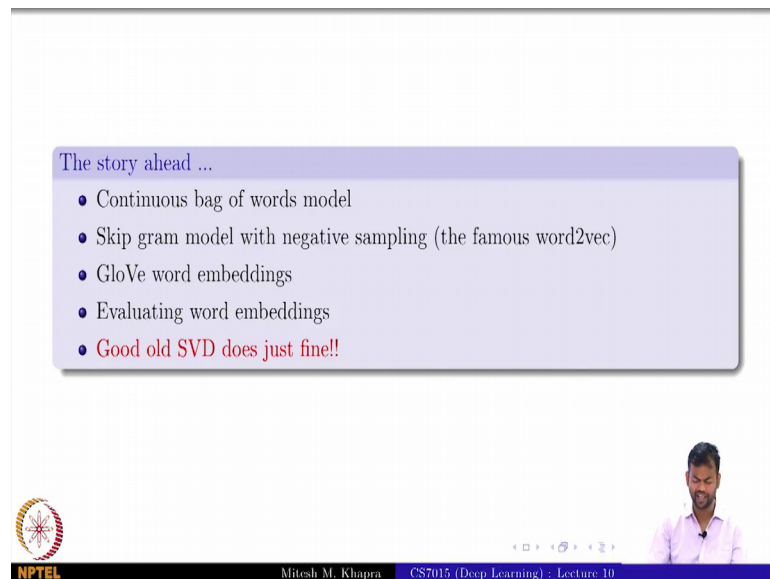


**Deep Learning**  
**Prof. Mitesh M Khapra**  
**Department of Computer Science Engineering**  
**Indian Institute of Technology, Madras**

**Module – 10.10**  
**Lecture – 10**  
**Relation between SVD and word2Vec**

Now, and later on actually the same guys, they also came up with this formal relation between SVD and word2vec which is again under some assumptions.

(Refer Slide Time: 00:20)



The story ahead ...

- Continuous bag of words model
- Skip gram model with negative sampling (the famous word2vec)
- GloVe word embeddings
- Evaluating word embeddings
- **Good old SVD does just fine!!**

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

(Refer Slide Time: 00:21)

$W_{context} \in \mathbb{R}^{k \times |V|}$   
 $h \in \mathbb{R}^k$   
 $W_{word} \in \mathbb{R}^{k \times |V|}$   
 $x \in \mathbb{R}^{|V|}$

$M = W_{context} * W_{word}$

where  
 $M_{ij} = PMI(w_i, c_j) - \log(k)$   
 $k = \text{number of negative samples}$

- Recall that SVD does a matrix factorization of the co-occurrence matrix
- Levy et.al [2015] show that word2vec also implicitly does a matrix factorization
- What does this mean ?
- Recall that word2vec gives us  $W_{context}$  &  $W_{word}$ .
- Turns out that we can also show that

So essentially, word2vec factorizes a matrix M which is related to the PMI based co-occurrence matrix (very similar to what SVD does)

NPTEL Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10 70/70

But I am not going to do the proof here. I am just going to give you the intuition. So, recall that SVD does a matrix factorization of the co occurrence matrix levy et al showed that word2vec also does such a implicit matrix factorization. So, what does this mean? So, recall that word2vec gives us  $W_{context}$  and  $W_{word}$ . It gives us these 2 parameters. So, they say that there exist a matrix M such that this is wrong, just be the product of 2 matrices right this is the product of 2 matrices. It should be  $W_{context}^T$ ,  $W_{word}$  or just see which way the transpose should be.

So, it is actually a product of these 2 matrices that we have learnt and what is m m is actually nothing, but the PMI matrix minus this  $\log k$ , where does the k come from? What was k? The negative samples that you have taken; so, they actually showed that whatever representations word2vec runs; it is actually doing a factorization of this matrix, where this matrix has a strong connection to the PMI matrix. And SVD also works with the PMI matrix, right.

If you take SVD matrix and do these modifications to it; that means, you take every value which is the PMI and then subtract this  $\log k$  from that. And then just do an SVD of that you will essentially get back the same word representations as word2vec.

There was some certain assumptions made in the paper, but that is I mean, I do not want to go into those, but the key idea here is that, you can actually show that SVD and word2vec are actually connected. And if you think about it at an intuitive level, though

these methods are relying on the same underlying principle that words appear together. Based on that, the word representations get updated or an SVD based on there the counts get updated and you then eventually end up with certain representation.

Next the underlying principle is the same. So, there has to be a connection right it is not that they are doing fundamentally something different both of them are relying on the idea of co occurrence or the idea of distribution right. So, they have to at some level be similar in some ways right. So, that is what they finally, showed and so, now, but still in most applications word2vec is preferred.

So, one reason for that is, that this is an iterative training position right as compared to SVD and I come back to your question, right? How do you do that? How do you compute the eigenvectors of  $X^T X$ ? And the answer is, there is no simple way of doing that and you have to do that expensive matrix multiplication.

And then rely on various very smart libraries for computing the eigenvectors which are still order  $n^2$  point something or something like that they not order  $n^3$ , but they are still order  $n^2$  point something, means they are still expensive. And then of course, you have this memory issue that if you have a very large vocabulary, your PMI matrix is going to be very high dimensional. And then you need to do the factorization of that high dimensional vectors right. So, that runs into these computational efficiency issues.

On the other hand, word2vec by design is an iterative algorithm because, you are going to grade gradient descent which is that every times that you are going to update some parameters of the model. You are not learning all the parameters together. You are only dealing with some parameters at every time set right. So, that is more computationally efficient. Especially, if you do the contrastive divergents or the negative sampling or the hierarchal sample; so, that is why, perhaps it is still more popular than SVD.

So, that is where we end this lecture.