Imperial College Business School

Imperial means Intelligent Business

Machine Learning in Finance

Lecture 5
Practical Implementation : Word Vectors

Arnaud de Servigny & Jeremy Chichportich

Outline:

- Introducing the Problem

- Word Embedding Methods
  - The GloVe approach
  - The Word2vec approach

- Programming Session: Implementation of the GloVe approach

Part 1 : Introducing the problem

Why do we need vectors to represent words ?

- We have a collection of sentences in the form of a corpus and aim to undertake a classification task, for example.

- Certainly, we cannot input words directly into a model; it only processes numerical data.

- The question is : How do we represent the words of our corpus in a way that can be feeded in a Machine Learning Algorithm ?

- It’s clearly an Unsupervised Learning task.

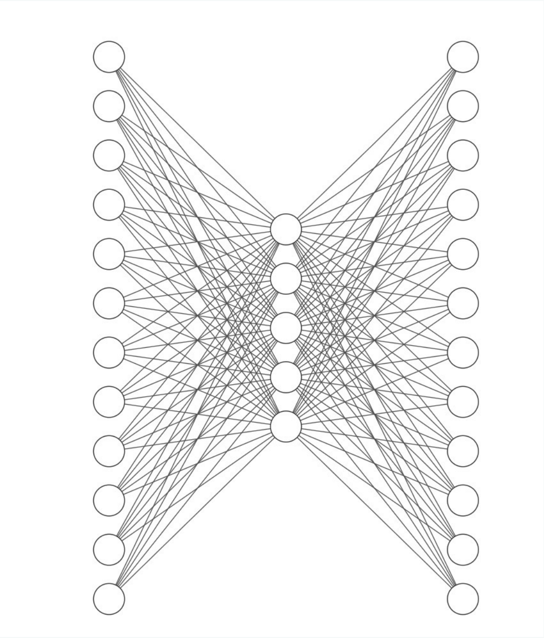
DATA

- Document 1 : « The sole evidence it is possible to produce that anything is desirable is that people actually to desire it. »
- Document 2 : « In law a man is guilty when he violates the rights of others. In ethics he is guilty if he only thinks of doing so. »
- Document 3 : « Always recognize that human individuals are ends, and do not use them as means to your end. »

  …
- Document N : « Justice is a name for certain moral requirements, which, regarded collectively, stand higher in the scale of social utility and are therefore of more paramount obligation than any others. »

?

Model



Review : Words as discrete symbols:

- What we have seen so for (in Lecture 5) is the possibility to turn each word into a discrete symbol.

- For that, we create a dictionary to map each word present in our corpus to a unique discrete index.

DATA

- Document 1 : « The sole evidence it is possible to produce that anything is desirable is that people actually to desire it. »
- Document 2 : « In law a man is guilty when he violates the rights of others. In ethics he is guilty if he only thinks of doing so. »
- Document 3 : « Always recognize that human individuals are ends, and do not use them as means to your end. »

  …
- Document N : « Justice is a name for certain moral requirements, which, regarded collectively, stand higher in the scale of social utility and are therefore of more paramount obligation than any others. »

Code

```
index = 1
word_index = {}
for sentence in sentences:
    for word in sentence:
        if word not in word_index:
            word_index[word] = index
            index += 1
```

word_index =

{
'the' : 1 ,
'sole' : 2 ,
'evidence' : 3 ,

'any ' : 934233
}

Transitioning from discrete symbols to one hot vectors

- After the initial preprocessing step, we obtain lists of integers representing the words as follows :

Corpus

- Document 1
- Document 2
- Document n
- Document N

Discretize via word_index

- Document 1 : [23, 43, 12, …, 2343, 1]
- Document 2 : [12, 1, 23453, …, 123]
- Document n : [1234, 1, 23]
- Document N : [1, 1232, …, 12322]

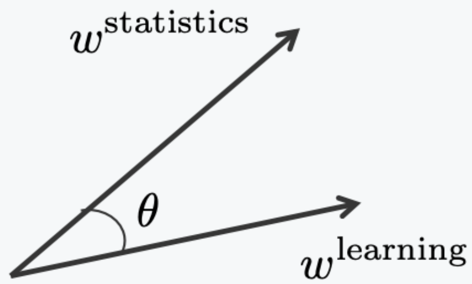
- Rather than representing a word by its index in the **word_index** dictionary, it is equivalent to represent it as a vector of size V (where V is the vocabulary size, i.e., the number of distinct words in the entire corpus), with a value of 1 at the index position and zeroes at all other positions.
  - Example : Let's suppose the word « equity » is of index 134 and V = 100 000.
  - Then, the word « equity » will be represented by the following vector of size V:

$$[0, \ldots, 0, \underset{\substack{\uparrow \\ \text{position 134}}}{1}, 0, \ldots, 0]$$

  - This vector is commonly referred to as a **one hot vector**.

Limitations of one hot vectors:

- The **One-hot-vector** is the easiest way to represent words as vectors.
- In this encoding scheme, each word is treated as a completely independent entity, without any consideration for similarity between words, even if they share the same meaning
- A method for adressing the similarity between two vectors involves utilizing the dot product.
- The dot product is just the cosine similarity:



$$\text{similarity}(w^{\text{statistics}}, w^{\text{learning}}) = \cos(\theta) = \frac{< w^{\text{statistics}}, w^{\text{learning}} >}{||w^{\text{statistics}}||\ ||w^{\text{learning}}||}$$

- Even though "statistics" and "learning" may share some meaning, their similarity when represented as two one-hot vectors will be zero.

Part 2 : Word Embedding Methods

Creating Word Embedding

- We need to find a subspace that encodes the relationships between words.
- As there are millions of tokens in any language, we can try to reduce the size of this space from R^V (where V is the vocabulary size) to some D-dimensional space (such that D << V) that is sufficient to encode all semantics of the language.
- Each dimension would encode some meaning (such as tense, count, gender ...)
- We are going to introduce 2 approaches:
  - **GloVe** (Global Vectors for Word Representation, Pennington, Socher and Manning, 2014).
  - The **Word2vec approach**: introduced by Mikolov, Sutskever, et al. (2013).
- Both algorithms take their inspiration from an English linguist, named John Ruper Firth, known for his famous quotation:

« *You shall know a word by the company it keeps* » (J.R. Firth 1957:11)

The GloVe approach - Introduction -

- **GloVe** (Global Vectors for Word Representation) is an unsupervised algorithm developed at the Stanford NLP lab. It learns embedding vectors based on word-word co-occurrence statistics

- The GloVe algorithm involves applying **Matrix Factorization Methods** to a matrix that encapsulates the **co-occurrence statistics** of the corpus.

- Each entry X_{ij} in the co-occurrence matrix represents the number of times the word j occurs in the context of word i, thereby implying the definition of a context size (or window size).

- For example, if i represents the index of the word "enjoyed", we would include the index of the word "I" twice.

$$\overbrace{\underbrace{I}_{X_{ij}+=1}\ \boxed{\text{enjoyed}}\ \text{the}}^{window\ size=1}\ \text{research project.}$$

$$\overbrace{\underbrace{I}_{X_{ij}+=1}\ \boxed{\text{enjoyed}}\ \text{NLP}}^{\text{window size=1}}$$

The GloVe approach - The co-occurrence matrix -

- Let's construct the co-occurrence matrix using a simple corpus consisting of three documents with a vocabulary size of 10 tokens, i.e $V = 10$ The co-occurrence matrix has then a shape (V, V)

- The corpus comprises the following documents:
  - Document 1: I enjoyed the research project .
  - Document 2: : I like Deep Learning .
  - Document 3: I enjoyed NLP .

- The final co-occurrence matrix:

$$
X = \begin{array}{l} \\ I \\ enjoyed \\ the \\ research \\ project \\ like \\ Deep \\ Learning \\ NLP \\ . \end{array}
\begin{array}{c}
\overset{V}{\longleftrightarrow} \\
\begin{array}{cccccccccc} I & enjoyed & the & research & project & like & Deep & Learning & NLP & . \end{array} \\
\begin{pmatrix}
0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\end{array}
\updownarrow V
$$

The GloVe approach - SVD based methods -

- To create **embedding vectors** from the **co-occurrence matrix**, one approach can be to use a **Singular Value decomposition** (SVD) of the co-occurrence matrix:

$$\underset{(V \times V)}{X} = \underset{(V \times V)}{W_1} \ \underset{(V \times V)}{\Omega} \ \underset{(V \times V)}{W_2^T}$$

- Then, we reduce the dimensionality by selecting the first D singular vectors (with **D << V**)

$$\hat{X} = \hat{W}_1 \ \hat{\Omega} \ \hat{W}_2^T$$

$(V \times V)$ $(V \times D)$ $(D \times D)$ $(D \times V)$

- Let $\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_V)$, such that $\omega_1 > \omega_2 > \cdots > \omega_V$

- We select D so that we can capture the desired amout of variance we want: $\dfrac{\sum_{i=1}^{D} \omega_i}{\sum_{i=1}^{V} \omega_i}$

The GloVe approach - Matrix Factorization instead of SVD -

- The SVD approach does not work well in practice for several reasons:
  - The dimensions of the matrix change very often (new words are added very frequently and the corpus changes in size).
  - The matrix is extremely sparse (i.e, it contains a lot of zero values) since most words do not co-occur.
  - The matrix is very high dimensional as the vocabulary size is usually huge.

- We are going to introduce another way of performing the factorization: **Matrix Factorization Methods** are widely used for generating **meaningful** and **low-dimensional word representation**.

- In the GloVe approach, since non-zero values are very large, we factorize the logarithm of X (denoted $\log X$) instead of factorizing X.
- Remark: Obviously, as we can't apply the logarithm function on the entries with a zero value, we add 1 to all the element of the matrix before applying the logarithm).

$$\forall (i,j) \in V^2 \quad X_{ij} \leftarrow X_{ij} + 1$$

- We want to factorize $\log X$ into 2 matrices: $\log X \approx W\tilde{W}^T$

- We want to estimate $W, \tilde{W} \in \mathbb{R}^{V \times D}$ with $D << V$

Matrix Factorization for Collaborative Filtering

- Let us introduce the concept of Matrix Factorization in the context of **Collaborative Filtering**.

- Let us imagine that we have users who rate movies on some platform.
  - The number of users is N
  - The number of movies is K
  - Each rating is a real number

	Movie 1	…	Movie k	…	Movie K
User 1	-2				-1
User n	5		4		
User N	1				

- We obtain a highly sparse matrix R of shape (N, K) reflecting the fact that most users have seen very few movies.

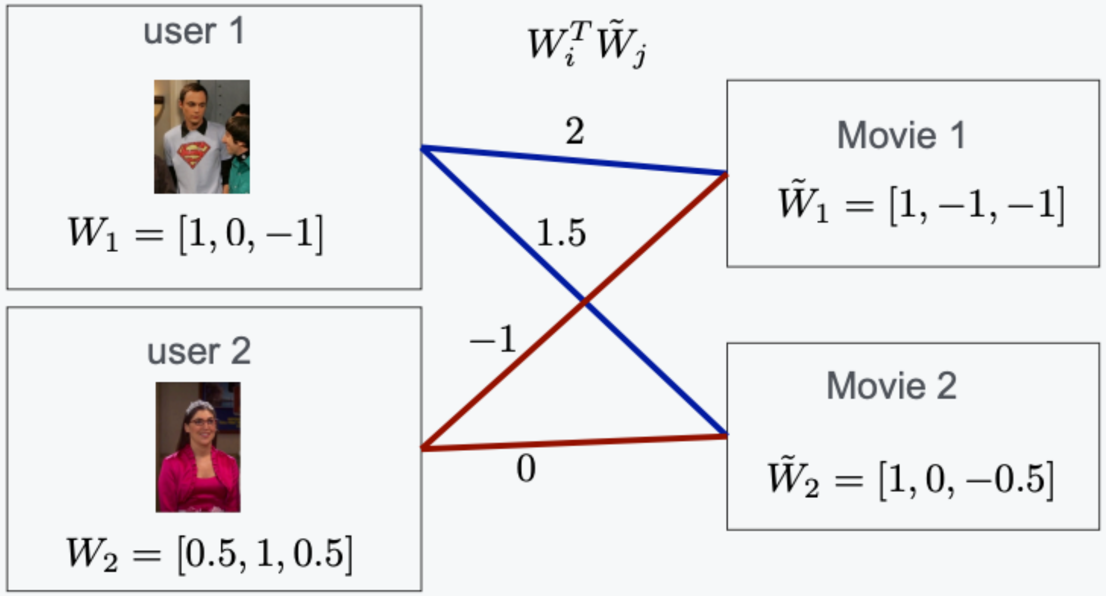
- Collaborative Filtering involves leveraging the ratings provided by other users to predict the rating of a movie for a user who has not yet seen it.

Matrix Factorization for Collabative Filtering

- In order to approximate R , we factorize the matrix into two matrices $R \approx \hat{R} = W\tilde{W}^T$
- We want to estimate the parameters W and $\tilde{W}$ by minimizing $J = \sum_{i=1}^{N}\sum_{j=1}^{K}(R_{ij} - \hat{R}_{ij})^2 = \sum_{i=1}^{N}\sum_{j=1}^{K}(R_{ij} - W_i^T\tilde{W}_j)^2$

$$W = \begin{pmatrix} - & W_1 & - \\ \vdots & \vdots & \vdots \\ - & W_N & - \end{pmatrix} \in \mathbb{R}^{N \times D} \qquad \tilde{W} = \begin{pmatrix} - & \tilde{W}_1 & - \\ \vdots & \vdots & \vdots \\ - & \tilde{W}_K & - \end{pmatrix} \in \mathbb{R}^{K \times D}$$

- Each row i of the W matrix is a D-dimensional vector representing the user i. Each dimension encodes a latent meaningful information about the user.
- Each row j of the $\tilde{W}$ matrix is a D-dimensional vector representing the movie j. Similarly, each dimension encodes a meaningful information about the movie.
- As an example: let us consider D=3 latent dimensions:
  - Sci-fi
  - Comedy
  - Romance



The GloVe approach - Summary

- In the GloVe approach, we are going to approximate the logarithm of the co-occurrence matrix by using the same factorization method.

- We also add a bias term for the matrix W and a bias term for $\tilde{W}$

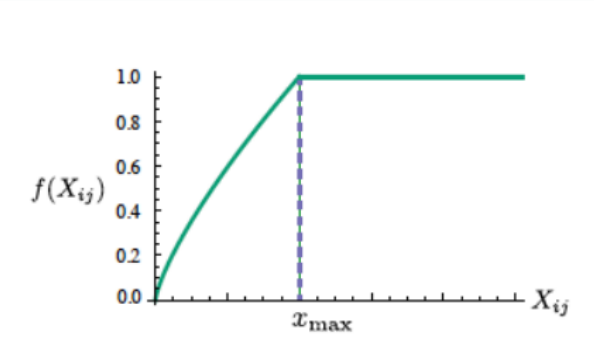
$$\forall (i,j) \in \{1, \ldots, V\}^2 \quad \log X_{ij} \approx W_i^T \tilde{W}_j + b_i + \tilde{b}_j$$

- The parameters of the model:

$$W = \begin{pmatrix} - & W_1 & - \\ \vdots & \vdots & \vdots \\ - & W_V & - \end{pmatrix} \in \mathbb{R}^{V \times D} \quad \tilde{W} = \begin{pmatrix} - & \tilde{W}_1 & - \\ \vdots & \vdots & \vdots \\ - & \tilde{W}_V & - \end{pmatrix} \in \mathbb{R}^{V \times D} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_V \end{pmatrix} \in \mathbb{R}^V \quad \tilde{b} = \begin{pmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_V \end{pmatrix} \in \mathbb{R}^V$$

- The cost function of the weighted least squares regression model is:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) (\log X_{ij} - W_i^T \tilde{W}_j - b_i - \tilde{b}_j)^2$$



- The weights $f(X_{ij})$ are added because we consider that rare occurrences are noisy and carry less information than the more frequent ones.

The Word2vec approach – The idea –

- To construct word embeddings using the Word2vec methodology, the key concept involves defining a word by its contextual usage across all documents within the training corpus.

- For example, consider the word "economy". Clearly, it will not appear in the same context as the word "rock".

- The Word2vec approach consists in creating word embedding vectors by using a shallow neural network in order to:
  - Predict the center word (« economy » in our example ») from the context words. It’s called The Continuous Bag of Words method (CBOW)
  - Predict the context words from the center word. It’s called the Skipgram method.

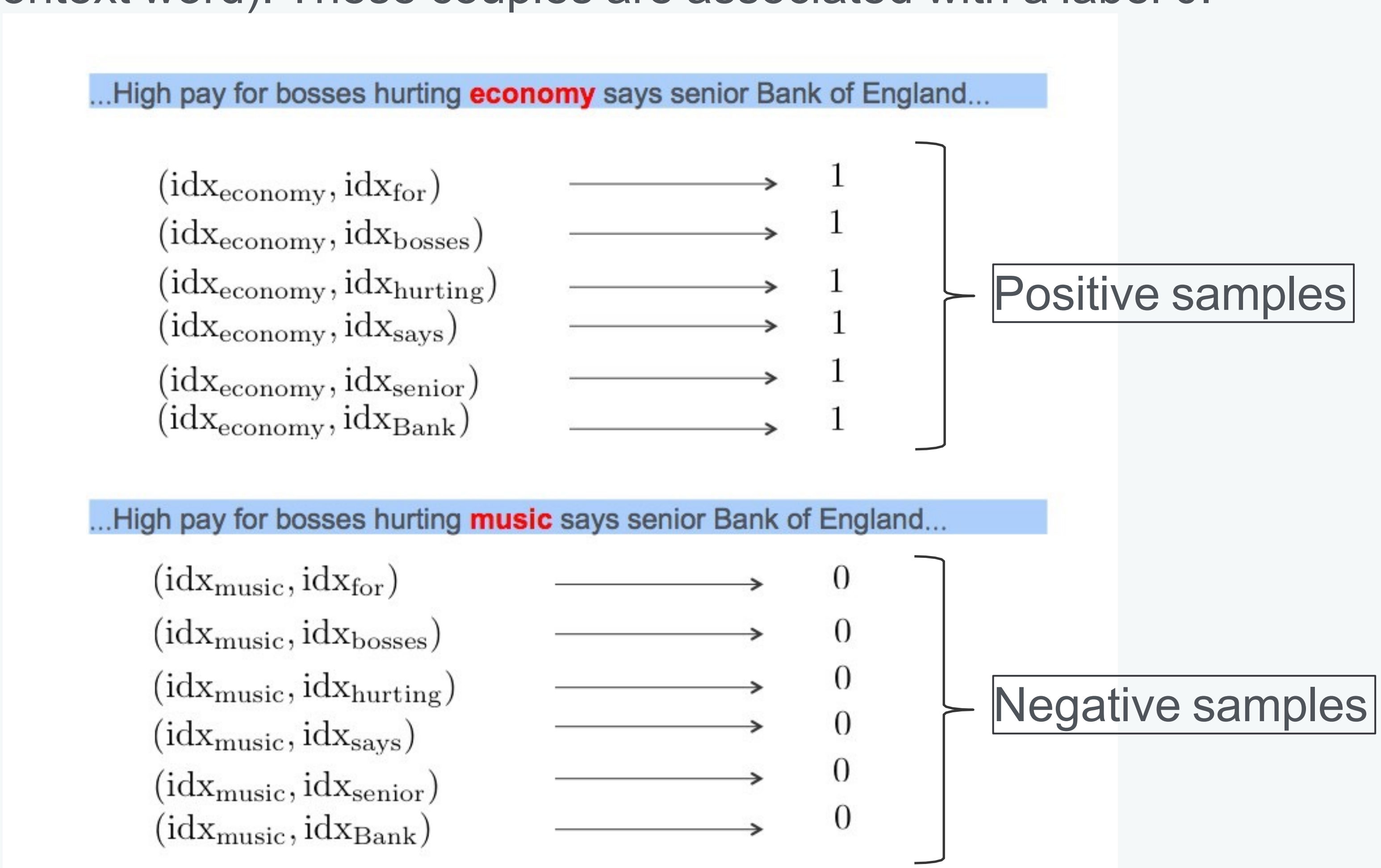
The Word2vec approach - The data -

- Let us consider a window size of D.

- For each center word in our corpus, we have a list of 2*D context words associated with this center word (the ones on the right and the ones on the left).

- We can then define 2*D couples of (center word, context word) as shown in the figure with « economy » as a center word. These couples are associated with a label 1.

- By sampling a random word in the corpus, we can create other false couples of (center word, context word). These couples are associated with a label 0.



The Word2vec approach - The Forward Propagation -

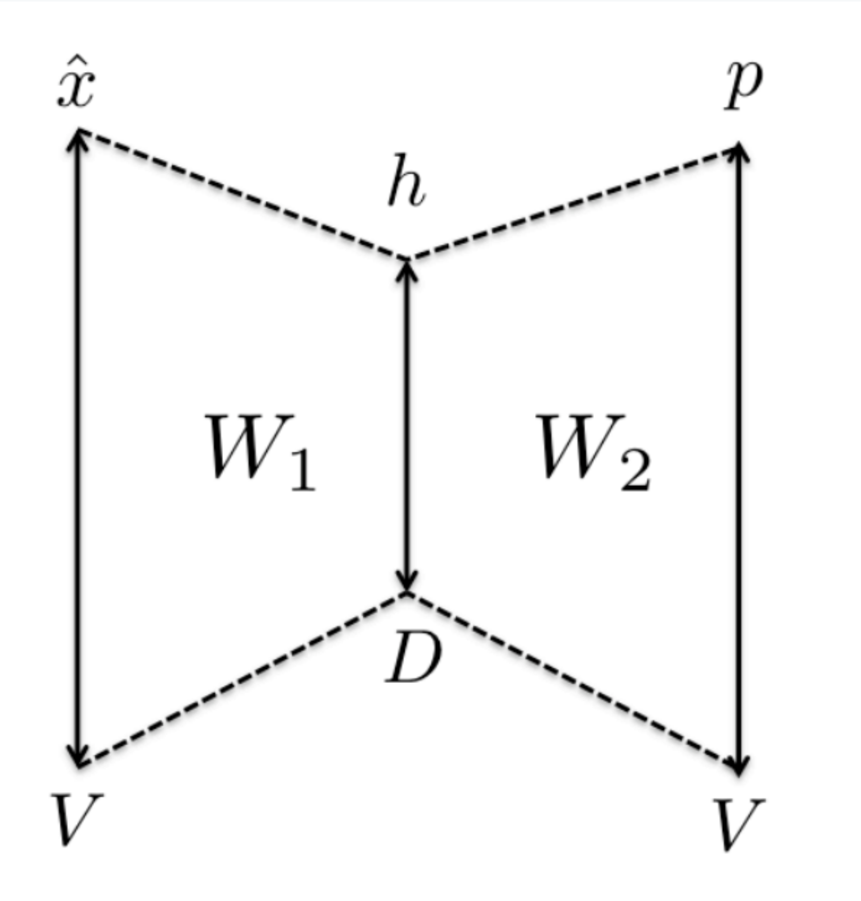
- A one hot vector $\hat{x}$ representing the center word is feeded to the neural network.

- A first linear transformation maps $\hat{x}$ to the D-dimensional vector h as follows:

$$h = W_1^T \hat{x}$$

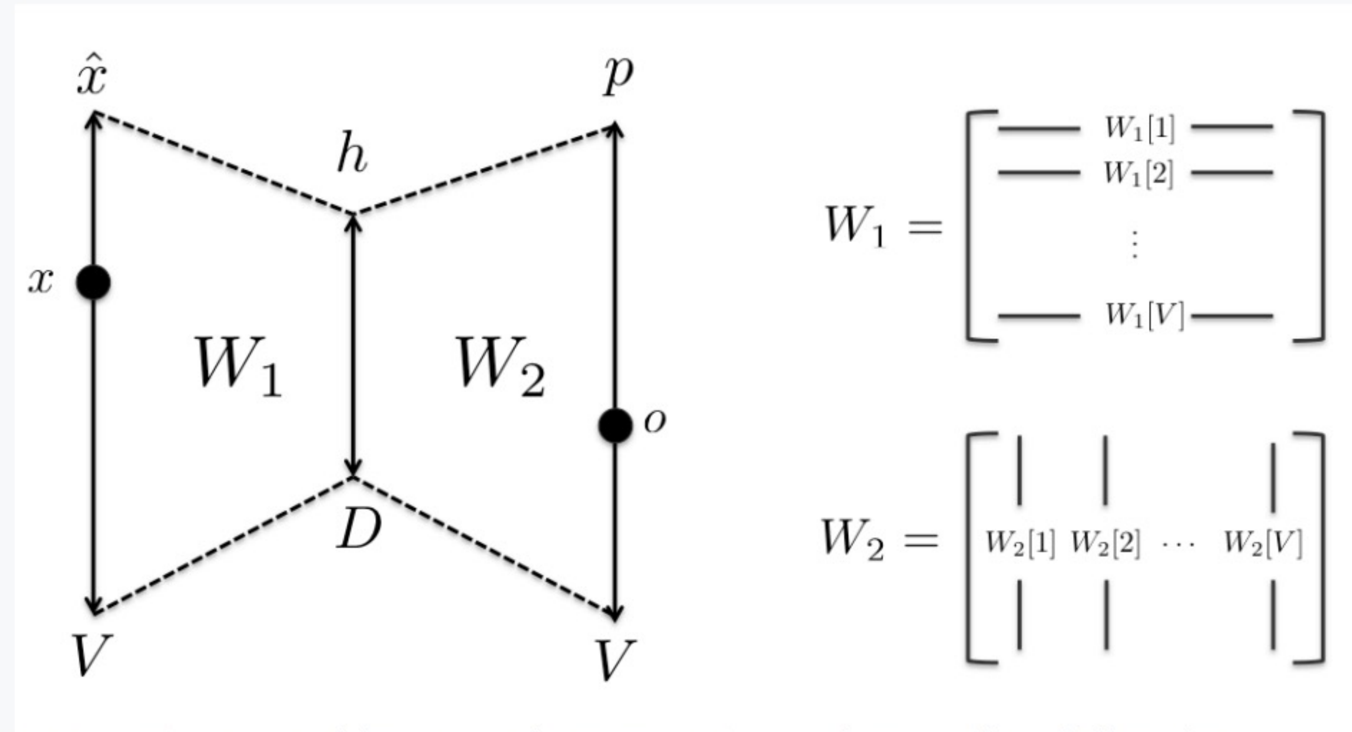
- A second transformation maps the hidden vector h to the prediction vector p as follows:

$$p = \text{sigmoid}(W_2^T h)$$



The Word2vec approach - The Forward Propagation -

- Let x be the non zero position in the one hot vector $\hat{x}$ and o be one of the V dimensions of the prediction vector p

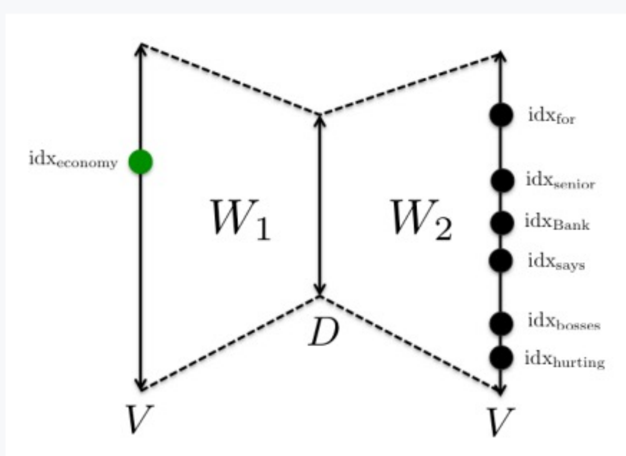


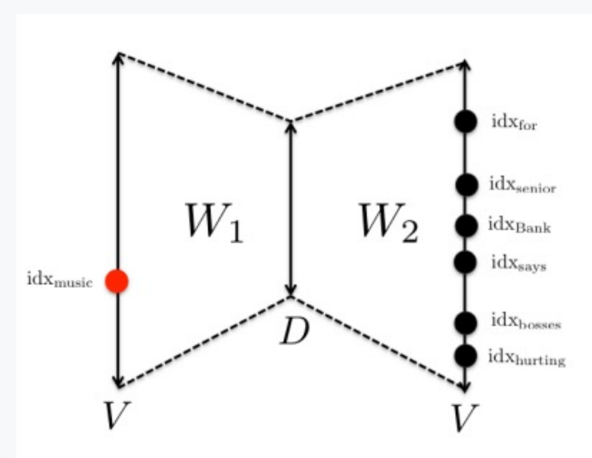
- We can easily prove that:

$$p_o = \text{sigmoid}(W_1[x]^T W_2[o])$$

- In other words, to predict p_o, we only need the row x of the matrix W_1 and the row o of the matrix W_2

- Let us consider the example of « economy » as a center word in one document, we have tho following couples of (center word, context word) , for center word in $\mathcal{C} = \{\text{idx}_{\text{for}}, \text{senior}_{\text{Bank}}, \text{idx}_{\text{says}}, \text{idx}_{\text{bosses}}, \text{idx}_{\text{hurting}}\}$





- The loss associated with these 12 samples is:

$$J = -\sum_{c \in \mathcal{C}} \left[\log(\sigma(W_1[\text{idx}_{\text{economy}}]^T W_2[c])) + \log(1 - \sigma(W_1[\text{idx}_{\text{music}}]^T W_2[c]))\right]$$

The Word2vec approach - The Learning Process -

Pseudo code:

- Initialize W_1 and W_2 randomly.
- Initilize an empty list of losses.
- For each epoch:
    - Shuffle the sequences.
    - For each sequence in sequences:
        - For each position in the sequence
            - Get the true center word (corresponding to the position).
            - Get the context of the true center word.
            - Get the fake center word.
            - Do one step of SGD for the true center word to update W_1 and W_2
            - Do one step of SGD for the fake center word to update W_1 and W_2
    - Keep track of the loss function by appending the list of losses

Programming Session