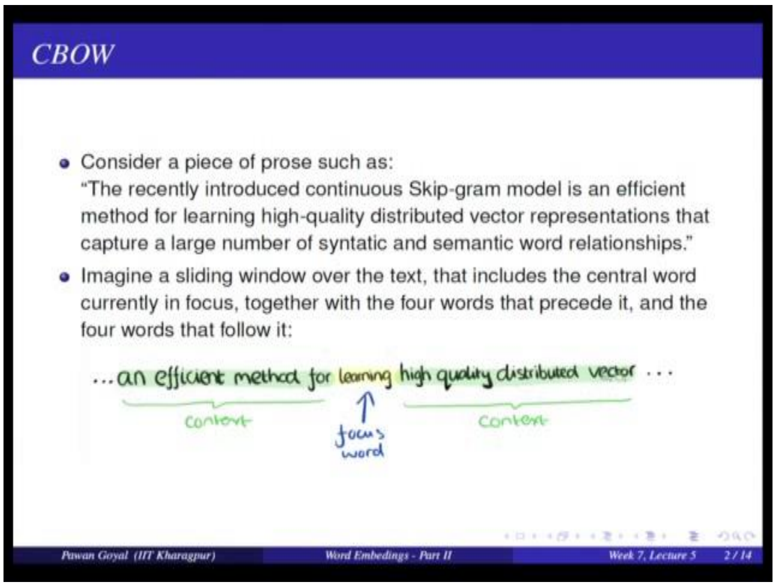
Natural Language Processing
Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 35
Word Embeddings – Part II

Hello everyone, welcome to the fifth lecture of this week. So we are talking about word embeddings. So in the last lecture we discussed the what is the different ideas behind using word embeddings, what can be some interpretation given to the different dimensions here and what are different tasks where you can use these word embeddings. But we do not go through the learning part, how do we actually obtained these word embeddings for different words. So today in this lecture we will try to discuss in detail how do we obtain these embeddings.

(Refer Slide Time: 00:53)



CBOW

- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of syntatic and semantic word relationships."
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:

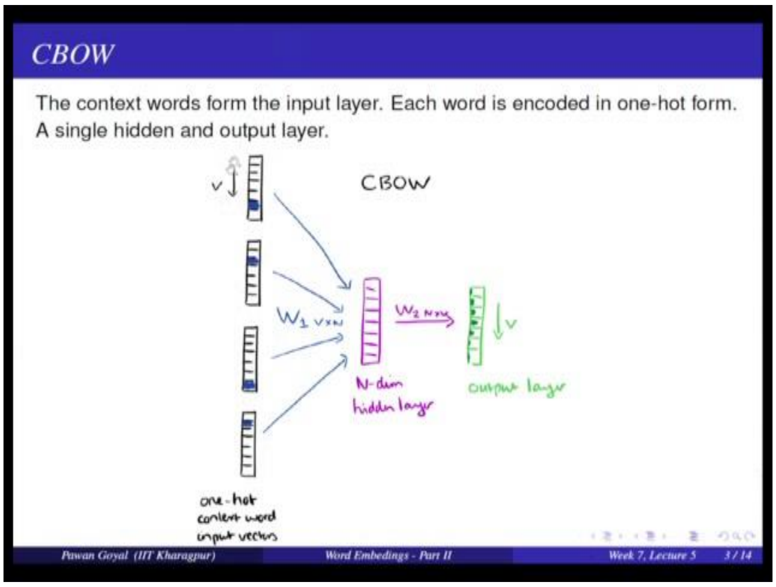
...an efficient method for learning high quality distributed vector ...
Context
focus word
Context

So in the last lecture we discussed that there are 2 different models for word embeddings. So that mean two main models one is continuous back away words model and another is skip gram model. What is the idea there? In continuous back of words model, using the neighboring words, I am trying to predict the center word. In a skip gram model using the center word I am trying to break the neighboring words.

So let us see what we do in CBOW model. So here let us say we have a prose like this. The recently introduced continuous skip gram model is an efficient method for learning high quality distributed vector representation and so on. So that is the test data that I have so much and I am trying to learn representation for various words embeddings.

So at a time I will go through one particular window here. So I will focus on a word. So let us say I am focusing on one word like learning. So what I will do? I will take a window around that word. Now window size can vary and this can be a hyper parameter that you will choose. So you can choose a I will take forward before there and forward after that. So using these 8 words around this word learning, I will try to predict this word learning. So how do we do that? So imagine is sliding window over the text, that includes the central word and with 4 words that precede and the 4 words that follow it. Like here learning is my focus word, and there are 4 words before it and for words after that. And you can also call that as the context words. And usually the context words you are trying to break the focus word.

(Refer Slide Time: 02:40)



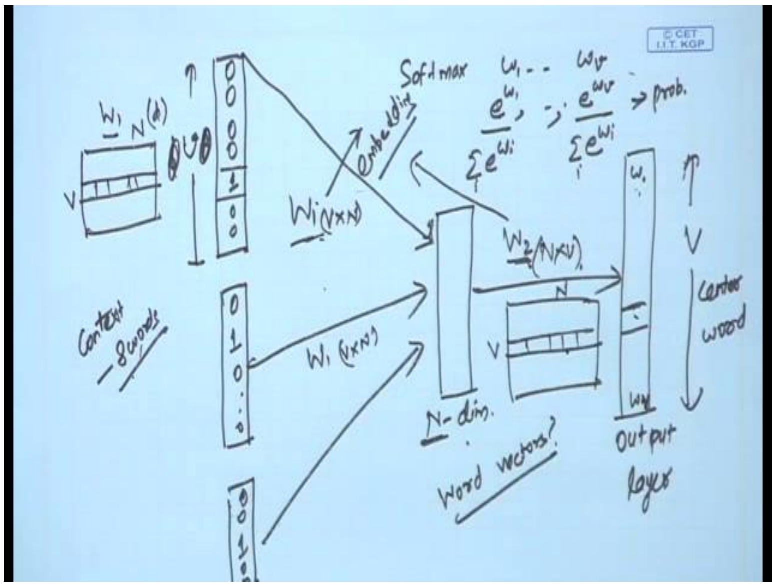
CBOW

The context words form the input layer. Each word is encoded in one-hot form. A single hidden and output layer.

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part II Week 7, Lecture 5 3 / 14

So this is some sort of network representation and how this learning will take place. So you are seeing here in the input you have, input you have various column vectors. So what do I mean by this? So what you are doing here you are putting in one hot encoding as the input. So I have this column vector of size v.

(Refer Slide Time: 03:09)



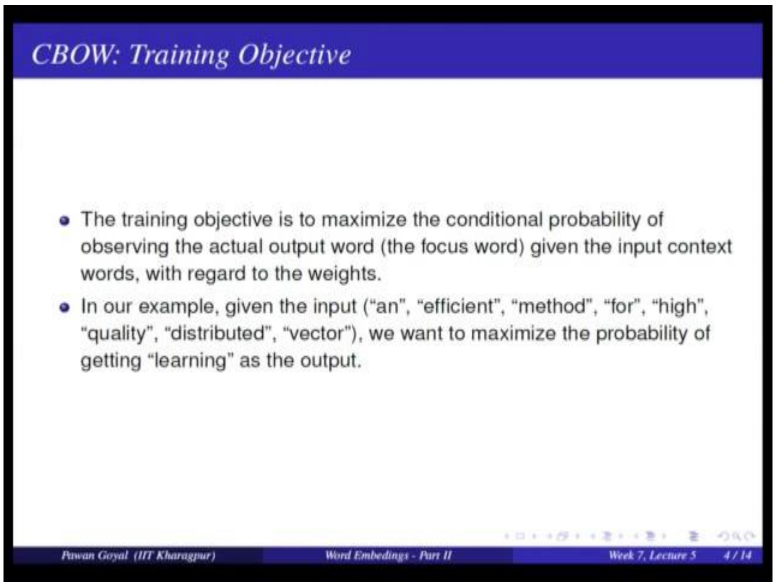
So the column of size V now, so V is the size of my vocabulary. I have that many words my vocabulary. So I may not use this only V, V is the size of my vocabulary. Now in your context you are heading having certain word that will be there in the vocabulary. So let us say this is the index of that word. So this input would be everything else a 0 and this will be 1. So one hot encoding of that word, so column vector; so like that in your context suppose you have here 8 words. So we will feed all these 8 one hot forms. So here it might be one here 0 here it might be one somewhere and everything else 0. That is your input that you are feeding. Now using this input then you are having a hidden layer this is N dimension. I remember this N will be important we will see what is this N.

So now the size V. Now I want to map these 2 this N dimensions. So I will have a weight matrix. Let us say w1 of dimension. So this is V cross 1 sorry. So this you can take it as 1 cross V. So you will have it as V cross N, so that the final representation is N dimensional now. So you will have the same weight matrix at each input. So you will get and then you can take the average of all these now from these N dimension you again go to your V dimension. This is you are output layer. So what will be the second weight here? This will be N cross V. See I starting with a V dimension going to N dimension and hidden and then going to the V dimension in the output now. So what you are doing here? So before going into what are the connections a network you are putting some input there are your context words. So suppose you are having 8 different one hot vectors and whatever happens here finally, you are trying to create your center word. This should be your center word. So what will happen here? You will have various weights W 1, W n some numbers you will W V some V different numbers you will get.

And you want to ensure that the actual center word. Suppose this is my center word. This is the highest probability. And if it is not having the highest probability, you will try to modify the weights in your network. So that it gets a higher probability then what is getting right now.

So now let us see what do we mean by all these connections. So what would happen from here? So let us go back to the slides. So each word is encoded in one hot form that is here. We have a single hidden layer and an output layer. So you are having 2 different weight matrixes one is V cross N another N cross V. So from input you go to hidden from hidden you go to output. Output you are trying to predict the center word.

(Refer Slide Time: 07:15)



CBOW: Training Objective

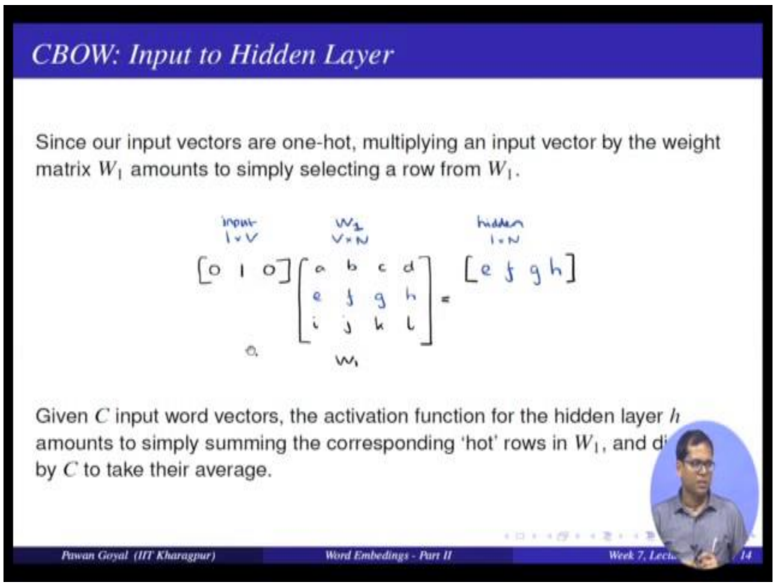
- The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words, with regard to the weights.
- In our example, given the input ("an", "efficient", "method", "for", "high", "quality", "distributed", "vector"), we want to maximize the probability of getting "learning" as the output.

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lecture 5 4 / 14

So what is my training object is here? So you see I am getting some word, some numbers for different all my V words at the output layer. And this you can think of as the probability value. And you might have a question that how do I convert these numbers to probability we will see that. So assume that you having V different probability values, now you want to maximize the conditional probability of observing the actual output word given the input context word with regard to weights.

So you say that I want to maximize the probability that I will obtain the actual what that I saw in my input, and not any other word. So whatever probability I am getting that becomes my objective function. So I want to maximize this condition probability of output word given all my input words. And this probability will be expressed in terms of all these weights that I am having in my network. So in our example what would happen, I am giving this input all these 8 words an efficient method for high quality distributed vector, and I want to maximize the probability of getting learning as the output this is my center word.

(Refer Slide Time: 08:24)



CBOW: Input to Hidden Layer

Since our input vectors are one-hot, multiplying an input vector by the weight matrix W_1 amounts to simply selecting a row from W_1.

$$\underset{\substack{\text{input}\\1\times V}}{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}} \underbrace{\overset{\substack{W_1\\V\times N}}{\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}}}_{W_1} = \overset{\substack{\text{hidden}\\1\times N}}{\begin{bmatrix} e & f & g & h \end{bmatrix}}$$

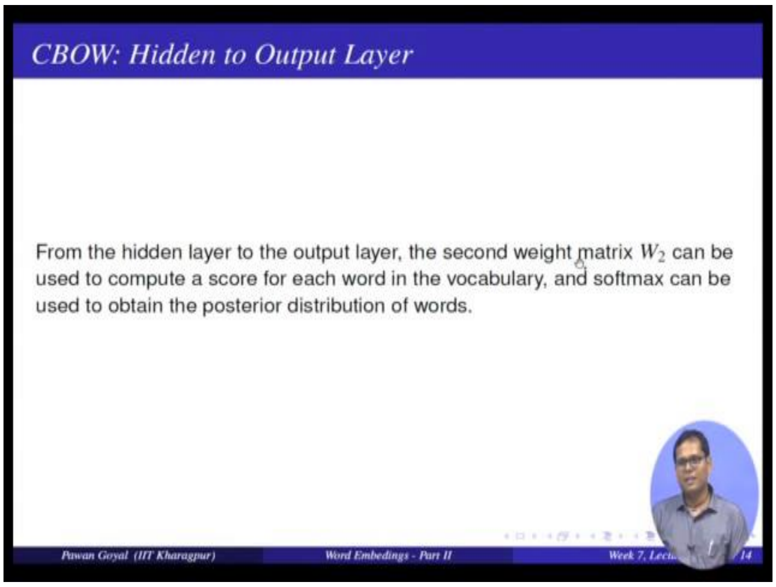
Given C input word vectors, the activation function for the hidden layer h amounts to simply summing the corresponding 'hot' rows in W_1, and di by C to take their average.

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lectu 14

Now what happens from input to the hidden layer? So input we are saying you are feeding one hot vectors. So if there are 8 different input words I am having 8 different one hot vectors. Now what do I mean by multi bank this one hot vector with my first weight matrix that is of dimension V cross N. So you can think of it like that. So you put using an input. So this is one word and having a corresponding element is one that that is my input word and you are multiplying it with V cross N weight matrix. So this operation is nothing like this nothing, but you are taking the corresponding row of this matrix. This matrix has V different rows. So these V rows you can think of as if corresponding to V different words in my vocabulary. So whenever you are feeding a one hot form of one word you are picking up that row and so you are doing it for 8 different words. So you are picking 8 different rows in my initial weight word weight for matrix, and then you are having given c input words. So activation function for the hidden layer will be simply summing the corresponding hot rows. So I will pick up the hot rows here that correspond to the input words and divide by c, so that I have an average representation.

So now you understand what is going from input to hidden layer. You are taking c different one hot c different rows from your weight matrix and averaging those that is what goes in your hidden layer.

(Refer Slide Time: 09:59)



CBOW: Hidden to Output Layer

From the hidden layer to the output layer, the second weight matrix W_2 can be used to compute a score for each word in the vocabulary, and softmax can be used to obtain the posterior distribution of words.

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lectu 14

Now, what happens from hidden to output layer? So remember we have a second weight matrix W 2 that goes from hidden to output layer and it is dimension is N cross V. So now, a hidden layer dimension each 1 cross M. So if I multiply this 1 cross a matrix with this N cross V matrix I will get a 1 cross V matrix and that is my output layer. So this 1 cross V matrix even think of as having weights for different words in my vocabulary. And I want to maximize weight for my center word. So from the hidden layer to output layer the second word matrix W 2 can be used to compute a score for each word in my vocabulary and now you can obtain the weights. How do we convert them to probability values and for that you use soft max? What is the idea of soft max?

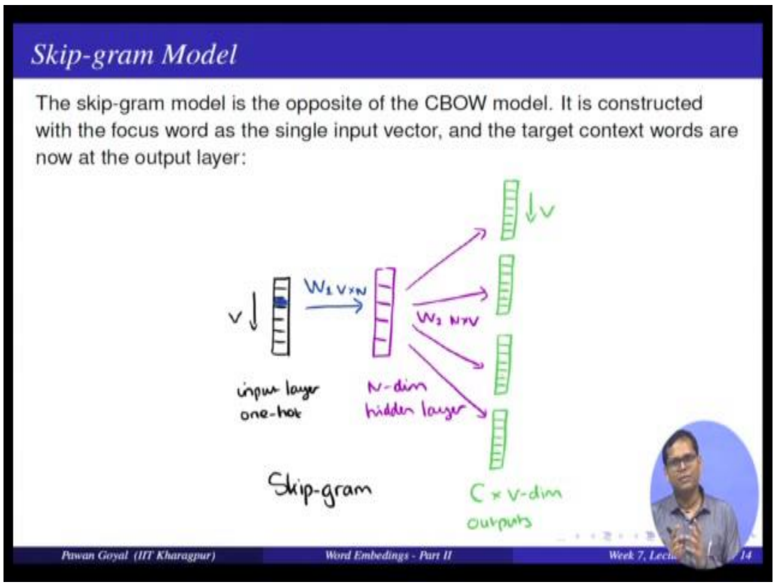
So here you will you get the weights W 1 to W V. They can be any real numbers. Now how do you convert that to probability distribution? So this is a simple idea that is in soft max. So in soft max what you do is you are given W 1 to W v and you want to convert that to a probability distribution. So you will see that I simply multiply all these by. So I will put an exponent over these, so e to the power W 1 e to the power W v. Now they are all positive numbers and then to convert them to probability I will just divided by summation over e W i for all i.

So now this is this will some off to 1 because this is normalize and these are my probability distribution. So I have these weights I use a soft max to convert them to probability values. And then I will try to maximize the probability of my actual center word. This will become my training objective and based on that I will learn my weights. So I will talk a bit about this learning problem in the when I go to the next model of escape ground, but suppose we have some way of learning these weights, and finally, I have the optimal set of weights W 1 and W 2. Now where are the word vectors, what are your word vectors. Here I am feeding one hot vector, this is not being learned. This is the same. So where are my word vectors being learned now if you think about it. So what I am learning are the weight matter research. This is off V cross N this is N cross V. So I can take a transfer this also will become V cross N. So you can think of it as if for each word you are learning a N dimension representation. So after you learn this matrix these weight one and weight 2 will correspond to your word embeddings.

So W 1 will be of size, So W 1 will be of size V cross N. So you can take any vector i and get a N dimensional representation and the sign you can also think of as your d, d dimensional representation. So for each word you are getting a d dimensional vector. Similarly, for W 2 you will get similarly if you take a transpose you will get a V cross N representation. And in general what you do, you can take a, so you are getting 2 vectors: one from W 1 from W 2. So we can finally, combine these 2 vectors you can either concatenate these vectors or the same word or some these over or taken average and that works fine. So this is the addition about what is the kind of network that you use for learning these embeddings. So these weights are nothing, but my embeddings. That I am learning. So this is about continuous back of words model. Now what will happen skip gram, in a skip gram model the network will slightly change. So now, I will feed only the center word here only one input vector, but I will predict multiple contexts words.

So now from input to hidden there are only one. So the only one input vector, but output there are multiple vector.

(Refer Slide Time: 14:41)



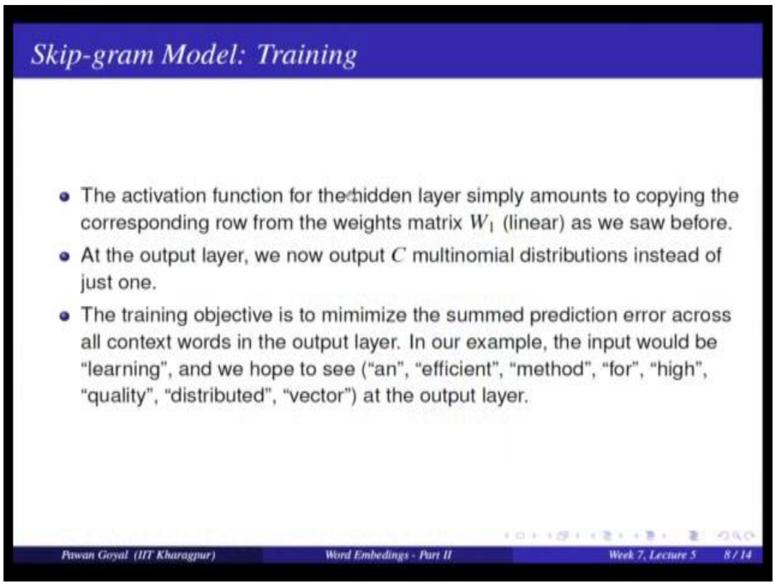
Skip-gram Model

The skip-gram model is the opposite of the CBOW model. It is constructed with the focus word as the single input vector, and the target context words are now at the output layer:

So let us look at the skip gram model. Skip gram model is the opposite of CBOW model. So you have the center word, as the sing single input vector and the target context words are at the output layer. This is my output layer. And you can now very easily correlate with what we did in CBOW. So the weights correspond to roughly the same idea.

So now let us formally define so we see how in the using the network we can think about the learning. How the learning will take place and how the weight updates will take place, but suppose you want to write it mathematically and how do we do that?

(Refer Slide Time: 15:19)

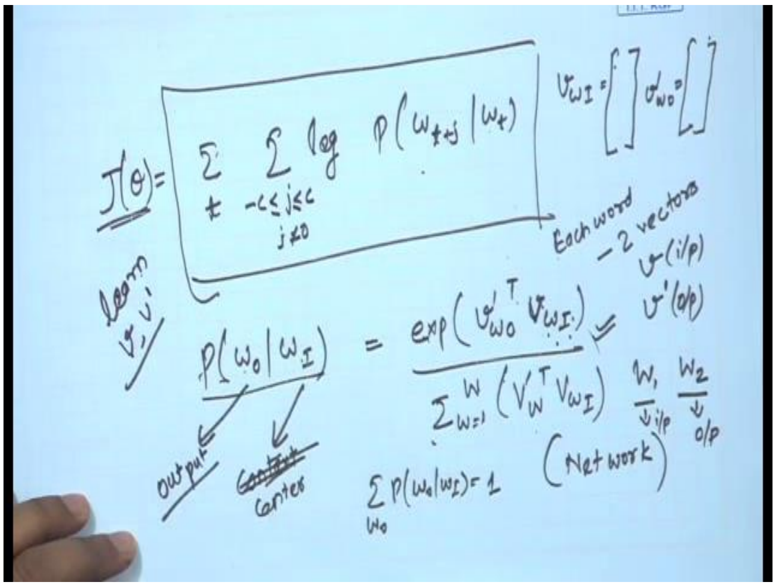


Skip-gram Model: Training

- The activation function for the hidden layer simply amounts to copying the corresponding row from the weights matrix W_1 (linear) as we saw before.
- At the output layer, we now output C multinomial distributions instead of just one.
- The training objective is to mimimize the summed prediction error across all context words in the output layer. In our example, the input would be "learning", and we hope to see ("an", "efficient", "method", "for", "high", "quality", "distributed", "vector") at the output layer.

So yeah this is very analogues to what we find the case of CBOW. So here from input to hidden layer I am simply copying the row from the weight matrix W 1. So I am having only one input. So I will copy only one row and that will go to my hidden layer. Now at the output layer we will have see different distributions. And I will try to predict the see different context words using my hidden layer and objective is to maximize some prediction errors across all the context version my output layer. So you want to predict although. So I want to maximize the probability for observing my actual context words. So let us say we have a window, where I am having small c words in the left small c words in the right.

(Refer Slide Time: 16:20)



$$\underline{J(\theta)} = \boxed{\sum_{t} \sum_{\substack{-c \le j \le c \\ j \ne 0}} \log p(w_{t+j} \mid w_t)}$$

$v_{w_I} = \begin{bmatrix} \vdots \end{bmatrix}$ $v'_{w_O} = \begin{bmatrix} \vdots \end{bmatrix}$

Each word – 2 vectors

v (i/p)

v' (o/p)

learn v, v'

$$\underbrace{p(w_O \mid w_I)}_{} = \frac{\exp(v'^{T}_{w_O} v_{w_I})}{\sum_{w=1}^{W} (v'^{T}_{w} v_{w_I})}$$

output

~~Context~~ center

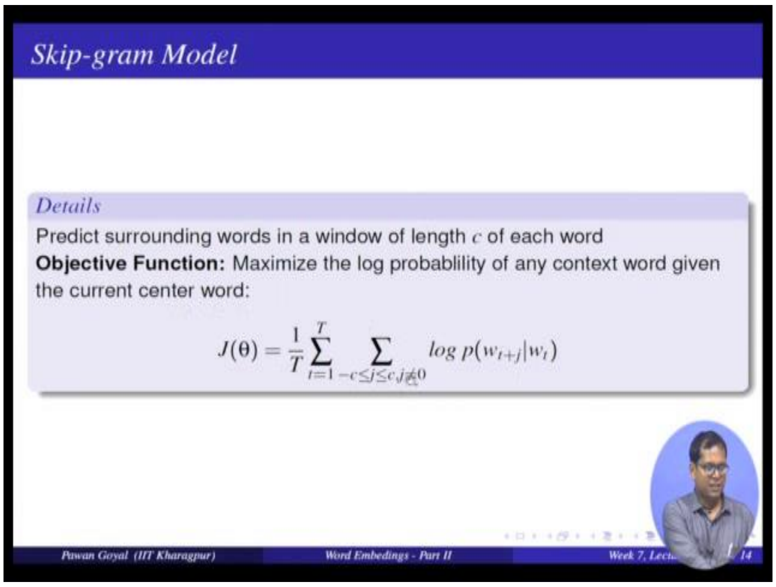
$\underset{\downarrow \text{i/p}}{W_1}$ $\underset{\downarrow \text{o/p}}{W_2}$

(Network)

$$\sum_{w_O} p(w_O \mid w_I) = 1$$

So what can be my training objective? It can be I am trying to maximize, suppose I take a log probability W t plus a given W t. Wt is the center word and I am going from minus c less than equal to j that is equal to c. Window of size c around j and j is not equal to 0. And this I can do for all possibility. All possible center words and this becomes my training objective.

(Refer Slide Time: 16:56)



Skip-gram Model

Details

Predict surrounding words in a window of length c of each word

Objective Function: Maximize the log probablility of any context word given the current center word:

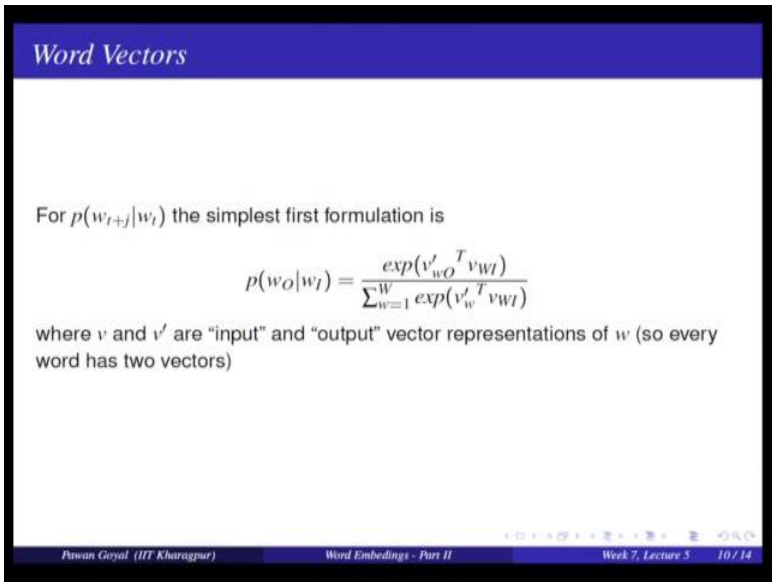
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} log\ p(w_{t+j}|w_t)$$

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lectu 14

So we will see that. So I am predicting surrounding words in the window of length c of each. Word my objective function is I want to maximize the log probability of any context word given the current center word. So I am trying to maximize this probability. Probability of W t plus j given W t and sum over all the possible context words and then I can sum it over all the possible words in my input, and this becomes my overall objective, soj theta and theta of all the parameters all the weight matrix W1 and W2 that I am trying to learn.

So now this is my objective function and I want to maximize that and by doing that I want to learn my parameter theta and how do we do that.

(Refer Slide Time: 17:51)


Word Vectors

For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(w_O|w_I) = \frac{exp({v'_{wO}}^T v_{WI})}{\sum_{w=1}^{W} exp({v'_w}^T v_{WI})}$$

where v and v' are "input" and "output" vector representations of w (so every word has two vectors)

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lecture 5 10 / 14

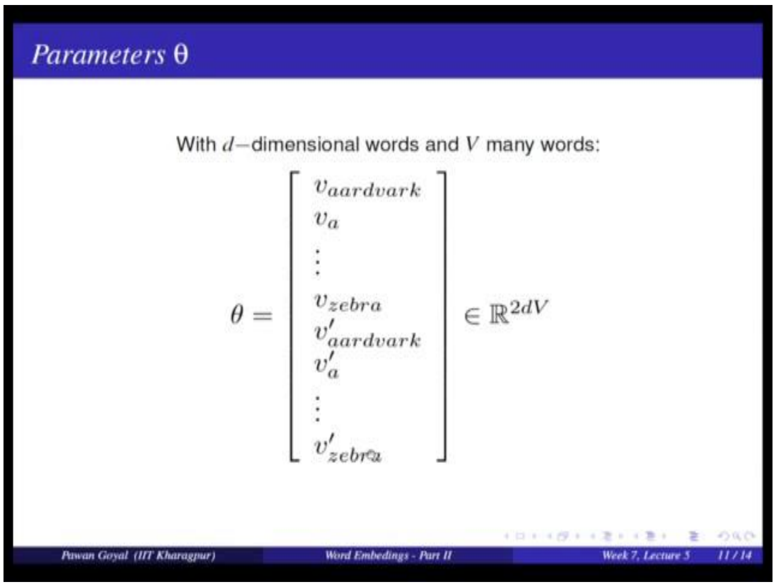
Firstly, let us see how will be compute these probabilities. Probability Wt plus j given Wt; now this will be actually you have already seen that in the case of using the network but using which some mathematics can be show that. So idea is that let us say I am having a probability for P Wo given WI, WI is my context world, and W is the output word. That is the; I am sorry so I should say as the center word. And this is my output word these are all the context words. Let us say for a given context words how do we write it. And I am saying we can write it in this form. That is exponent V prime Wo Transpose V WI divide by sum over W is equal to one to capital W V prime W transpose V WI. And these are formulation of this probability. And let us try to understand that. So whatever different is we have written here. So one thing you are seeing there is a V and there is a V prime. So for each word you have 2 vectors. One is V another is V prime. So V is for the so V is use only for the input center word. And V prime is use for output.

So when I am trying to use the input word to pre the output word I will simply take a dot product of these. Now both is vector like column vectors. So V WI will be of this form, and V prime W will also be as a column vector. So how do I get a number by multiplying these 2? I will take a transpose of this and multiplied with this and that is what I am doing here. V prime W transpose times V WI. That gives me single number. Now this number I want to convert to a probability value. So then I am using a softmax over that. Exponent over this number divide by I will do it for all the words in output. So that is why I have a summation over all the word, W is the total number of words in my vocabulary. This is nothing about a probability distribution, and you can see that summation P Wo given WI for all words Wo will add to 1 because of this normalization factor. So this is added to word this is giving me a condition probability of Wo given WI. Now as a simple exercise you can also try to see how this number comes from the network that we talked about. So we had a networking interpretation. Now we have a simple matrix short of interpretation.

So how they correspond to each other? And they use the idea that word one sorry weights 1 and weighs 2. What are these? Weights weight 1 correspond to the input and way to correspond to output. Use this idea and see that by using the network also are you getting the same form of this probability by putting x softness over the output layer and you will you will see that you are actually getting the same form. So now, I have this formulation Wo given WI this particular formulation and I put it here for all the words in my context and this becomes my j theta. This is my objective function.

Now what is my objective? I want to learn my V an V prime. So I will try to learn my V and V prime such that this is optimized and for that there is a simple way that is you can use getting descent algorithm. So you can try to take partial derivative of j theta with respect to each of these parameters and update your weights accordingly. So this is the probability condition, probability distribution function that we have found and V and V prime are input and output vector representation of the same of the same word W. So every word has now 2 vectors.

(Refer Slide Time: 23:01)



Parameters θ

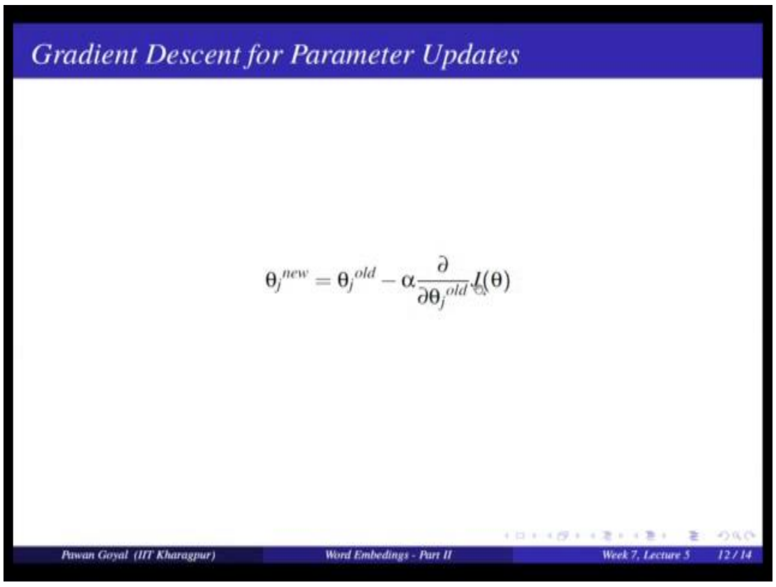
With $d-$dimensional words and V many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ v'_{aardvark} \\ v'_a \\ \vdots \\ v'_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lecture 5 11 / 14

So suppose I have d dimensional vectors. So d is the dimensional of my hidden layers in network terms or for each word what is the representation that we using we using dimension representation. So I have the many words. So what is the size of my parameter theta? So I have capital V words for each word I have been input vector and output vector. So there are 2 V vector and each vector of dimension d. So I have 2d times capital V that many parameters slower this should my whole set theta.

(Refer Slide Time: 23:41)



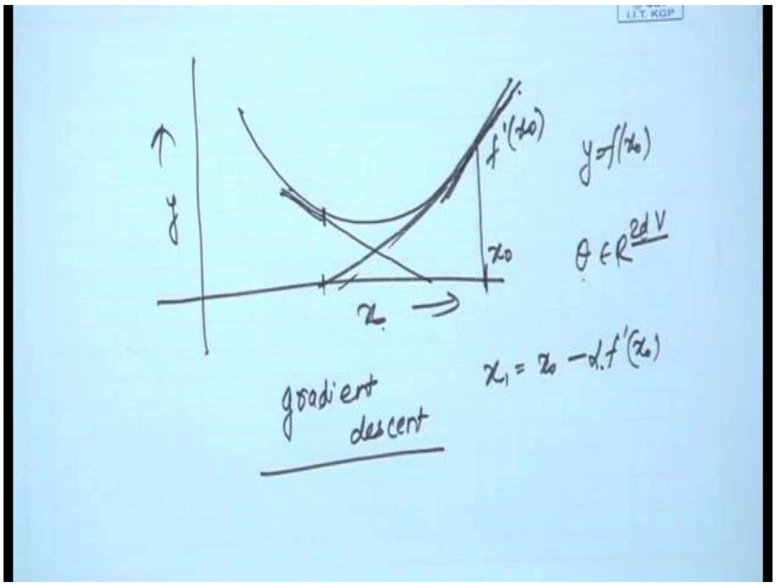
Gradient Descent for Parameter Updates

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lecture 5 12 / 14

And what is the simple gradient descent formula. So this is like you take the derivative with respect to that particular parameter. So theta j new is theta j old minus alpha is my learning date and you take a partial derivative of j theta with respect to the parameter when putting the old values.

(Refer Slide Time: 24:20)

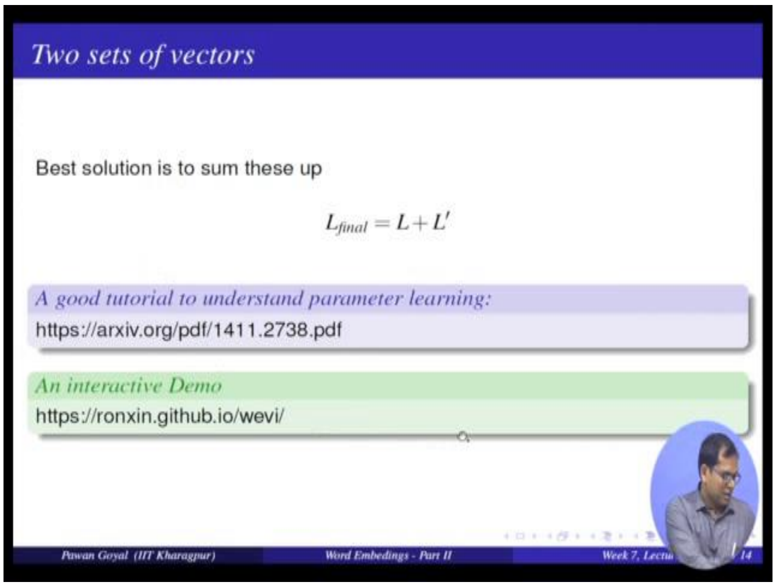


Now, I am supposing that you know the idea gradient descent that is used to optimize yours minimize a particular function. So if I want to give this idea very briefly. So this is like suppose you have a simple function like this. And I want to find out for what parameter theta, of what parameter x will suppose, this is a function over x and this is my y. For what value of x this is minimized. So what I will do? I will start with N e x. So suppose I am starting with this x 0. I will find out the value of function. So I go to the function. So why is suppose f x 0.

Now I want to know what is the value of x where there is minimized. So what I will do? I will take the derivative of the function at this point f prime at x 0 and now, if I have to minimize if this direction of my gradient I have to go in the opposite direction of my gradient. So my new value should be x naught minus f prime x naught. So I will go into opposite direction and I will use some learning rate with what rate you will go in this direction and this is simply the idea of gradient. You keep on doing that. So you go somewhere again you find our suppose you are going at this point. So now, your gradient will point you to this direction. So will try to go in this direction again gradient will point you some direction and finally, you will converge. So this is a very simple idea of gradient descent, but I will suggest that you have a look at it that what is actually the gradient descent.

So this is simple function, but your function can be over multiple parameters. So like here I have my theta with in 2d V dimension. So I have 2d times capital V many parameters. My theta is a function my j theta is a function of all these parameters. So what I will do, I take a partial derivative with respect to each of these parameters and accordingly update those parameters and this is how it will look like. So we are not going in in doing the derivation because that would not be in the scope of this course, but if you are interested you can try to find out if I derive this form what update values do I get for different what vectors how do I update these vectors.

(Refer Slide Time: 26:48)



Two sets of vectors

Best solution is to sum these up

$$L_{final} = L + L'$$

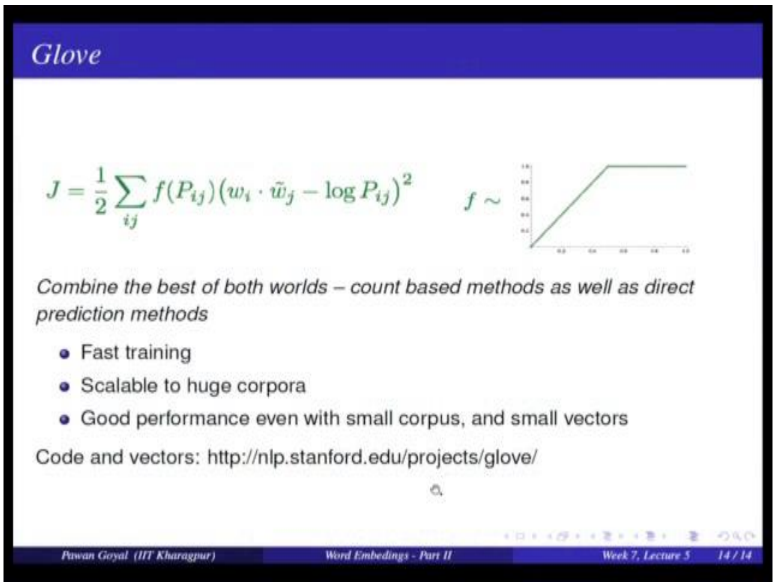
A good tutorial to understand parameter learning:
https://arxiv.org/pdf/1411.2738.pdf

An interactive Demo
https://ronxin.github.io/wevi/

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lectu 14

And now once I have done that, I have 2 different vectors V and V prime. And I can simply some these over. And that will give me the final embedding for each word. And if you want to understand more about how do we learn these parameters. So you might have a look at this paper and this gives the nice tutorial for parameter learning and also if you want to try out an interactive demo, so you can try out here.

(Refer Slide Time: 27:23)



Glove

$$J = \frac{1}{2}\sum_{ij} f(P_{ij})(w_i \cdot \tilde{w}_j - \log P_{ij})^2 \qquad f \sim$$

Combine the best of both worlds – count based methods as well as direct prediction methods

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

Code and vectors: http://nlp.stanford.edu/projects/glove/

Pawan Goyal (IIT Kharagpur) Word Embedings - Part II Week 7, Lecture 5 14 / 14

So what to work is one famous model and such 2 variation c V W and skip gram. And both are used. So in some tasks skip gram has shown to be better in some CBOW has shown to be better. And there are many other tricks that I used for learning these parameters that we have not covered. I have only tried to give you the intuition. So I hope you will at least have an idea that how these word vectors are learned from my data, by just predicting neighboring words from the context or context from the neighboring word. And what is the embedding? So, how these weight vectors for my embeddings.

So now there are some other models, so one other popular model is glove model. So what is a basic cognition of glove model? So in this skip gram I am trying to learn my all my embedding from scratch. So in glove model what they are saying. So from a corpus you can kind of count the co-occurrence. See know with that is what we did in the distribution semantics, we would be counting co-occurrence. Now what they are saying? So this gives a very good idea about the words. So which words are similar to what are the words? And suppose you find out what is the co-occurrence probability for any towards. Now try to make word vectors such that when you do a dot product between towards you get close to the actual co-occurrence that you are seeing in the corpus.

And by doing that you will get a low dimensional representation and also it will take care of the words that are coming very sparsely, where you do not have enough data in the corpus to predict the co-occurrence. And this is the simple objective function. So that is if you look at the objective function. So forget word f, f is a few simple function that that make sure that if certain words are certain co-occurrence, we are very popular you do not base your learning towards those. See you just clip those at certain point. So everything above that will be converted to this one.

So this is the main idea here of this glove vectors. So you are having in word definition for i and j that you want to learn. An idea is try to learn them that are that such that they are resembling their whatever you get from the co-occurrence probability. So P i j is from the co-occurrence. You can use either P m i condition probability or many other things. So find out w i, w j such that they are close to this co-occurrence probability. And then we have these objective function and you optimize this objective function and try to learn your different weights. So you are trying to combine the best of both words. Using the count based methods and the direct prediction methods. And using both of these you are trying to come up with your word vectors. And these vectors have also become very popular because the training is much faster than then my skip gram model because now you are not going to each individual word in your and all the windows. You have already computer the counts now you are only looking at the pairs word pairs and one maximizing that. And even with the small corpus and small vectors they have shown very good performance.

So one good thing is that suppose you want to use this vector finding of your task, so either word vectors or glove vectors are freely available on different websites. So what do we already told the website in the previous lecture for glove, you can go to this Stanford website and there you can download code as well as the vectors. And these vectors you can use for many of your tasks, where you are trying to find out whether 2 words are similar, or you are trying to find out some analogy task a is to b then c is to what. And many other applications they have been used in. So I think this is what I had to say for this tropical distribution semantics, this is a very well growing research field and lot of new thinks happening.

So I hope whatever we have discussed will help you to also understand the papers in this field if you want to going in further depth, and also you can try out these ideas for many of your applications. So in the next week what we will do. So we will start with a separate notion of semantics that is Leska semantics. So how to use lexicon to find semantics between words.

Thank you.