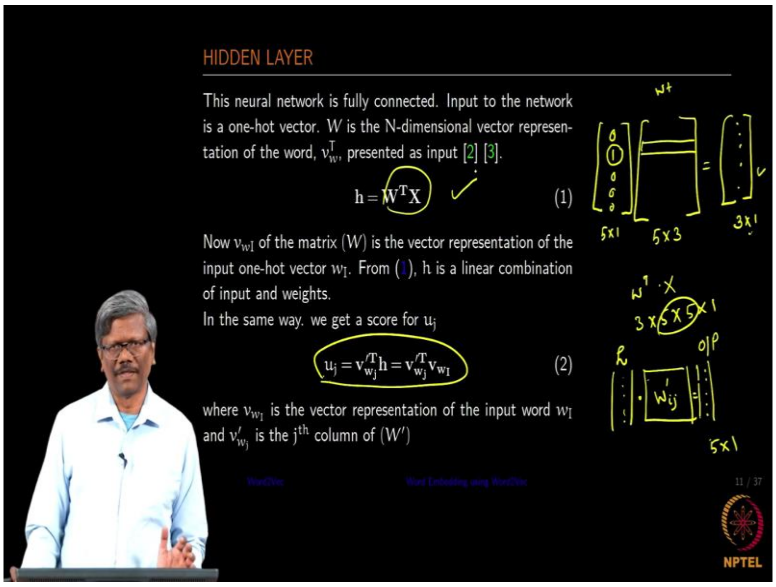
Applied Natural Language Processing

Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute

Lecture – 42
Forward pass for Word2Vec

(Refer Slide Time: 00:15)



HIDDEN LAYER

This neural network is fully connected. Input to the network is a one-hot vector. W is the N-dimensional vector representation of the word, v_w^T, presented as input [2] [3].

$$h = W^T X \qquad (1)$$

Now v_{wI} of the matrix (W) is the vector representation of the input one-hot vector w_I. From (1), h is a linear combination of input and weights.
In the same way. we get a score for u_j

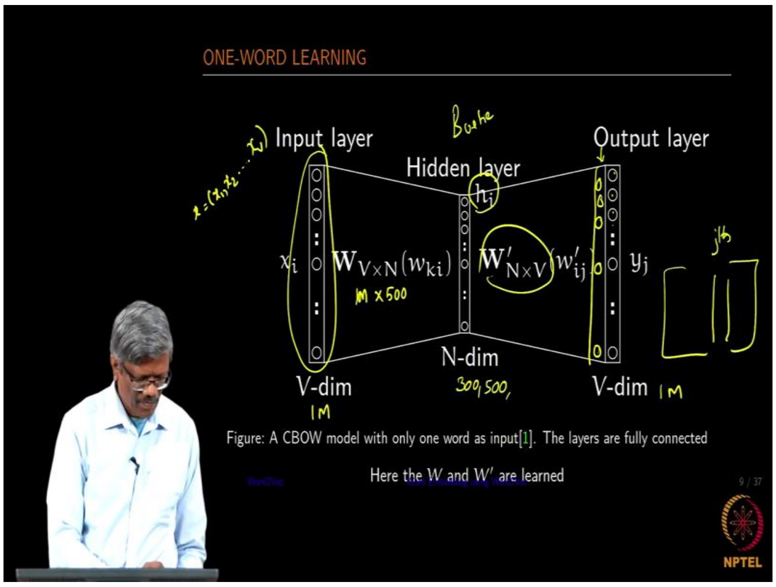
$$u_j = v'^T_{w_j} h = v'^T_{w_j} v_{w_I} \qquad (2)$$

where v_{w_I} is the vector representation of the input word w_I and v'_{w_j} is the jth column of (W')

11 / 37

Let us now see how the hidden layer values are computed. In this case, the network is fully connected, the input to the network is a one-hot vector as I mentioned earlier.

(Refer Slide Time: 00:31)



ONE-WORD LEARNING

Figure: A CBOW model with only one word as input[1]. The layers are fully connected

Here the W and W' are learned

9 / 37

Again W is the weight on the input hidden layer side and then W dash is the weights on the hidden layer to the output layer side. When we compute the hidden values in the hidden layer, we use a dot product to get that right. So, this is represented in the vector form right now, so the bold letters we have the vector representation.

As I mentioned earlier in the earlier session use linear algebra packets, so that you can directly compute this instead of going through the loop method ok. So, in this case, what happens is you have the hot vector like this and then you have the weight vector. And what you want to find is your h right. So, let us say this is 5 by 1 and then we want to have the weight victor which is of size, we want to have the hidden layer of size let us first define that as 3 by 1. Then what is going to happen is this is going to be a 5 by 3 matrix right.

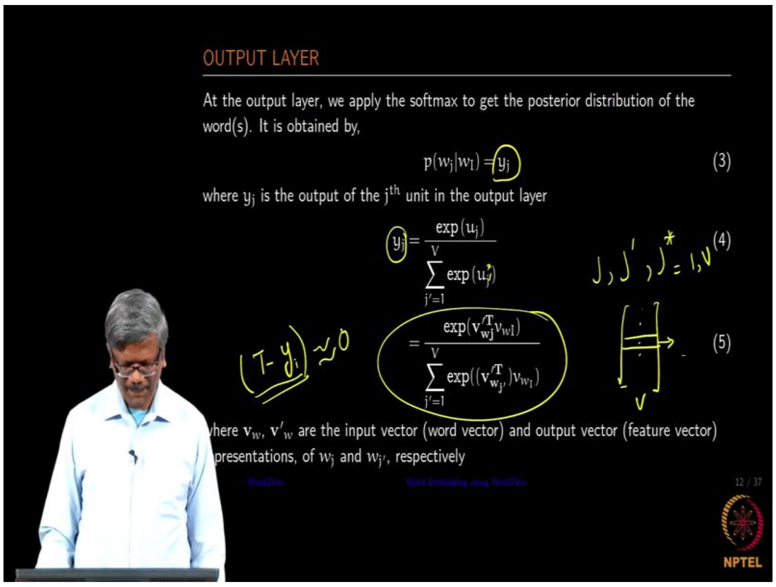
So, what you do is you just do the W transpose into X that is 3 by 5 into 5 by 1 will give you 3 by 1 here ok. So, when you do this operation, what happens is it actually copies the row in the weight vector and then places them here ok. Since all the values are here, so it does not matter whatever value that you have in the weight, it only takes the row corresponding to the value of 1 in the one-hot vector and copies that particular row into the hidden layer is ok.

So, now what we going to do is, we going to be creating some more representations ok. So, we have done the calculation of h using this model. So, now, we have the other side from the hidden layer to the output. So, we apply a similar mechanism to compute u j, u j is equal to the v w j dash transport and h. So, you remember there is a hidden layer, and then there is an output layer and then we have another weight matrix right. So, this is computed.

So, we want to do the dot product of these two and then create the output layer right. And this is what is shown here. U j is computed by taking these values and then using the W is I j and we get the output values as u j. And the size of this would be equal to 5 by 1 ok. Clear with this? So, we have computed first the hidden values. And then from the hidden values to the output, again we have a matrix here. So, we have computed this and then using this, you are going to be computing another set here right.

So, let me use one more layer to clearly explained what happens. In this output layer, the values are computed. And then later we apply the softmax to get these values ok. And this is what is shown in this equation ok. So, that output values are computed before applying the softmax and u j represents that output values ok. So, in this case v w is the vector representation of the input word that we have provided as the one-hot vector and then v w j is the jth column of the w. So, in this case here j the column. So, you have this matrix here and then we are taking the jth column of the ok.

(Refer Slide Time: 05:53)



OUTPUT LAYER

At the output layer, we apply the softmax to get the posterior distribution of the word(s). It is obtained by,

$$p(w_j|w_I) = y_j \tag{3}$$

where y_j is the output of the j^{th} unit in the output layer

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{4}$$

$$= \frac{\exp(v'^{T}_{wj} v_{wI})}{\sum_{j'=1}^{V} \exp((v'^{T}_{w_{j'}}) v_{w_I})} \tag{5}$$

where $\mathbf{v}_w$, $\mathbf{v}'_w$ are the input vector (word vector) and output vector (feature vector) representations, of w_j and $w_{j'}$, respectively

12 / 37

And then later what you do is as I mentioned earlier in the output layer, we are going to be applying the softmax that is where we are going to get the real values of the predicted word correct. So, the real value would be computed using this $u_j = V'^{T}_{w_j} \; h = V'^{T}_{w_j} \; v_{w_1}$.

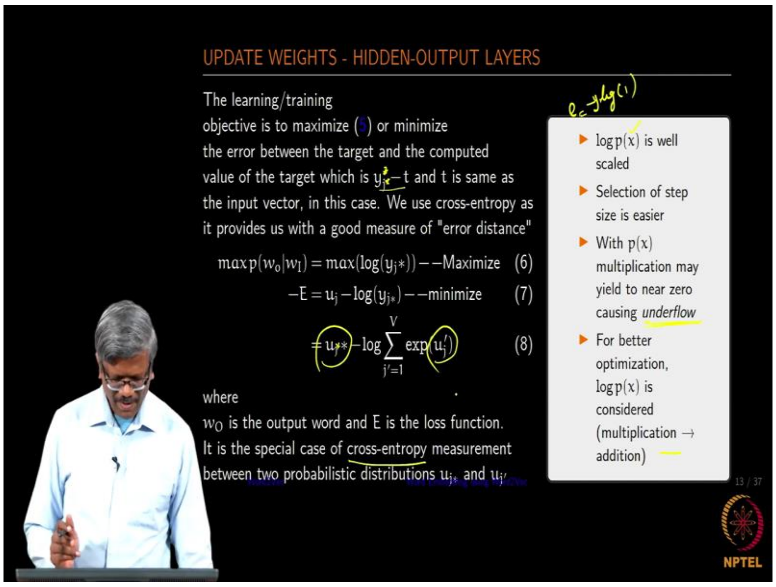
So, these are all these indexes that I will be using. And each one would be having the value from 1 to v. Just to distinguish between the input, the output and the predicted elements, we will be using three different indexes here ok.

So, we are going to be predicting the value of y_j using this model. So, in this case, we go to get 1 vector of size v ok. And then we want to find out what is the actual error, what is the actual value predicted. And then if the actual value predicted is the same as what is input, then there is nothing much, we can do the system has learned it. If there is a difference, we will take the error and then do the backpropagation algorithm ok. So, it is very similar to what we had seen earlier. Again I am going to repeat this one more time.

So, we will compute the values in the hidden layer using this formula right. And this is nothing but just a copy of one of the rows corresponding to the input hot one-hot vector.

And then it from the hidden layer to the output layer, we have j, using that value we are going to be computing the output values and u_j is our output value ok. And then each column of the w_{1_j} is used for the computation of this u j values ok. And then later in the output layer, we apply the softmax, so that the probability is distributed across the vocabulary z all right.

(Refer Slide Time: 08:59)



UPDATE WEIGHTS - HIDDEN-OUTPUT LAYERS

The learning/training objective is to maximize (5) or minimize the error between the target and the computed value of the target which is $y_{j*} - t$ and t is same as the input vector, in this case. We use cross-entropy as it provides us with a good measure of "error distance"

$$\max p(w_o|w_I) = \max(\log(y_j*)) -- \text{Maximize} \quad (6)$$

$$-E = u_j - \log(y_{j*}) -- \text{minimize} \quad (7)$$

$$= u_{j*} - \log \sum_{j'=1}^{V} \exp(u_j') \quad (8)$$

where

w_O is the output word and E is the loss function. It is the special case of cross-entropy measurement between two probabilistic distributions u_{j*} and $u_{j'}$

- $\log p(x)$ is well scaled
- Selection of step size is easier
- With $p(x)$ multiplication may yield to near zero causing *underflow*
- For better optimization, $\log p(x)$ is considered (multiplication $\rightarrow$ addition)

So, the next question is how do we really update the weights? So, this is very similar to what we had seen in the backpropagation model. There is no change in the way we are operating right now. The only change that we are going to be introducing is bringing a different error mechanism ok or bring in a different model for the error and use it for making the corrections in the weights as we move along right. So, we have the value that is computed. Supposing, if this is the target and we have y_j, I want to make sure that this value is minimized ok. Is given equation

$$\max \quad p(\frac{w_0}{w_1}) = \max(\log(y_j *))$$

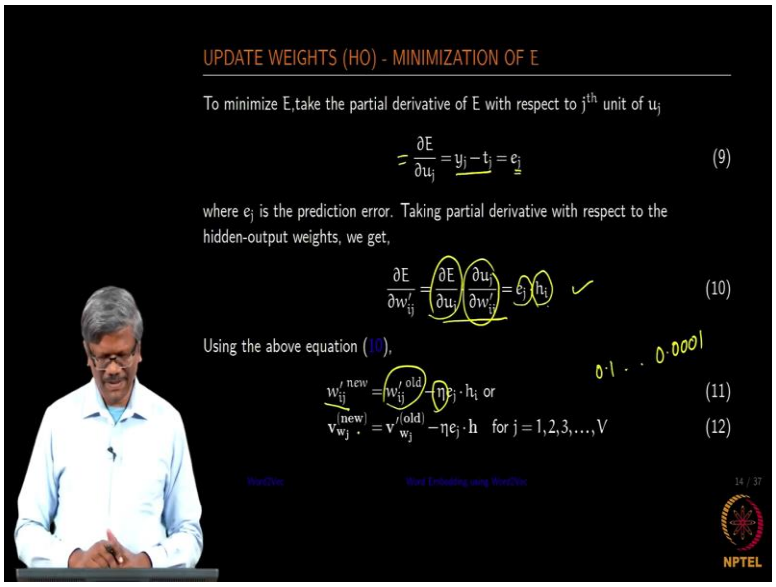
$$-E = u_j * - \log \sum_{j'}^{v} \exp\ (u_j^1\ )$$

So, using y_j we are going to be minimizing the error and then backpropagate the error into the weights, and then keep repeating that until it becomes very close to 0 or certain condition that we have defined ok. So, in this case, as I mentioned earlier there is a separate index created for the softmax layer as well that is called j^* ok, $y_{j^*} - T$ is going to be our error value. So, we want to be able to minimize that error right.

So, minimizing the error is the same as maximizing the probability of that predicted word. So, we are going to be using a different scale here in terms of the errors. We are going to be using loss function which is called the cross-entropy measurement. So, why do you want to use this mechanism? So, if we look at the log p x or the predictive value that we have here it is very well scaled ok. Selecting the step size to move forward in terms of the error correction is easier in this. And then when we introduce the probability and then when you start multiplying them you know the values become smaller and smaller as we progress and sometimes it leads into the underflow.

So, instead of multiplication, now we try to get into the logging framework, where the multiplication becomes an addition to right, so that is the reason why we want to have this. The one more advantage also, it provides a good measure of error distance. For example, if the value becomes 1 ok, so the error value would be which would be equal to 0 ok. So, we normally do it y into this. So, if this is if there is an error value which is not equal to 1, which is smaller than 1, this value will be larger. So, it gives us a good visual feeling with respect to how the error is getting reduce, so that is the reason y also we use the cross-entropy measurement for error reduction. So, we can also see that it is a very special case of cross-entropy measurement between two probability distributions here all right.

(Refer Slide Time: 12:41)



UPDATE WEIGHTS (HO) - MINIMIZATION OF E

To minimize E,take the partial derivative of E with respect to j^th unit of u_j

$$= \frac{\partial E}{\partial u_j} = y_j - t_j = e_j \tag{9}$$

where e_j is the prediction error. Taking partial derivative with respect to the hidden-output weights, we get,

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial w'_{ij}} = e_j h_i \quad ✓ \tag{10}$$

Using the above equation (10),

$$0.1 \ldots 0.0001$$

$$w'^{\,new}_{ij} = w'^{\,old}_{ij} - \eta e_j \cdot h_i \text{ or} \tag{11}$$

$$\mathbf{v}^{(\mathrm{new})}_{\mathbf{w}_j} = \mathbf{v}'^{(\mathrm{old})}_{\mathbf{w}_j} - \eta e_j \cdot \mathbf{h} \quad \text{for } j = 1,2,3,\ldots,V \tag{12}$$

14 / 37

NPTEL

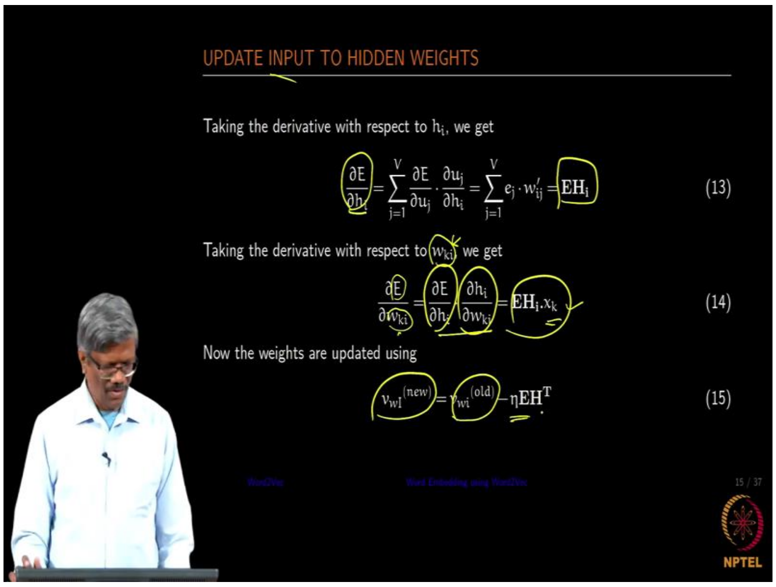
So, how do we now update the weights? So, we update the weights using the partial derivative that we have seen. So, we know that the error in the predicted value is given by y minus t j which is equal to e j and this is what we want to minimize correctly. So, e j is our prediction error. So, we want to be able to find out with respect to the w'_{ij} that we find between the output and the hidden layer right. So, when we want to do that, we use the chain rule which goes like this

$$\frac{\partial E}{\partial u_j} = y_j - t_j = e_j$$

$$\frac{\partial E}{w'_{ij}} = \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial w'_{ij}} = e_j \, h_i$$

So, now, we want to update the weights. How do we update the weights? So, now, we have some learning parameter as we had mentioned earlier in the previous backpropagation network in the same case, we will have a learning parameter and we will have the value between 0, 1, 2 some depending on the network weights and all that ok. So, in this case, we are going to be computing the new weights using our old values ok. So, once this is computed, now we have a new set of a matrix in the hidden to the output layer correct. So, based on the error reduction mechanism, the values have changed all right.

(Refer Slide Time: 14:53)



UPDATE INPUT TO HIDDEN WEIGHTS

Taking the derivative with respect to h_i, we get

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} = \mathbf{EH_i} \qquad (13)$$

Taking the derivative with respect to w_{ki} we get

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w_{ki}} = \mathbf{EH_i}.x_k \qquad (14)$$

Now the weights are updated using

$$v_{wI}{}^{(new)} = v_{wi}{}^{(old)} - \eta \mathbf{EH}^T \qquad (15)$$

Word2Vec Word Embedding using Word2Vec 15 / 37

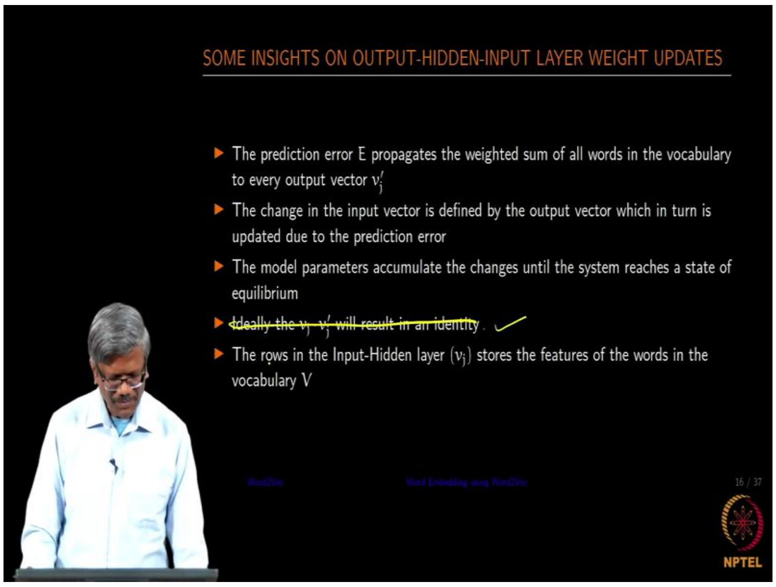
So, in the same fashion, we should be able to reduce the waste between the input and the hidden weight. So, we have moved from the error from the output layer to the softmax layer and then propagated the error back to the weights. And now the weight vectors are changed. From the changed weight vectors, now we have to move that backroom to the input to the hidden layer so that the values there in the input of the hidden layer weights are also changed ok.

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w_{ki}} = EH_i\, x_k$$

So, again taking the derivative with respect to h_1, we get the value of EH i. We are going to be taking the derivative with respect to w_{k1} which is nothing but the input to the hidden layer weights right, so that we are going to be changing. For us to change that weight, so we need to differentiate that or do the partial differentiation with respect to w_{k1},and we have the error and we have to change it with respect to this right. Using the rules, chain rule, again we will rewrite this equation in this fashion and then find the values that we want to use to update the input weight ok.

So, since this is known and then if you do the partial differentiation with respect w_k, what you will get is the input value right. And then the weights are updated using the old values and the here that we have computed which will be multiplied by the learning parameter in this case ok. There is this clear? All right. So, let us move on to the next one.

(Refer Slide Time: 16:57)



SOME INSIGHTS ON OUTPUT-HIDDEN-INPUT LAYER WEIGHT UPDATES

- The prediction error E propagates the weighted sum of all words in the vocabulary to every output vector v'_j
- The change in the input vector is defined by the output vector which in turn is updated due to the prediction error
- The model parameters accumulate the changes until the system reaches a state of equilibrium
- ~~Ideally the $v_j \cdot v'_j$ will result in an identity~~. ✓
- The rows in the Input-Hidden layer (v_j) stores the features of the words in the vocabulary V

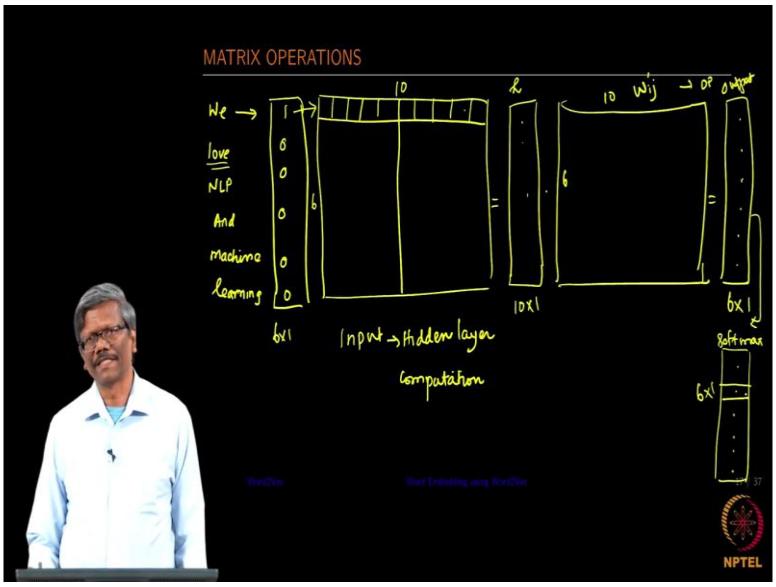
Word2Vec Word Embedding using Word2Vec 16 / 37

NPTEL

So, these are some of the insights that you will have. The prediction error propagates the weighted sum of all words in the vocabulary to every output vector right. The change in the input vector is defined by the output vector which in turn is updated due to the prediction error correct. The model parameters accumulate the change until the system reaches a state of equilibrium, which means, the model parameters are those which are computed using the error propagation.

And finally, when there is no more change that can be accommodated, these freezes that that is when the model parameters are created and ready for testing our model. This particular bullet point is meant for the same word learning in terms of learning the same vocabulary words and then creating the word vector form, but in our case this is not true. The roles in the input-hidden layer store the features of the words in the vocabulary ok. So, once the training is completed, what we have is the embedding of that particular vocabulary that we try to learn ok.

(Refer Slide Time: 18:39)



So, let us now look at some of the operations that we had seen earlier in terms of the matrix form. So, what I am going to be doing is, I am going to be really drawing those matrices, so that whatever equation that we get seen and whatever error propagation that we try to define using the equation will become very clear ok. So, what we let us take one small example actually I have notes taken for this so that I do not make any mistakes while repeating this. So, I am going to be copying those values from the notes here.

Let us say we have the words that we want to train ok. So, this is the sentence that I want to train. And then if you want to look at the corresponding one-hot vector let us say we're going to be taking this word, it is going to be like this correct. So, this is going to be our one-hot vector corresponding to we. So, this is of size 6×1. And then let us say that we are going to be constructing a matrix of size 6×10. So, this is just an example.

So, I am just taking a matrix of size. So, this is our w_{k1} in the earlier case remember. So, if you look at this, we will have about I am going to draw only 1 row here ok. So, we have 6 rows in this. So, the corresponding row will be just copied as I mentioned earlier into h ok. So, when we do the linear combination of this, this will be copied into this. So, this would be 10×1. So, how do you obtain this? You take a transpose of this and then do a dot product with this input value and you get this 10×1 matrix ok. So, this is our input to right.

So, what is the next step? Using this we use another to create our output correctly. So, we use this the matrix size is the same it is 10 × 6 ok. So, we have 10 elements in this. And then doing a dot product of this would give you the output. Since the size of the output will be very same as the input which will be equal to 1 ok. So, in this case since we are using a bag of words, for the input word v this network should learn love right. So, we have some values 6 values are computed. And then for this will take a softmax ok and then the softmax will do the property distribution between 0 and 1 for all the values. And we need to find the corresponding context word right ok.

So, let us see how that happens. So, in this case let me draw the softmax here. So, after this, we compute the softmax again of the same size 6 × 1 ok. So, we have computed this softmax for this and this softmax will again give you 6 values. So, let us assume that this value is what you want to predict right. So, it would have given values for each of this since the initial values of the weights are random, we cannot expect this word to have the maximum value. So, it will have some random values because of the weights that we have created in these two cases ok. So, now, once this is computed, we want to find out the error between the predicted value and the target value.