APAN PS5430
Applied Text & Natural Language Analytics
Week 6: Vector Space Modeling using Neural Networks

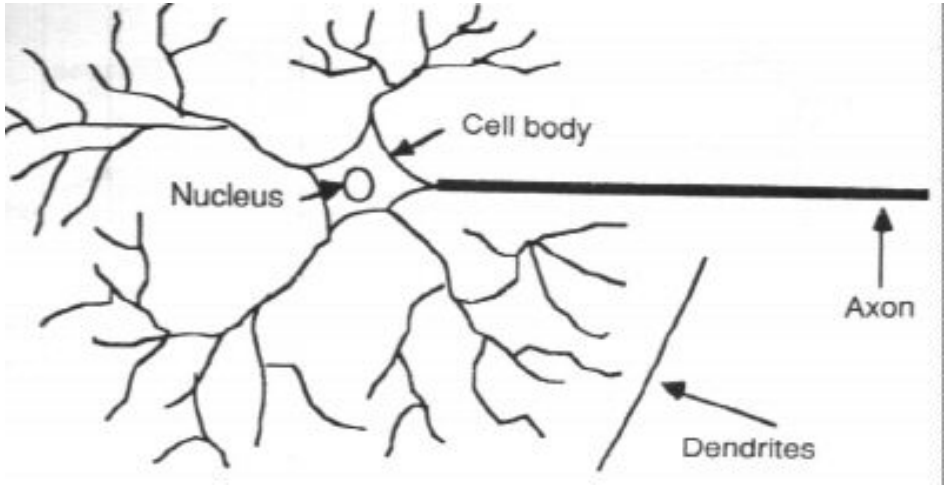
Javid Huseynov, Ph.D.
Wednesday, July 1, 2020

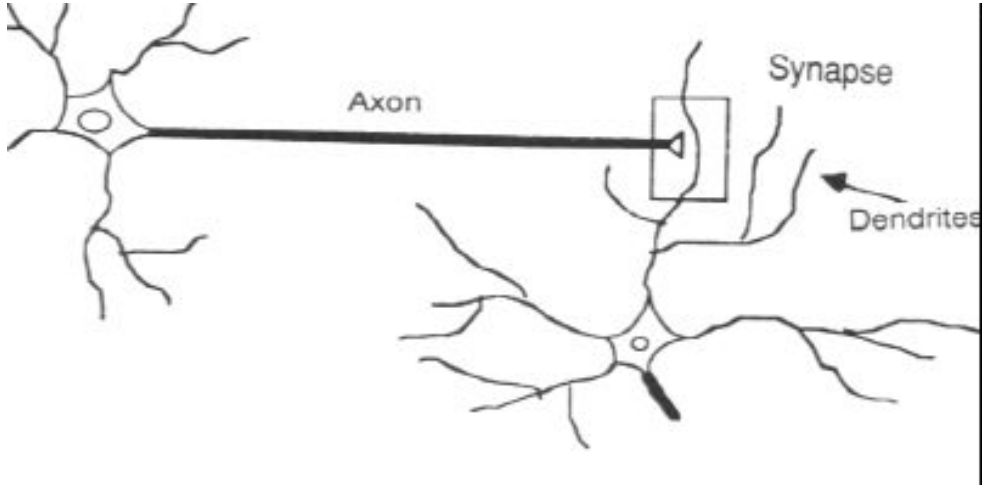
Week 6 Agenda

- Neural Networks Overview
- Vector-Space Model
- Sentence Vectors
- Word Vectors: Concept
- Co-occurrence Matrix
- Singular Value Decomposition (SVD)
- Word2Vec
- Class Exercises

Neural network: Bio inspiration

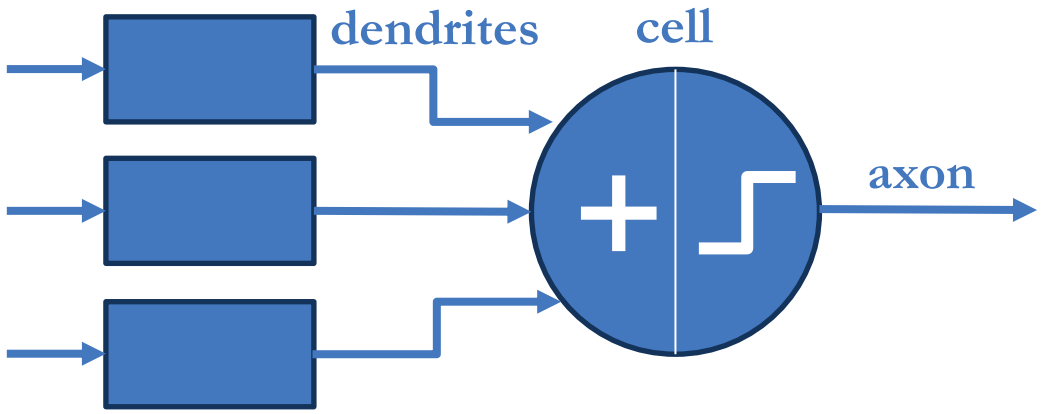
- Almost all living species can learn and react to changes in their environment using their nervous system



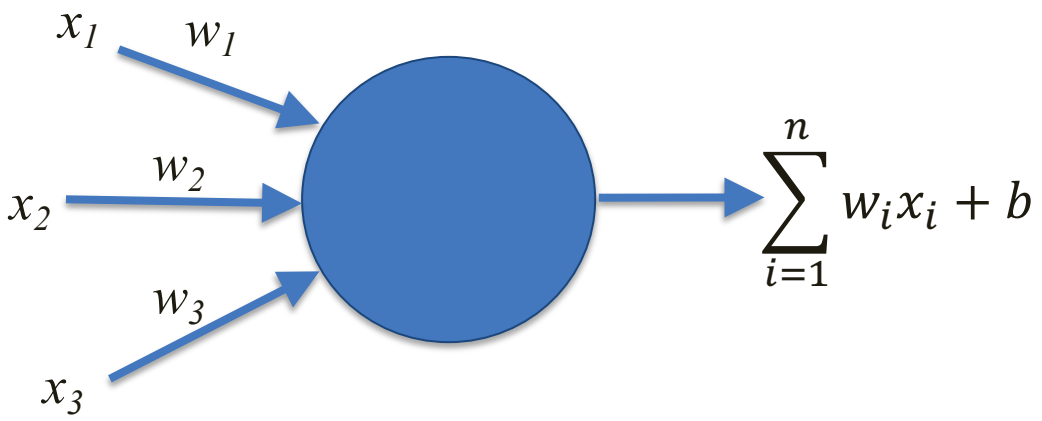


- An artificial mathematical model can reproduce the behavior of the central nervous system

From biological to artificial neurons



Simple Neuron – Perceptron Model (Rosenblatt 1958)

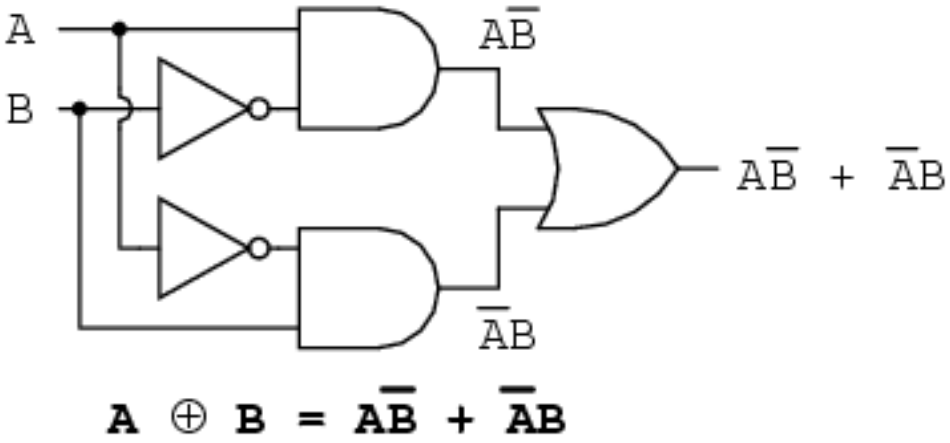


- Similar to linear regression
- Perceptron can learn fundamental logical functions: AND, OR, NOT

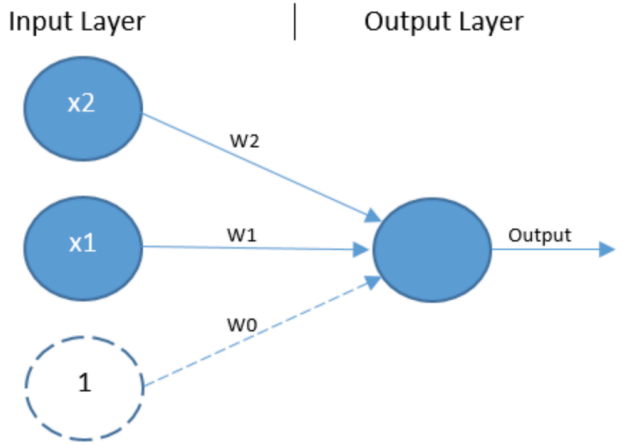
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Perceptron Limitation: Exclusive-OR (XOR) problem

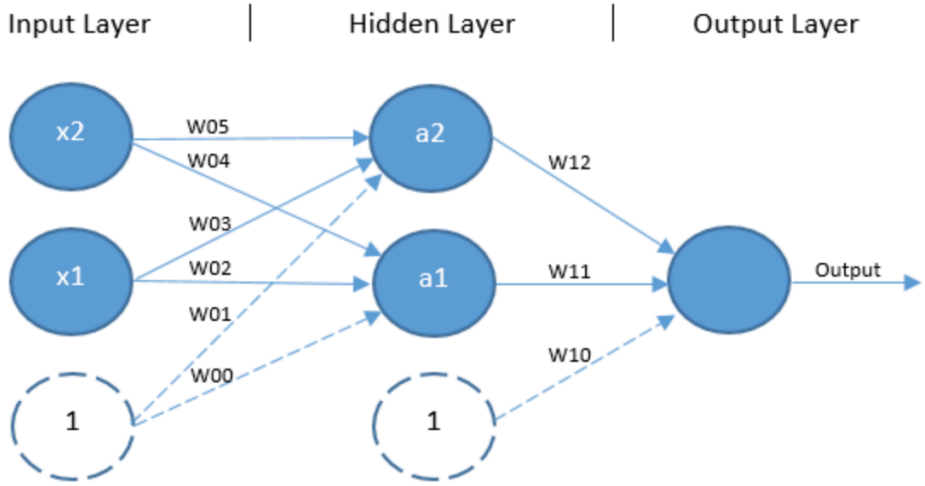
In 1969, Minsky & Pappert showed that Simple Perceptron cannot learn XOR logic



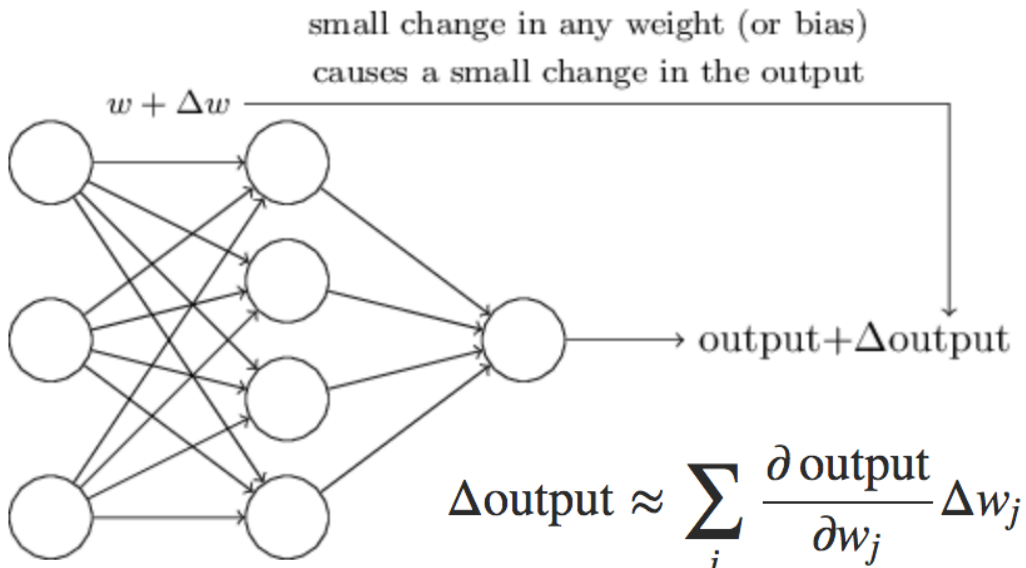
Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0



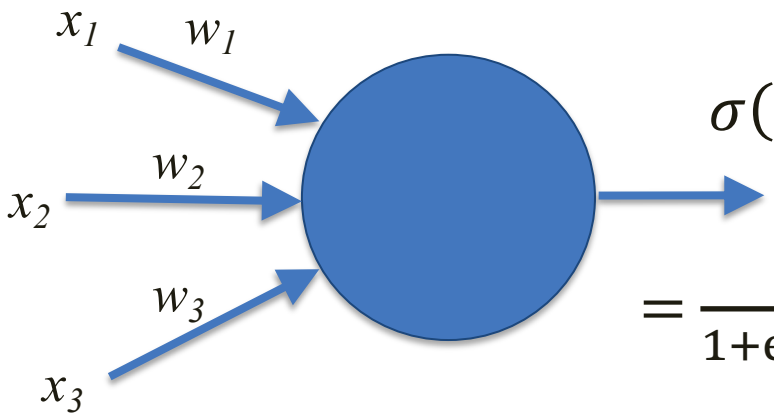
In 1985, Rumelhart & Hinton. demonstrated a solution to XOR through Backpropagation and non-linear activation function



From Perceptron to Activation Function



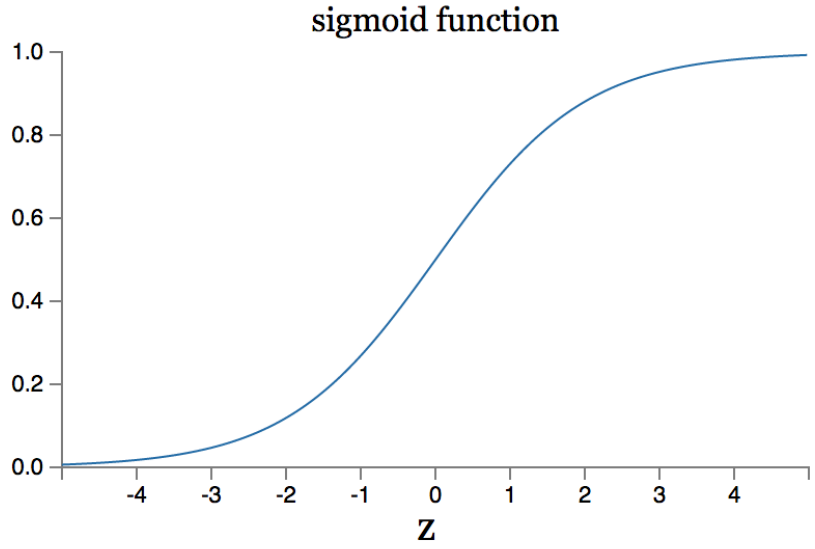
$$\Delta \text{output} \approx \sum_j \frac{\partial \,\text{output}}{\partial w_j} \Delta w_j + \frac{\partial \,\text{output}}{\partial b} \Delta b,$$



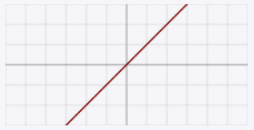
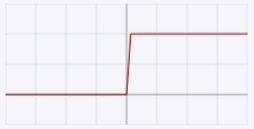
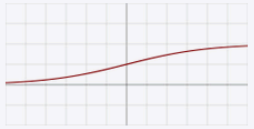
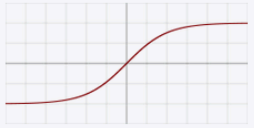
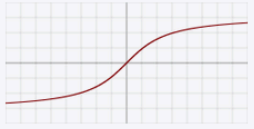
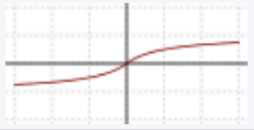
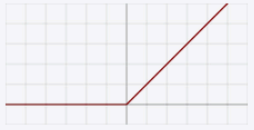
$$\sigma(z) = \frac{1}{1 + e^{-z}} =$$

$$= \frac{1}{1+\exp(-\sum_{i=1}^{n} w_i x_i - b)}$$

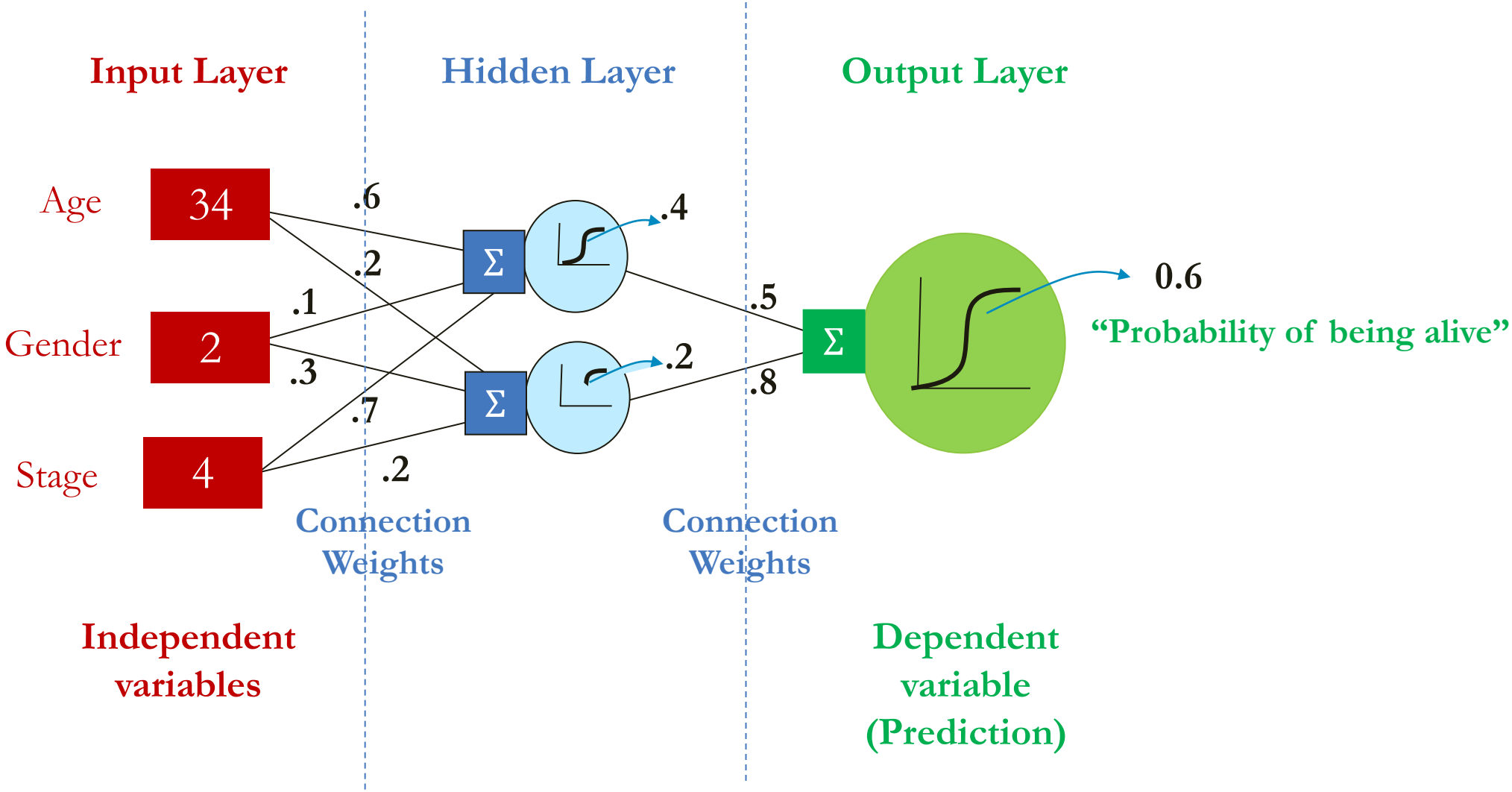
- Set of small changes in connection weights can alter the output significantly
- To smooth this impact and handle non-linearity, sigmoid (logistic) activation function is used



Types of Activation Functions

Name (Sort ascending)	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Softsign [7][8]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$

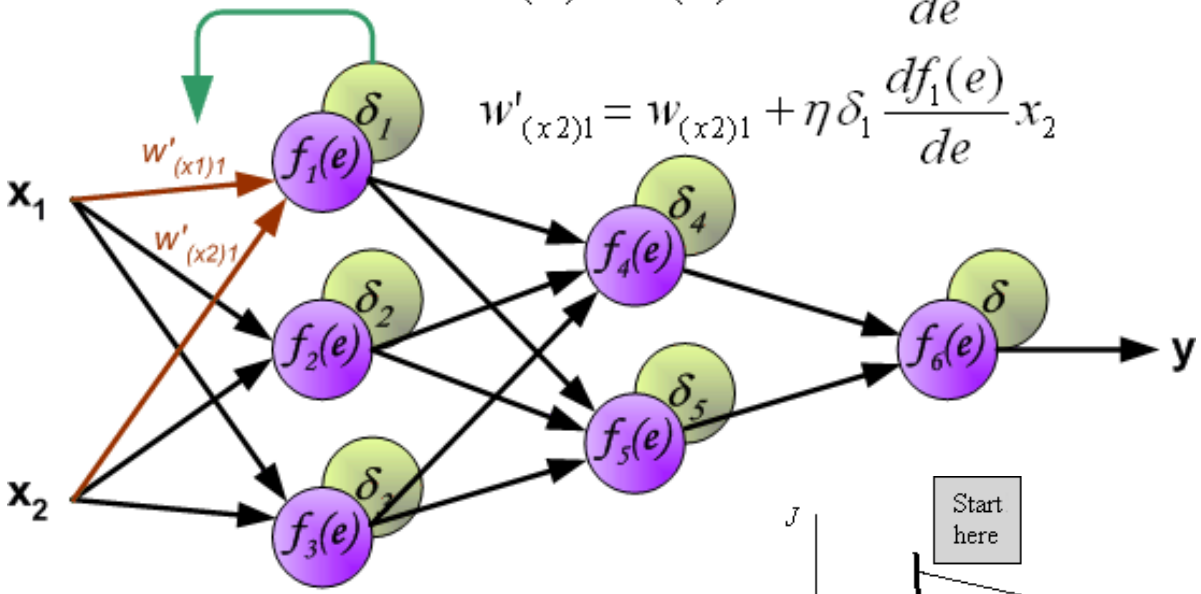
Feedforward Neural Network Example

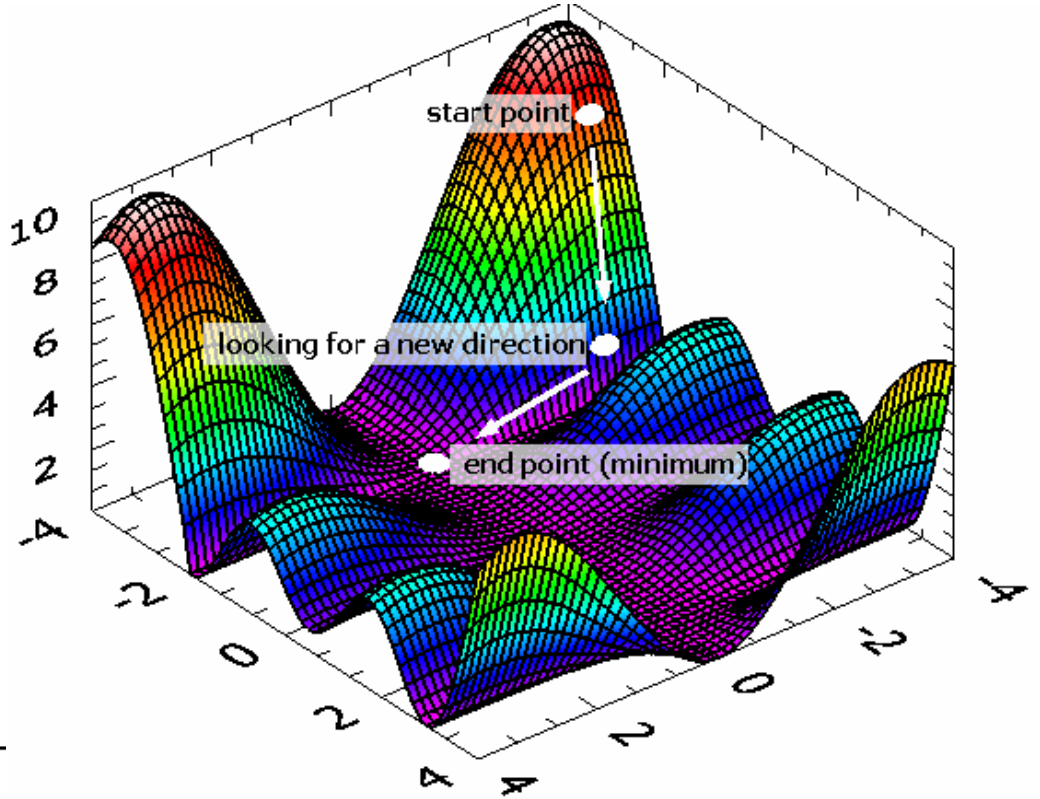


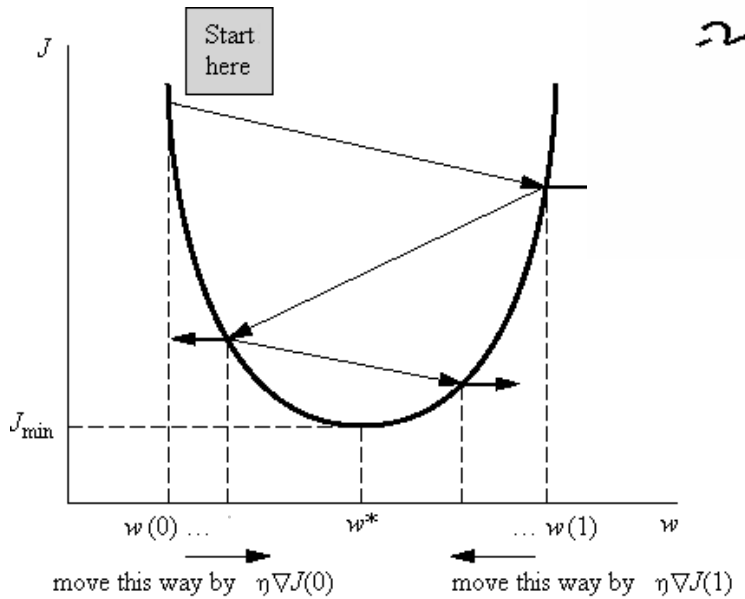
Backpropagation & conjugate gradient

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$







Backpropagation Step-by-Step

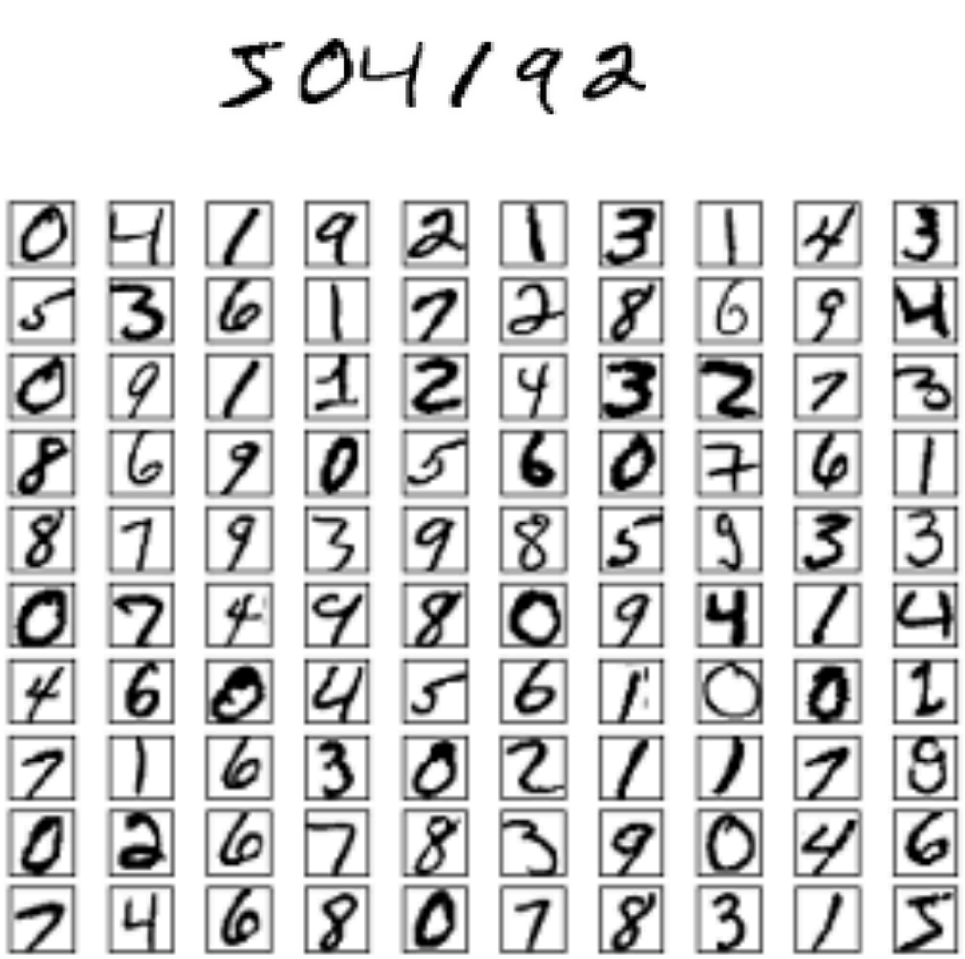
- Visit the following site to see a simple but great overview of the operation of the backpropagation algorithm with numbers:
- **https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/**

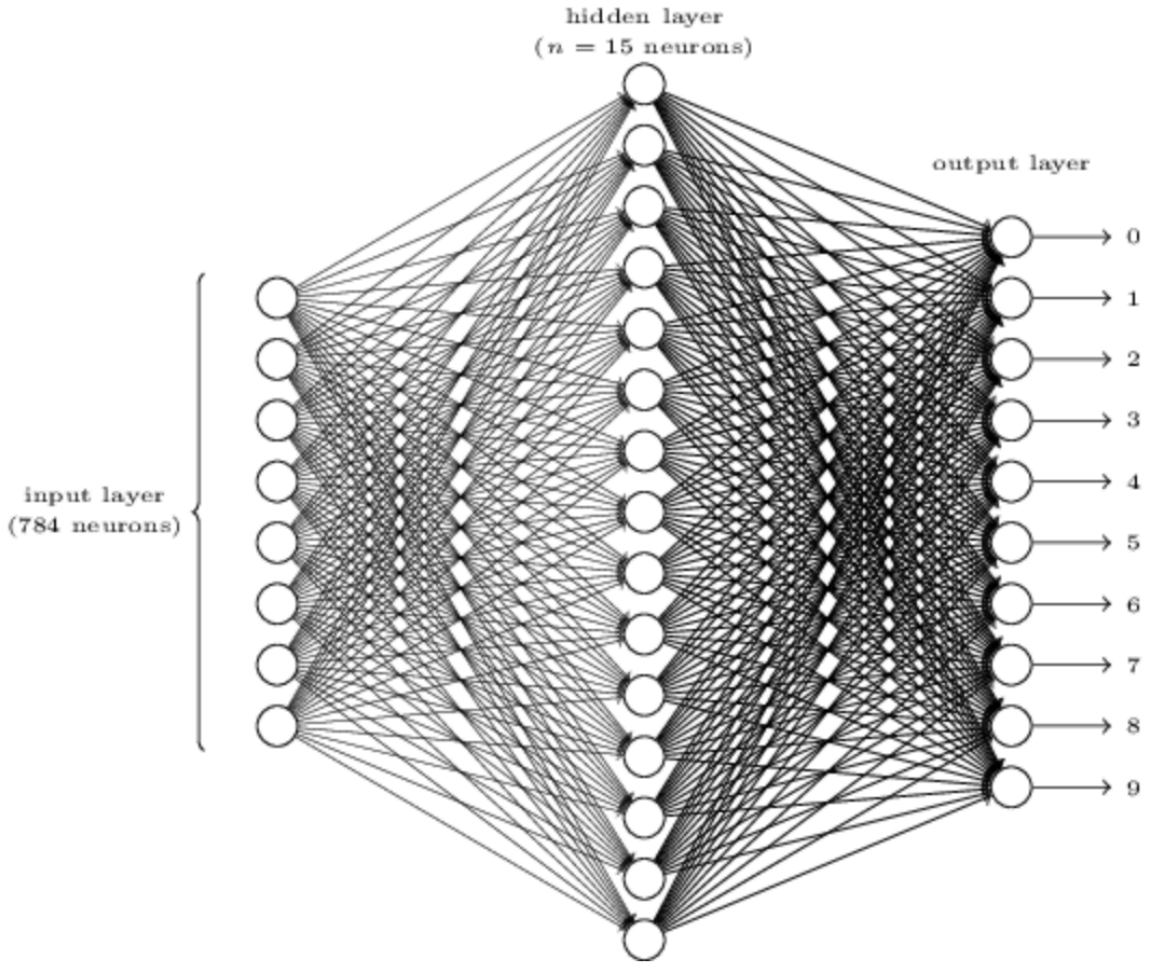
Neural Network Applications

- Neural networks in pop culture

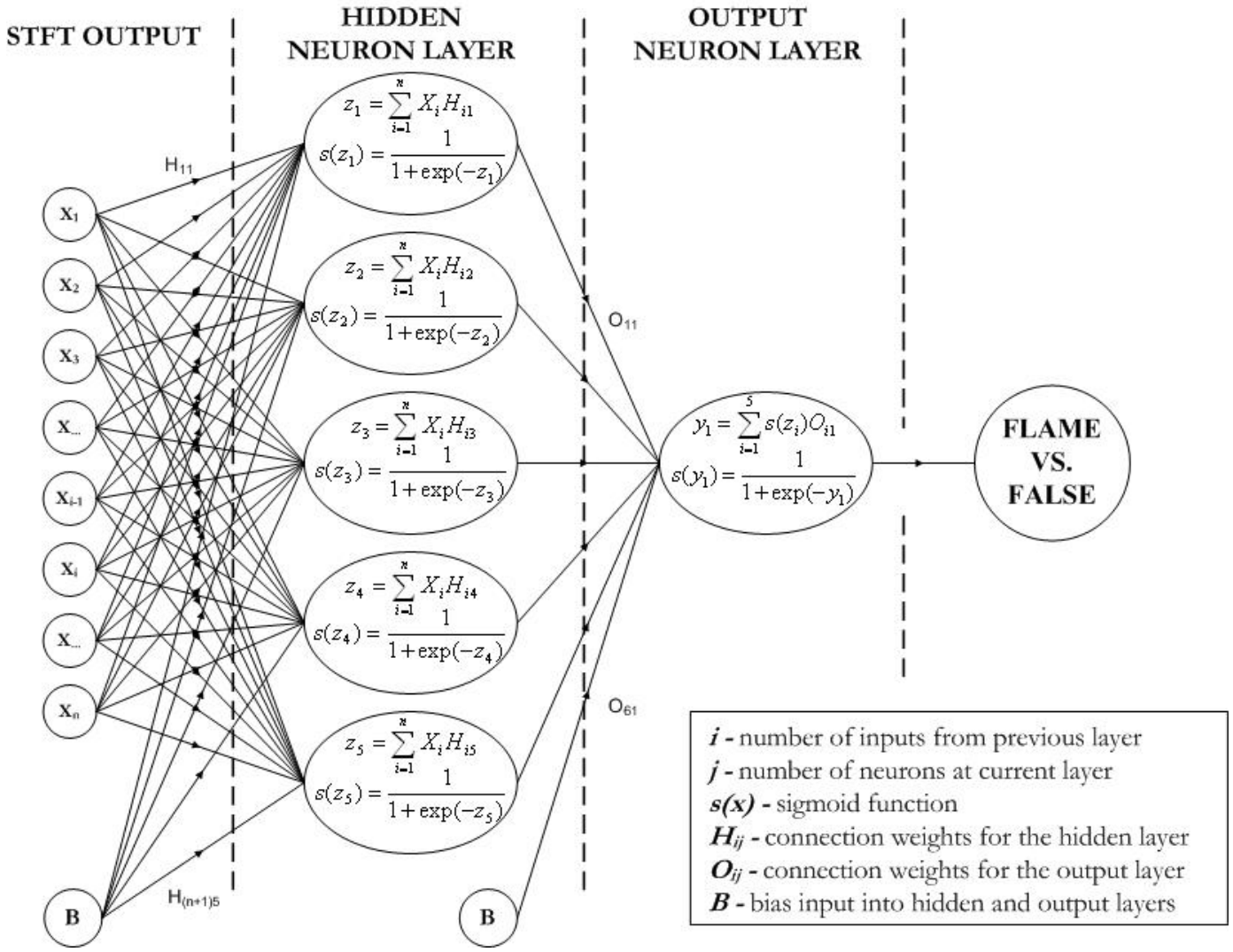
- Natural Language Processing
- Signal Processing | Time Series
- Stock Market Forecasting
- Credit Risk Assessment
- Image Processing | Character Recognition
- Speech Recognition
- Self-driving cars
- Sports Prediction
- Any other field where extra brain can be useful ☺

Use Case: Scanned Digit Recognition with Neural Networks





Another Use Case: Flame Detection using Neural Networks

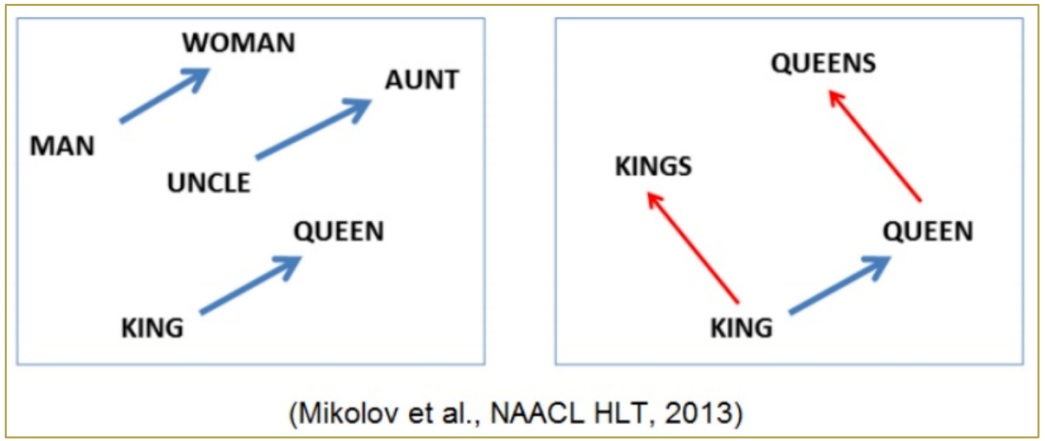


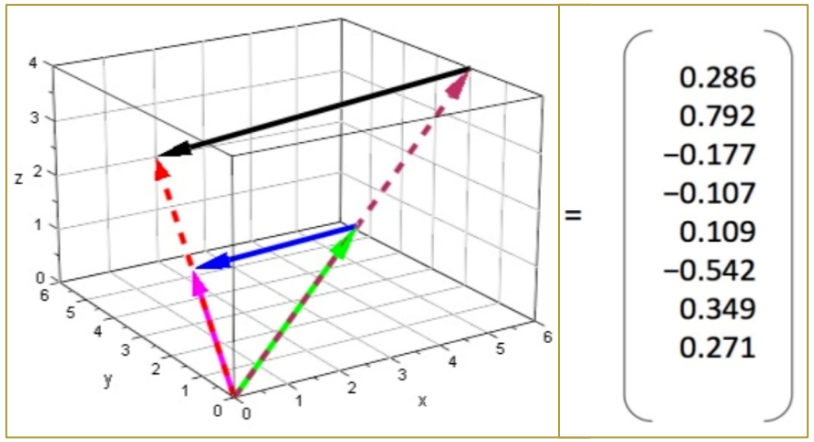
VSM + Neural Networks

Word vectors: concept

"You shall know a word by the company it keeps"
~ J.R. Firth 1957

- Vectors are directions in space, which can also encode relationships:
  - e.g. **man** is to **woman** as **king** is to **queen**





- One of the most successful ideas in modern statistical NLP
  - …government debt problem turning into banking crises as has happened in…
  - …saying that Europe needs unified banking regulation to replace the hodgepodge…
  - **surrounding words capture the context of the word *banking***

Word vectors: Learning with Neural Networks

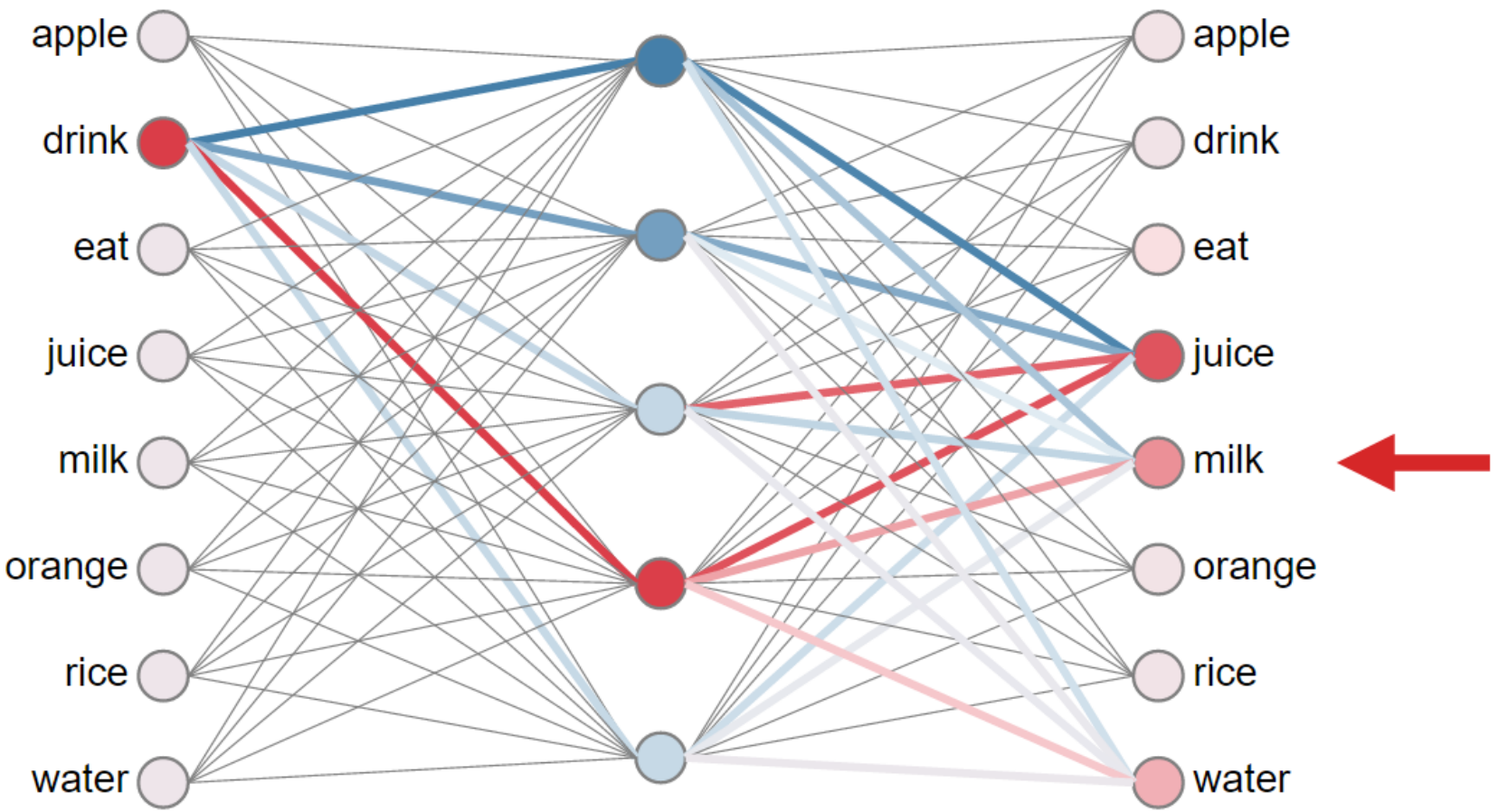
Neural Network-based Word Embedding Models

Google

Word2vec

facebook

FastText



IR Vector-Space Model

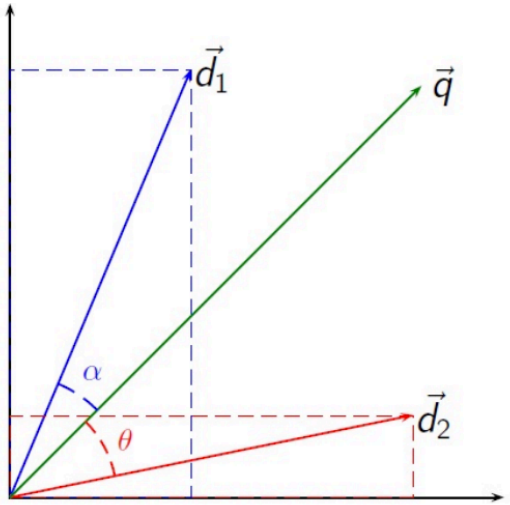
- An algebraic representation of text documents or queries as vectors

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$
$$q = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

- Cosine similarity

$$\|\mathbf{q}\| = \sqrt{\sum_{i=1}^{n} q_i^2}$$

$$\cos\theta = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\|\mathbf{d_2}\| \, \|\mathbf{q}\|}$$



$$\mathbf{sim}(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

One-Hot Encoding (Word Embedding)

- Given "`Can I eat the pizza`" of `N=5`
  1. Convert to lower case
  2. Sort the words in alphabetical order
  3. Give numerical labels to each word:
     `can:0, i:2, eat:1, the:4, pizza:3`
  4. Transform to binary vectors

```
[[1. 0. 0. 0. 0.] #can
 [0. 0. 1. 0. 0.] #i
 [0. 1. 0. 0. 0.] #eat
 [0. 0. 0. 0. 1.] #the
 [0. 0. 0. 1. 0.]] #pizza
```

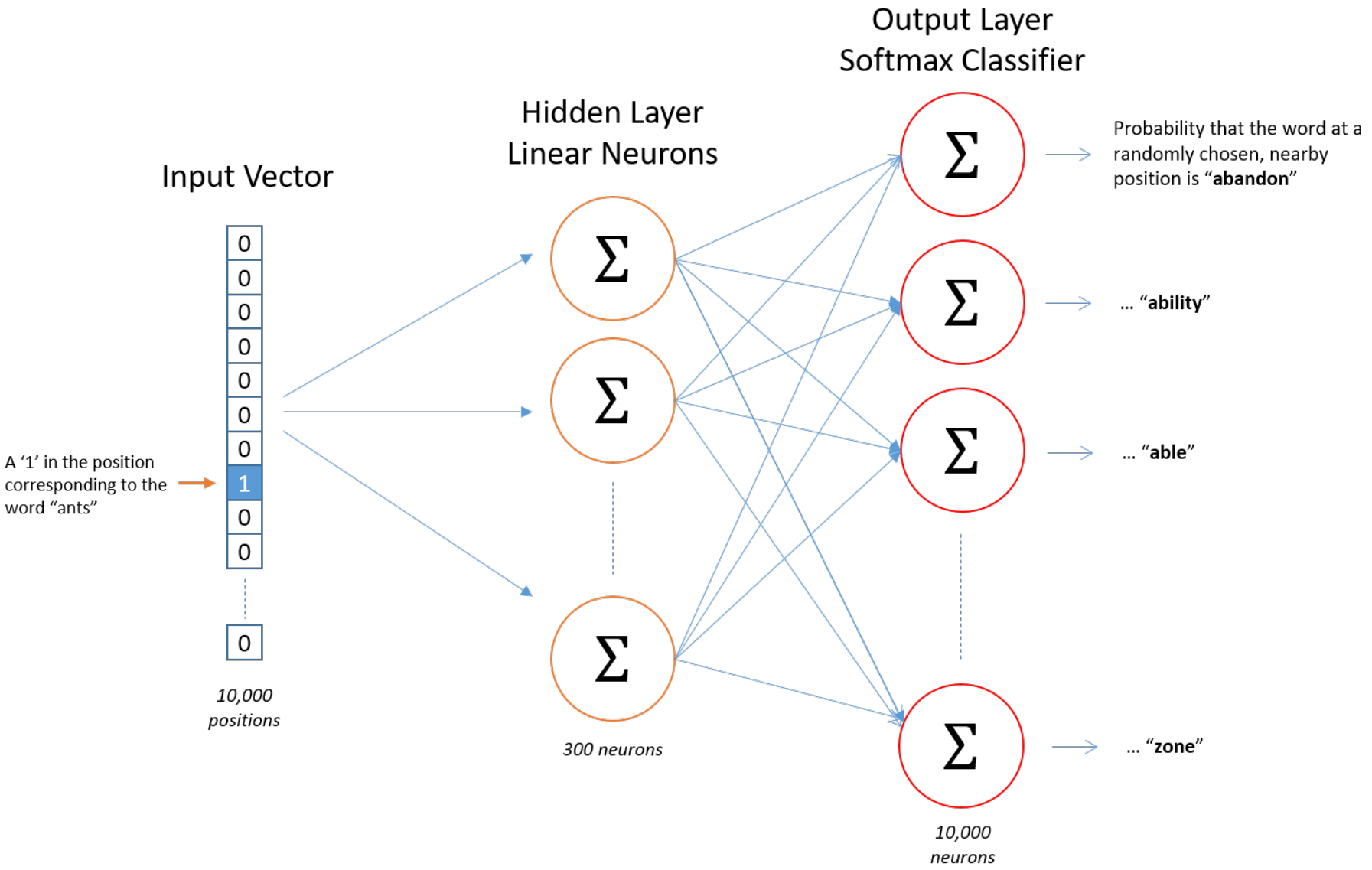
Co-occurrence Matrix

Corpus = {"I like deep learning"
"I like NLP"
"I enjoy flying"}

Context = previous word and next word

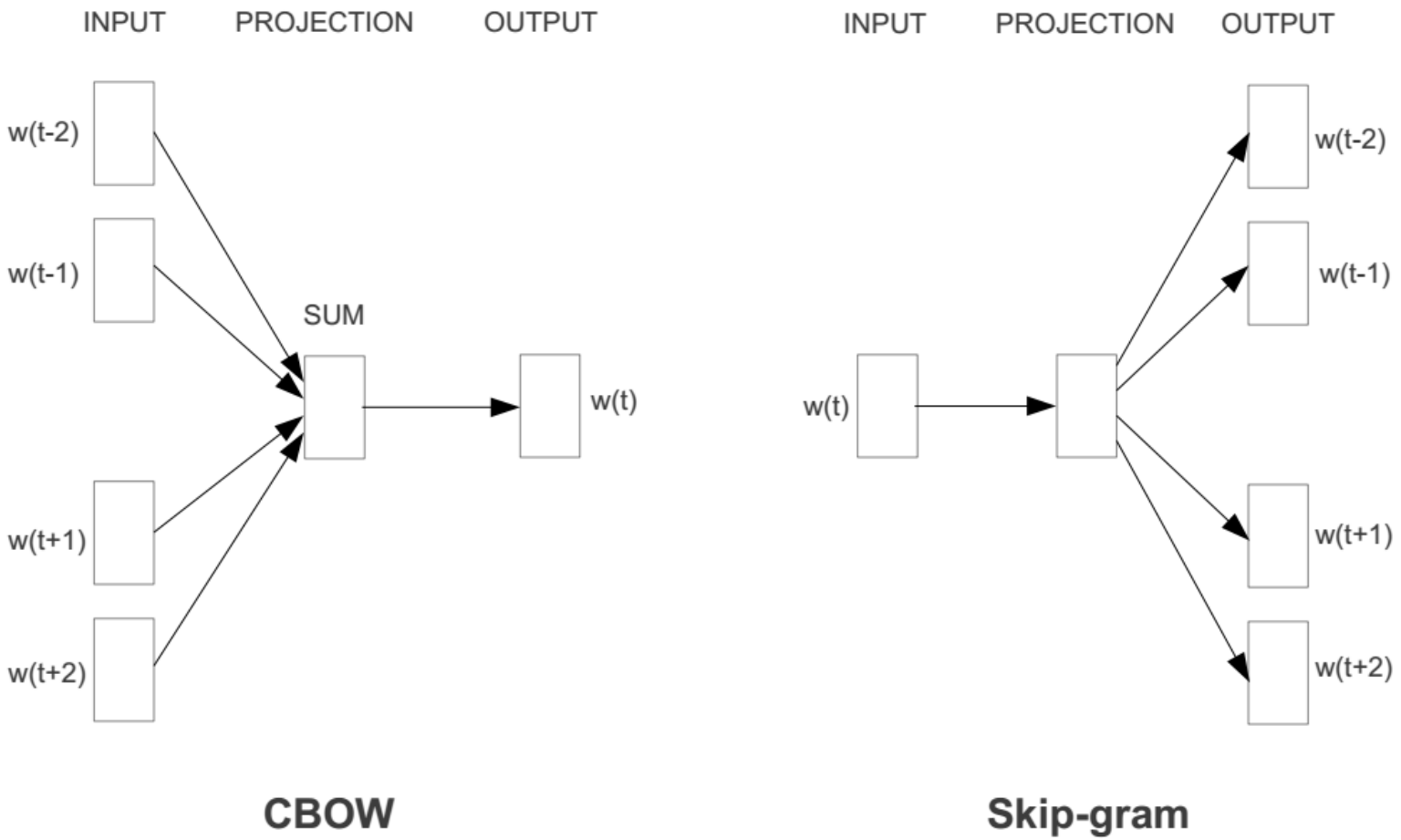
counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Word2Vec Neural Network Architecture



Word2vec

- Represent the meaning of the word by its context



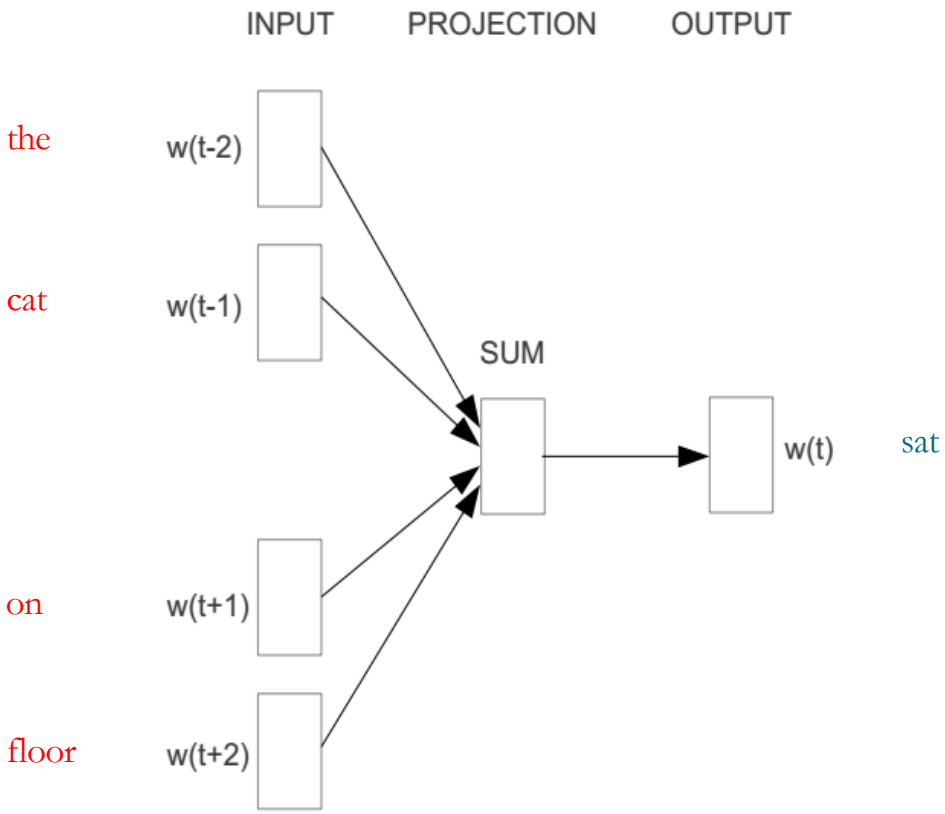
Word2Vec Learning models

- Continuous Bag of Words (CBOW):
  - Given the context predict the word:
    - $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
  - Example: **The cat ate** _____.
    - Fill in the blank, e.g. "food".
  - Faster to train
  - Works well for large amount of training data

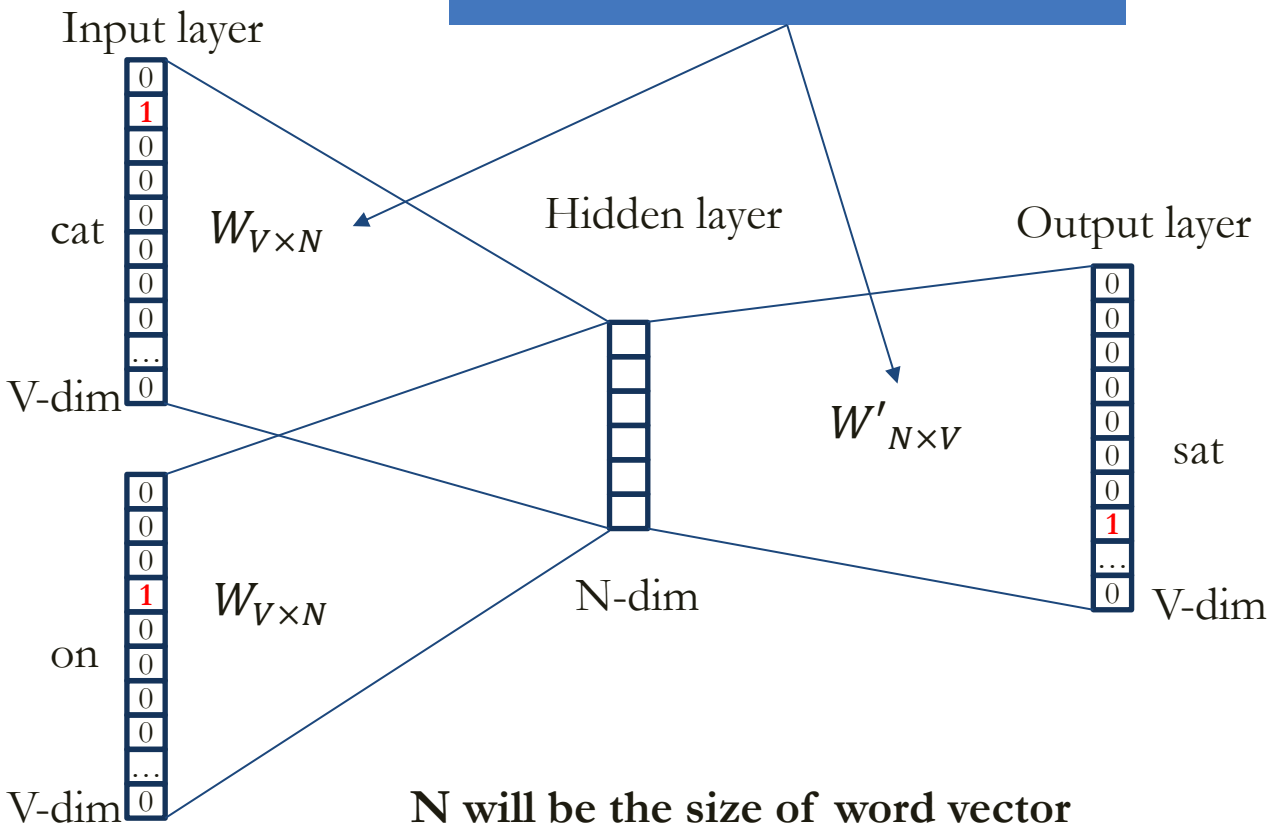
- Continuous Skip-Gram:
  - Given the word predict the context:
    - $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
  - Ex: ___ ___ ___ food.
    - Fill in the blank, e.g. "The cat ate"
  - Slower to train
  - Works better for infrequent words

Word2Vec: CBOW Example

- ”the cat ___ on floor”



We must learn W and W’



Learning Connection Weights