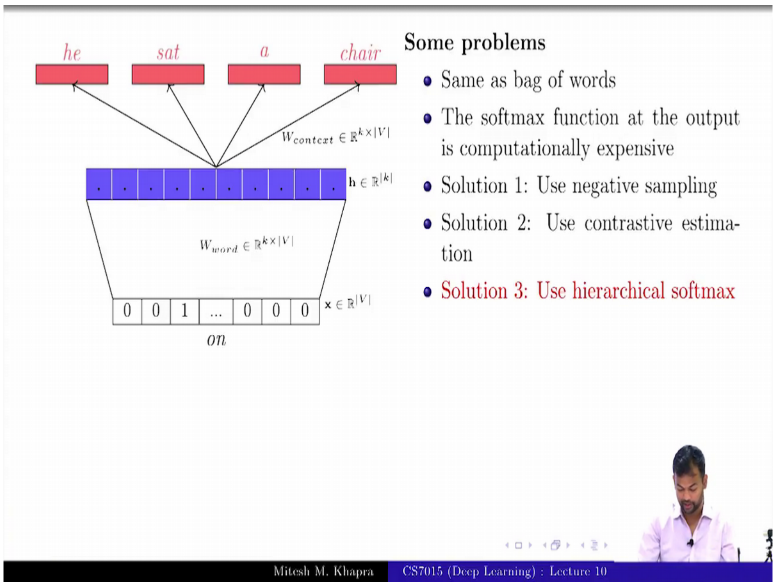
Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 10.7
Lecture – 10
Hierarchical Softmax

(Refer Slide Time: 00:16)

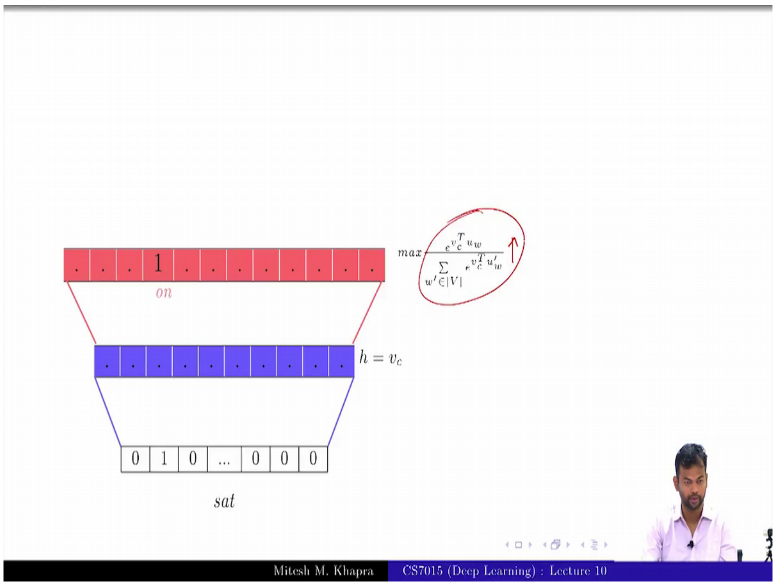

he
sat
a
chair
$W_{context} \in \mathbb{R}^{k \times |V|}$
$h \in \mathbb{R}^{|k|}$
$W_{word} \in \mathbb{R}^{k \times |V|}$
0 0 1 ... 0 0 0
$x \in \mathbb{R}^{|V|}$
on
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 10

Some problems

- Same as bag of words
- The softmax function at the output is computationally expensive
- Solution 1: Use negative sampling
- Solution 2: Use contrastive estimation
- Solution 3: Use hierarchical softmax

The next one is a bit tricky so, the third solution is to use something known as Hierarchical Softmax. This is a bit counterintuitive in the sense it is a very smart trick, but it is not something which is very obvious. So, just pay a bit attention on this it is a neat way of handling this large vocabulary thing. And this I think used in various and it will be applications, where speed is important not often, but wherever speed is important.

(Refer Slide Time: 00:35)

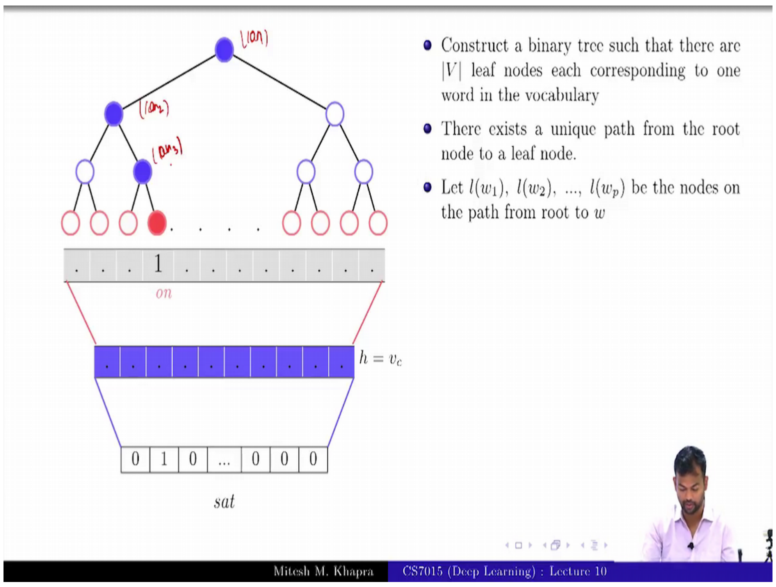


$max \frac{e^{v_c^T u_w}}{\sum_{w' \in |V|} e^{v_c^T u_{w'}}}$
1
on
$h = v_c$
0 1 0 ... 0 0 0
sat
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 10

So, this is what our original network was. This was the either you take it as a skip gram model or you take it as a continuous bag of words model, right. Let us take it as a continuous bag of words model.

You had a word as the input, and then you had this large prediction, and you had this softmax computation which gives you the probability, and you are trying to maximize this probability for the correct word, right where V w is the correct word?

(Refer Slide Time: 01:04)

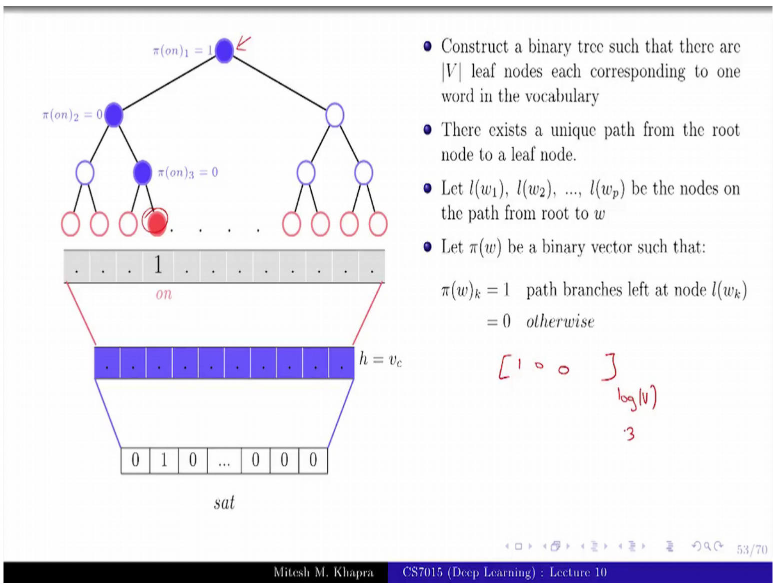


- Construct a binary tree such that there are $|V|$ leaf nodes each corresponding to one word in the vocabulary
- There exists a unique path from the root node to a leaf node.
- Let $l(w_1)$, $l(w_2)$, ..., $l(w_p)$ be the nodes on the path from root to w

Now, instead of this the hierarchical softmax says that you construct a binary tree such that your tree has how many nodes? V nodes, it has one node corresponding to every word. And there exist a unique path from the root node to every leaf node. Every leaf node corresponds to a word and there is a unique part from the root node to leaf node. Of course, there will be overlapping things for example, for this word the path is these nodes, and for this word also the path is like there is some overlap in the path.

But for every word there is a unique path, how many if you get that set up. Now let lw 1 lw 2 up to lw p be the nodes on this path. So, I am calling this as lw 1 lw 2 lw 3 sorry, sorry, sorry, sorry yeah actually it is. So, actually this is l on 1, l on 2, l on 3; that means the third node on the path of on, the second node on the path of on and so on, right that is how it is going to be and let pi w be a binary vector

(Refer Slide Time: 02:15)



- Construct a binary tree such that there are $|V|$ leaf nodes each corresponding to one word in the vocabulary
- There exists a unique path from the root node to a leaf node.
- Let $l(w_1)$, $l(w_2)$, ..., $l(w_p)$ be the nodes on the path from root to w
- Let $\pi(w)$ be a binary vector such that:

$$\begin{aligned} \pi(w)_k &= 1 \quad \text{path branches left at node } l(w_k) \\ &= 0 \quad \textit{otherwise} \end{aligned}$$

$[1\ 0\ 0\ ]$

$\log(V)$

3

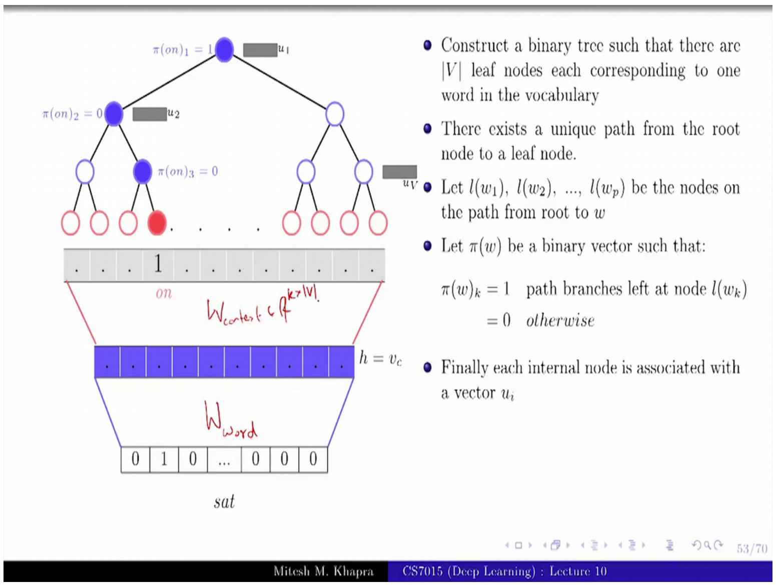
53/70

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

. So, what is the size of pi w actually binary tree log of v, right? So, the size of pi w vector is going to be log of v. So, if there are 8 leaf nodes, you will have 3 nodes as the size of the vector. So, for each of these things, this vector takes on a value 1 so, here the value would be one, because the path branches to the left if the path branches to right, then the value is going to be 0, right. So, for every node or every word I have this way of uniquely defining it is path, I can say that the path is 1 0 0, is that fine? For the word on the path is 1 0 0, if I consider some other word the path would be different, is that fine?

And of course, I have assumed there are only 8 words here, right that is why this holds if there are either otherwise I would have a vector whose size is log v, right now my V is 8 so, it is just 3.

(Refer Slide Time: 03:25)



- Construct a binary tree such that there are $|V|$ leaf nodes each corresponding to one word in the vocabulary
- There exists a unique path from the root node to a leaf node.
- Let $l(w_1)$, $l(w_2)$, ..., $l(w_p)$ be the nodes on the path from root to w
- Let $\pi(w)$ be a binary vector such that:

$$\pi(w)_k = 1 \quad \text{path branches left at node } l(w_k)$$
$$= 0 \quad otherwise$$

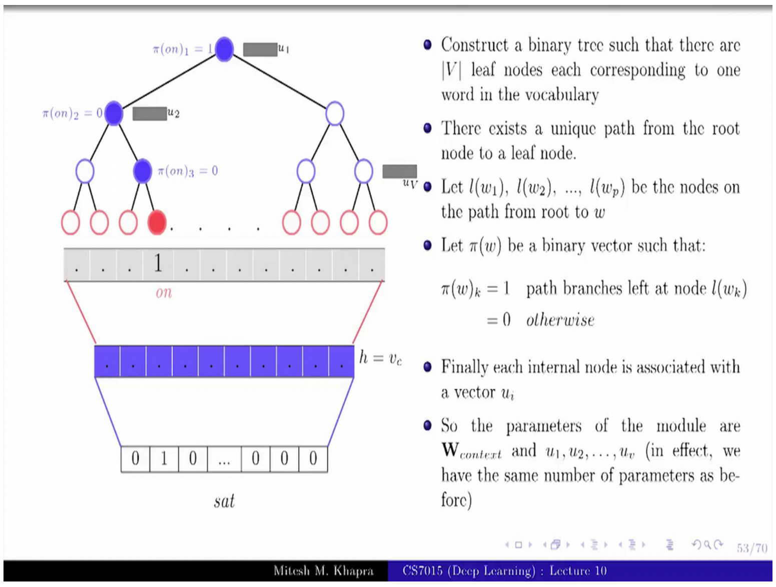
- Finally each internal node is associated with a vector u_i

Finally, each of these internal nodes is associated with a vector. So, I have u 1 u 2 u 3. So, how many of these would I have? If there are V nodes at the leaf, how many non-leaf nodes do you have in the binary tree, V you all know this, right?

So, if you have V nodes at the leaf then you will have V nodes internally. So, for each internal node, I have a vector associated with it. So, how many vectors do I have in all? U V and my input side is still the same, right I have this w word or w context depending on whether it is a skip gram or by or continuous bag of words model. How many of you get this set up? Why we are doing this is not clear? But at least the setup is clear, what we are trying to I mean what is the setup is clear, right.

So, how many parameters does this model have? Is it same as the bag of words model or less than the bag of words model or more than the bag of words model? This is how you will think, you will see how many input parameters do the poo 2 models have, how many output parameters to the 2 models are input parameters, same output parameters, how many vectors do you have? U 1 to uv each of size k, same as the original model, right it is just as an original model I had put everything inside as w context which was k cross V right. So, it is the same number of parameters, is that fine? Everyone gets that.

(Refer Slide Time: 04:57)



- Construct a binary tree such that there are $|V|$ leaf nodes each corresponding to one word in the vocabulary
- There exists a unique path from the root node to a leaf node.
- Let $l(w_1)$, $l(w_2)$, ..., $l(w_p)$ be the nodes on the path from root to w
- Let $\pi(w)$ be a binary vector such that:

$$\pi(w)_k = 1 \quad \text{path branches left at node } l(w_k)$$
$$= 0 \quad otherwise$$

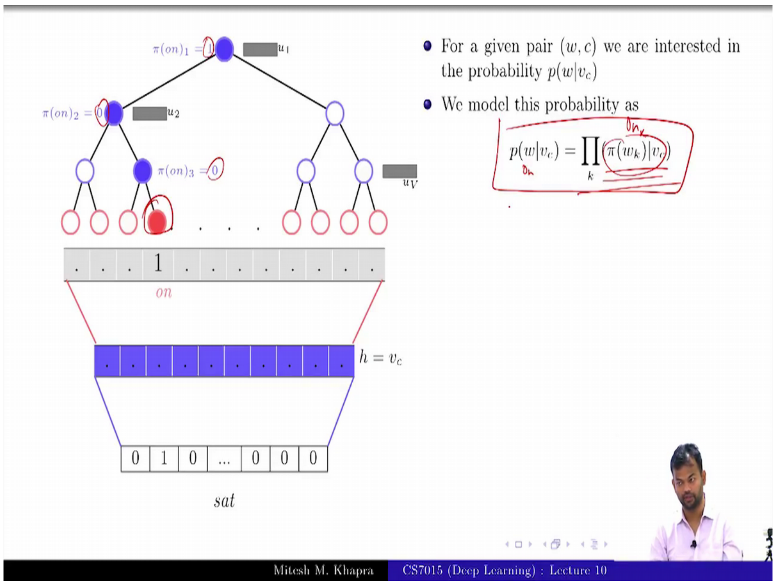
- Finally each internal node is associated with a vector u_i
- So the parameters of the module are $\mathbf{W}_{context}$ and $u_1, u_2, \dots, u_v$ (in effect, we have the same number of parameters as before)

53/70

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

So, the total number of parameters in the network is the same.

(Refer Slide Time: 05:01)



- For a given pair (w, c) we are interested in the probability $p(w|v_c)$
- We model this probability as

$$\boxed{p(w|v_c) = \prod_k (\pi(w_k)|v_c)}$$

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

Now, for a given pair w comma c, which is the correct path, we are interested in the probability p of w given vc, nothing great about this it is the same as I have been saying always that, we want the pa probability of w given c, what we are going to model as w? Given vc, because we c is the representation of c. And we model this probability now as the following thing, why does this make sense? You just assume this is on, and these are on k s right. So, on 1 on 2 on 3, why does this make sense?

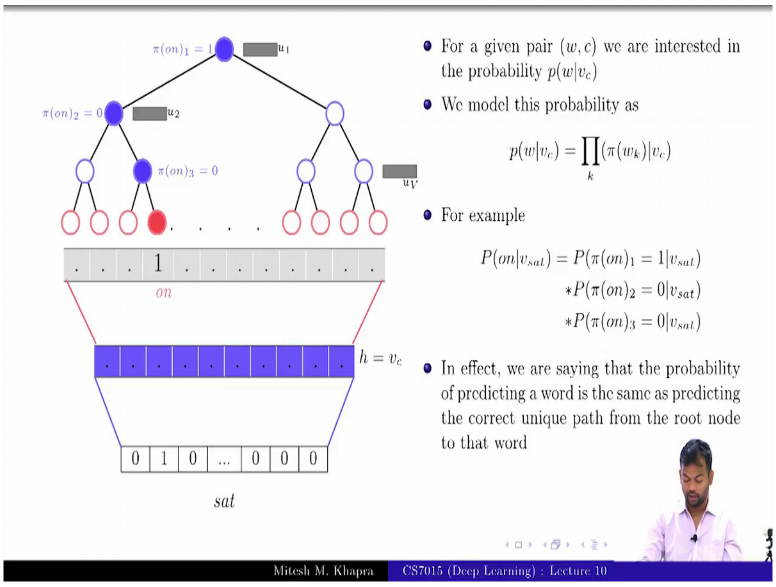
I will get the word on at the output only if the first element on the path was pi on 1 and the second element on the path was pi on 2 up to the k th element on the path was pi on k. How many forget that? Please raise your hands, right. So, that is how we are modeling it. Is it ? But what about pi on 1, pi on 2, pi on k, how do you model that? At least this form is clear to everyone, right if it is not let me know. Because then you not understand the rest of the stuff, yeah.

So now see that modeling part is always in your hands, right? You know that you want you are interested in a certain probability; it depends on you how to model it. So now, what you have done is you have con constructed a binary tree. Now I am interested in p of on given some word vc, right or some word vector vc. Now I can say that, but the way I am thinking about this is that, I get the word on only if the first if I started from the root node, the first vector took on the value 1 or the first branch took on the value 1, the second branch took on the value 0, and the third branch took on the value 0. So, that is exactly what I am saying here, right?

It is a probability that the first turn that I took was a left turn, then a right turn then a right turn, yeah the path is you have constructed the binary tree, and the path is fixed now for all the words. How to construct the binary tree is a separate thing, but the binary tree has been constructed and every word has a unique path associated with that. So, that word will occur only if that path is executed, right. So, I am just trying to find the probability of that path being executed.

Now, I need to tell you what does each; so, how many terms are there in this product? K terms, right how do I estimate each of these k terms is what I need to tell you, can you think of it? How would I model each of these probabilities, remember that every node has a vector associated with it? How many if you can think of an answer? I hope I are you saying what I think you are saying.

(Refer Slide Time: 07:48)



- For a given pair (w, c) we are interested in the probability $p(w|v_c)$
- We model this probability as

$$p(w|v_c) = \prod_k (\pi(w_k)|v_c)$$

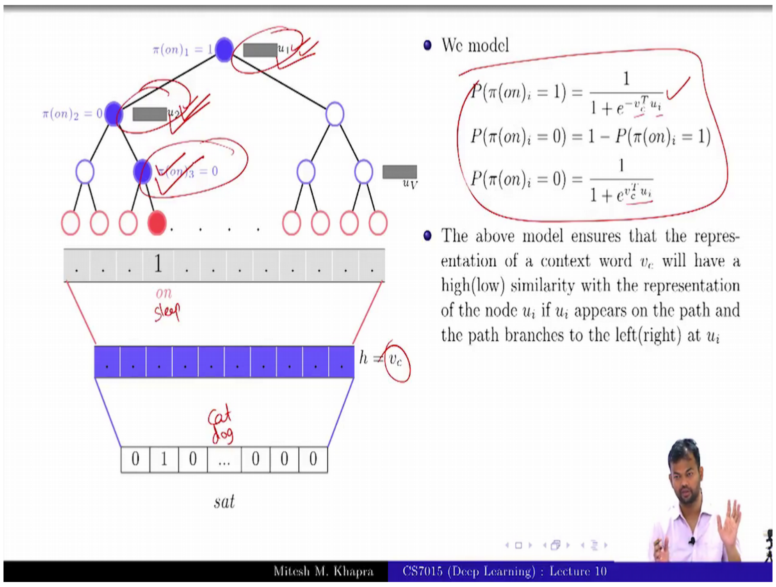
- For example

$$P(on|v_{sat}) = P(\pi(on)_1 = 1|v_{sat})$$
$$*P(\pi(on)_2 = 0|v_{sat})$$
$$*P(\pi(on)_3 = 0|v_{sat})$$

- In effect, we are saying that the probability of predicting a word is the same as predicting the correct unique path from the root node to that word

So, this is what I will do. So, as I said for the on example this is what you want, this is the path that you want to be executed.

(Refer Slide Time: 07:59)



- We model

$$P(\pi(on)_i = 1) = \frac{1}{1 + e^{-v_c^T u_i}}$$
$$P(\pi(on)_i = 0) = 1 - P(\pi(on)_i = 1)$$
$$P(\pi(on)_i = 0) = \frac{1}{1 + e^{v_c^T u_i}}$$

- The above model ensures that the representation of a context word v_c will have a high(low) similarity with the representation of the node u_i if u_i appears on the path and the path branches to the left(right) at u_i

And I am going to model it as this.

So, getting a left turn, I model it using this that dot product between the original word vector which was the input word vector which was vc, and the node representation of the node associated with that particular node. Does this make sense? So, I will tell you what we are trying to do. So, this path was clear that the probability is going to be a product of these probabilities.

Now I want how do I get each of these probabilities. So, that is again in my hand, right, I am going to say that, I am going to train my parameters vc and ui where ui is the parameter corresponding to every node. I am going to train it in a such a way that whenever I want this to take on the value 1, this should be close to 1, because I will set up my loss function accordingly, we will see the loss function.

But I am saying that whenever I want the probability to be equal to 1, I am going to use this to computed. And alternately when I want the probability to be 0, I am going to take 1 minus that which is just this, is that fine? Okay, let us go ahead a bit and then we will come back if you are still lost, right.

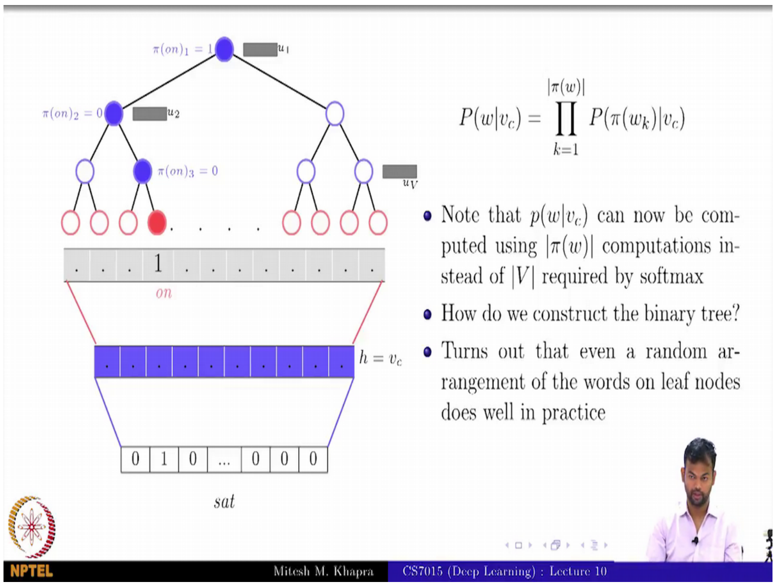
So, what does this actually ensure? This ensures that the representation of a context word vc will have a very high similarity with the node ui if the path takes a left turn there, and it will have a very low similarity with the node ui if the path takes a right turn their. How many if you get this part? Based on if you assume that this is how we are going to model it, when is this going to be high? When the dot product between vc and ui is high. When is this going to be low?

When the dot product between these 2 is look right there is a negative, yeah. So, we, sorry I or rather when is this going to be low, right. So, you get that, so, it is coming so, the word representation which is vc which is this guy would come to the re come close to all these representations, or move away from them depending on whether you want to take a left turn there or a right turn there, ?

Now, what would happen to words which appear in similar context? The same thing that we have been discussing so far, right they will come close to the node representations which are along the path, right is that fine? So, this is the context representation, right? This is actually you are representing every context word by these 3 representations. Now if a word appears in the same context, it is representation is going to either come close or move away from these representations, right. So, words appear in the same context, if you have cat and you had sleep here, then cat has to come close to this it has to move away from this, and it has to move away from this. Is that clear? That is how we have set up the probabilities.

Now, instead if I had dog and again you had the context word as sleep. Now the representation of dog also has to go close to this. It has to move away from this, and it has to move away from this. So, in effect again the same thing is happening that the representation of cat and dog are moving in the same directions. So, they will eventually come close to each other, how many if you get this intuition?

(Refer Slide Time: 11:25)



$$P(w|v_c) = \prod_{k=1}^{|\pi(w)|} P(\pi(w_k)|v_c)$$

- Note that $p(w|v_c)$ can now be computed using $|\pi(w)|$ computations instead of $|V|$ required by softmax
- How do we construct the binary tree?
- Turns out that even a random arrangement of the words on leaf nodes does well in practice

And how many computations do you need now to compute the probability of this. So, earlier you acquired that complex softmax computation, how many computations do you need? Now you definitely need these many computations. And each of these computations requires a sigmoid over or dot product, right. So, that is much much lesser than so, you just need these many dot products as compared to your expensive softmax computation earlier.

So, you see how you get the savings using the hierarchical softmax. So, this is as I said this is not very intuitive it is like a really smart trick, and it takes time to get your head around it, but I am sure if you go back and look at the slides you will get it, right? If it is if you have just got 50 percent of the idea here that is typically how it happens every time, but and I probably not figured out a better way of teaching this, but once you go back I am pretty sure that you will get to understand what is happening, right.

So now the question is how do we construct a binary tree. Anyone has any thoughts on that? Do we need to ensure certain things while constructing the binary tree? Okay, I will ask this as a quiz question, just note that. There is some subtlety here ah, in practice this is what is done. You just randomly arrange the nodes on the leaf nodes, and then you just construct a binary tree from there right. So, you have distributed all your leaf nodes randomly, and on top of that you have constructed a binary tree. My question is there a problem in doing that which I will ask you on.