way to translate this or for this 3rd correction, if we see this kind of sentence like 15 min from my house, then we probably know that 15 min would be far more probable than the 15 min. So it would be easy to correct this word.

282

01:10:44.780 --> 01:10:58.399

Jisun An: or even when you do speech recognition, if you heard something, iso event. But it, if you need to determine between these 2 sentences, say, you know that this iso event is far more probable than the other. So

283

01:10:58.790 --> 01:11:09.349

Jisun An: so basically, we want to. If we, you can assign a probability to a sentence, then this can be really applicable to to various applications.

284

01:11:10.400 --> 01:11:23.810

Jisun An: So the language modeling is essentially, we are computing the probability of a sentence or the sequence word. So and if the the sentence is composed of the a set of words in sequence.

285

01:11:24.262 --> 01:11:37.550

Jisun An: then we want to compute this probability, and this is also very related to a task of the next word prediction. So, given a a few words, what would be. The next words coming in is the very related task.

286

01:11:37.900 --> 01:11:49.389

Jisun An: and those either compute a model that computes either this px or pxi, given px. One to X minus one. It is called as a language model.

287

01:11:50.310 --> 01:11:51.280

Jisun An: So

288

01:11:52.476 --> 01:12:07.409

Jisun An: then how can we compute this joint probability? Any any idea how you can compute this probability of what is likely that you will see the its water is so transparent.

289

01:12:07.850 --> 01:12:13.560

Jisun An: How can you compute this probability in a very simple way?

290

01:12:14.360 --> 01:12:16.140

Jisun An: Let's say you are reading a book.

291

01:12:18.460 --> 01:12:20.270

Jisun An: How can you compute

292

01:12:29.060 --> 01:12:30.100

Jisun An: so?

293

01:12:31.520 --> 01:12:39.270

Jisun An: So if you think about how you usually compute the probability, then? You have all possible

294

01:12:39.690 --> 01:13:04.199

Jisun An: events. And then you count the number of events actually happened. Right? So if it's not possible, probably not possible. But if you can count how many times you saw like it's water is so transparent in a book divided by like a number of all possible combination of these sentences, then that that may give you the

295

01:13:05.246 --> 01:13:13.850

Jisun An: probability. But usually that is not possible, and that's the reason that we could rely on like a chain rule of the probability.

296

01:13:15.260 --> 01:13:16.330

Jisun An: And

297

01:13:17.050 --> 01:13:44.929

Jisun An: that's what is coming in the next lecture. So we will. Now talk about how we can model the language. And then I, and then so we. And we start from something very simple, which is like Unigram, or the diagram which we will talk more, and they will lead to how we can use the neural network to do the language modeling hopefully where we can combine all the concepts that we discussed in the previous lecture to be presented in that model.

298

01:13:46.490 --> 01:13:49.410

Jisun An: All right. Yeah, any questions.

299

01:13:52.010 --> 01:14:01.927

Jisun An: Okay, cool. I hope this was not too difficult. And hopefully you can go through those examples again by yourself, so that you can fully understand backward and forward.

300

01:14:02.480 --> 01:14:08.430

Jisun An: and let me know if you have any other questions. Yeah, thanks a lot. Have a great weekend, and I will see you next Tuesday.

301

01:14:11.936 --> 01:14:12.529

Jisun An: Thank you.

302

01:14:13.460 --> 01:14:14.240

Jisun An: You know.