Τι θα δούμε σήμερα

- Τα βασικά στοιχεία των word embeddings
- Ερωτήσεις, ασκήσεις
- Στατιστικά συλλογής (και ίσως συμπίεση)

Word embeddings II (basics)

Διανυσματική *αναπαράσταση* (representation) λέξεων – *κατανεμημένη (distributed) αναπαράσταση*

embedding

λέξη → διάνυσμα

dense, low dimension

Στόχος: όμοιες λέξεις -> όμοια διανύσματα

word2vec

Είσοδος κείμενα, προτάσεις

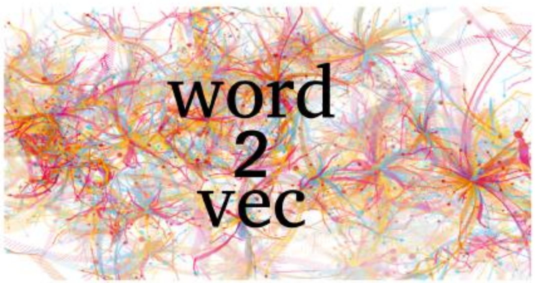
Text

feed in text

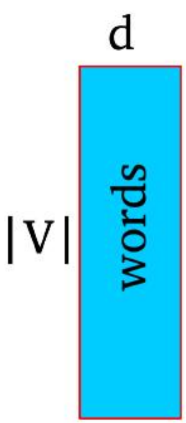
Wikipedia

Google news

…



wait a few hours



dog = (0.12,-0.32,0.92,0.43,-0.3 ...)
cat = (0.15,-0.29,0.90,0.39,-0.32 ...)

chair = (0.8,0.9,-0.76,0.29,0.52 ...)

get a |V|xd matrix W where each row is a vector for a word

Ομοιότητα/απόσταση

4

1 2 3 5

6

- Similarity is calculated using *cosine similarity*:

$$sim(\vec{dog}, \vec{cat}) = \frac{\vec{dog} \cdot \vec{cat}}{||\vec{dog}|| \, ||\vec{cat}||}$$

||(1, 2, 3||

- For normalized vectors ($||x|| = 1$), this is equivalent to a dot product:

$$sim(\vec{dog}, \vec{cat}) = \vec{dog} \cdot \vec{cat}$$

- **Normalize the vectors when loading them.**

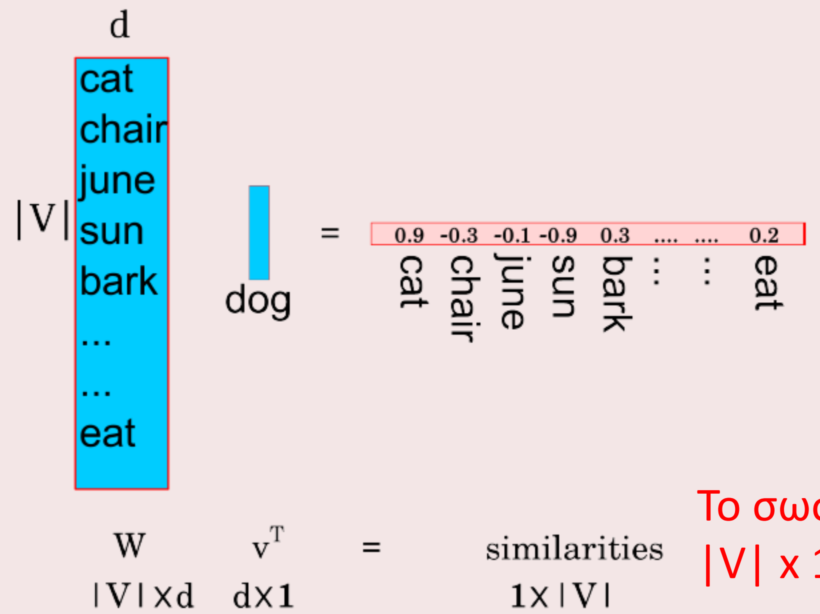
word2vec

- dog
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig
- sheep
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock
- november
  - october, december, april, june, february, july, september, january, august, march
- jerusalem
  - tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ramla, safed
- teva
  - pfizer, schering-plough, novartis, astrazeneca, glaxosmithkline, sanofi-aventis, mylan, sanofi, genzyme, pharmacia

Πως θα βρούμε τις ποιο όμοιες λέξεις με το dog;

TIP: Όπου μπορούμε χρησιμοποιούμε πράξεις πινάκων. Γιατί;

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^{\top}$



Το σωστό
|V| x 1

- Result is a $|V|$ sized vector of similarities.
- Take the indices of the k-highest values.

Λέξη ποιο όμοια σε πολλές άλλες;

- “Find me words most similar to cat, dog and cow”.
- Calculate the pairwise similarities and sum them:

$$W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow}$$

- Now find the indices of the highest values as before.

- Matrix-vector products are wasteful. **Better option:**

$$W \cdot (\vec{cat} + \vec{dog} + \vec{cow})$$

Σε προηγούμενα μαθήματα είδαμε

Lemmatization
Stemming

Λέξεις σημασιολογικά κοντινές

Πως θα πάρουμε αυτόν τον πίνακα;

Βασική ιδέα

Μία λέξη προσδιορίζεται από τις συμφραζόμενες της λέξεις (context)

Ο καθηγητής διδάσκει το μάθημα στους φοιτητές του στην αίθουσα.

Παράθυρο (window) = 3

Center word
Context word

Κάθε λέξη δύο αναπαραστάσεις: (1) center (2) context
Δηλαδή, έχουμε 2 |V| x d πίνακες

- Το center-διάνυσμα της center λέξης πρέπει να είναι *όμοιο* με τα context-διανύσματα (δηλαδή, το άθροισμα των context διανυσμάτων) των context λέξεων
- Και προφανώς το *συμμετρικό*

Learning: παραδείγματα κειμένου και προσπαθούμε να «μάθουμε» αυτά τα διανύσματα (βάρη)
Training examples – fix the matrices to work for them

How does word2vec work?

While more text:

w: center representation – c: context representation

- Extract a word window:

A springer is[a cow or **heifer** close to calving].

a	cow	or	**heifer**	close	to	calving
c_1	c_2	c_3	w	c_4	c_5	c_6

- Try setting the vector values such that:

$$\sigma(w\cdot\ c_1)+\sigma(w\cdot\ c_2)+\sigma(w\cdot\ c_3)+\sigma(w\cdot\ c_4)+\sigma(w\cdot\ c_5)+\sigma(w\cdot\ c_6)$$

is **high**

How does word2vec work?

While more text:

w: center representation – c: context representation

- Extract a word window:

```
A springer is[ a   cow   or   heifer   close   to   calving ].
```

$$\text{a} = c_1,\ \text{cow} = c_2,\ \text{or} = c_3,\ \textbf{heifer} = w,\ \text{close} = c_4,\ \text{to} = c_5,\ \text{calving} = c_6$$

- Try setting the vector values such that:

$$\sigma(w\cdot c_1)+\sigma(w\cdot c_2)+\sigma(w\cdot c_3)+\sigma(w\cdot c_4)+\sigma(w\cdot c_5)+\sigma(w\cdot c_6)$$

is **high**

Negative sampling (αρνητικά παραδείγματα)

- Create a corrupt example by choosing a random word w' (negative sample)

```
[ a   cow   or   comet   close   to   calving ]
```

$$\text{a} = c_1,\ \text{cow} = c_2,\ \text{or} = c_3,\ \textbf{comet} = w',\ \text{close} = c_4,\ \text{to} = c_5,\ \text{calving} = c_6$$

- Try setting the vector values such that:

$$\sigma(w'\cdot c_1)+\sigma(w'\cdot c_2)+\sigma(w'\cdot c_3)+\sigma(w'\cdot c_4)+\sigma(w'\cdot c_5)+\sigma(w'\cdot c_6)$$

is **low**

Word2Vec

Two **algorithms**

1. Continuous Bag of Words (CBOW)

   Predict center word from a bag-of-words context

2. Skip-grams (SG)

   Predict context words given the center word

Position independent (do not account for distance from center)

Two **training methods**

1. Hierarchical softmax
2. Negative sampling

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality.* NIPS 2013: 3111-3119

Βασική ιδέα

Το σκυλί __ την ουρά
Η γάτα __ το ποντίκι
Ο ήλιος __ το πρωί
Το φεγγάρι __ κάθε νύχτα

CBOW

__ __ κουνά __ __
__ __ κυνηγάει __ __
__ __ ανατέλλει __ __
__ __ δύει __ __

skipgram

Ας δούμε πάλι και κάποιες λεπτομέρειες

One-hot vectors

Έστω ότι υπάρχουν |V| διαφορετικές λέξεις (όροι) στο λεξικό μας

- Διατάσσουμε τις λέξεις αλφαβητικά
- Αναπαριστούμε κάθε λέξη με ένα $\mathrm{R}^{|V|x1}$ διάνυσμα που έχει παντού 0 και μόνο έναν 1 στη θέση που αντιστοιχεί στη θέση της λέξης στη διάταξη

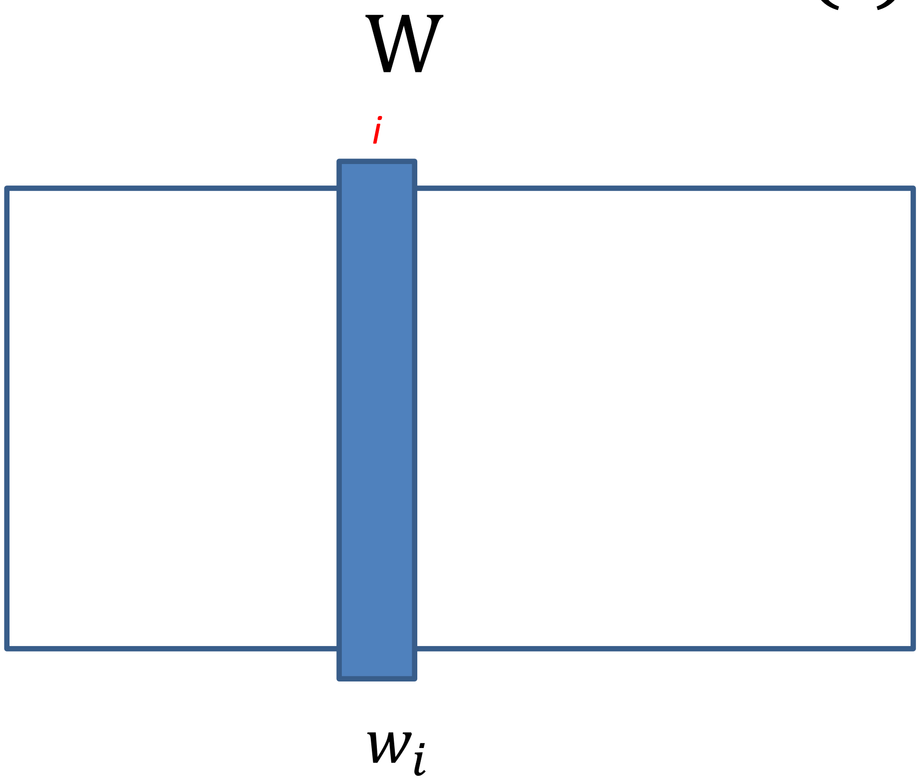
$$w^{aardvark} = \begin{bmatrix} \mathbf{1} \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{a} = \begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{at} = \begin{bmatrix} 1 \\ 0 \\ \mathbf{0} \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad \cdots \quad w^{zerba} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ \mathbf{1} \end{bmatrix}$$

- Καμία πληροφορία για ομοιότητα
- Πολλές διαστάσεις

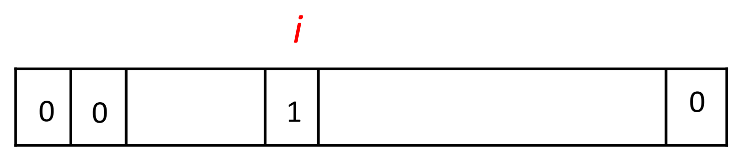
Given matrix W, πως παίρνουμε το embedding της i-οστής λέξης;

Lookup/project

$$ENC(i) = W\ I_i$$



$One\ hot\ vector\ I_i$



One-hot or indicator vector, all 0s but position *i*

CBOW

$|V|$ number of words
N size of embedding
m size of the window (context)

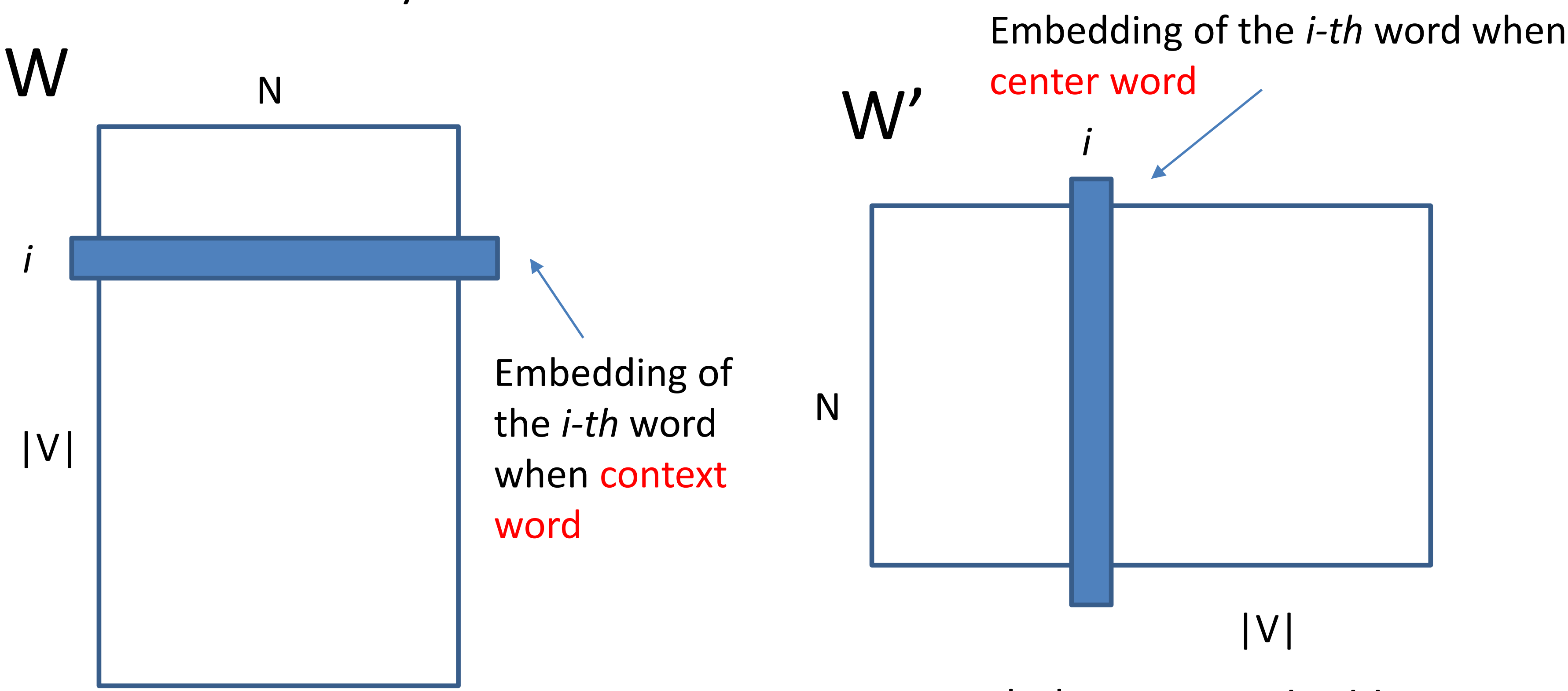
Use a window of context words to predict the center word

Input: 2m context words
Output: center word
each represented as a one-hot vector

CBOW

Use a window of context words to predict the center word

Learns two matrices (<u>two embeddings per word</u>, one when context, one when center)



|V| x N context embeddings when ***input***

N x |V| center embeddings when ***output***

CBOW

Intuition
The W'-embedding of the *center word* should be *similar* to the (sum of the) W-embeddings of its *context words*

We want similarity close to one for the center word and close to 0 for all other words

CBOW

Given window size m

$x^{(c)}$ one hot vector for context words, y one hot vector for the center word

1. **INPUT:** the *one hot vectors* for the *2m* context words

$x^{(c-m)}$, …, $x^{(c-1)}, x^{(c+1)}$, …, $x^{(c+m)}$

2. *GET THE EMBEDDINGS of the context words*

$v_{c-m} = Wx^{(c-m)}$, …, $v_{c-1} = Wx^{(c-1)}, v_{c+1} = Wx^{(c+1)}$, …, $v_{c+m} = Wx^{(c+m)}$

3. *TAKE THE SUM* these vectors

$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \cdots v_{c+m}}{2m}$, $\hat{v} \in R^N$

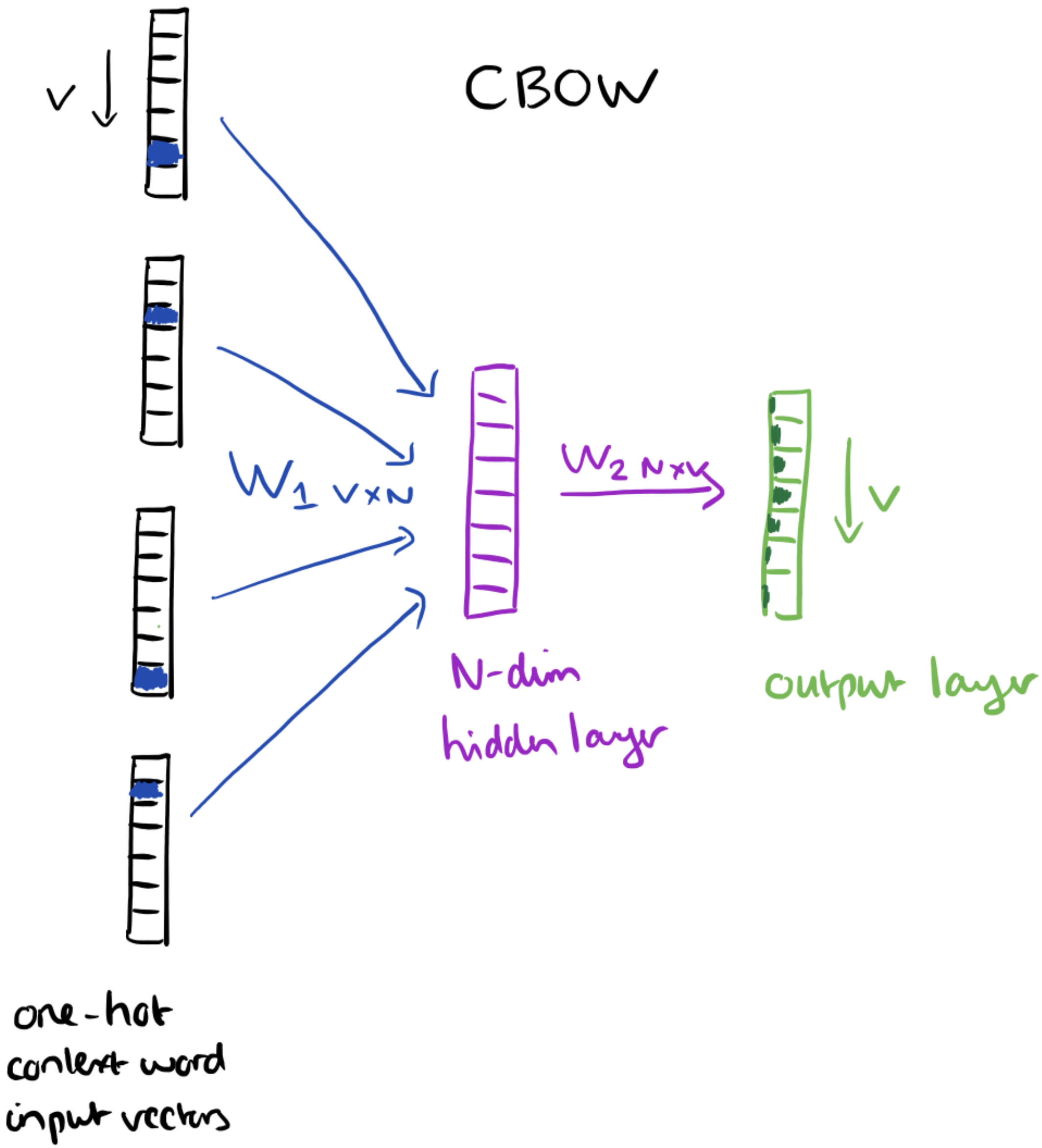
4. *COMPUTE SIMILARITY: dot produce W' (all center vectors) and context* $\hat{v}$

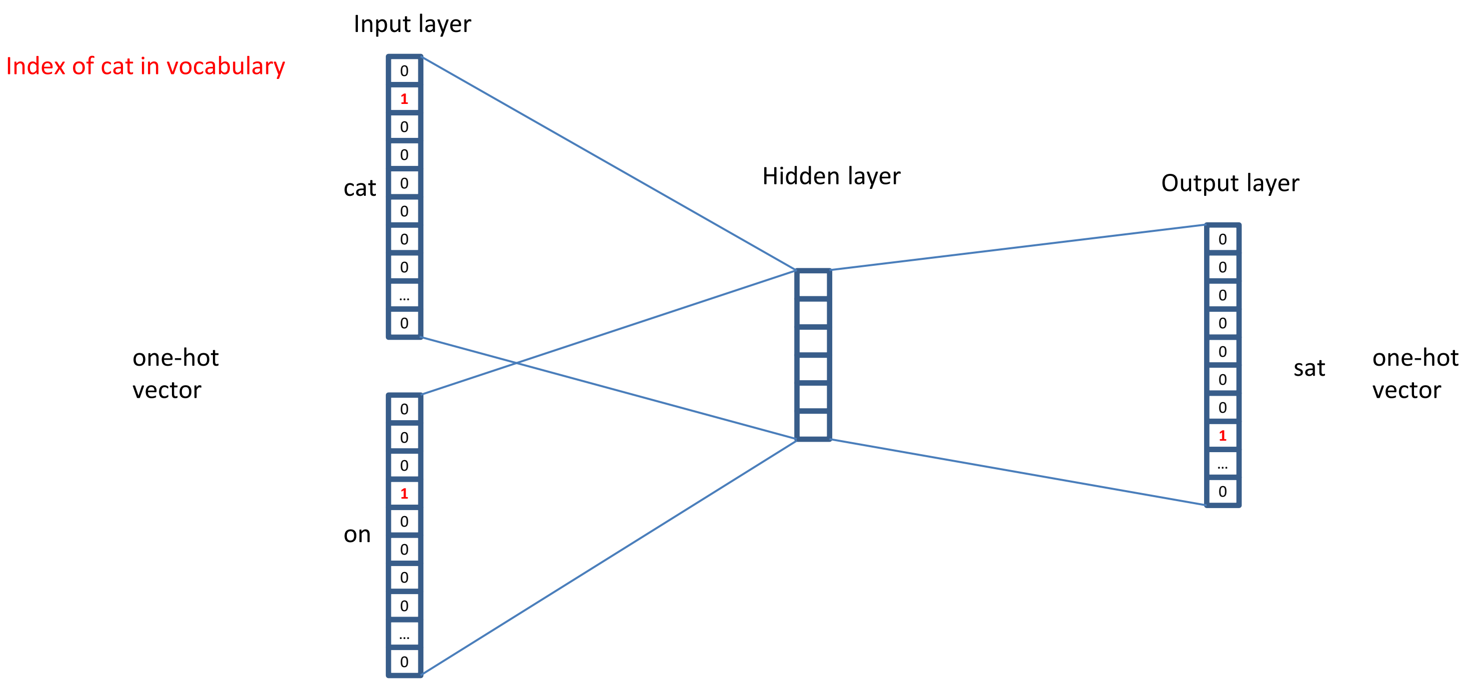
z = W' $\hat{v}$

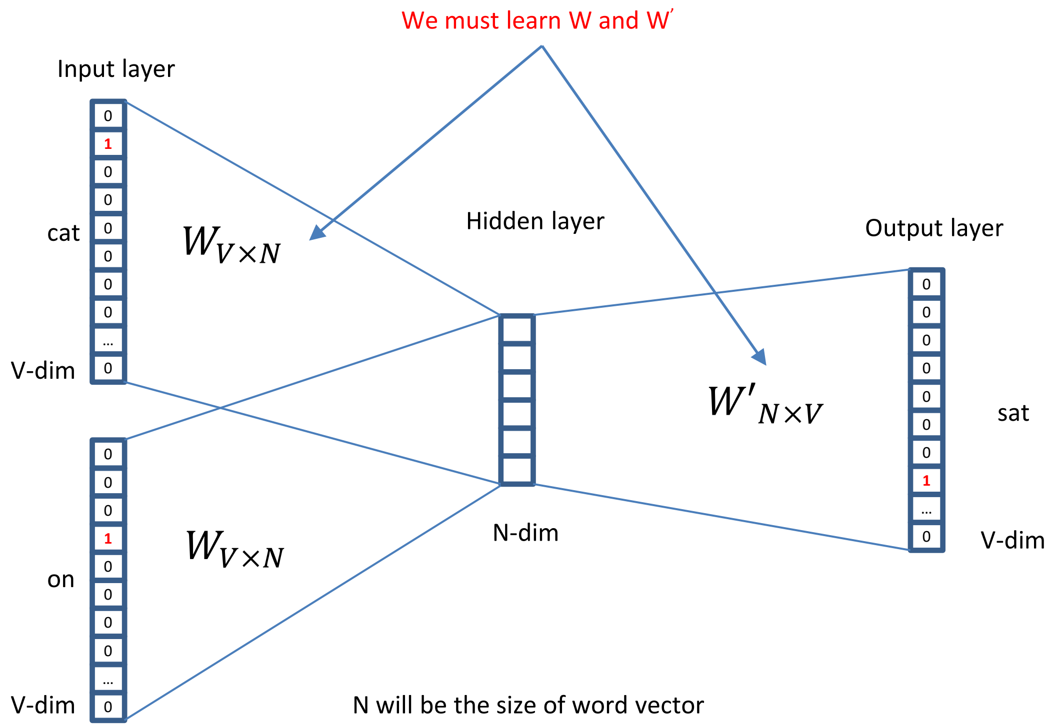
5. Turn the *score vector to probabilities*

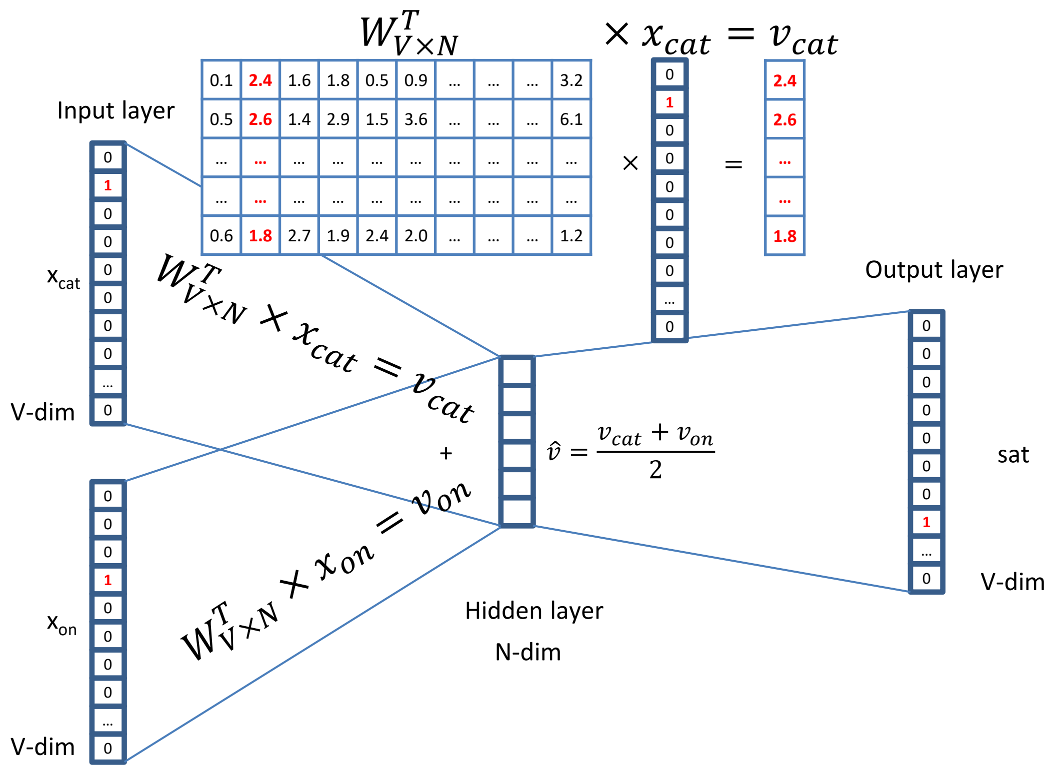
$\hat{y}$ = *softmax(z)*

We want this to be close to 1 for the center word

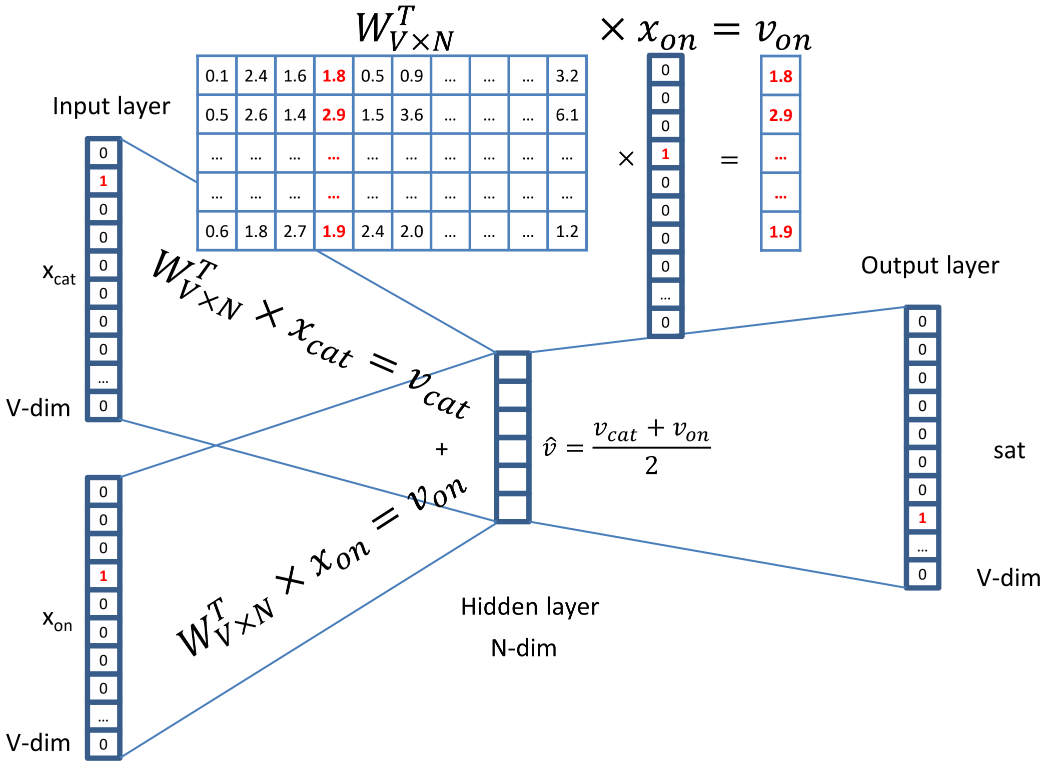

CBOW
V
W1 VxN
W2 NxV
V
N-dim hidden layer
output layer
one-hot context word input vectors

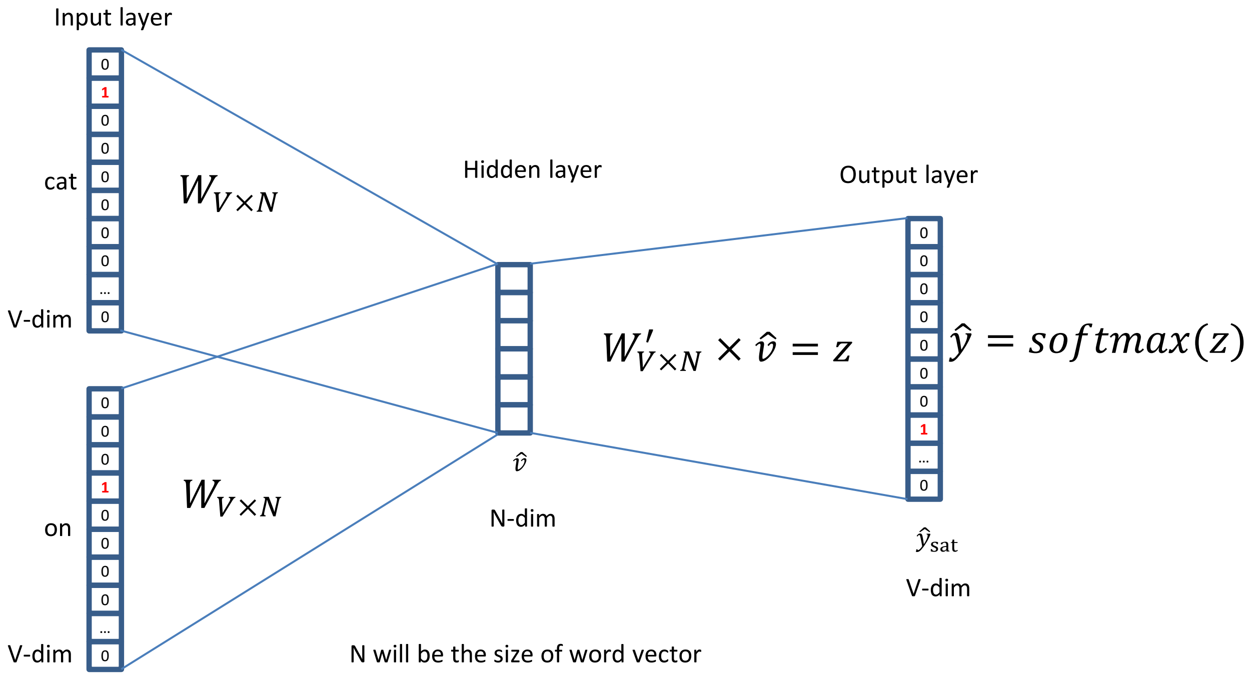

Input layer
Index of cat in vocabulary
0
1
0
0
0
0
0
0
...
0
cat
Hidden layer
Output layer
0
0
0
0
0
0
0
1
...
0
sat
one-hot
vector
one-hot
vector
0
0
0
1
0
0
0
0
...
0
on

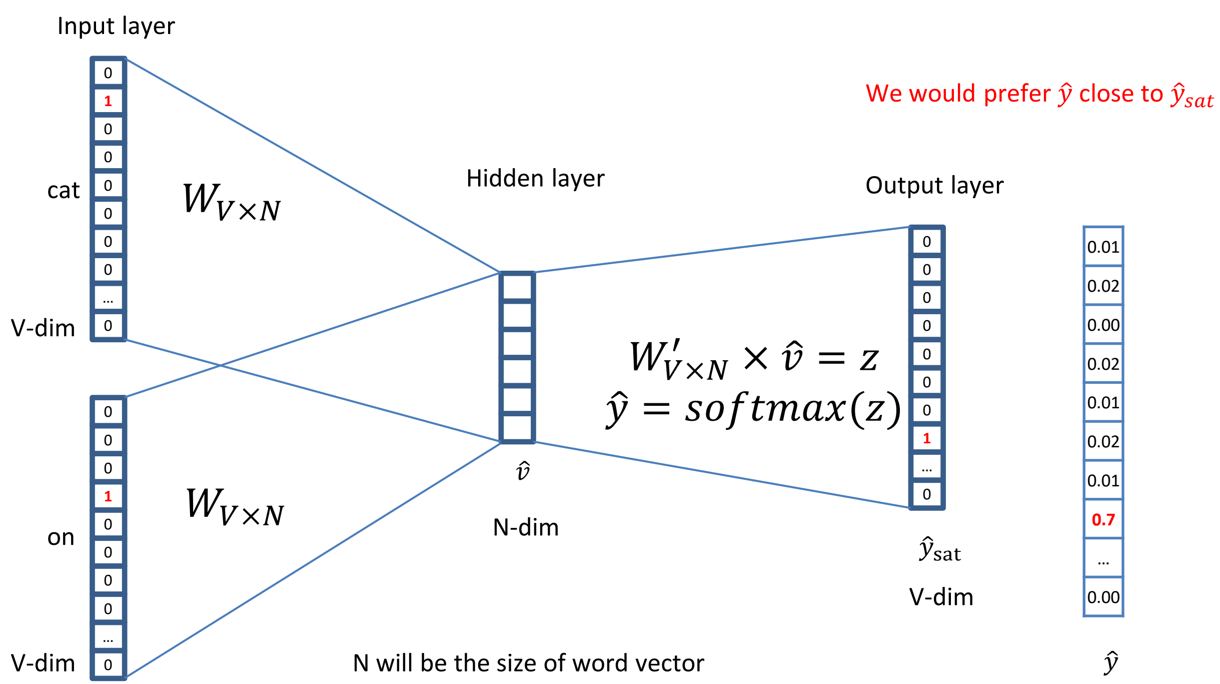

We must learn W and W′
Input layer
cat
$W_{V\times N}$
Hidden layer
Output layer
V-dim
$W'_{N\times V}$
sat
on
$W_{V\times N}$
N-dim
V-dim
V-dim
N will be the size of word vector



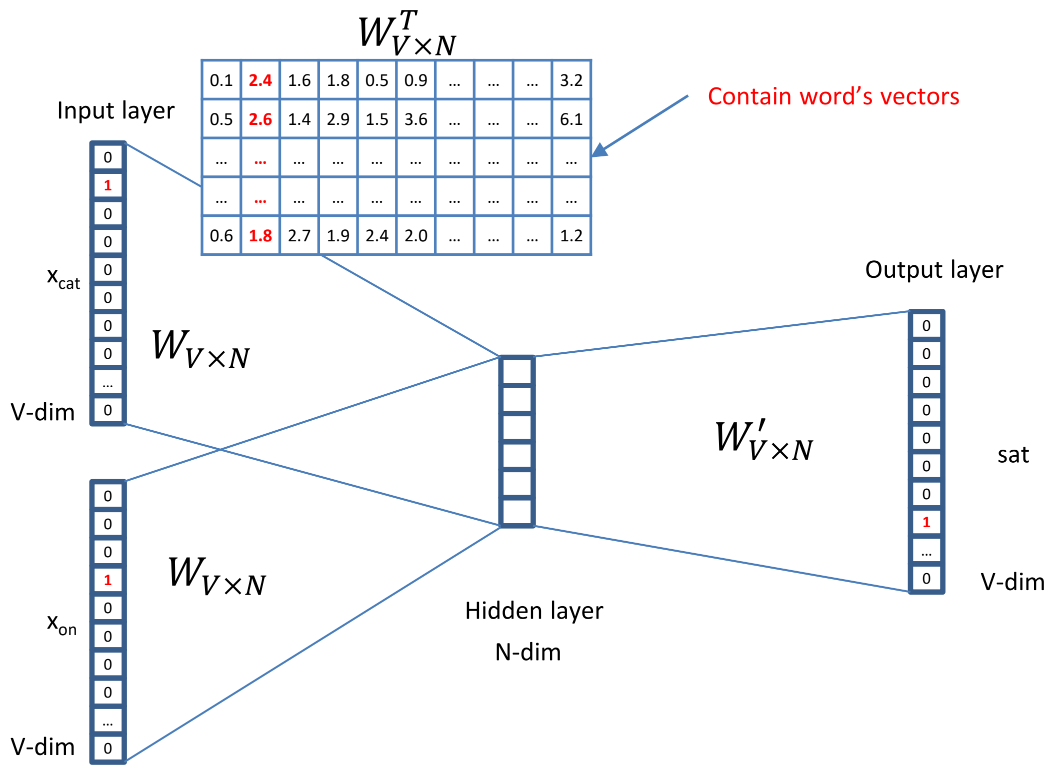
0.1	2.4	1.6	1.8	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	2.9	1.5	3.6	…	…	…	6.1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
0.6	1.8	2.7	1.9	2.4	2.0	…	…	…	1.2


$W_{V\times N}^T$
$\times x_{on} = v_{on}$
Input layer
x_{cat}
V-dim
x_{on}
V-dim
$W_{V\times N}^T \times x_{cat} = v_{cat}$
+
$W_{V\times N}^T \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim


Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N} \times \hat{v} = z$
Output layer
$\hat{y} = softmax(z)$
$\hat{y}_{\text{sat}}$
V-dim
N will be the size of word vector



Input layer
cat
V-dim
on
V-dim
0
1
0
0
0
0
0
0
...
0
0
0
0
1
0
0
0
0
...
0
$W_{V \times N}$
$W_{V \times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V \times N} \times \hat{v} = z$
$\hat{y} = softmax(z)$
Output layer
0
0
0
0
0
0
0
1
...
0
$\hat{y}_{\text{sat}}$
V-dim
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
...
0.00
$\hat{y}$
N will be the size of word vector



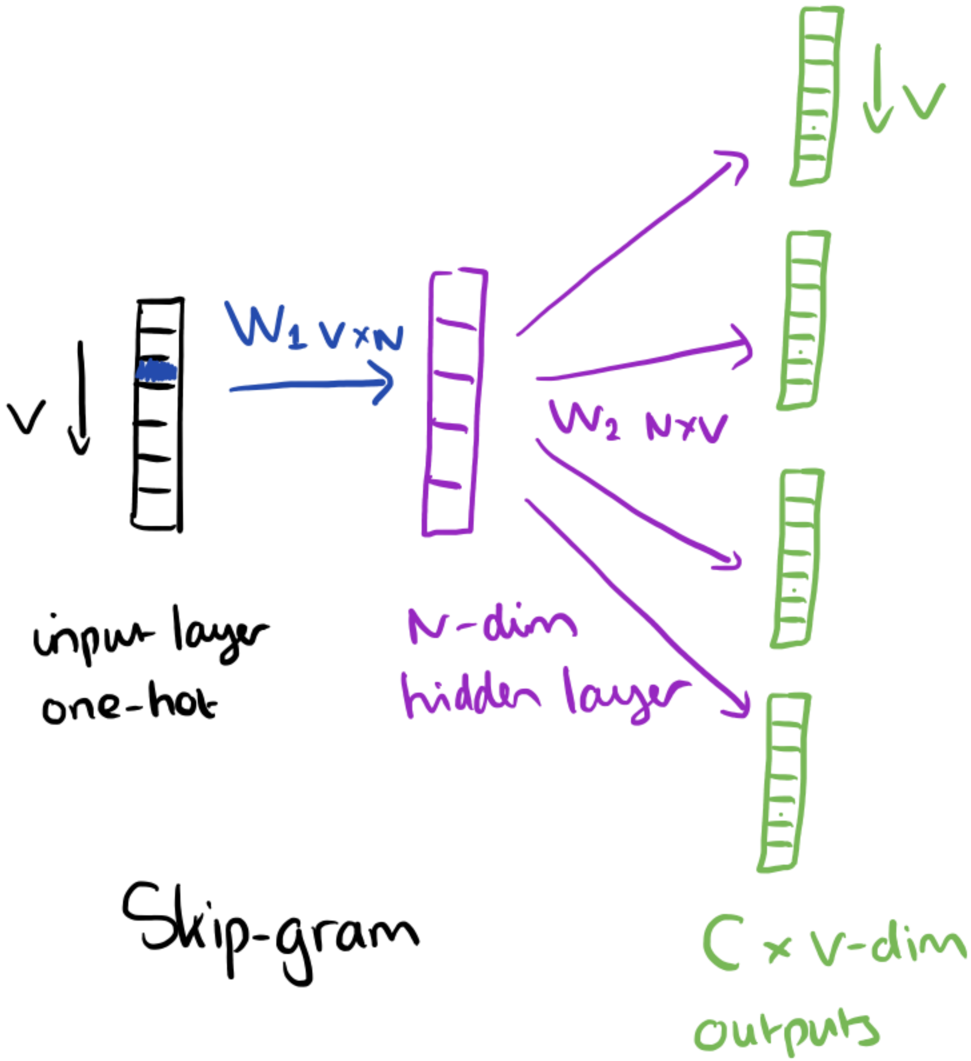
We can consider either W (context) or W' (center) as the word's representation.
Or even take the average.

Skipgram

Given the center word, predict (or, generate) the context words

Input: center word
Output: 2m context word
each represented as a one-hot vectors

Learn two matrices
W: N x |V|, input matrix, word representation as center word
W’: |V| x N, output matrix, word representation as context word


V
W1 VxN
W2 NxV
V
input layer
one-hot
N-dim
hidden layer
C x V-dim
outputs

Skip-gram

Skipgram

$y^{(j)}$ one hot vector for context words

1. Input: *one hot vector* of the center word

x

2. Get the *embedding of the center word*

$v_c = W\ x$

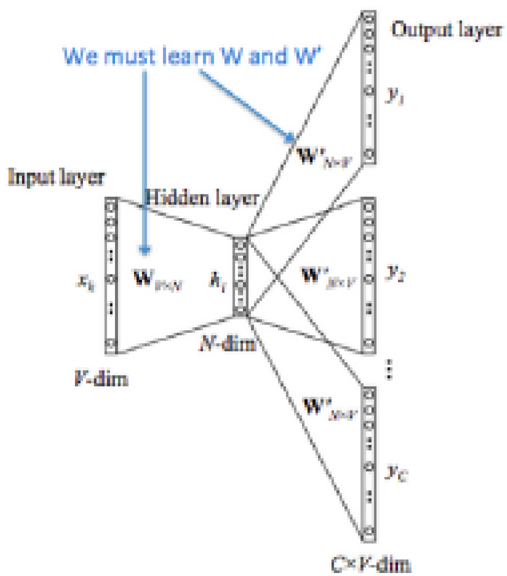
3. Generate a *score vector for each context word*

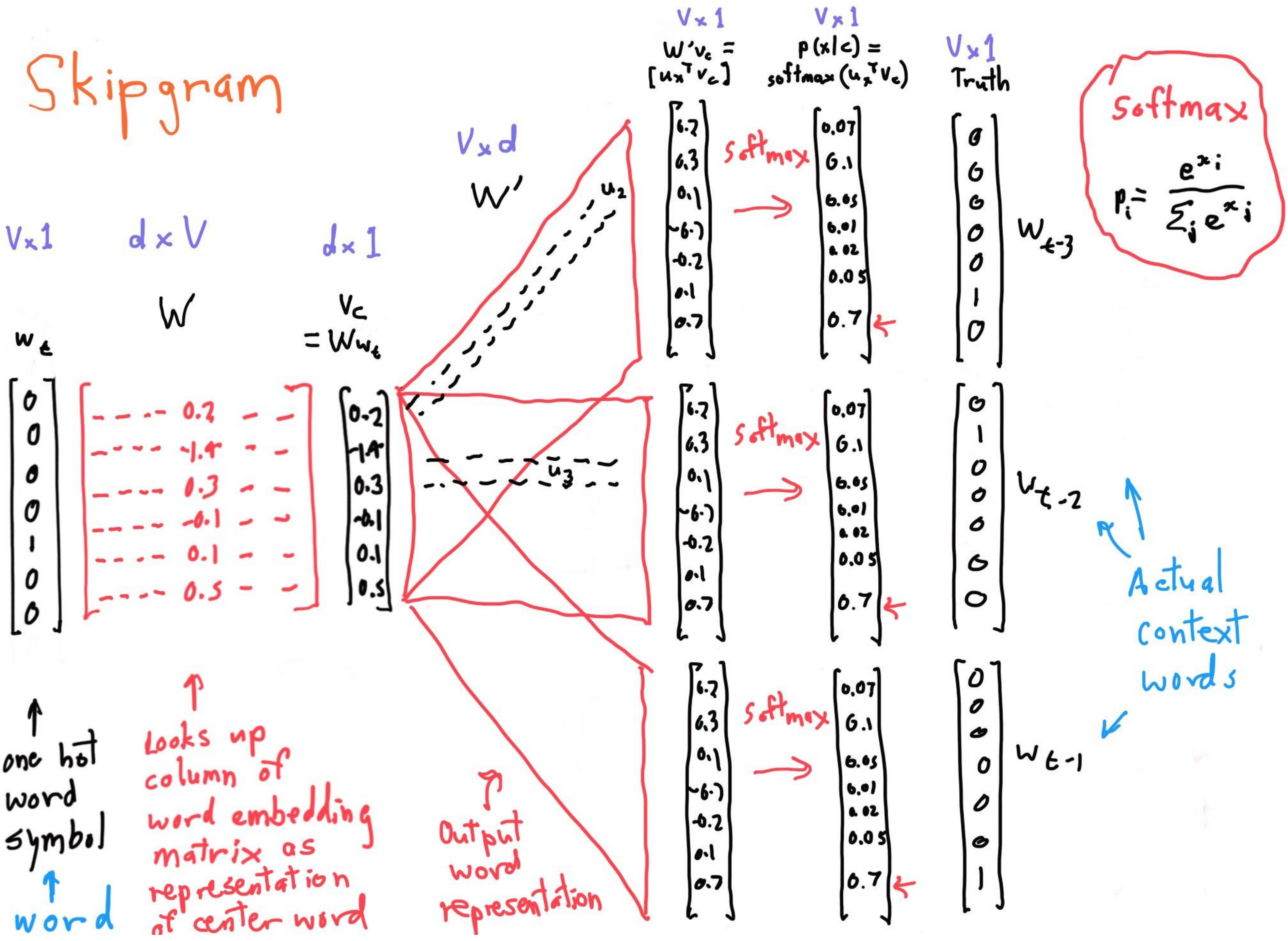
z = $W' v_c$

5. Turn the *score vector into probabilities*

$\hat{y}$ = *softmax(z)*

We want this to be close to 1 for the context words




Skipgram
V x 1
d x V
d x 1
V x d
W
W'
$v_c = W w_t$
w_t
u_2
u_3
V x 1
$W' v_c = [u_x^T v_c]$
V x 1
$p(x|c) = \text{softmax}(u_x^T v_c)$
V x 1
Truth
softmax
w_{t-3}
w_{t-2}
w_{t-1}
Softmax
$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
one hot word symbol
word
Looks up column of word embedding matrix as representation of center word
Output word representation
Actual context words

Εντυπωσιακά αποτελέσματα!

These representations are *very good* at encoding similarity and dimensions of similarity!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

Syntactically

– $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$

– Similarly for verb and adjective morphological forms

Semantically

– $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$

– $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

Improve language translation

bilingual embedding with chinese in green and english in yellow

By aligning the word embeddings for the two languages

End of lecture

Χρησιμοποιήθηκε υλικό από
- CS276: Information Retrieval and Web Search, Christopher Manning and Pandu Nayak, Lecture 14: Distributed Word Representations for Information Retrieval
- Jordan Boyd-Graber, UMD course Natural Language Processing,
- https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Μια περιγραφή του skipgram:
Chris McCormick
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
Δείτε και το
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/