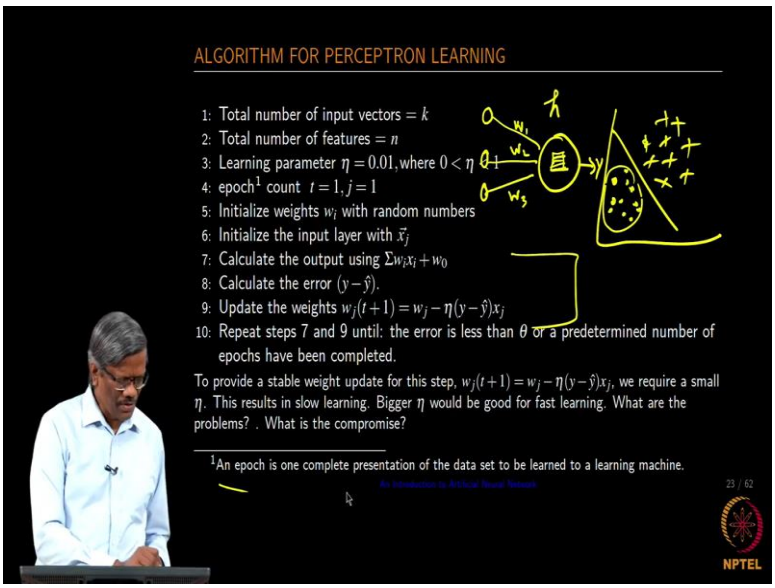
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 34
Perceptron Learning

(Refer Slide Time: 00:15)



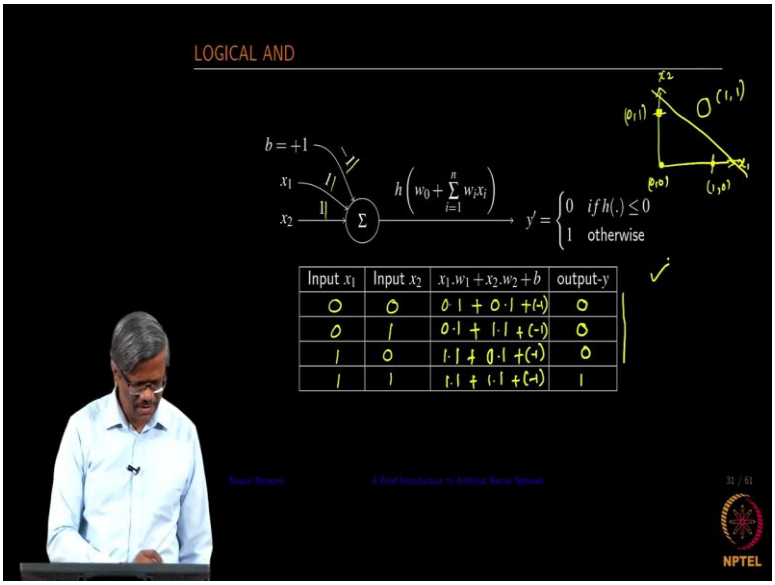
Alright, we will continue on the Perceptron Learning part of that is the last session I showed you the algorithm for perceptron learning. Step 7 to 9 of crucial right; so, it is a repeated operation. For what you can do with the perceptron is you know you would be able to classify objects that are in the linear space ok.

And one more thing is the objects that we are going to be classifying should be linearly separable as well right. So, once we satisfy this condition, it is possible for you to use the perceptron. So, remember in this perceptron we have only one neuron, even though we have neurons in the input space, they just pass on the value and only the computation is happening in this space right.

So, this is where we do the computations or we get the value of h, and then we run it through an activation function, and then finally, the output is computed. As I mentioned earlier knowing, we are not going to be just using one input we are going to be having lots of input in order for you to classify like we had shown here correct. So, in order to

create this boundary, you can either start with just one and then start learning that or present the input completely. So, this is what we saw in the previous session alright.

(Refer Slide Time: 01:53)



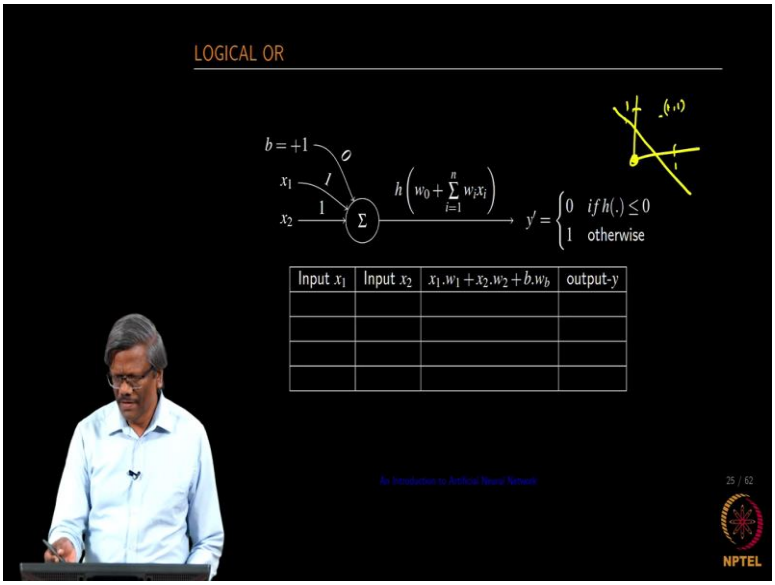
So, now let us take a very small example, and then see how perceptron behaves ok. Since this is a linear classifier we are going to be taking a logical AND right. So, we have the values here as 1, this is our 0, we have x_1 and x_2 ok. In the AND operations, we have 0s here at 0 0 1 0 and 01. And then we have a_1 here at 1 1 right. Let us see whether the neural net is able to compute this. So, in this case, the weights that are presented here already learned to help you understand that the perceptron really creates a linear boundary I am just using the weight.

So, this is one of the many combinations that you will find in the weights ok, there could be several combinations of the weights. It need not be minus 1 for the bias connecting to the neuron, and then one connecting the x to the neuron and so on. So, it could be any number. So, in this case, let us take those inputs, let me first write this. And then start combining this. So, this is 0 and 1 plus 0 1 plus, and b is minus 1 right. So, that means, when the values less than or equal to 0 we are going to make it 0 ok.

So, now, let us take this one 0. 1 plus I am doing a multiplication 1 .1 plus ok. So, this is again 0 right the same fashion 0, and here it is 1 1. 1 plus so and so this is 1. So, it is actually separating these from the real value right. So, we have elements here on the side

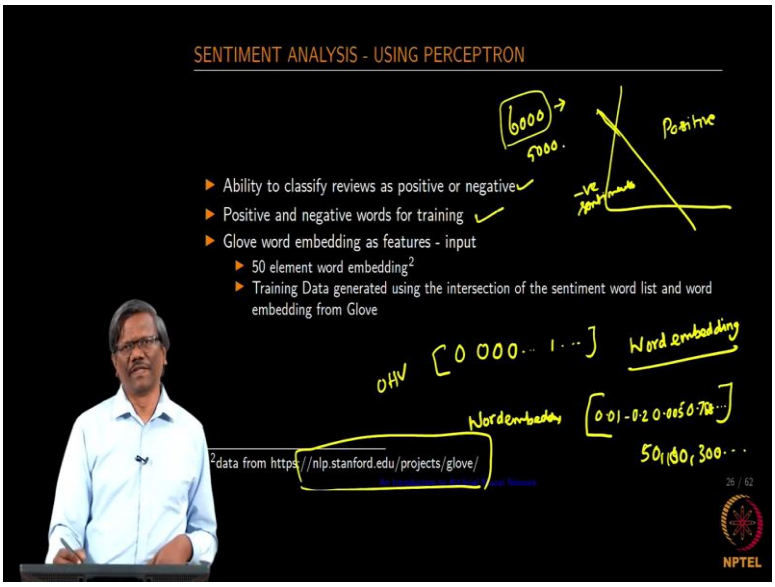
below this decision boundary, and there is one element which is 1 1 for this. So, our perceptron really classifies, it is able to solve this problem.

(Refer Slide Time: 04:54)



So, in the same fashion I am not going to be doing this for or you can do this. Again in this case you should be able to separate the 0 out, these are all values using a straight line in this fashion ok. So, try this operation and then using the weights given or try to find out it is really linearly separating this ok.

(Refer Slide Time: 05:25)



Let us take an example from the natural language processing, and then see how we could create a linear boundary between positive and negative sentiments. So, the idea is to really draw a boundary, so that all the positives sentiment words are on the right side of this boundary and the negative sentiments on the left side of the boundary ok.

So, on this website you have a lot of data related to words. So, we have word embedding. So, let we will talk about that in the next lecture or so. So, how the word embedding could be obtained using one heart vectors ok. So, we will talk about that later, but assume that we have captured this in some fashion. So, let us assume that instead of 0 or 1 given as the input, let us use the word embedding which contains values in this fashion.

Let us say your one heart where there would be like this, if you take some positive word so this is your one heart vector. And then word embedding for the same thing would be like this ok. So, this is a word embedding vector that contains about fifty elements I am only taking the lowest possible so that the computation can be done a lot faster. So, you have up to we have 50, 100, 300-word vectors are available as part of the glove implementation.

So, what they have done is they have taken close to 1 billion words and then provided the one rather the word embeddings for each of the words that they have computed. So, this we can use as your input ok. So, in this case we are going to be taking the positive or negative reviews, and then trying to classify them right. And then we also require positive and negative words for training. So, in here what I am doing is for the sake of convenience, I have downloaded positive and negative words, they're about 6000 in number.

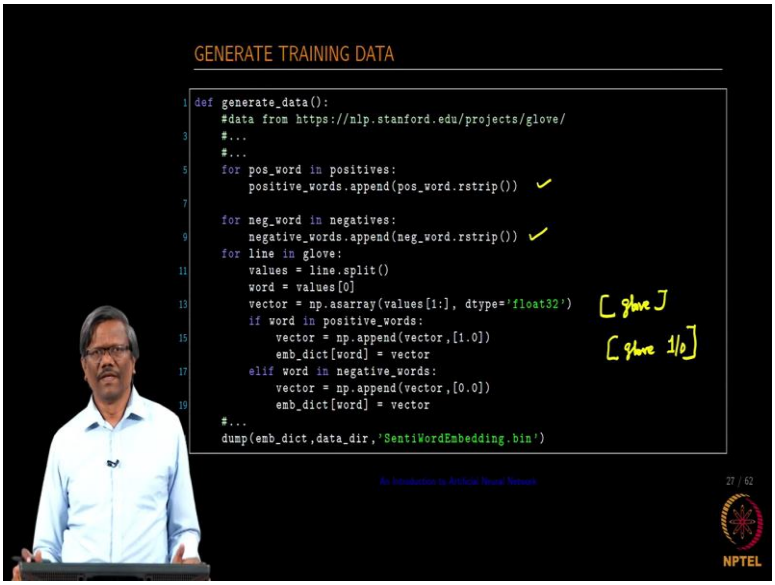
Again these are available as open-source you can capture them. So, what they have done is they have classified a set of words as positive words, set of words as negative words and you can take that. And one more thing that you may have to do is, you have to do some input preparation. So, out of these 6000 words you know it is either possible or not possible to find those words in the word embeddings that are available from the glove website.

So, make sure that the words that you are taking are present in the embedding that is one of the most important things that you have to do right. So, I have taken those 6000 words from the website from some website I am sorry I have not listed it here, but you can

search for positive or negative words you will find at least a couple of websites listing those words.

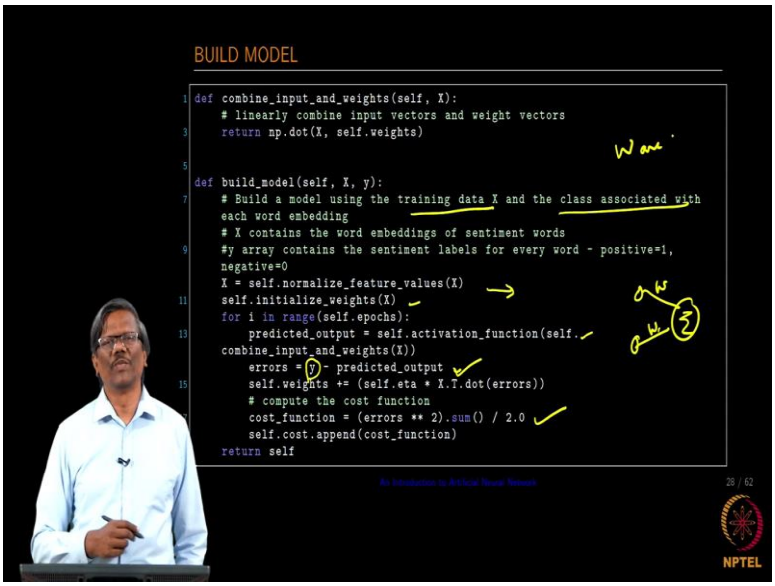
And then take each one of them, and try to find out whether the word embedding is available or not. And then again you get one more collection which could be equal to 6000 or could be less than 6000 words ok. So, that does not matter. So, for the sake of our understanding even if you get about 5000 words from the word embedding and if they are matching that is good enough for us. I am going to be using that set of words as input for this application ok.

(Refer Slide Time: 09:30)



So, this is what I explained to you know I need to generate the training data set first. So, what I am doing is, I am just taking the positive words, and then creating a list in the same fashion, creating a list here as well. And then using the glove data, I am actually separating them out. So, when I do this, what I create is, I just create a vector that is coming from the glove. And I know that it is a positive word or a negative word ok, and then I attach that element 1 or 0 to the end of that ok. So, in this way I am creating the dictionary of words and then start building that model.

(Refer Slide Time: 10:27)

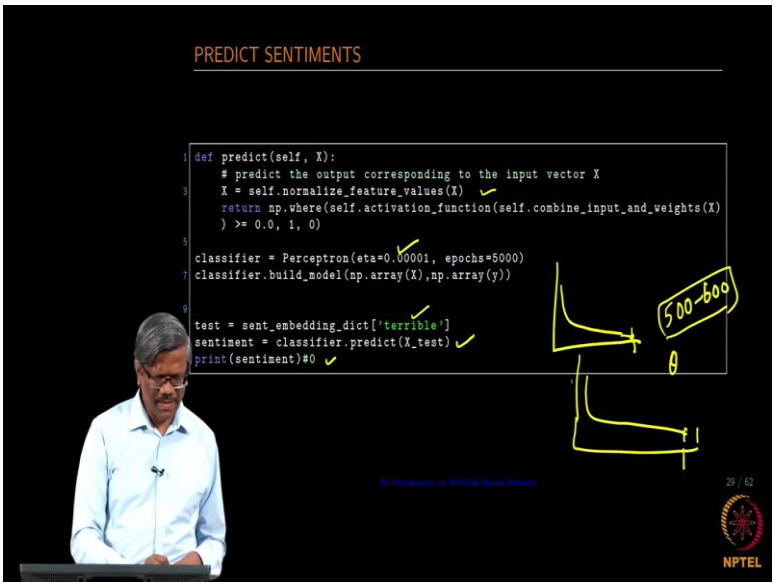


So, how do you build the model? So, build the model using the training data x and the class associated with each word in the embedding ok. You normalize the values sometimes it is required, sometimes it is not required, I am just doing the normalization of this using a function called normalized future values. And then I do the initialization of the weights that we have here, these weights I am initializing.

And then start computing the predicted output. And then find the errors. Since I know what is the value of the word right with respect to the sentiment, I find out whether it is close enough or not close enough; if it is not close enough, I compute the error and then adjust the weight, so that in the next iteration it becomes closer to the actual value ok.

Create the cost function; cost function is actually the sum of all the errors that you are getting every time right. So, when you are running through the entire (Refer Time: 11:54) for every word there is a loss created with respect to the actual target value and the predicted value. And then you keep adding that error into the cost function ok. And then finally, build the model wherein your Ws are computer appropriately ok.

(Refer Slide Time: 12:21)

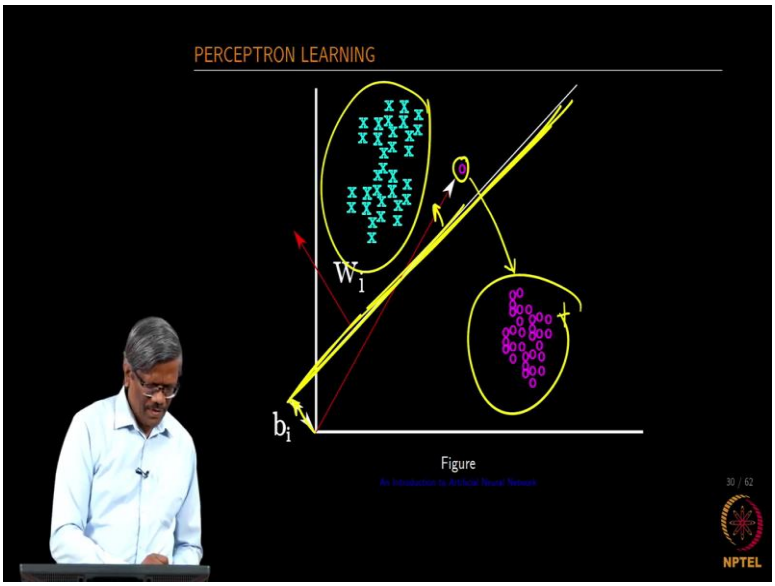


And then once the weights have stabilized, so that is when the perceptron is stable to predict the new input that you are going to be feeding in. You do the normalization very similar to what he had done during the model building. And then test the embedding word right. So, you take one word from the embedding and then feed that, and then find out whether the classifier really predicts the right value or not. So, in this case it has predicted. So, I think I have run close to about 500 to 600 iterations for all the input words.

And then when you see that the cost becomes closer to some value, so you set a threshold value ok, if the value is less than this, stop the iteration. Or if you think that, it is good enough, it is not going to improve beyond the point, for example, you will see some flattening happening in this fashion. So, when you see that the error between the previous and the current one is extremely small and it is not improving further, you can stop.

Or you can keep a fixed iteration number or the (Refer Time: 13:47) number or (Refer Time: 13:48) count and then stop the iteration at that point ok. So, this is pretty much you can do with perceptron where if you have linearly separable values. It is possible to use the perceptron to separate to create the boundary and then separate them off.

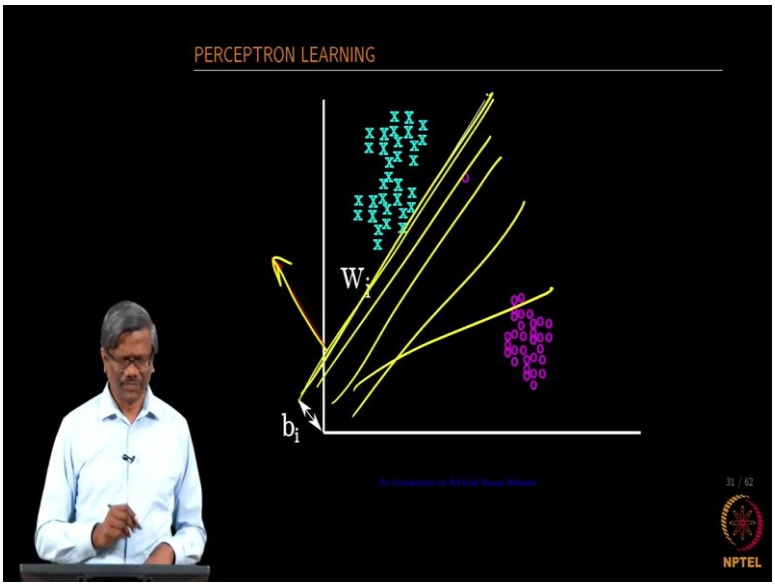
(Refer Slide Time: 14:04)



And you can see when you do the iteration right through the (Refer Time: 14:11) count, you can find that the decision boundary keeps shifting between values, sometimes you know it will start with this because the majority of the values provided here or here, and then you also have this one here, and then this one here. So, it tries to separate this first.

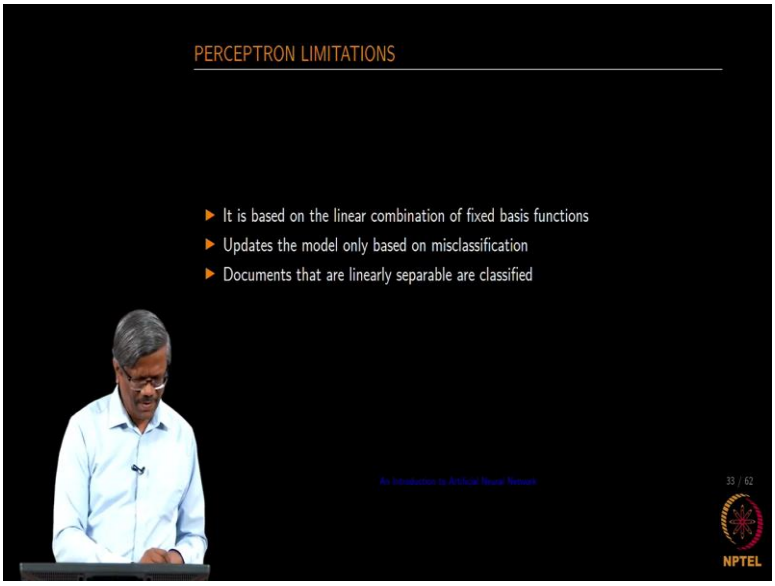
And then suddenly when it encounters a value which is belonging to the negative class or the positive class, in this case let us assume that this belongs to the negative class. It will start moving the decision boundary from here to another place. And, then you can see that the bias is actually pushing the boundary from the origin so that it can align with the right boundary line ok.

(Refer Slide Time: 15:03)



So, now, it will try to move it and your W would be in this fashion. You can keep looking at that right, there is a way you can plot how the error lines are really moving up and down. You can see that it will start from here, and then slowly adjust, and finally, move to the right spot ok. So, this is what the final one that you will see ok.

(Refer Slide Time: 15:32)



So, what are the limitations as we have been talking about this, it can only separate linearly separable asset ok. So, we can only do it based on the miss classification right.

So, it is a very simple computation model, where you can use it for a certain purpose. So, it is well-suited for your sentiment analysis in natural language processing.