Semantics
&
Word2Vec

- Understand Semantic Analysis
- Introduce Word Vectors
- Discuss Word2Vec

Semantics and Word Vectors

- Why we need more dimensions to a Word, hence Word Vector ?
- How word vectors are created ?

Word to Numbers

In order for Computer to Understand Context, we need to convert document into numbers. One way is just assign an arbitary number.

- *Antalya is great => 4.2 -1.9 12*
- *Antalya is awesome => 4.2 -1.9 -3.04*

Consider giving «great» and «awesome» a number close enough to indicate that they are similiar in meaning.

How about «Pet» and «Cat» vs «Cat» and «Tiger» ?

Because there are many dimensions to similarity. It may be better to assign more than 1 number.

Word to Numbers

- A number may indicate if it is an animal
- Another number may indicate if it is domesticated

Because there are many possible contextes. This is a lot of work.

But the good news is that we can develop a ANN to learn these contextes for us.

Semantics and Word Vectors

- Word2vec is a two-layer neural net that processes text.
- Its input is a text corpus and its output is a set of vectors:
  - Feature vectors for words in that corpus.

Word2vec Purpose and Usefulness

- Word2vec groups the vectors of similar words together in vectorspace.
- It detects similarities mathematically.

- Those similarities are used to establish a word’s association with other words (e.g. “man” is to “boy” what “woman” is to “girl”)

Word2vec creates vectors

- Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words.

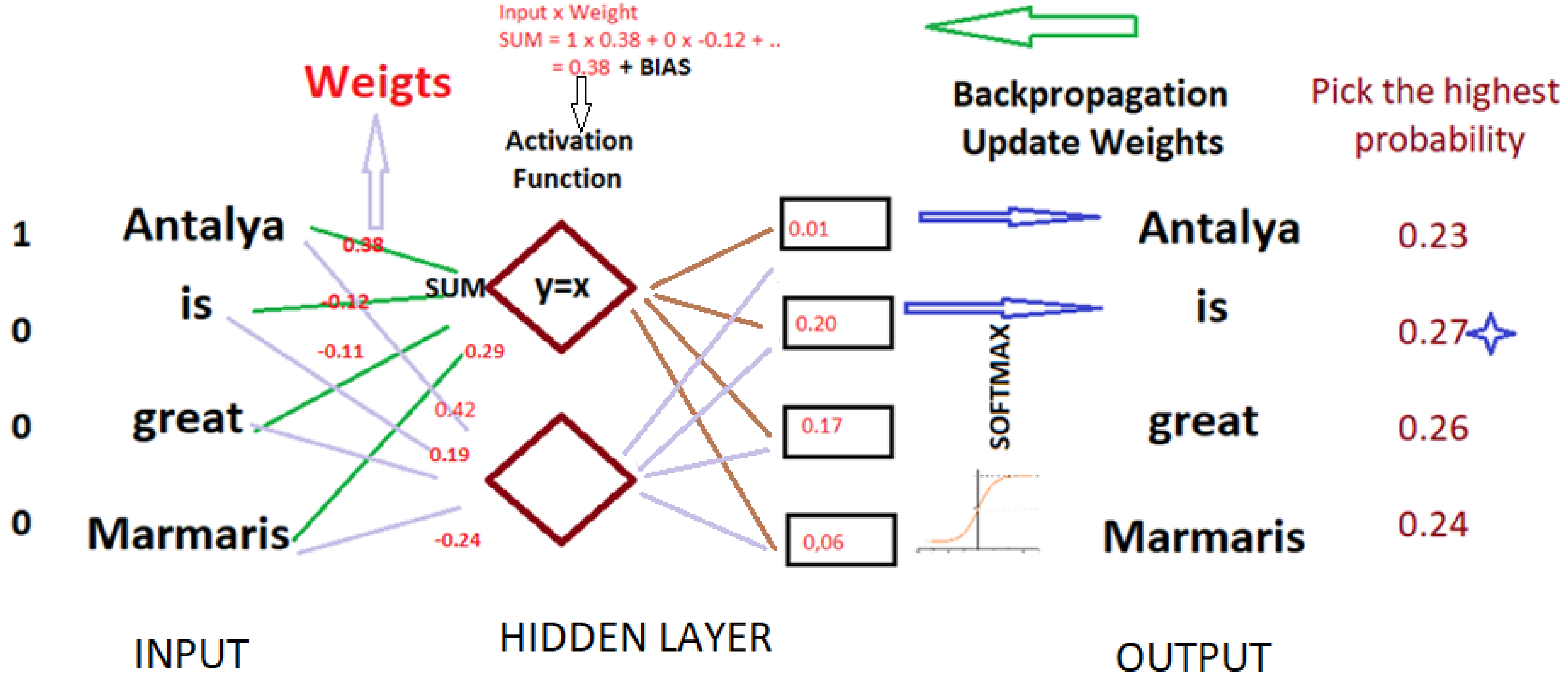
Word2vec power

- Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word’s meaning based on past appearances.

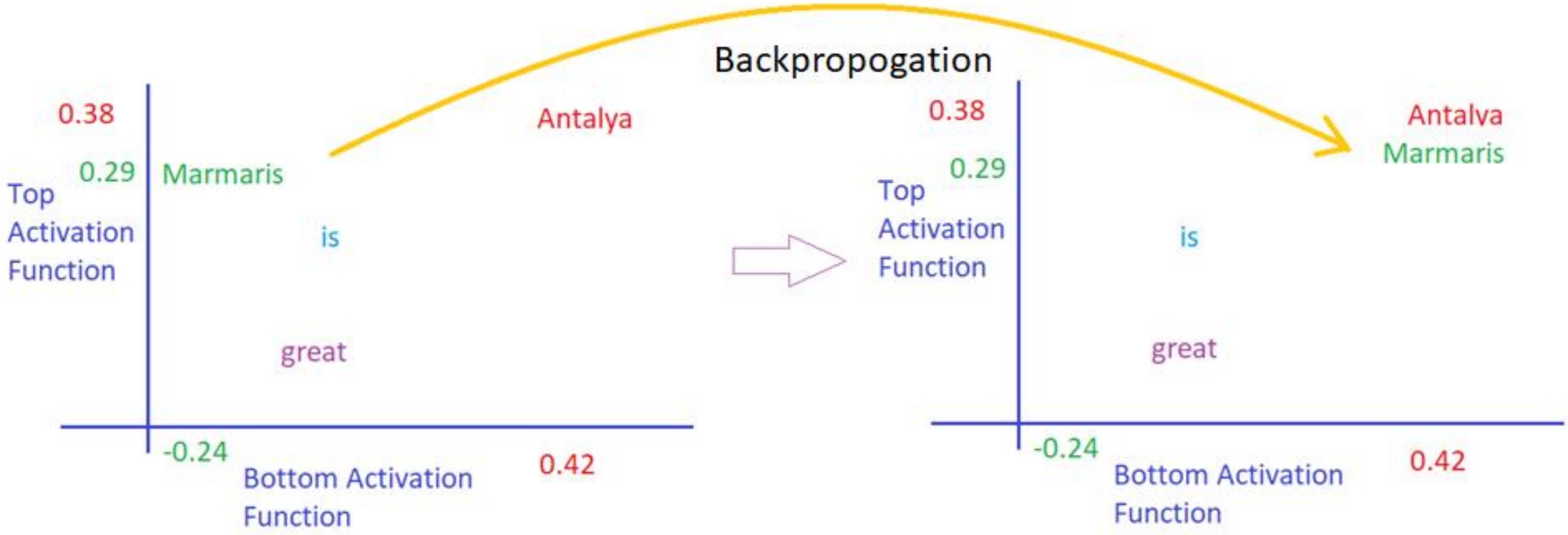
- It does so without human intervention.

ANN for Word2Vec - Predict the next Word

Using 2 Dimensions / Embedding for each Word { Antalya (0.38, 0,42) }
Window Size = 2 (1 input + 1 output)

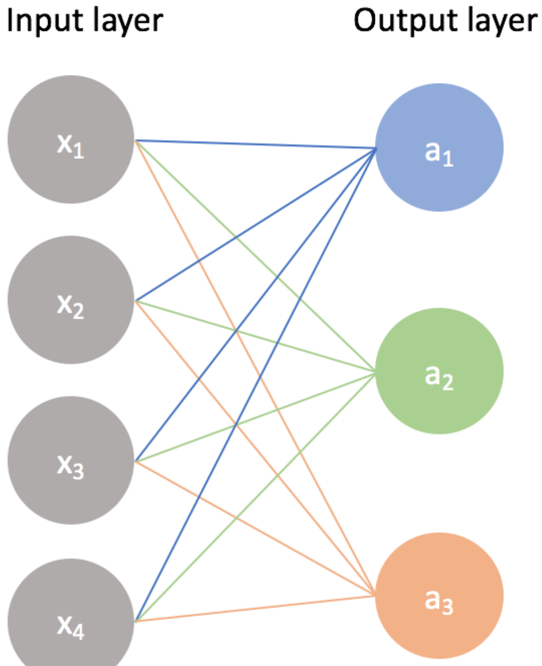


Backpropogation for better embeddings



Row Data for training

- Antalya (1, 0, 0, 0) => is (0, 1, 0, 0)
- is (0, 1, 0, 0) => great (0, 0, 1, 0)
- great (0, 0, 1, 0) => ?? EOS
- Marmaris (0, 0, 0, 1) => is (0, 1, 0, 0)



$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{array}{c} \begin{array}{cccc} \text{Observation 1} & \text{Observation 2} & \text{Observation 3} & \text{Observation 4} \end{array} \\ \begin{bmatrix} x_1 & x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 \end{bmatrix} \end{array} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} \xrightarrow[\text{activation}]{} \begin{array}{c} \begin{array}{cccc} \text{Observation 1} & \text{Observation 2} & \text{Observation 3} & \text{Observation 4} \end{array} \\ \begin{bmatrix} a_1 & a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 & a_3 \end{bmatrix} \end{array}$$

Using multiple observations

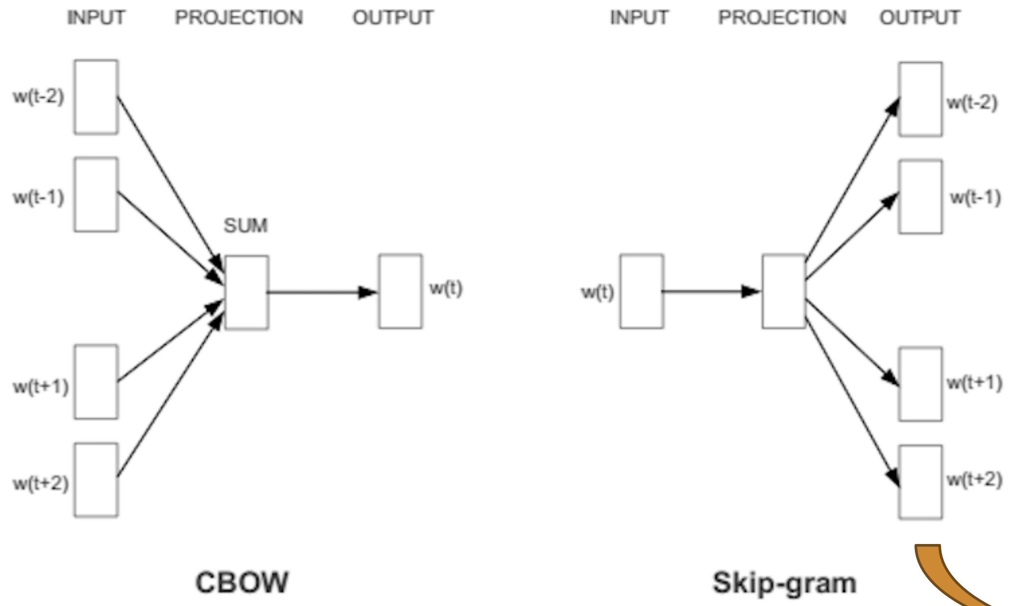
Instead of Next Word - Word2Vec trains

- Against other words that neighbor them in the input corpus.

It does so in one of two ways, either;

- context to predict a target word (a method known as Continuous Bag of Words, or cBoW),
- a word to predict a target context, which is called Skip-gram.

Two Possible Approaches



Antalya	1	0
is	0	1
Great	1	0
Marmaris	1	0

Antalya	0	0.33
is	1	0
Great	0	0.33
Marmaris	0	0.33

Window size of 5 - CBOW

4 inputs + 1 output
4 Rows (Data)

1. The quick brown fox jumped over the fence
2. The quick brown fox jumped over the fence
3. The quick brown fox jumped over the fence
4. The quick brown fox jumped over the fence

BOW: the, quick, brown, fox, jumped, over, fence

CBOW:

3 Dimensions per Word
Vocab size #7
Window Size 5

the	1
quick	0
brown	0
fox	0
jumped	0
over	0
fence	0

the	0
quick	1
brown	0
fox	0
jumped	0
over	0
fence	0

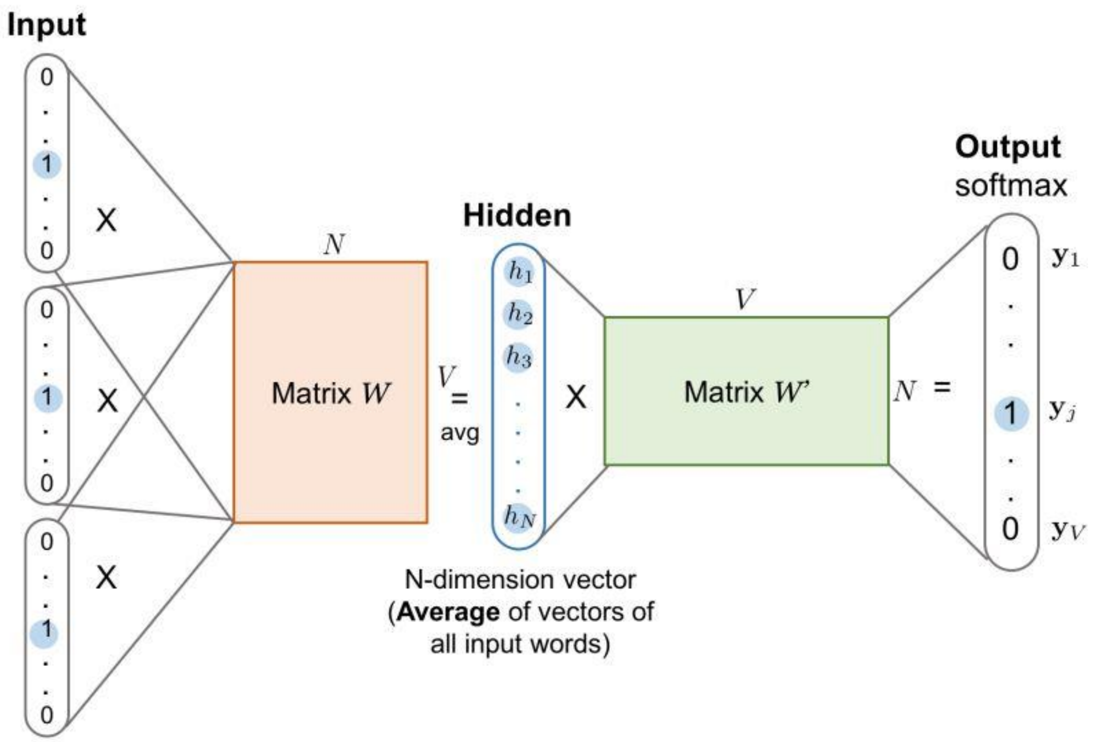
the	0
quick	0
brown	0
fox	1
jumped	0
over	0
fence	0

the	0
quick	0
brown	0
fox	1
jumped	0
over	0
fence	0

7x3x7

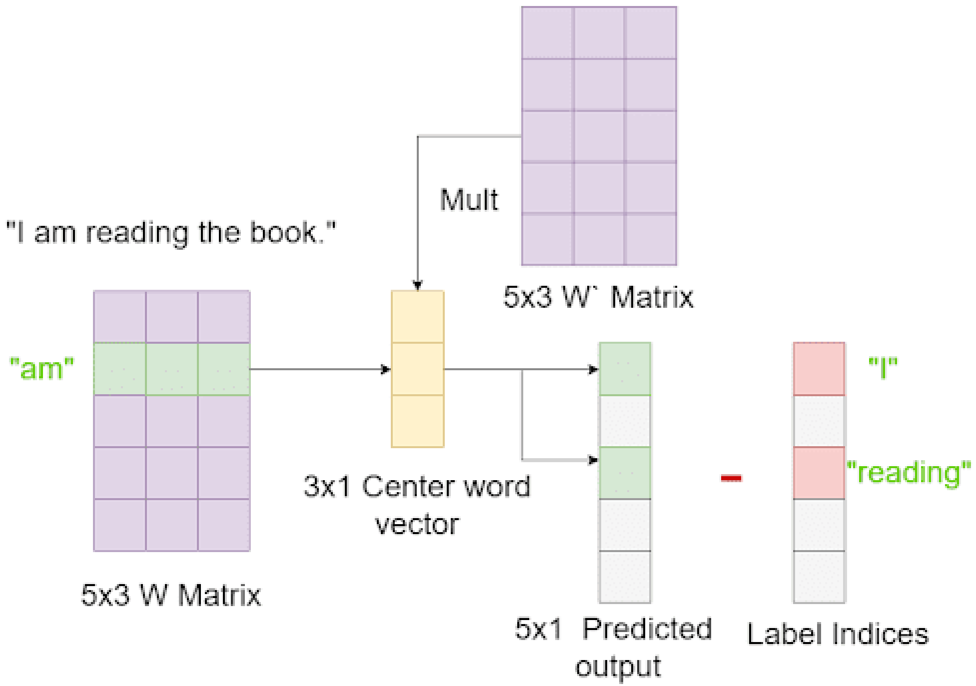
the	0
quick	0
brown	1
fox	0
jumped	0
over	0
fence	0

In our example, N = 3, V=7



Skip-gram

Window size 3
N=3 Dimensions
Vocab Size = 5

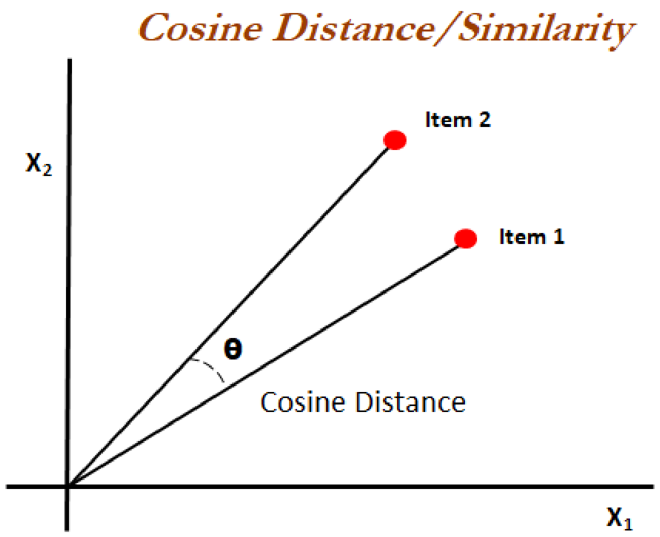


Each Word is a Vector

- Recall that each word is now represented by a **vector.**
- In Spacy each of these vectors has 300 dimensions. That is 300 Embeddings (Activation Functions)
- Vocab Size is 100Ks to 1 Million
- 1000.000 x 300

Cosine Similarity

This means we can use Cosine Similarity to measure how similar word vectors are to each other.



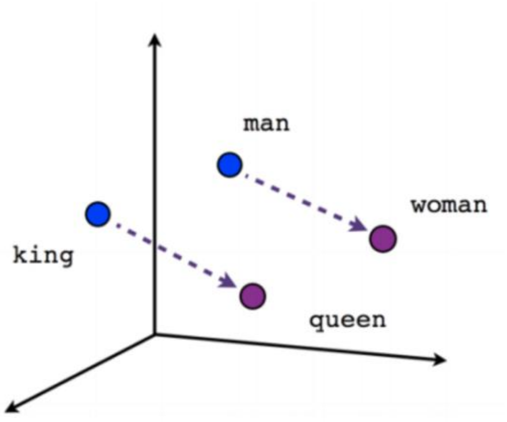
Vector Arithmetic

- This means we can also perform vector arithmetic with the word vectors.
  - **new_vector = king - man + woman**

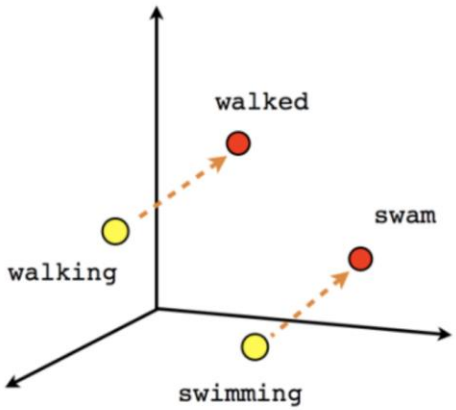
- This creates new vectors (not directly associated with a word) that we can then attempt to find most similar vectors to.
  - **new_vector closest to vector for queen**

Word Vector Relationships

Interesting relationships can also be established between the word vectors



Male-Female



Verb tense

Let’s begin to explore Spacy Word Vectors with Python!

Semantics and Word Vectors

- In order to use Spacy's embedded word vectors, we must download the **larger** spacy english models.
- Full details can be found at:
  - **https://spacy.io/usage/models**

Semantics and Word Vectors

- At the command line download the medium or large spacy english models:
- **python -m spacy download en_core_web_md**
- **python -m spacy download en_core_web_lg**