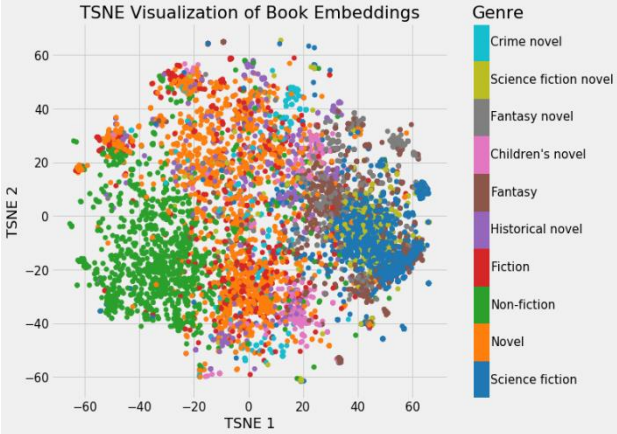
Word embeddings

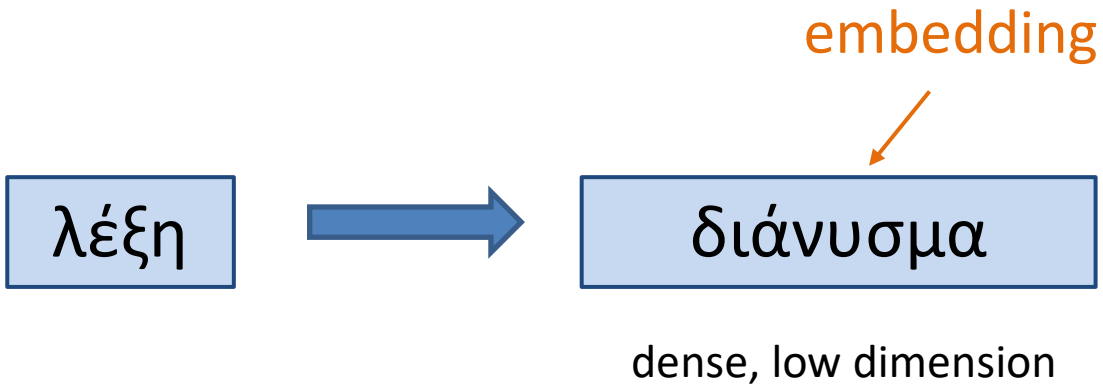
Τι θα δούμε σήμερα

- Διανυσματική αναπαράσταση λέξεων (word embeddings)



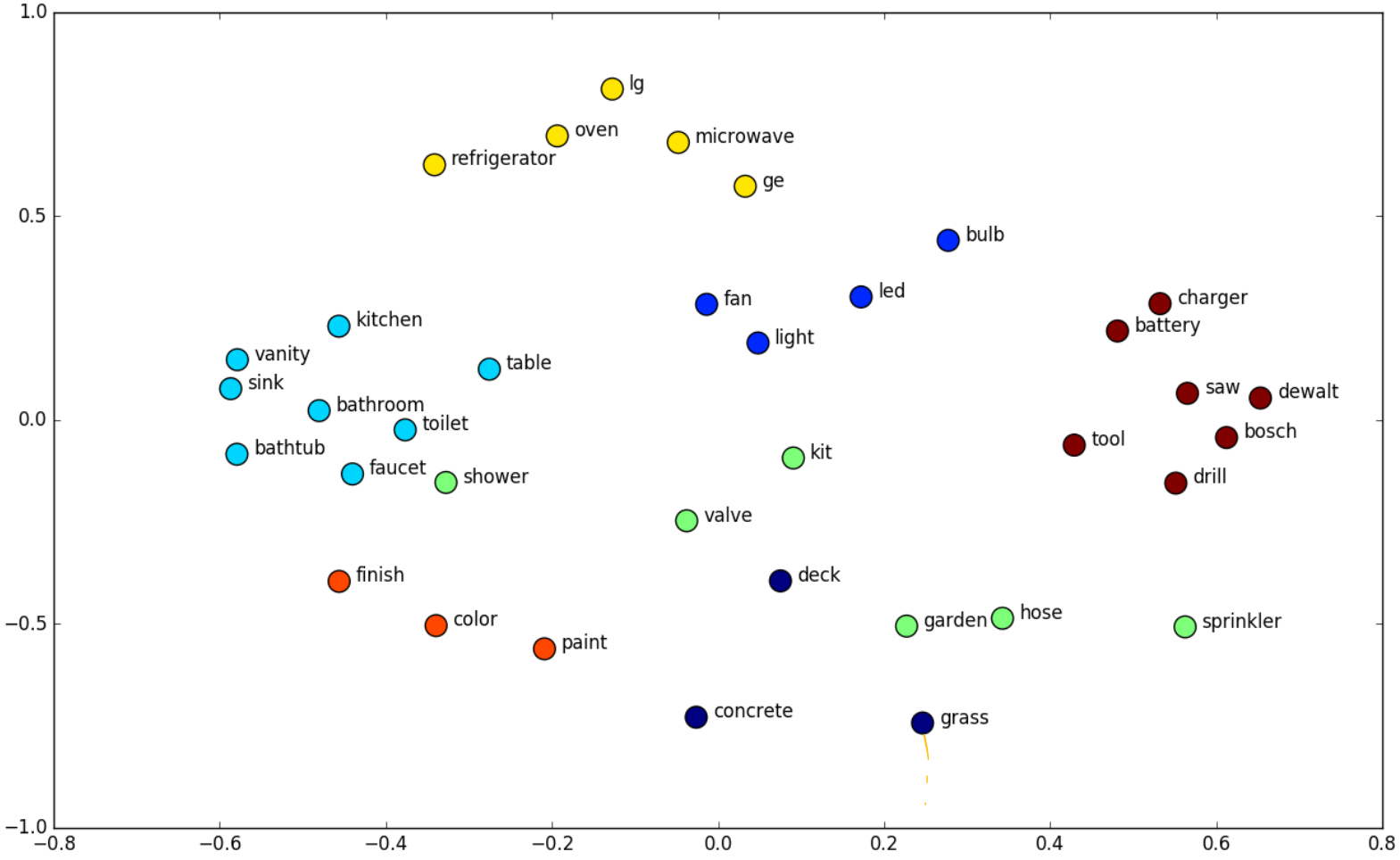
Books on Wikipedia colored by genre in two dimensions. Source: Koehrsen 2018

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



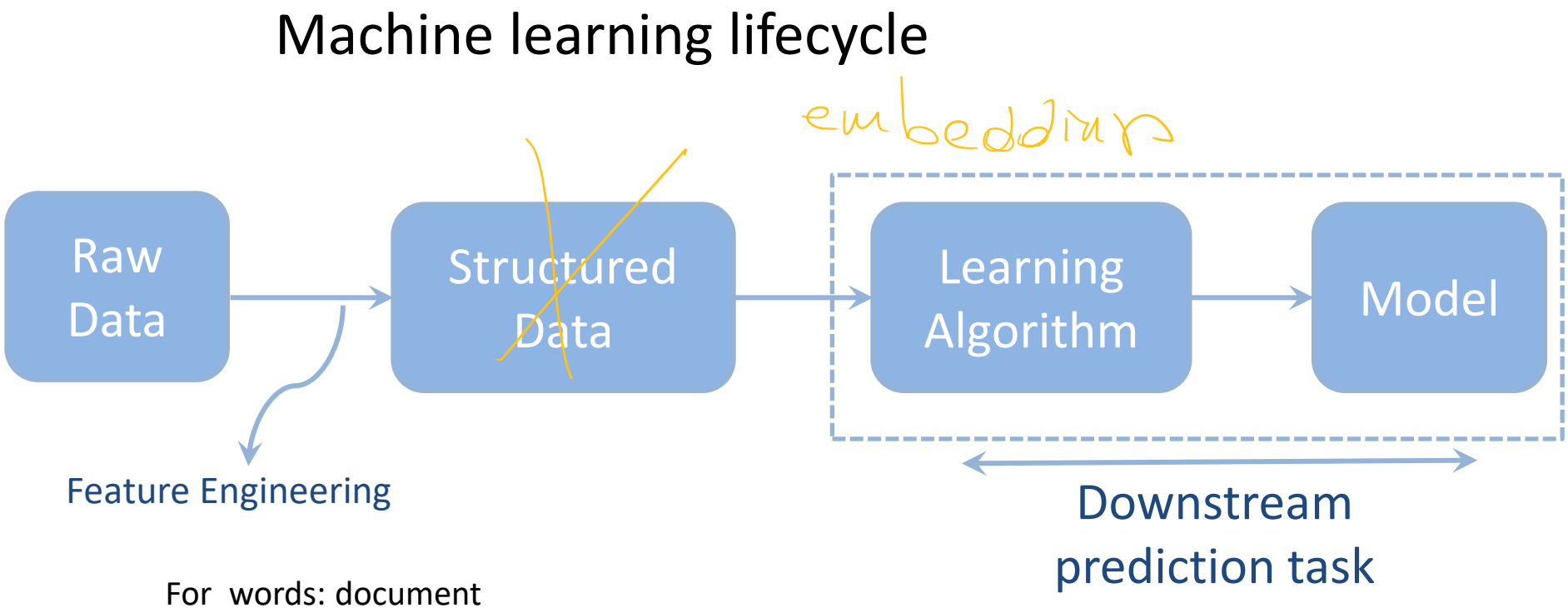
Στόχος: παρόμοιες λέξεις -> παρόμοια διανύσματα

Παράδειγμα: 2-διάστατα embeddings



http://suriyadeepan.github.io

Embeddings: why?



For words: document occurrences, k-grams. etc

For documents: length, words, etc

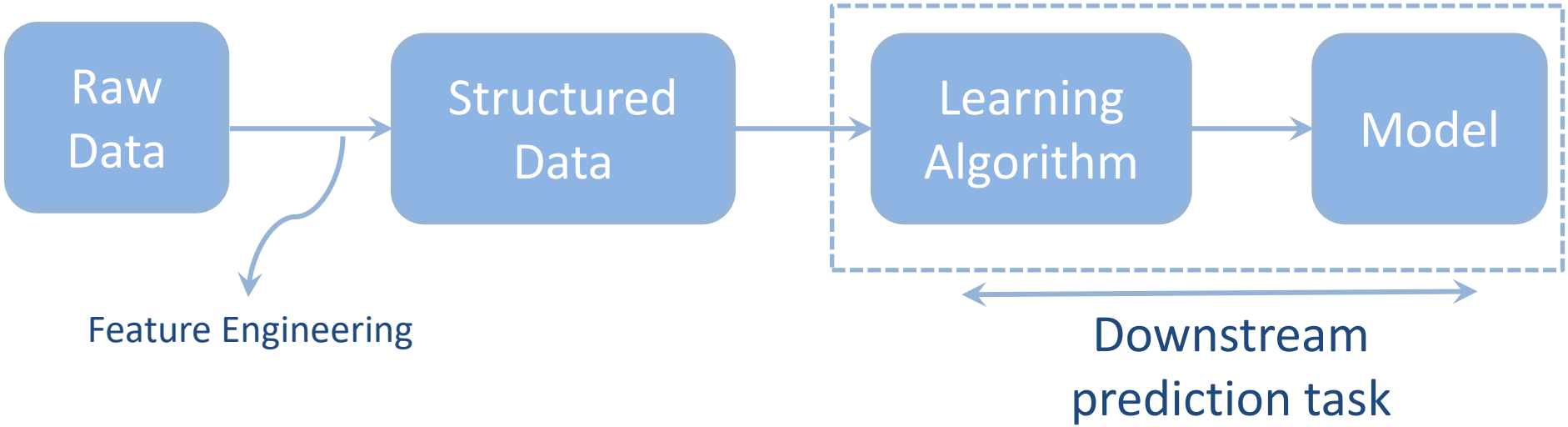
For graphs: degree, PageRank, motifs, degrees of neighbors, Pagerank of neighbors, etc

Classification
Learning to rank
Clustering

Query answering, machine translation

Embeddings: why?

Machine learning lifecycle



For words: document occurrences, k-grams. etc

For documents: length, words, etc

For graphs: degree, PageRank, motifs, degrees of neighbors, Pagerank of neighbors, etc

Automatically learn the features (embeddings)

Πως θα πάρουμε αυτά τα διανύσματα;

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

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

- Κάθε λέξη ένα διαφορετικό σύμβολο
- Πολλές διαστάσεις (|V| όσες οι λέξεις)
- Καμία πληροφορία για ομοιότητα

Term-Document co-occurrence matrix

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

- Κατασκευάζουμε ένα |V| x M πίνακα με τις εμφανίσεις των λέξεων στα έγγραφα
- Αναπαριστούμε κάθε λέξη με ένα $\mathrm{R}^{|\mathrm{M}|x1}$

Παράδειγμα:

d1: a b c
d2: a d a b
d3: a c d e c a f
d4: b e a b
d5: a b d c a

|V| = 6, |M| =5

	d1	**d2**	**d3**	**d4**	**d5**
a	1	1	1	1	1
b	1	1	0	1	1
c	1	0	1	0	1
d	0	1	1	0	1
e	0	0	1	1	0
f	0	0	1	0	0

Word vector for c

Term-Document co-occurrence matrix

Μπορούμε αντί για 0-1 να έχουμε το tf ή και το tf-idf βάρος

Παράδειγμα:

d1: a b c
d2: a d a b
d3: a c d e c a f
d4: b e a b
d5: a b d c a

	d1	**d2**	**d3**	**d4**	**d5**
a	1	2	2	1	2
b	1	1	0	2	1
c	1	0	2	0	1
d	0	1	1	0	1
e	0	0	1	1	0
f	0	0	1	0	0

Word vector for c

- Πολλές διαστάσεις
- Πρόβλημα κλιμάκωσης με τον αριθμό των εγγράφων

Window-based co-occurrence matrix

- Κατασκευάζουμε ένα $|V|x|V|$ affinity-matrix για τις λέξεις: για δύο λέξεις, μετράμε τον αριθμό των φορών που αυτές οι δύο λέξεις εμφανίζονται μαζί σε έγγραφα
- Συγκεκριμένα, μετράμε τον αριθμό των φορών που κάθε λέξη εμφανίζεται μέσα σε ένα *παράθυρο* συγκεκριμένου μεγέθους γύρω από τη λέξη ενδιαφέροντος

Παράδειγμα:

d1: a b c
d2: a d a b
d3: a c d e c a f
d4: b e a b
d5: a b d c a

W = 1 (σε απόσταση 1)

	a	b	c	d	e	f
a	0	4	3	1	1	1
b	4	0	1	1	1	0
c	3	1	0	2	1	0
d	1	1	2	0	1	0
e	1	1	1	1	0	0
f	1	0	0	0	0	0

Window-based co-occurrence matrix

- Κατασκευάζουμε ένα $|V|x|V|$ affinity-matrix για τις λέξεις: μετράμε τον αριθμό των φορών που δυο λέξεις εμφανίζονται μέσα σε ένα *παράθυρο* συγκεκριμένου μεγέθους

Λέξεις όπως apple, orange, mango, κλπ μαζί με λέξεις όπως eat, grow, cultivate, slice, κλπ και το ανάποδο

Παράδειγμα:

d1: I enjoy flying.
d2: I like NLP.
d3: I like deep learning.

W = 1

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

- Πολλές διαστάσεις

Θα μπορούσαμε να χρησιμοποιήσουμε μια τεχνική για να μειώσουμε τις διαστάσεις (dimensionality reduction) (πχ PCA analysis)

Singular Value Decomposition

From dimension d to dimension r

$$A = U \quad \Sigma \quad V^T = [\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_n] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}$$

$$[n \times n] \qquad [n \times n] \qquad [\times n]$$

- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$: singular values (square roots of eigenvals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$: left singular vectors (eigenvectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$: right singular vectors (eigenvectors of A^TA)

- Cut the singular values at some index r (get the largest r such values)
- Get the first r columns of U to get the r-dimensional vectors

Singular Value Decomposition

Ar best approximation of A
(Frobernius norm)

$$A = U \quad \Sigma \quad V^T = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_r \end{bmatrix}$$

$$[n \times r]\,[r \times r]\,[r \times n]$$

- **r** : rank of matrix A
- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$: singular values (square roots of eigenvals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_r$: left singular vectors (eigenvectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_r$: right singular vectors (eigenvectors of A^TA)

$$A_r = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$$

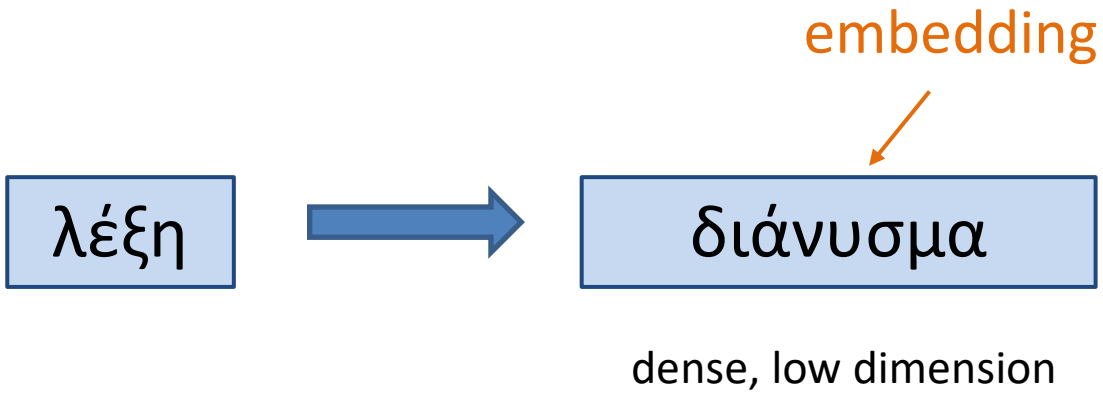
Αλλά

- Δύσκολο να ενημερώσουμε, πχ, αλλάζουν οι διαστάσεις συχνά
- Αραιός πίνακας
- Πολύ μεγάλες διαστάσεις

Θα δούμε μια τεχνική που βασίζεται σε επαναληπτικές μεθόδους

word2vec

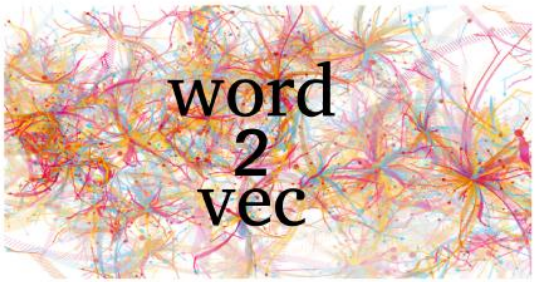
Στόχος: Διανυσματική *αναπαράσταση* (representation) λέξεων – *κατανεμημένη (distributed) αναπαράσταση*



Στόχος: παρόμοιες λέξεις -> παρόμοια διανύσματα

word2vec

Είσοδος κείμενα, προτάσεις



feed in text

Text

WIKIPEDIA

Google news

...

wait a few hours

d

|V| words

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

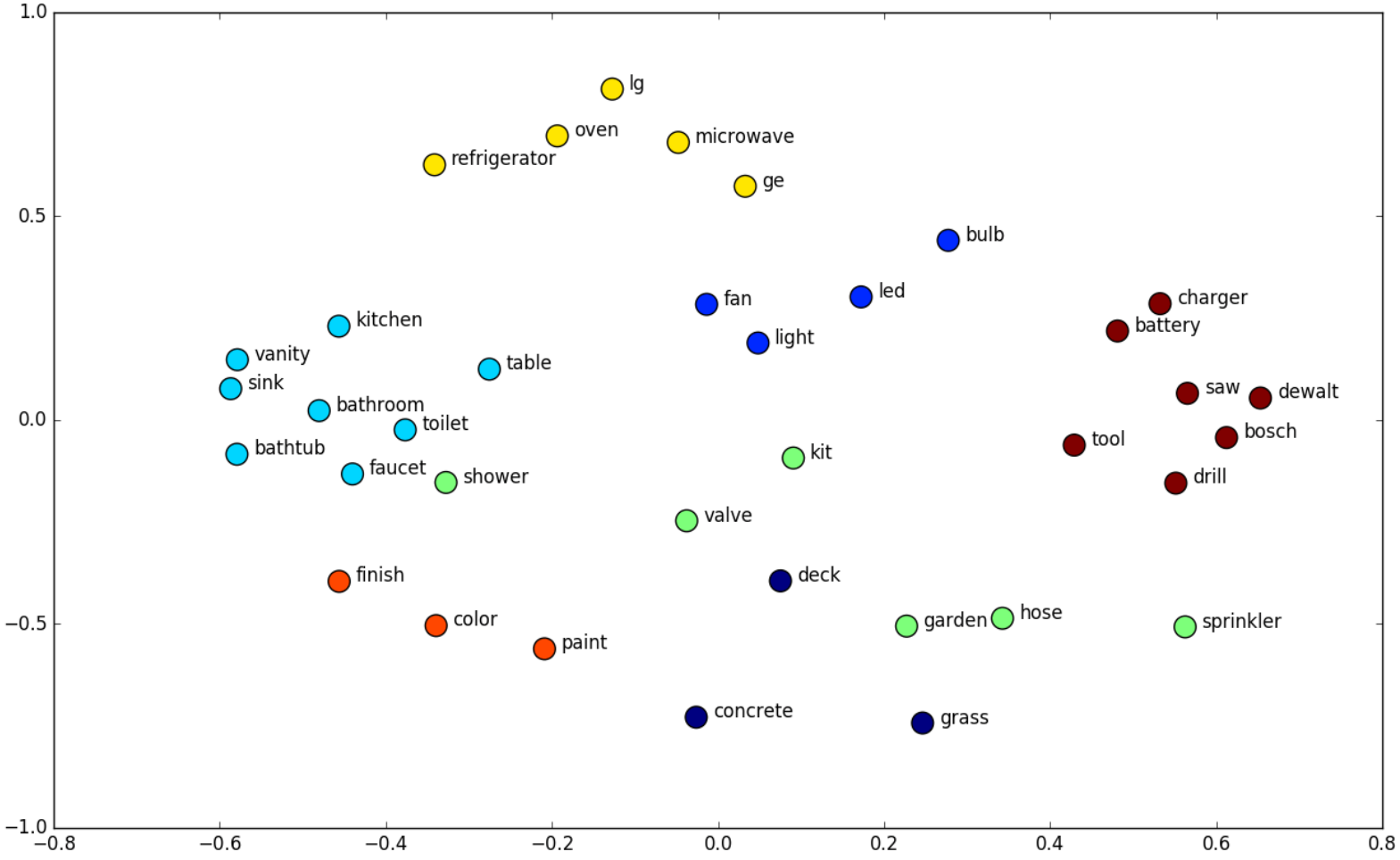
chair = (0.8,0.9,-0.76,0.29,0.52 ...)

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

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

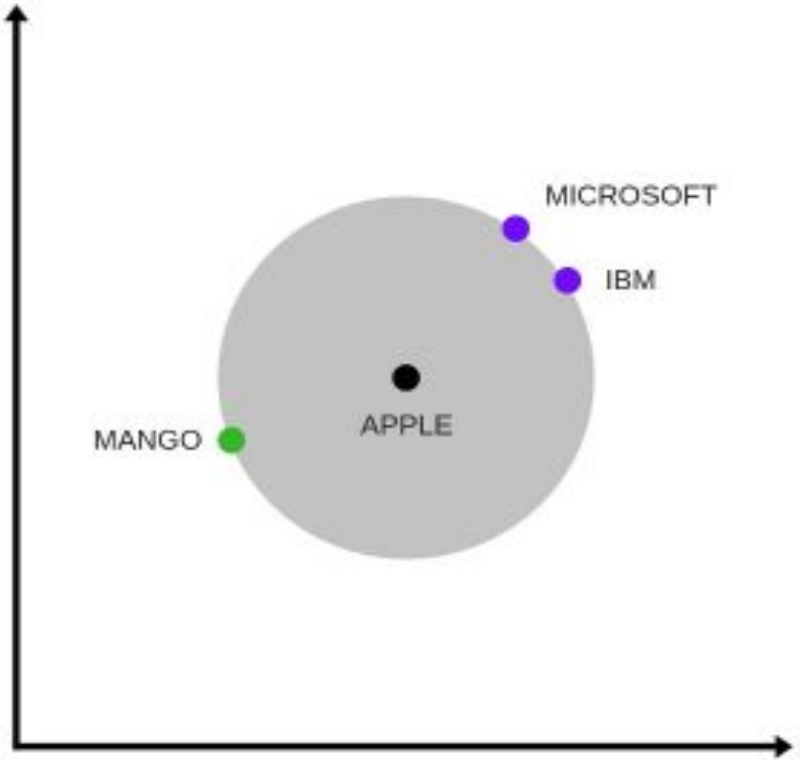
Παράδειγμα: 2-διάστατα embeddings



t-SNE plot

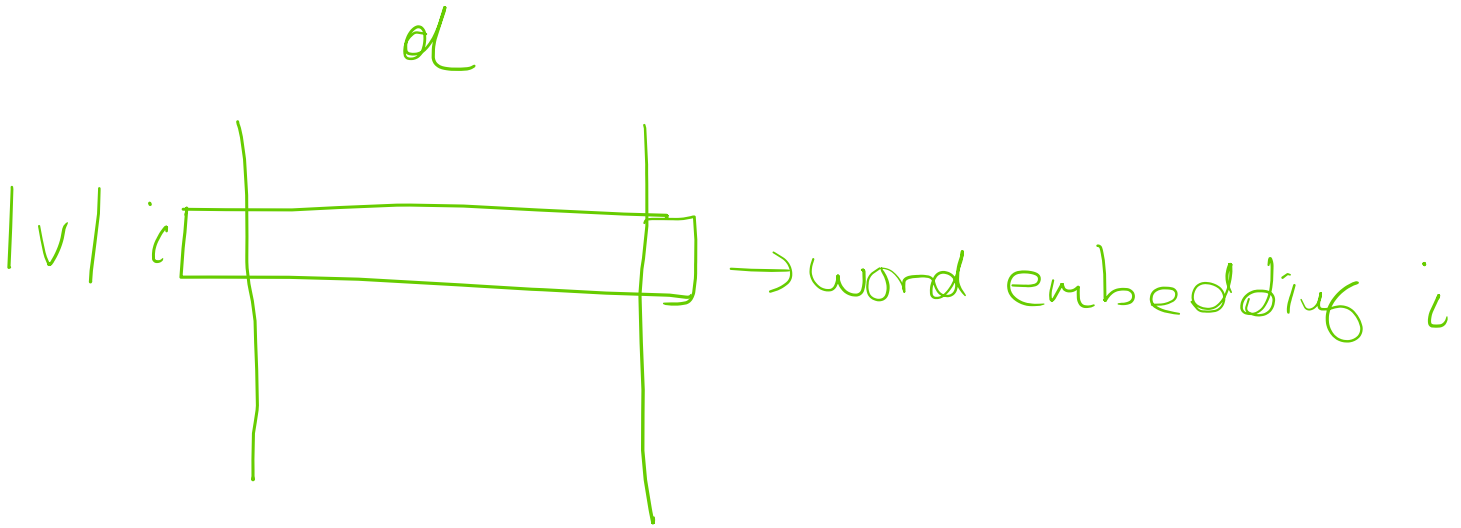
http://suriyadeepan.github.io

Apple: φρούτο και εταιρεία



https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Παράδειγμα: Έστω ότι έχουμε τον πίνακα W, πως θα βρούμε την ποιο όμοια λέξη με το “dog”?



διαυσμα για το dog $\vec{dog}$

Για κάθε λέξη w, $\vec{w}$

similarity($\vec{w}$, $\vec{d}$)

cosine($\vec{w}$, $\vec{d}$)

$\vec{w} \cdot \vec{d}$

Ομοιότητα/απόσταση

1 2 3 4 5 6

||(1, 2, 3||

- Similarity is calculated using *cosine similarity*:

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

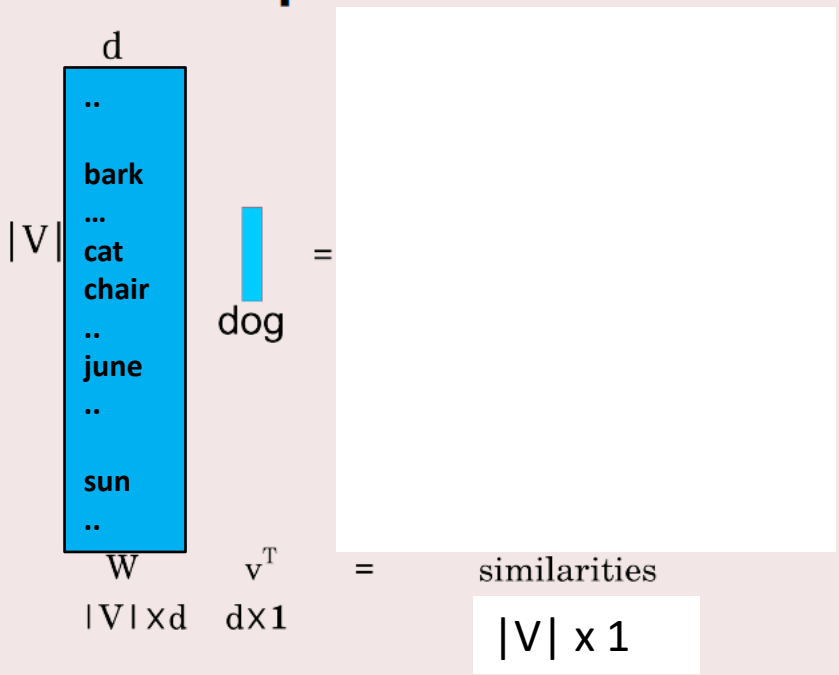
- For normalized vectors ($||x|| = 1$), this is equivalent to a dot product:

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

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

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**: $W \cdot \vec{v}^{\top}$



- 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

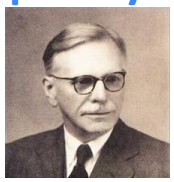
Lemmatization
Stemming

Λέξεις σημασιολογικά κοντινές

Πως θα πάρουμε αυτόν τον πίνακα;

Basic Idea

- You can get a lot of value by representing a word by means of its neighbors (distributional semantics)
- "You shall know a word by the company it keeps"



(J. R. Firth 1957: 11)

- One of the most successful ideas of modern statistical NLP
- Context: words that appear in a fixed length window around the word

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Use the many contexts of w to represent a word

Basic idea

Center word
Context words

.. problems turning into banking crises as ..

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*

Βασική ιδέα

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

Learning: παραδείγματα κειμένου και προσπαθούμε να «μάθουμε» αυτά τα διανύσματα (βάρη) – τους 2 πίνακες

Training examples – fix the matrices to work for them

Word2Vec

Predict between every word and its context words

Two **algorithms**

1. Skip-grams (SG)

   Predict context words given the center word

2. Continuous Bag of Words (CBOW)

   Predict center word from a bag-of-words context

Position independent (do not account for distance from center)

Two **training methods**

1. Hierarchical softmax
2. Negative sampling

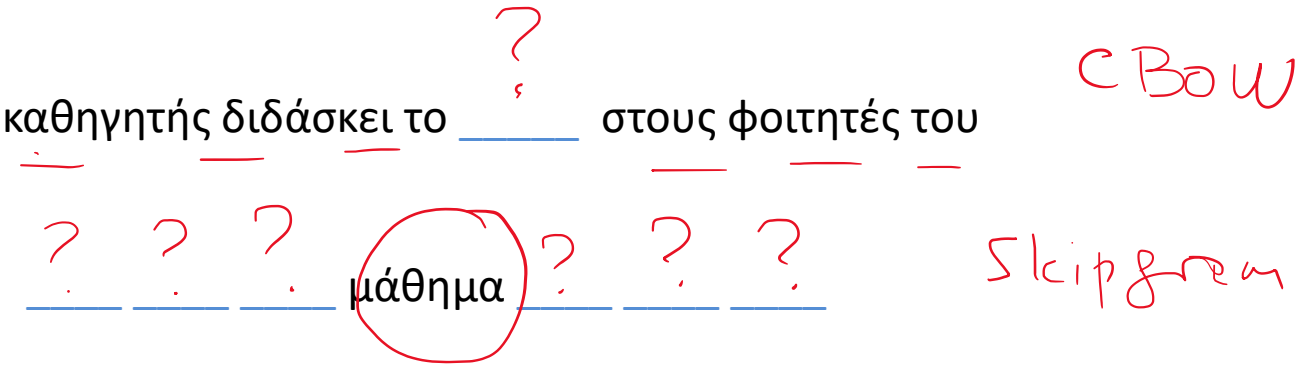
Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality.* NIPS 2013: 3111-3119

Βασική ιδέα

Μία λέξη προσδιορίζεται από τις συμφραζόμενες της λέξεις (context)

Ο καθηγητής διδάσκει το μάθημα στους φοιτητές του στην αίθουσα.

Παράθυρο (window) = 3



Συμβολισμοί

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

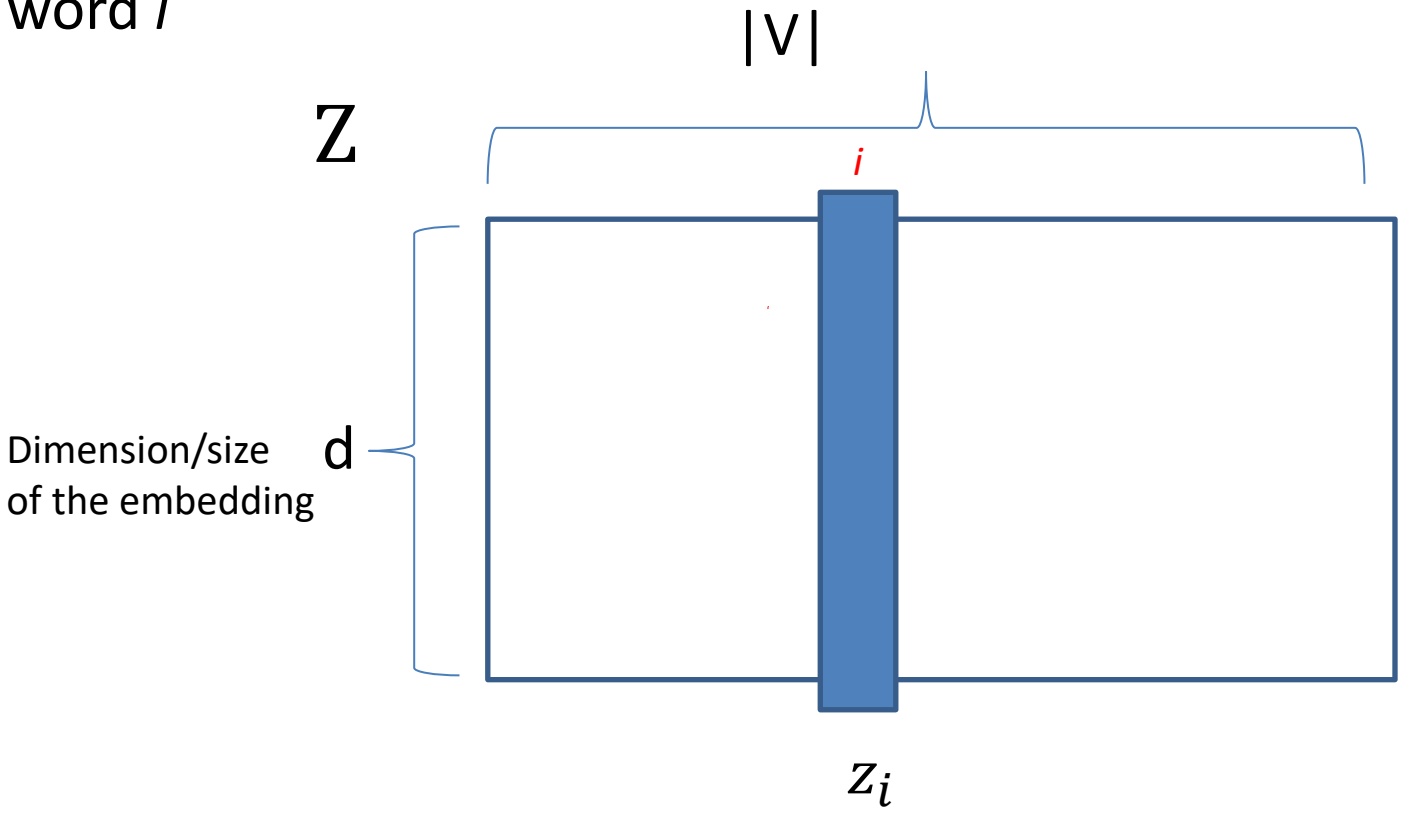
$$w^{aardvark} = \begin{bmatrix} \mathbf{1} \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{a} = \begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad w^{at} = \begin{bmatrix} 1 \\ 0 \\ \mathbf{0} \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad \cdots \quad w^{zerba} = \begin{bmatrix}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

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

Look up

Each word is assigned *a d-dimensional vector*

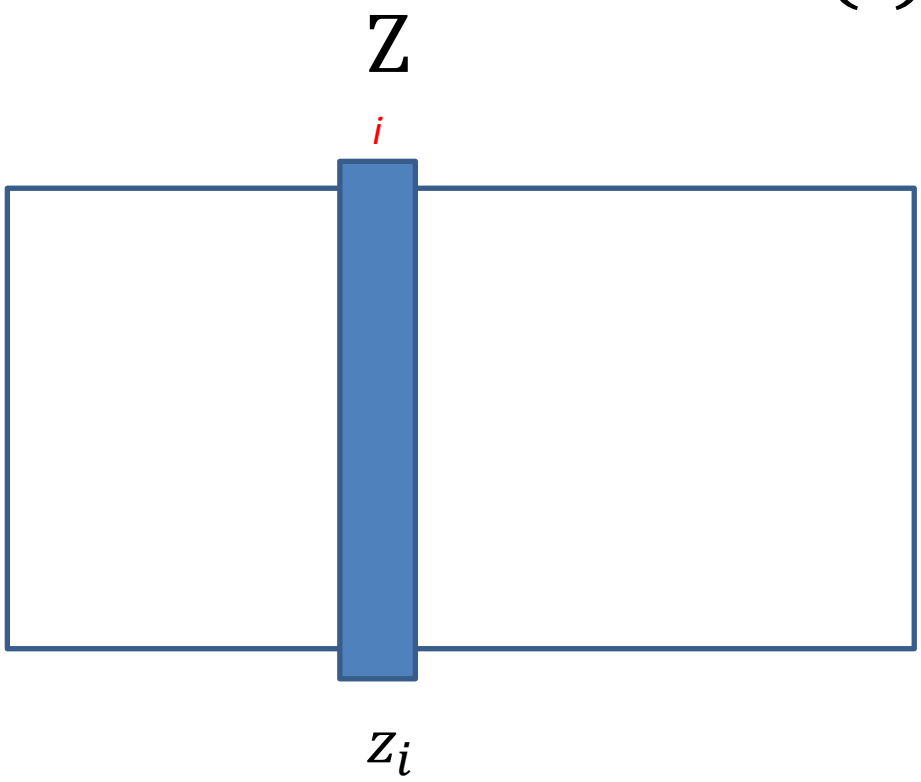
Learn embedding matrix Z: each column *i* is the embedding z_i of word *i*



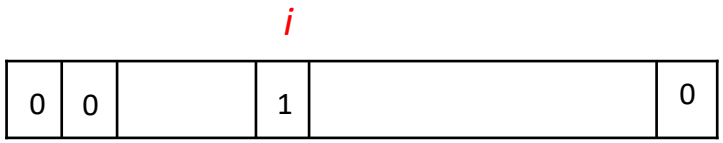
Look up

Encoder is an embedding lookup

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



$One\ hot\ vector\ I_i$



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

CBOW

CBOW

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

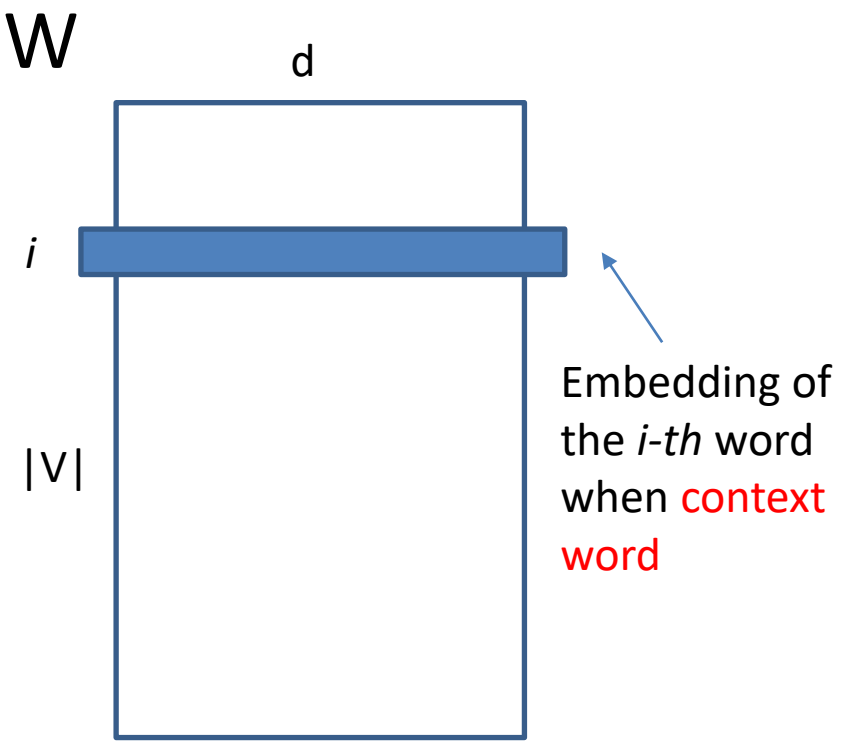
Use a window of context words to predict the center word

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

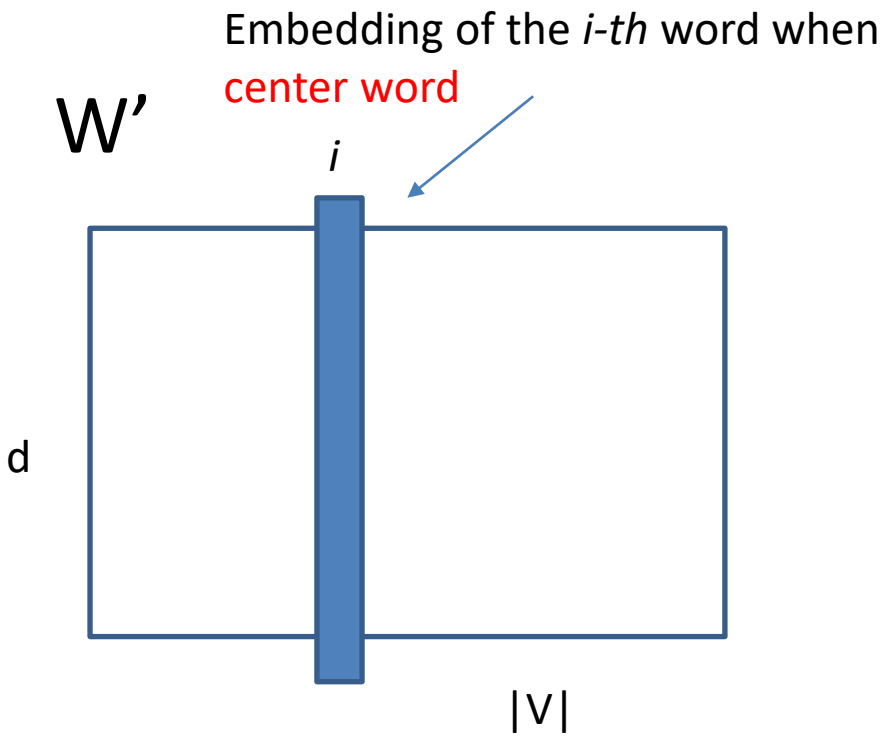
CBOW

Use a window of context words to predict the center word

Learns two matrices (two embeddings per word, one when context, one when center)



|V| x d context embeddings when *input*



d x |V| center embeddings when *output*

CBOW

Use a window of context words to predict the center word

Intuition

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

- For similarity, we will use cosine (dot product)
- We will take the average of the W-embeddings of the context word

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

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

2. *GET THE EMBEDDINGS of the context words*

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

3. *TAKE THE SUM* these vectors (average)

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

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

z = W' $\hat{v}$

5. Turn the *score vector to probabilities*

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

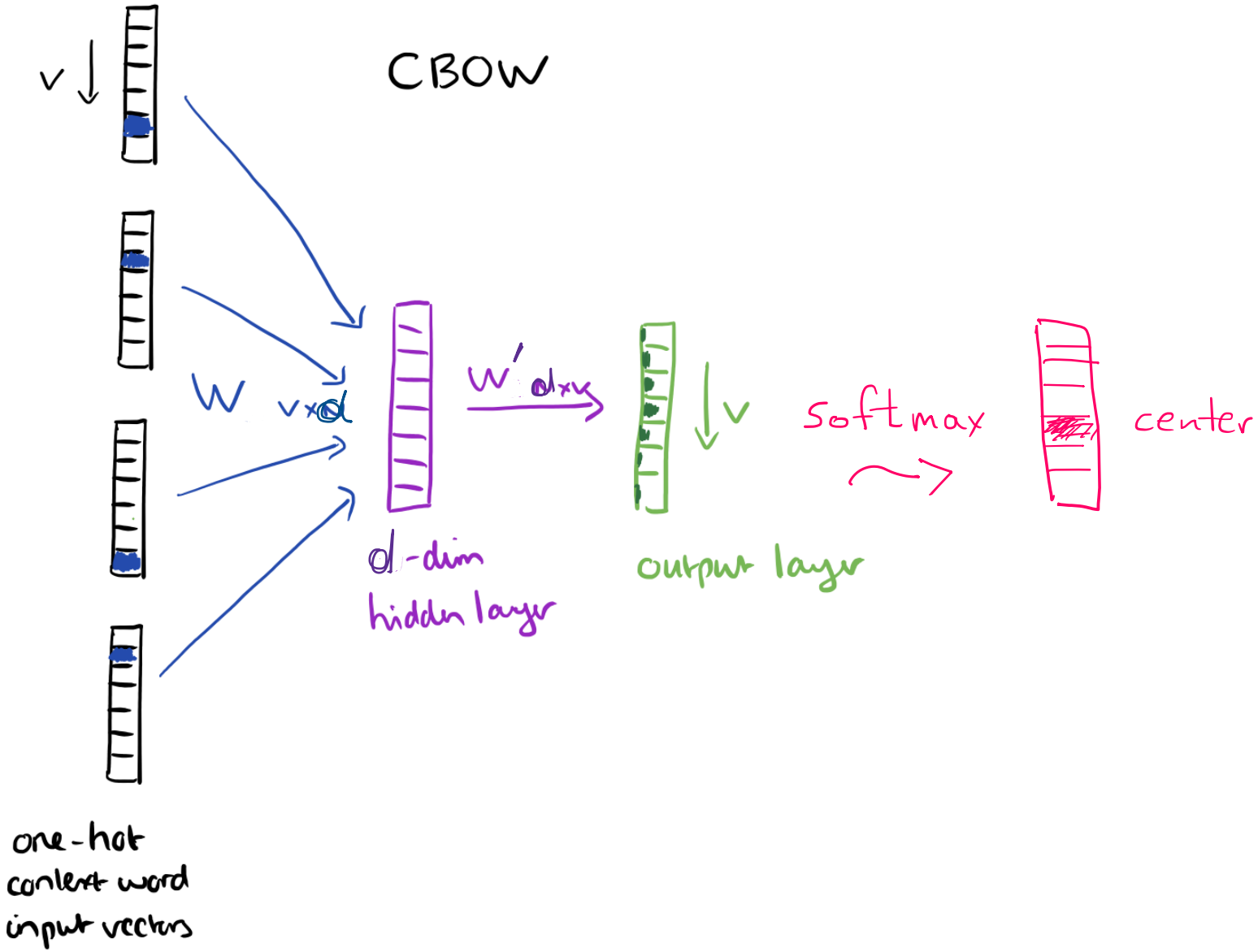
We want this to be close to 1 for the center word

Απεικόνιση κατανομών οποιοδήποτε τιμών σε κατανομές πιθανοτήτων

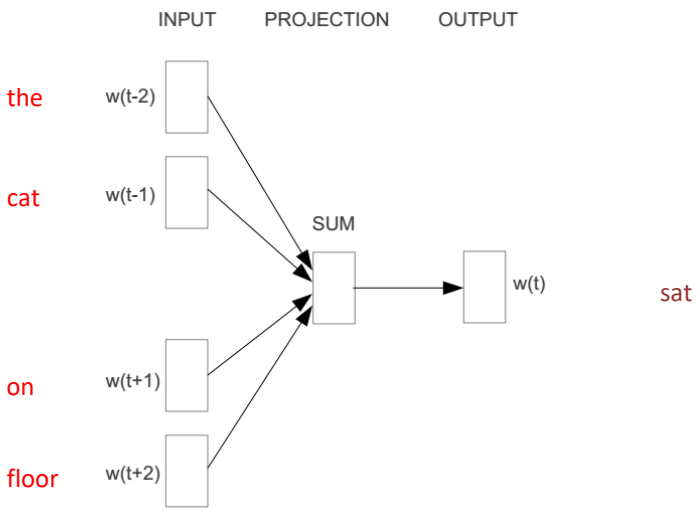
$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

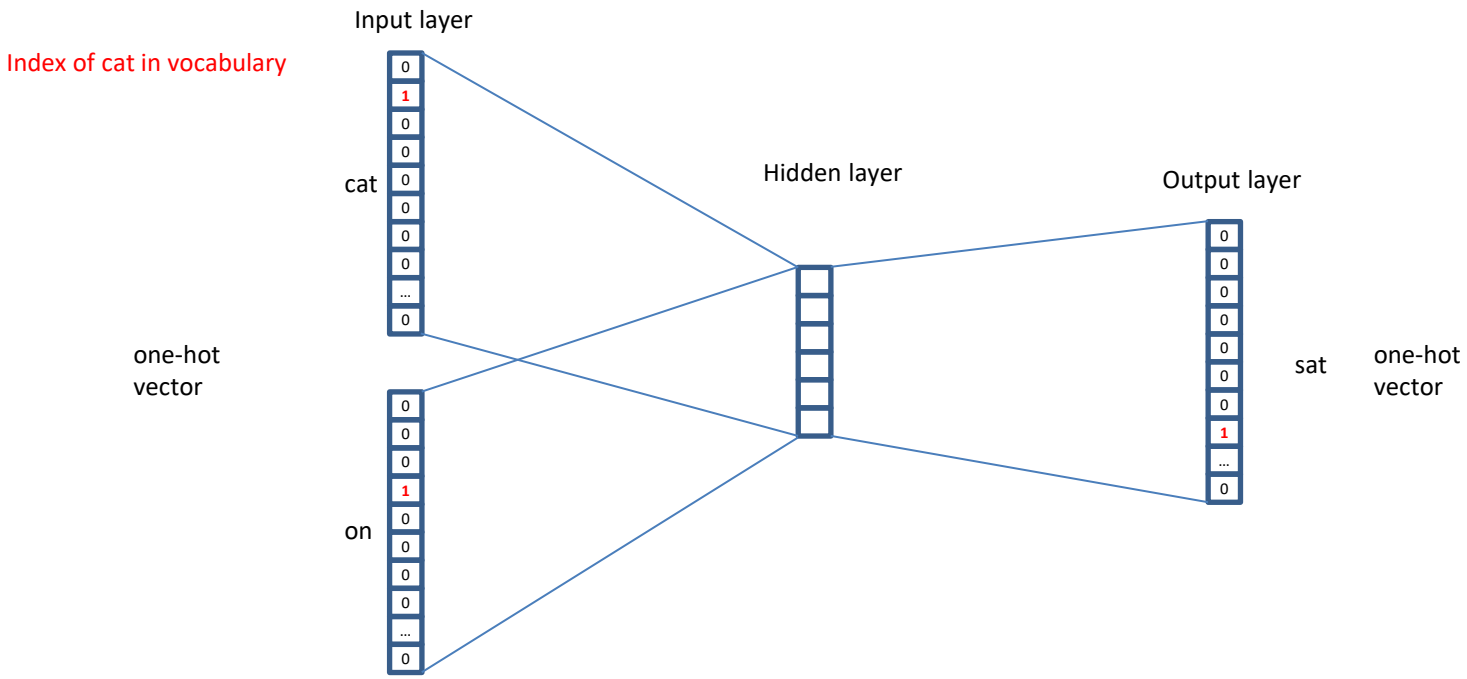
Εκθετικό για να είναι θετικό
Ενισχύει την πιθανότητα της μεγαλύτερης τιμής (max)
Κάποια πιθανότητα και στις μικρότερες τιμές (soft)

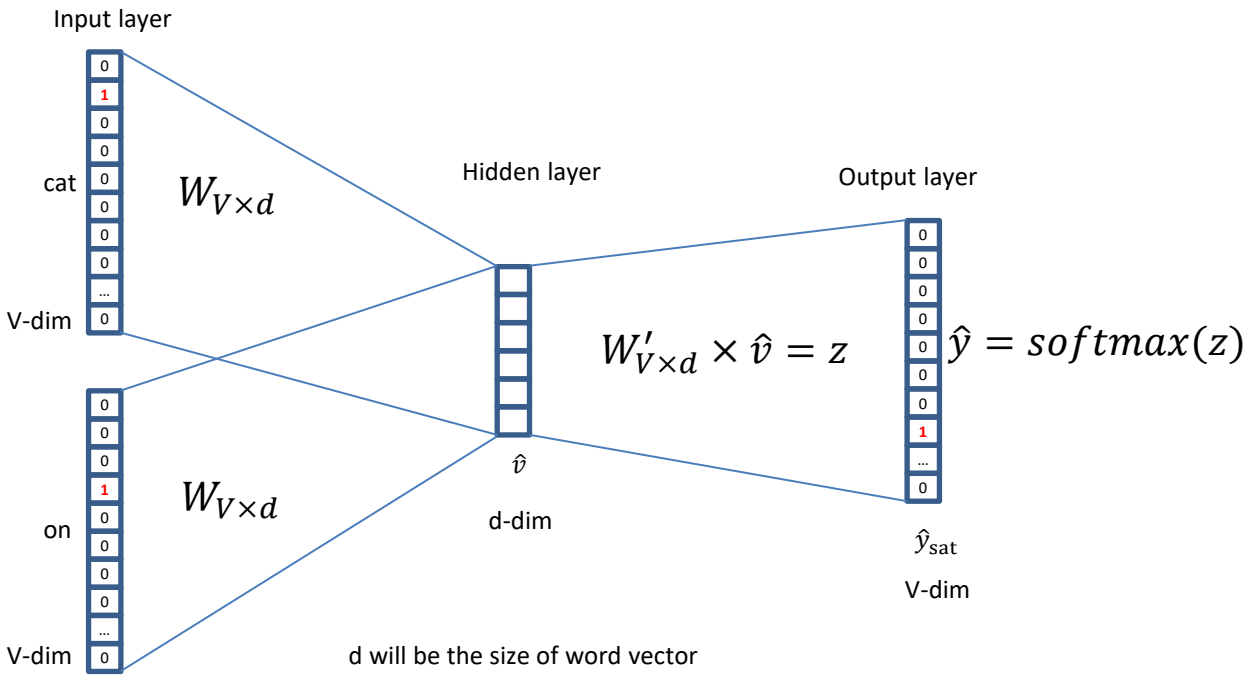
m=2

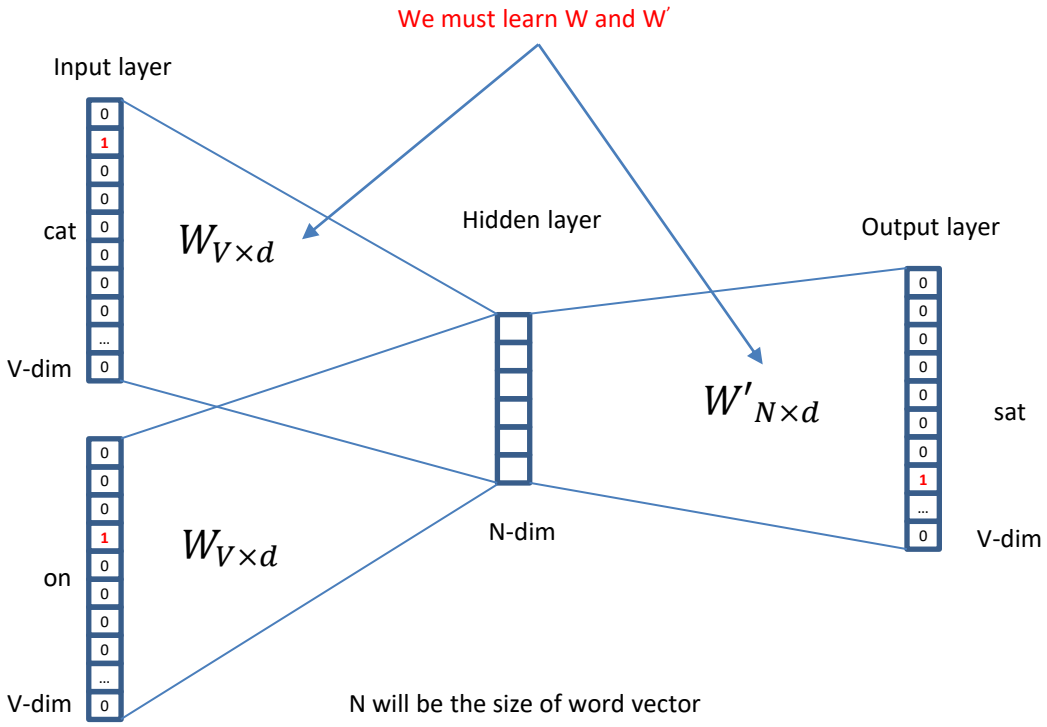

CBOW
V
W
V×d
W'
d×V
V
softmax
center
d-dim
hidden layer
output layer
one-hot
context word
input vectors

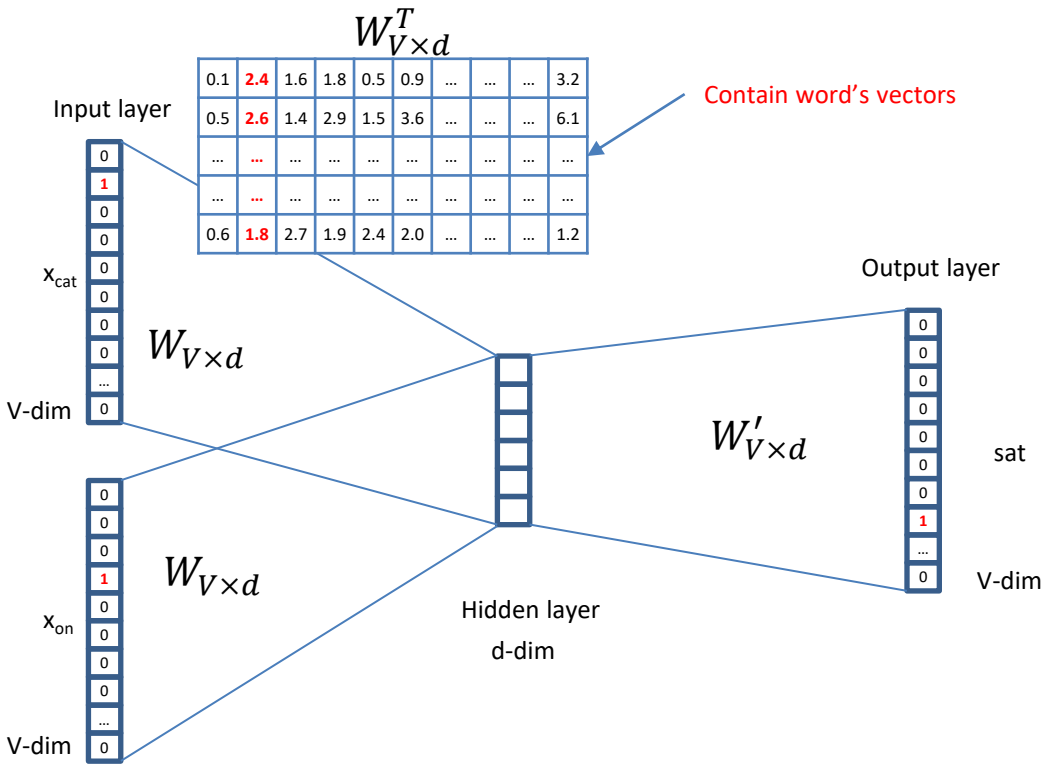
- E.g. “The cat sat on floor”
  - Window size = 2



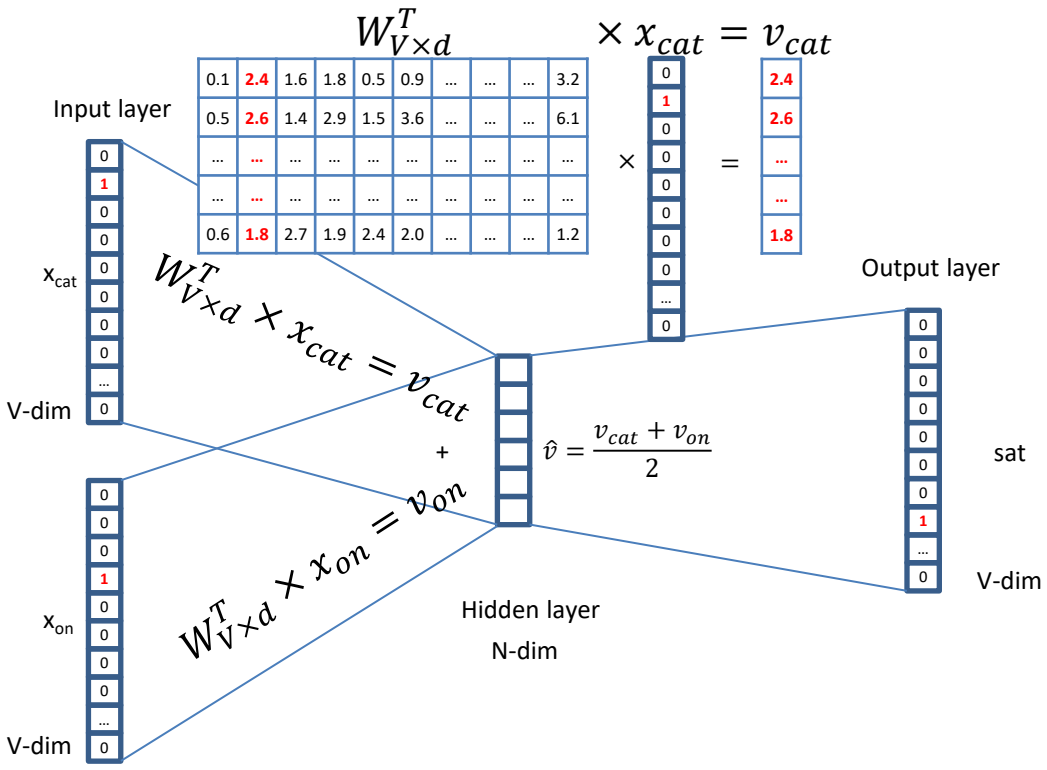

Index of cat in vocabulary
Input layer
cat
on
one-hot vector
Hidden layer
Output layer
sat
one-hot vector

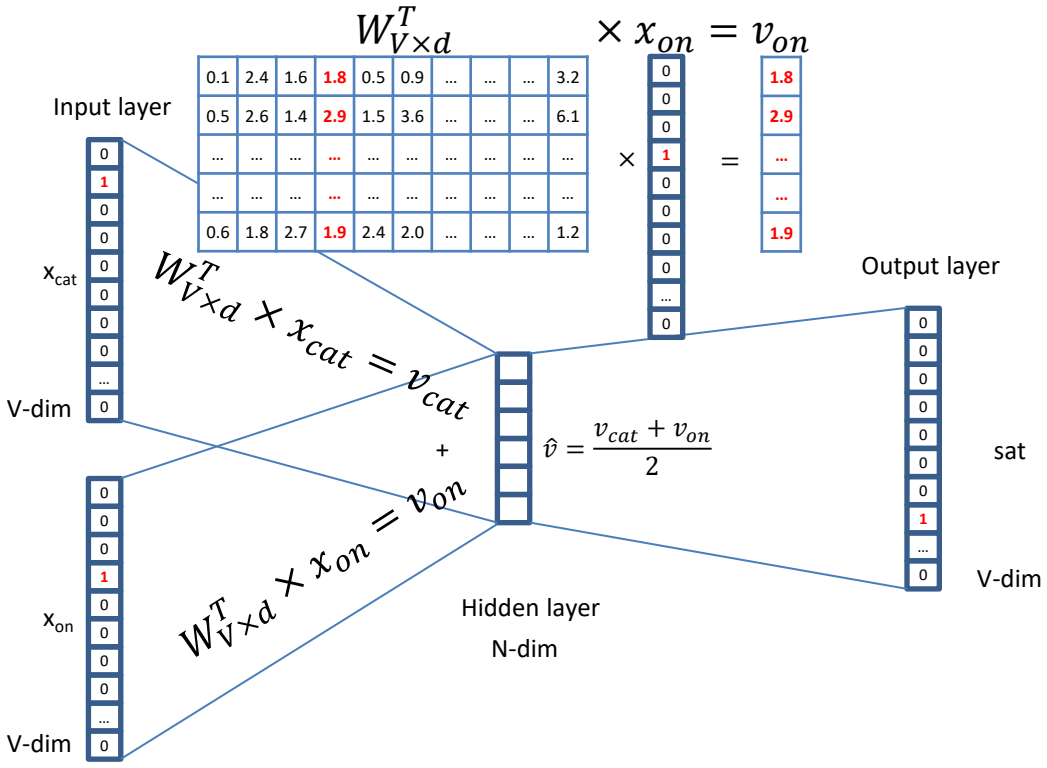

Input layer
cat
V-dim
on
V-dim
$W_{V\times d}$
$W_{V\times d}$
Hidden layer
$\hat{v}$
d-dim
$W'_{V\times d} \times \hat{v} = z$
Output layer
$\hat{y} = softmax(z)$
$\hat{y}_{sat}$
V-dim
d will be the size of word vector

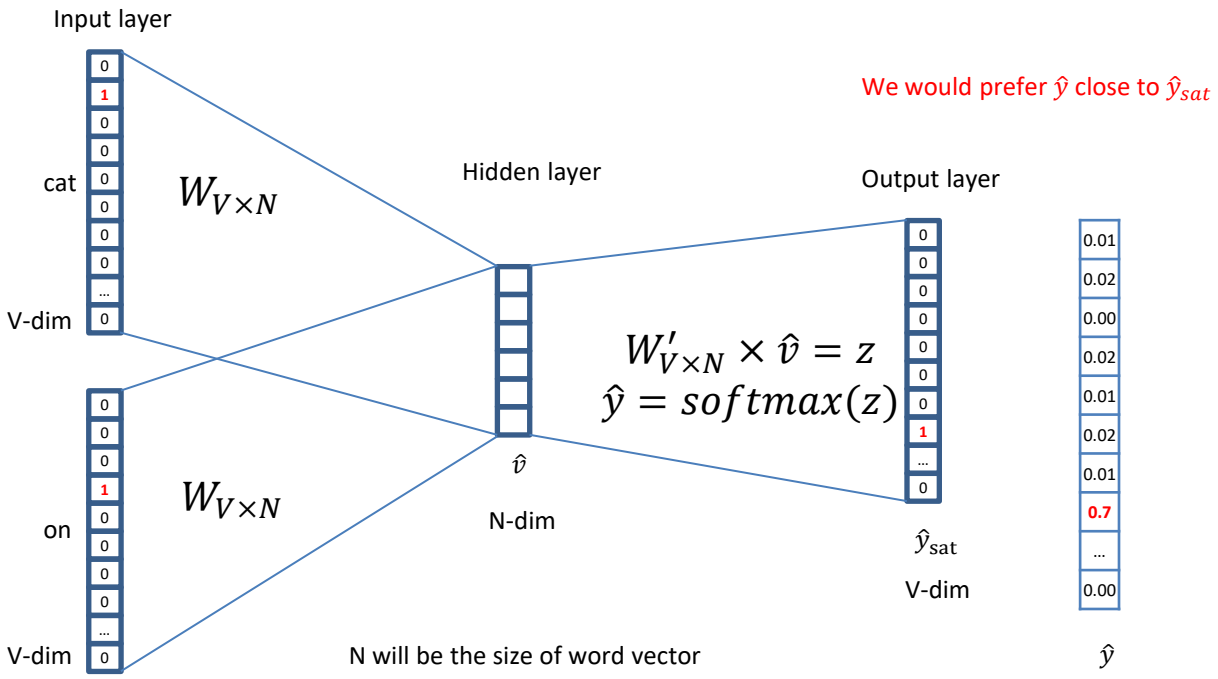

We must learn W and W′
Input layer
cat
V-dim
on
V-dim
$W_{V\times d}$
$W_{V\times d}$
Hidden layer
N-dim
$W'_{N\times d}$
Output layer
sat
V-dim
N will be the size of word vector



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


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


$W^T_{V\times d}$
$\times x_{on} = v_{on}$
Input layer
x_{cat}
V-dim
x_{on}
V-dim
$W^T_{V\times d} \times x_{cat} = v_{cat}$
+
$W^T_{V\times d} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
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
$W'_{V\times N} \times \hat{v} = z$
$\hat{y} = softmax(z)$
Output layer
$\hat{y}_{\text{sat}}$
V-dim
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
$\hat{y}$
N will be the size of word vector

Αύριο

Παρασκευή 21 Μάϊου, 6μμ-8μμ

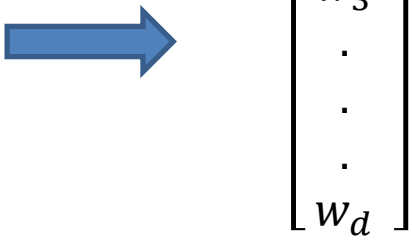
zoom link
https://us02web.zoom.us/j/84437398146?pwd=cCtJemtCSDVTUEhOR0lCOVh4NFJqQT09

Meeting ID: 844 3739 8146
Passcode: cse-w-2021

Word2vec (recap)

Word embedding (distributed representation)

term



$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_d \end{bmatrix}$$

d-dimensional vector

|V| embeddings

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
window

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*

Βασική ιδέα

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

term

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_d \end{bmatrix}$$

center

$$\begin{bmatrix} w'_1 \\ w'_2 \\ w'_3 \\ . \\ . \\ . \\ w'_d \end{bmatrix}$$

context

Βασική ιδέα

Learning:
παραδείγματα κειμένου (training examples) και προσπαθούμε να «μάθουμε» αυτά τα διανύσματα (βάρη) – τους 2 πίνακες ώστε να δουλεύουν για αυτά

Word2Vec

Predict between every word and its context words

Two **algorithms**

1. Skip-grams (SG)

   Predict context words given the center word

2. Continuous Bag of Words (CBOW)

   Predict center word from a bag-of-words context

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

Μία λέξη προσδιορίζεται από τις συμφραζόμενες της λέξεις (context)

Ο καθηγητής διδάσκει το μάθημα στους φοιτητές του στην αίθουσα.

Παράθυρο (window) = 3

center

?

CBOW

καθηγητής διδάσκει το _____ στους φοιτητές του

? ? ? μάθημα ? ? ?

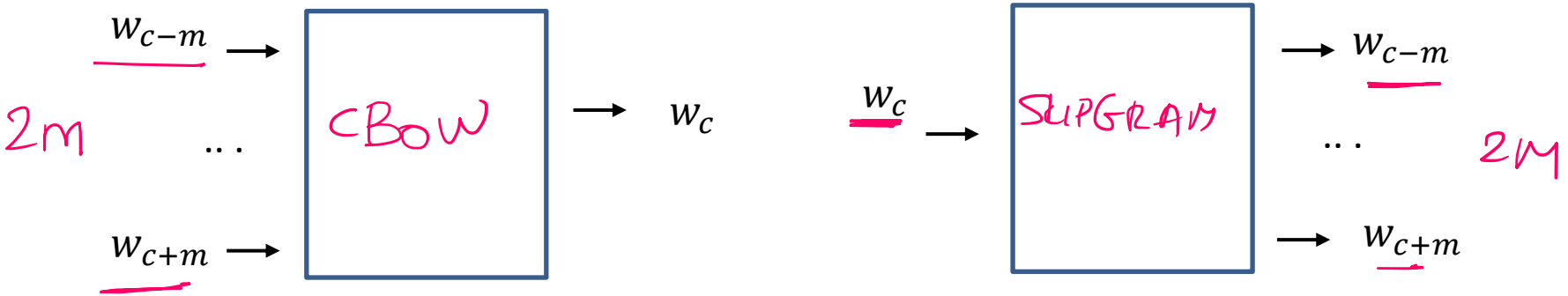
SKIPGRAM

Βασική ιδέα

m c m

2m

Learning:
παραδείγματα κειμένου (training examples) και προσπαθούμε να «μάθουμε» αυτά τα διανύσματα (βάρη) – τους 2 πίνακες ώστε να δουλεύουν για αυτά



One-hot vectors

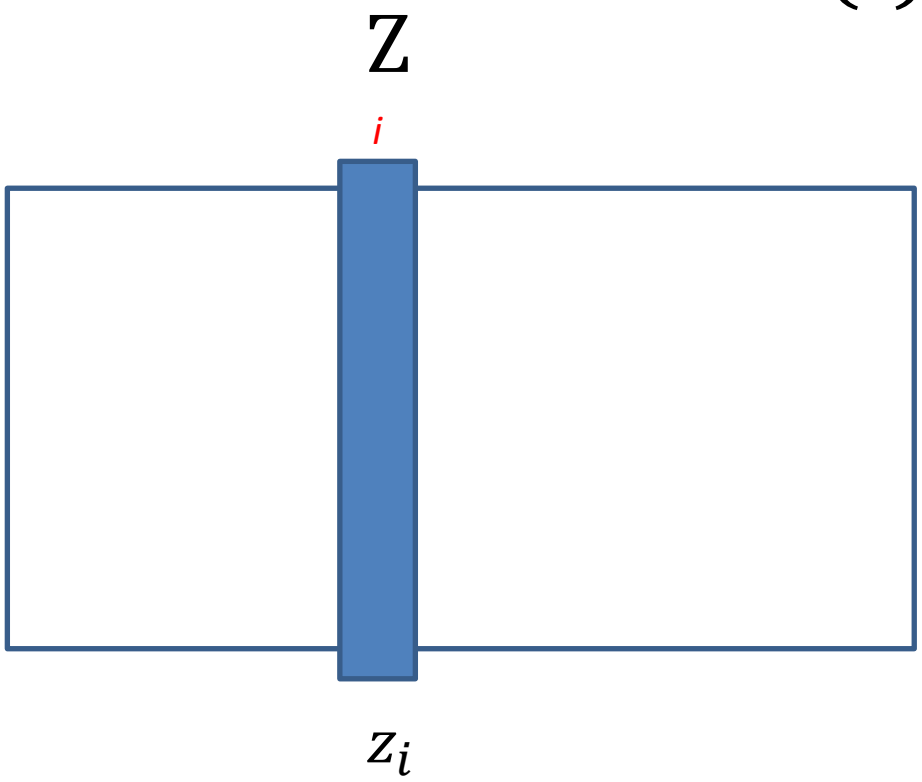
Είσοδος/έξοδος με μορφή one hot vectors

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

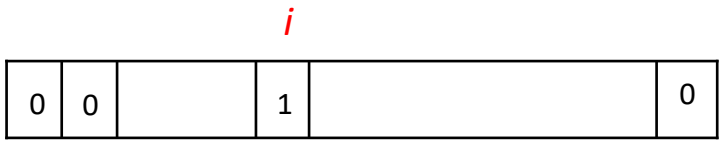
Look up

Encoder is an embedding lookup

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



$One\ hot\ vector\ I_i$



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

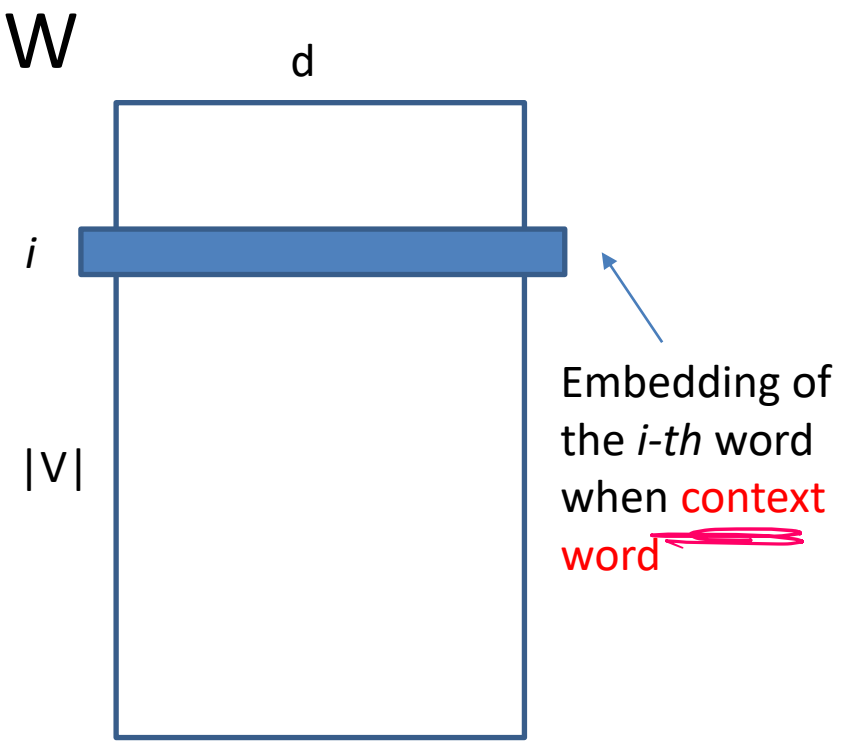
CBOW

Use a window of context words to predict the center word

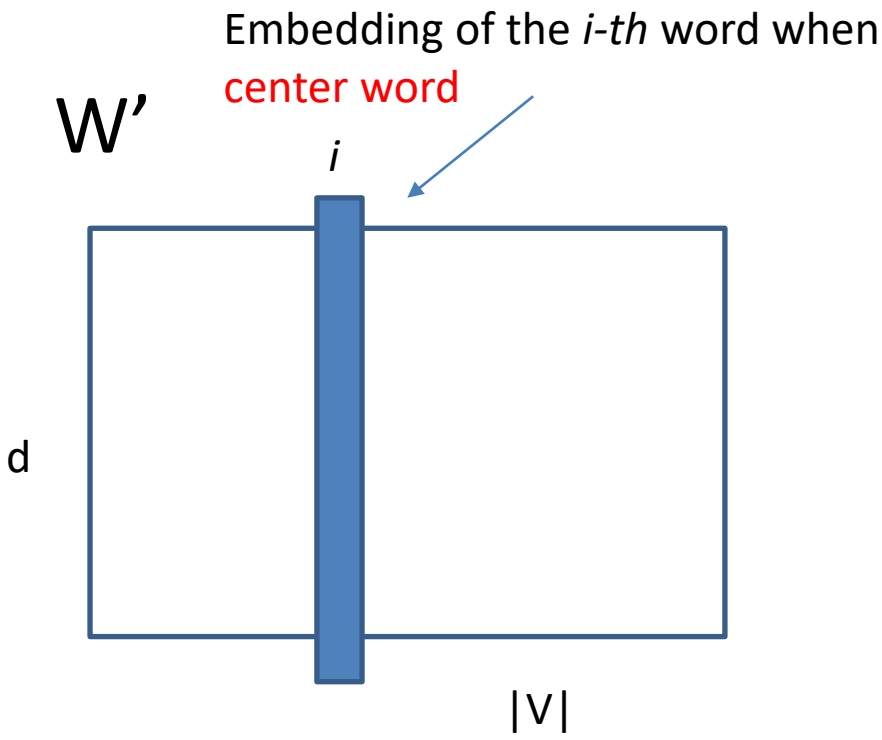
Input: 2m context words
Output: center word
each represented as a one-hot vector

CBOW

Learns two matrices (two embeddings per word, one when context, one when center)



|V| x d context embeddings when *input*



d x |V| center embeddings when *output*

CBOW

Intuition

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

- For similarity, we will use cosine (dot product)
- We will take the average of the W-embeddings of the context word

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

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

2. *GET THE EMBEDDINGS of the context words*

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

3. *TAKE THE SUM* these vectors (average)

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

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

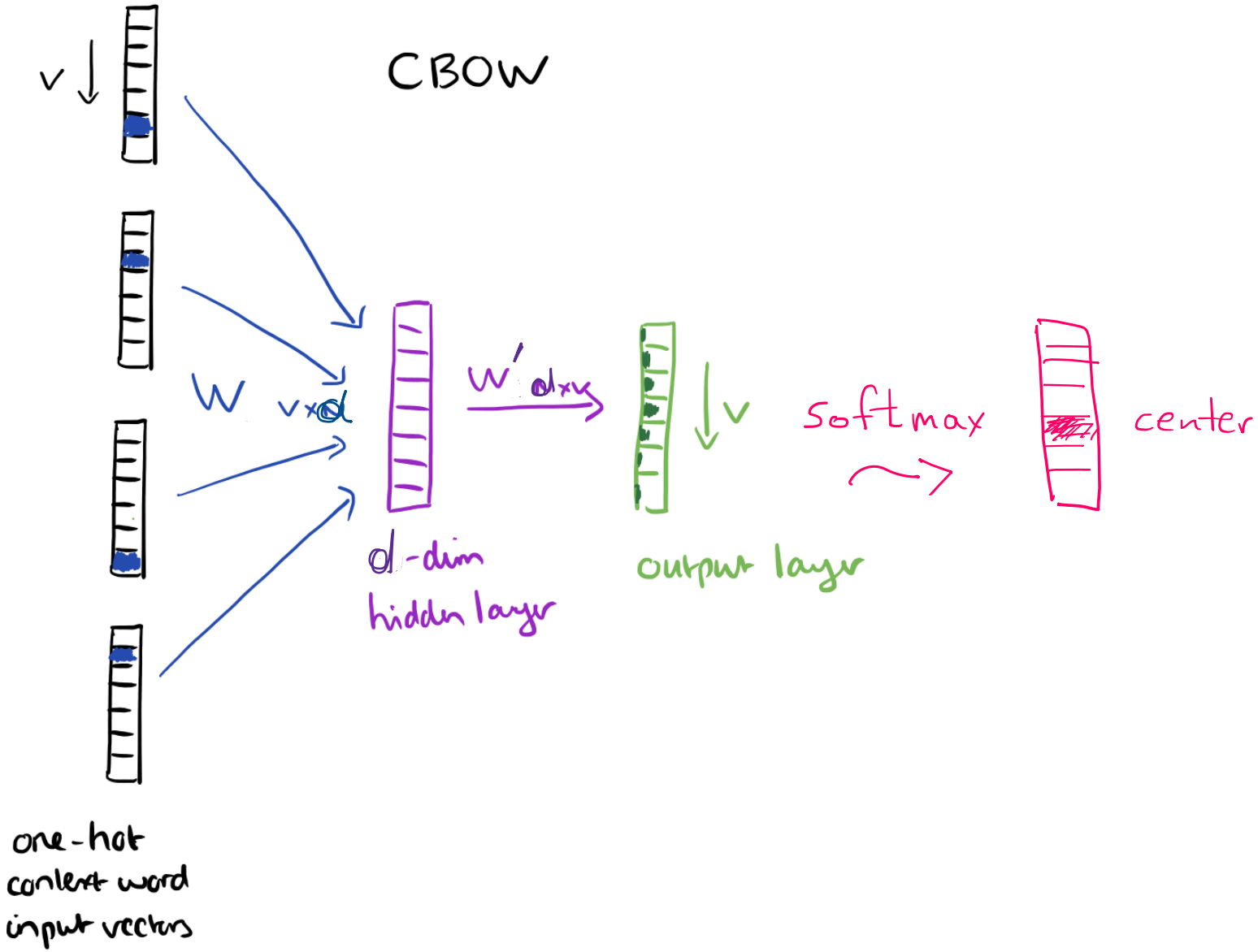
z = W' $\hat{v}$

5. Turn the *score vector to probabilities*

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

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

m=2


CBOW
V
W
V×d
W'
d×V
V
Softmax
center
d-dim
hidden layer
output layer
one-hot
context word
input vectors

SKIPGRAM

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: d x |V|, input matrix, word representation as center word
W': |V| x d, output matrix, word representation as context word

Skipgram

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

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

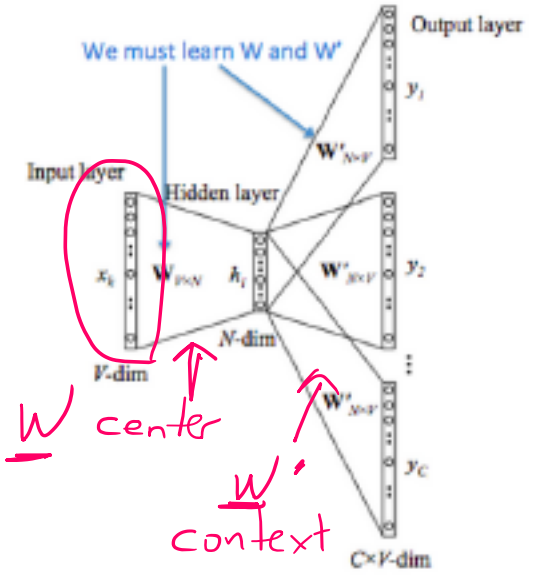
1. Input: *one hot vector* of the center word
x

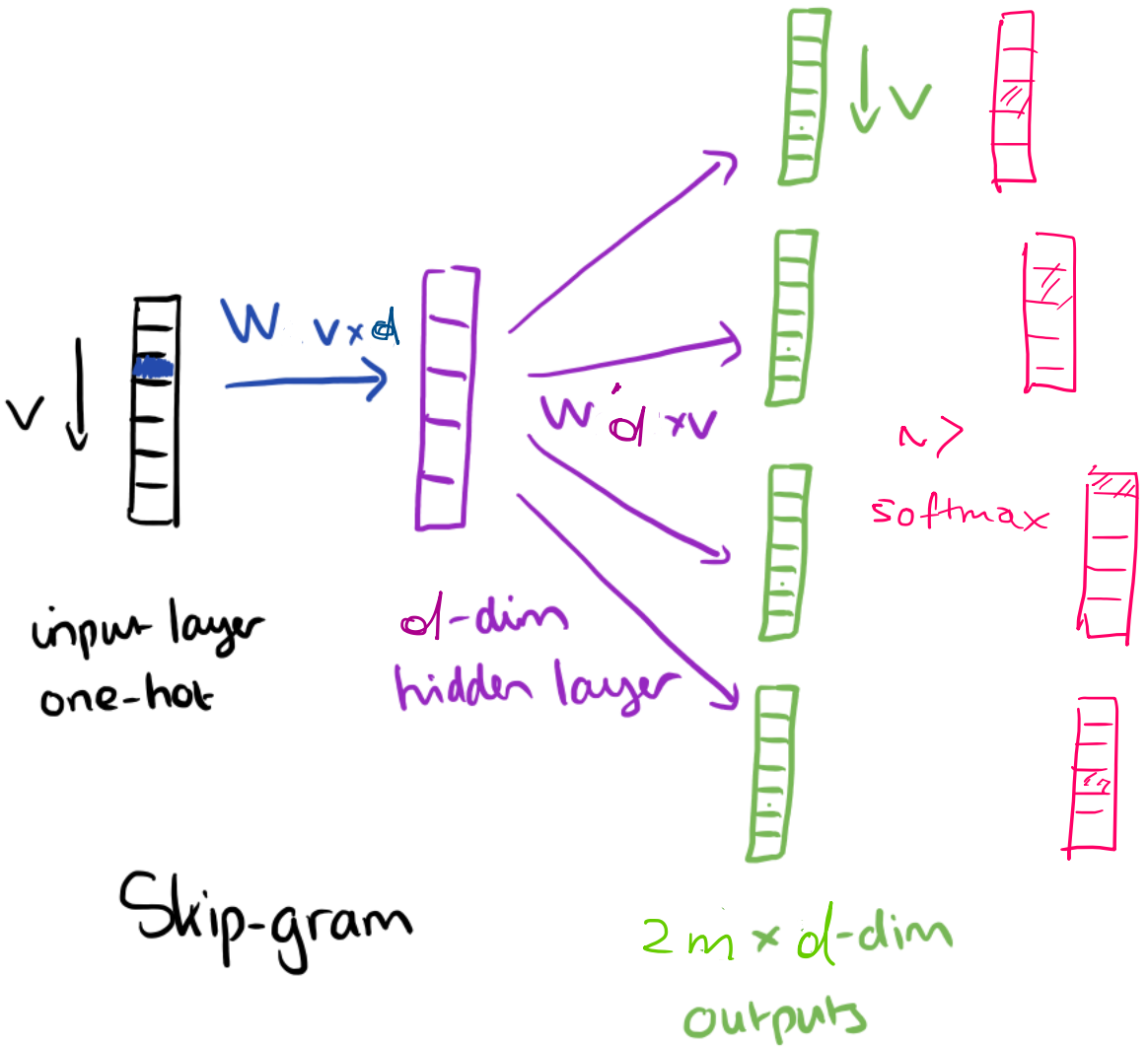
2. Get the *embedding of the center word*
$v_c = W\ x$

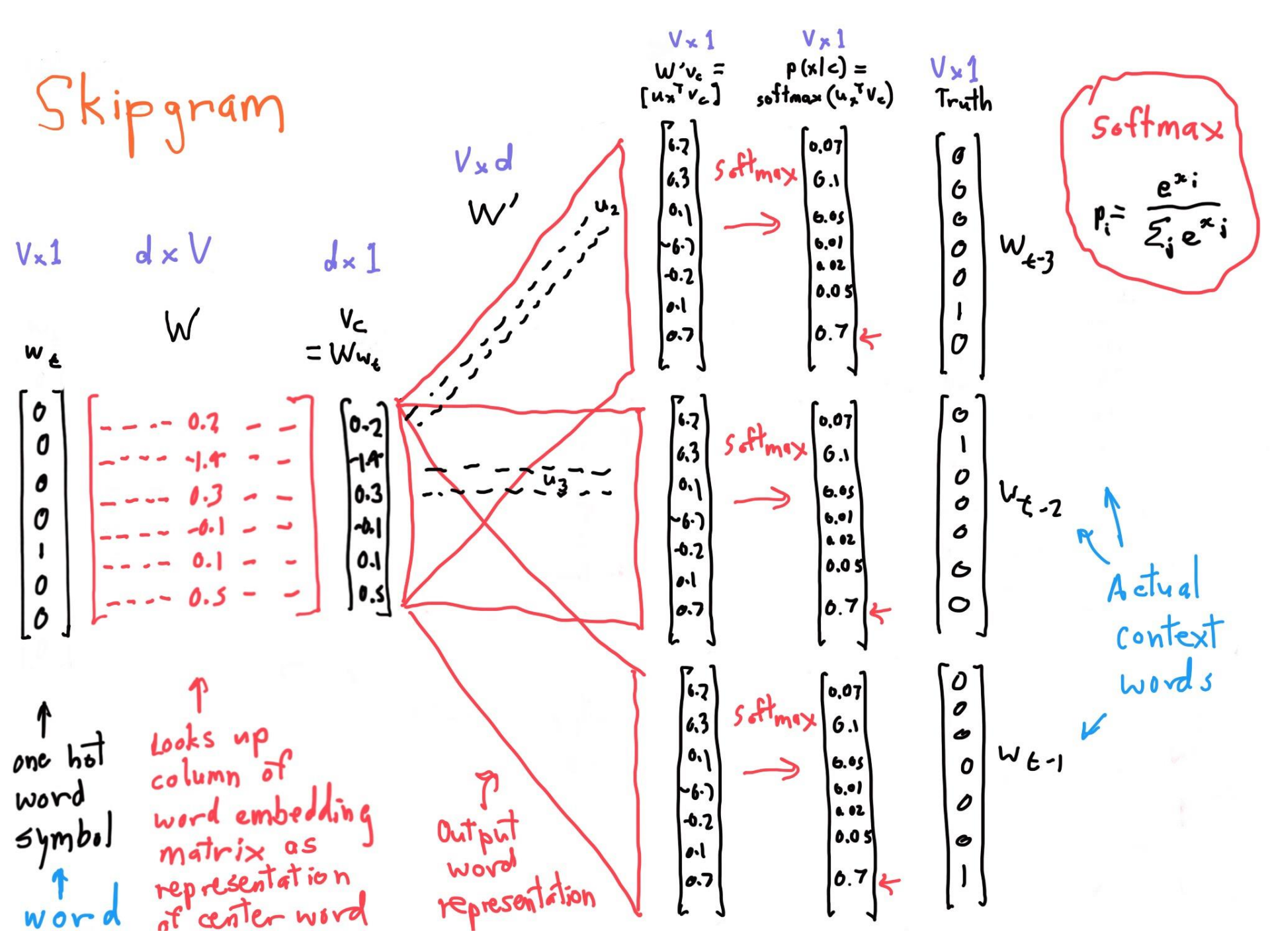
3. Generate a *score vector for each context word*
z = *W'* v_c

5. Turn the *score vector into probabilities*
$\hat{y}$ = *softmax(z)*

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




V
W V×d
input layer
one-hot
d-dim
hidden layer
W' d×V
V
V
softmax
2m × d-dim
outputs
Skip-gram


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

Αποτελέσματα

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

- Analogies

A is to B what C is to D

- 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

  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

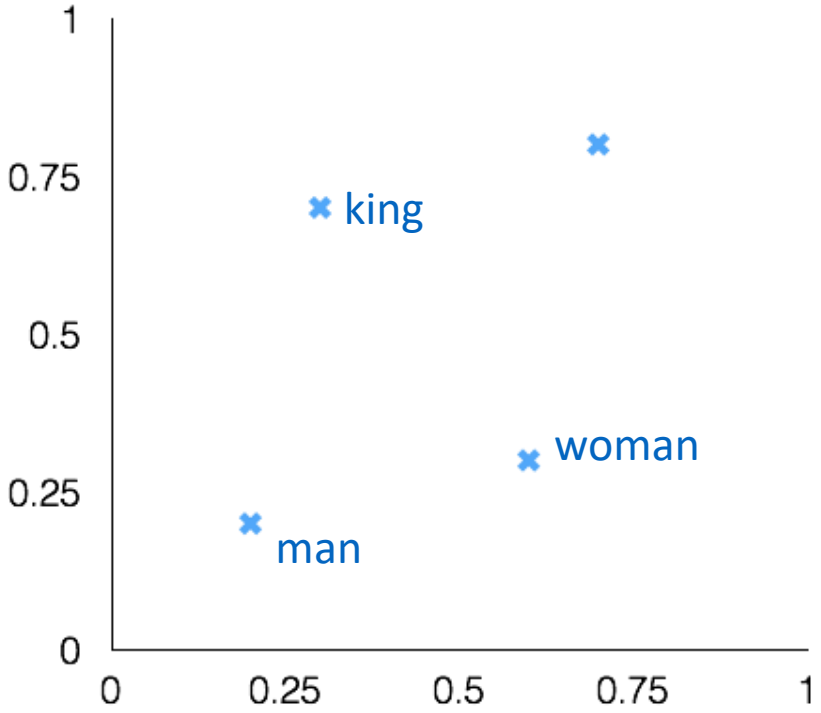
Test for linear relationships, examined by Mikolov et al.

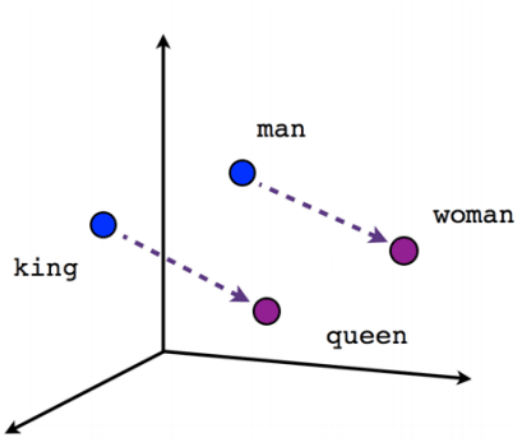
a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

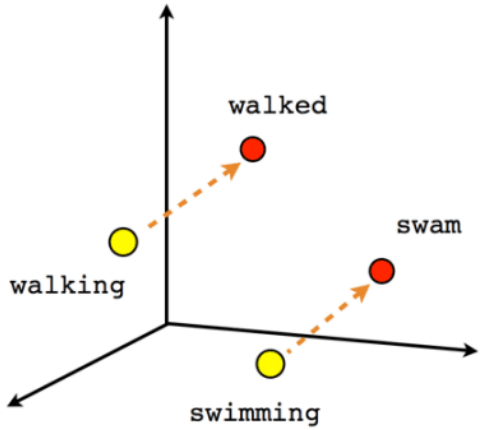
man:woman :: king:?

+	king	[0.30 0.70]
−	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

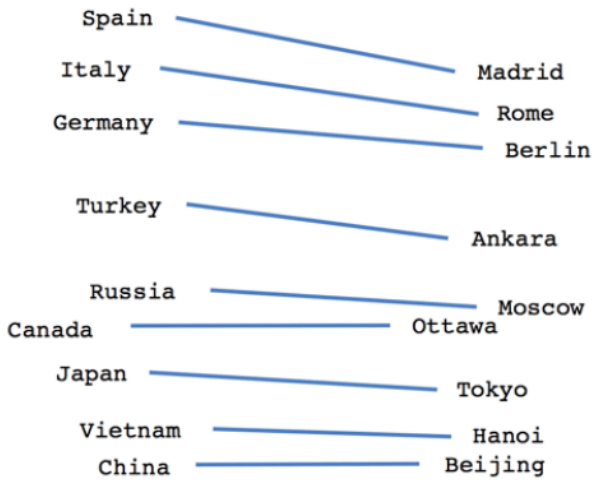




Male-Female



Verb tense



Country-Capital

Κάποιες εφαρμογές

- Στην *εργασία*, σας ζητείτε να χρησιμοποιείστε word embeddings
  - Πρέπει να επιλέξετε πως
  - Θα αξιολογηθεί και η καταλληλότητα/πρωτοτυπία/χρησιμότητα

- Στη συνέχεις θα δούμε
  - pretrained embeddings
  - μερικές εφαρμογές

Global vs. local embedding

global	local
cutting	tax
squeeze	deficit
reduce	vote
slash	budget
reduction	reduction
spend	house
lower	bill
halve	plan
soften	spend
freeze	billion

Πάνω σε ποια συλλογή (corpus) φτιάχνουμε τα embeddings;
Προτάσεις από ποια κείμενα θα χρησιμοποιήσουμε;

[Diaz 2016] Terms similar to 'cut' for a word2vec model trained on a general news corpus and another trained only on documents related to 'gasoline tax'.

1. Train and create embeddings based on a local collection

Python implementation in gensim

https://radimrehurek.com/gensim/models/word2vec.html

Tensorflow

https://www.tensorflow.org/tutorials/text/word_embeddings

2. Use pretrained embeddings

Pretrained embeddings for 157 languages

https://fasttext.cc/docs/en/crawl-vectors.html

Google

https://code.google.com/archive/p/word2vec/

Finding the degree of similarity between two words.

```
model.similarity('woman','man')
0.73723527
```

Finding odd one out.

```
model.doesnt_match('breakfast cereal dinner lunch';.split())
'cereal'
```

Amazing things like woman+king-man =queen

```
model.most_similar(positive=['woman','king'],negative=['man'],top
n=1)
queen: 0.508
```

Probability of a text under the model

```
model.score(['The fox jumped over the lazy dog'.split()])
0.21
```

ανεκτική ανάκτηση: (1) επέκταση ερωτήματος ή/και (2) context-dependent διόρθωση λάθους, όπου θα μπορούσαμε να χρησιμοποιήσουμε και το query log και γενικά query suggestions

Improve language translation

bilingual embedding with chinese in green and english in yellow

By aligning the word embeddings for the two languages

Χρήση στη διάταξη των εγγράφων του αποτελέσματος μιας ερώτησης

Είδαμε διάταξη με $q^T d$

Μπορούμε να χρησιμοποιήσουμε embeddings;

Πολλές εναλλακτικές, για παράδειγμα (aboutness)

$$\sum_{w \in q} w d'$$

Όπου w το *embedding των λέξεων της ερώτησης* και d' το embedding του εγγράφου (π.χ., το μέσο των embedding των λέξεων του εγγράφου)

- Στόχος: Σχέση ερώτησης με το όλο το περιεχόμενο του εγγράφου
- Input (center word) embedding ή output (context) word embedding; in-query, out-document
- Σε συνδυασμό με άλλα κριτήρια

Και άλλα embeddings

sentence2vec, doc2vec
node2vec

BERT (transformers)

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
- 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/