Word2vec embeddings: CBOW and Skipgram

VL Embeddings

Uni Heidelberg

SS 2019

Representation of word meaning for NLP

How can we represent word meaning?

- Lexical resources
  - Dictionaries
  - WordNet, GermaNet
  - FrameNet
  - ...

Problems with lexical resources

Problems with lexical resources

- Creation is time-consuming and costly
- Who decides what to include in the resource?
  $\rightarrow$ subjective
- Probably not complete
  - rare words, new terminology, slang
- Hard to compute accurate word similarity

Word representations for NLP

- Most work in rule-based and statistical NLP regards words as atomic symbols
  - one-hot encoding: sparse vector with one 1 and many zeros
    [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
  - Dimensionality: vocabulary size
    e.g.: 20K (speech) – 50K (PTB) – 500K (large corpus) – ...

    Hotel [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . 0 0 0 1 0 0 0]
    Motel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0]
- Cannot compute similarity between those representations

Solution: Representations based on distributional similarity

"You whall know a word by the company it keeps"

(J.R.Firth, 1957)

Die Bewegung der Maus, ausgeführt mit der Hand auf dem Arbeitsplatz ...
Kommerziell verwendet wurde die Maus 1981 im Rechner Xerox Star, ...
Die Maus (Mus) gehört zur Nagetiergattung aus der Gruppe der Altweltmäuse ...
Die Gattung umfasst knapp 40 Arten, von denen die Hausmaus die bekannteste ...

- Represent each word by its context (1st or 2nd order)

Example: Window-based co-occurrence matrix

- Window length 1 (more common: 5 - 10)
- Symmetric (not sensitive to position: left or right context)
- Toy corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

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

Problems with co-occurrence vectors

- Vectors are **high-dimensional** (require lots of storage)
- Vectors are very **sparse**
- When used for classification
  - Sparsity issues $\rightarrow$ models not robust

Solution: low-dimensional word vectors

- Build a dense representation for each word
- Only store the "most important" information in a fixed-size vector of lower dimensionality (usually 25 – 1000 dimensions)
- Words appearing in similar contexts should have similar vectors

$$\text{Rauhaardackel} = \begin{bmatrix} 0.348 \\ 0.758 \\ 0.399 \\ 0.183 \\ -0.252 \\ -0.234 \\ 0.434 \\ -0.215 \end{bmatrix}$$

- Terminology: Word vectors, word embeddings, word representations

How can we reduce the dimensionality?

- Well-known solution: Singular Value Decomposition (SVD)

  $\Rightarrow$ count-based (count co-occurrences of words in the corpus)

- Problems with SVD
  - computationally expensive (for *nxm* matrix: $O(mn^2)$)
    $\rightarrow$ bad for large corpora
  - hard to incorporate new words or documents

Idea: learn low-dimensional vectors directly

- Instead of capturing cooccurrence counts directly, predict the surrounding words for every word in the corpus

- Advantages:
  - faster
  - can easily incorporate new sentences/documents
  - can easily add new words to the vocabulary

- Not that new:
  - Rumelhart et al. (1986): Learning representations by back-propagating errors
  - Bengio et al. (2003): A neural probabilistic language model
  - Collobert and Weston (2008): NLP (almost) from scratch

  - Recently: Mikolov et al. (2013): Efficient Estimation of Word Representations in Vector Space

Neural Networks in a nutshell

From Perceptrons to Dense Neural Networks

- Basic introduction of the
  - Perceptron algorithm
  - Multi-Layer Perceptron (MLP)

The Perceptron algorithm

Rosenblatt (1958):

- supervised, binary linear classifier
  - learns a separating hyperplane
  - using the normal vector w of the plane, a vector x is classified as being above or below the plane, based on the dot product between x and w:

$$\hat{y} = w \cdot x$$

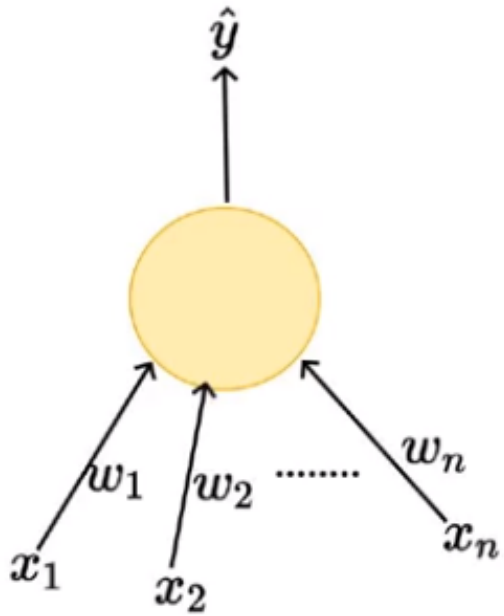
  - $\hat{y} > 0 \Rightarrow$ vector lies above the hyperplane
  - $\hat{y} < 0 \Rightarrow$ vector lies below the hyperplane

- Problem: hyperplane can only rotate around the origin

  $\Rightarrow$ add bias term b $\quad \boxed{\hat{y} = w \cdot x + b}$

- Bias: threshold that the dot product of w and x has to reach to be classified as 1 or 0

The Perceptron algorithm



$$\boxed{\hat{y} = w \cdot x + b}$$

- $\hat{y} = 1$ if $\sum_{i=1}^{n} w_i x_i \geq b$
- $\hat{y} = 0$ otherwise

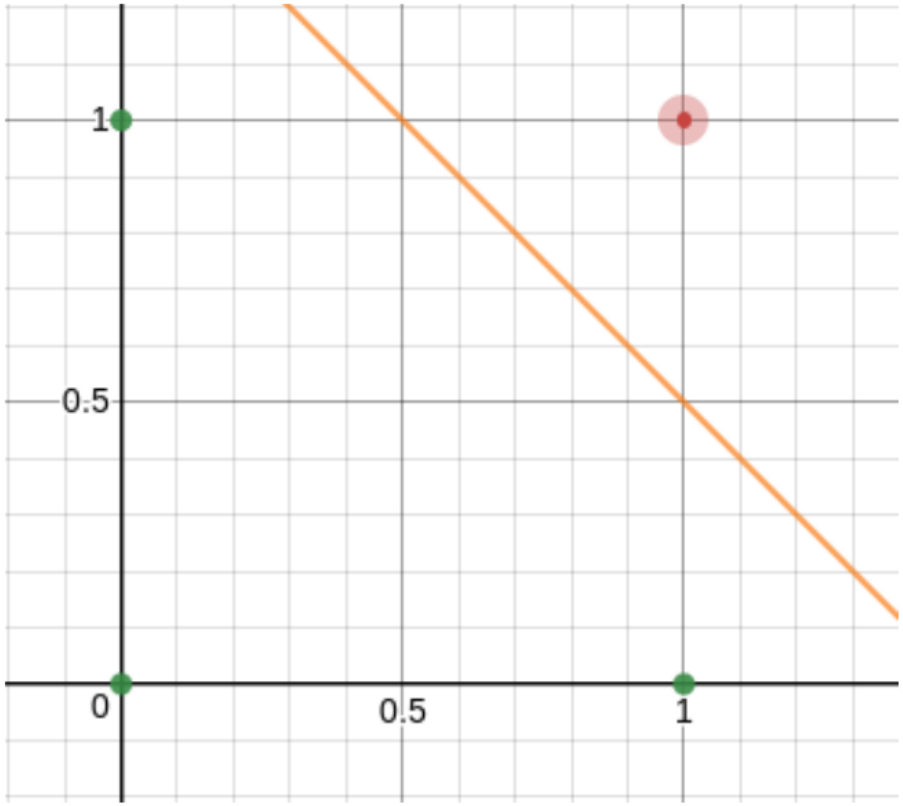
Image taken from `https://hackernoon.com/perceptron-deep-learning-basics-3a938c5f84b6`

Solving the AND problem with the perceptron

Solving the AND problem with the perceptron

x	y	$w \cdot x + b$
$\begin{pmatrix}0\\0\end{pmatrix}$	-1	-0.7
$\begin{pmatrix}0\\1\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\0\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\1\end{pmatrix}$	1	0.3

(Perceptron predictions with $w = (0.5, 0.5)$ and $b = -0.7$)

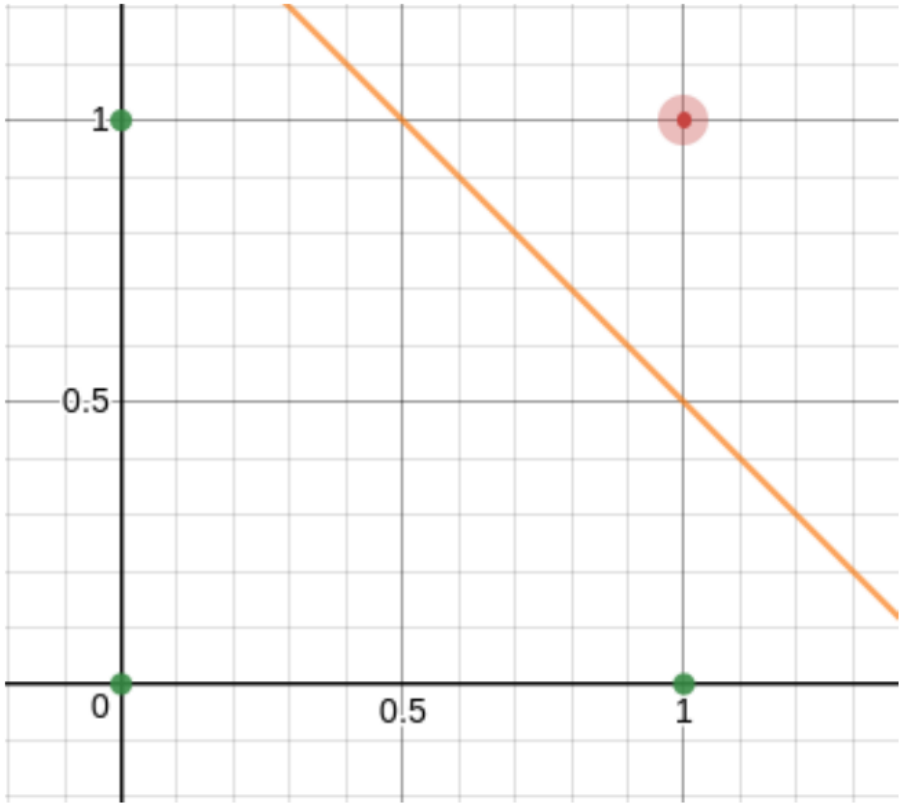


(Images from Michael Staniek)

Solving the AND problem with the perceptron

x	y	$w \cdot x + b$
$\begin{pmatrix}0\\0\end{pmatrix}$	-1	-0.7
$\begin{pmatrix}0\\1\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\0\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\1\end{pmatrix}$	1	0.3

(Perceptron predictions with $w = (0.5, 0.5)$ and $b = -0.7$)



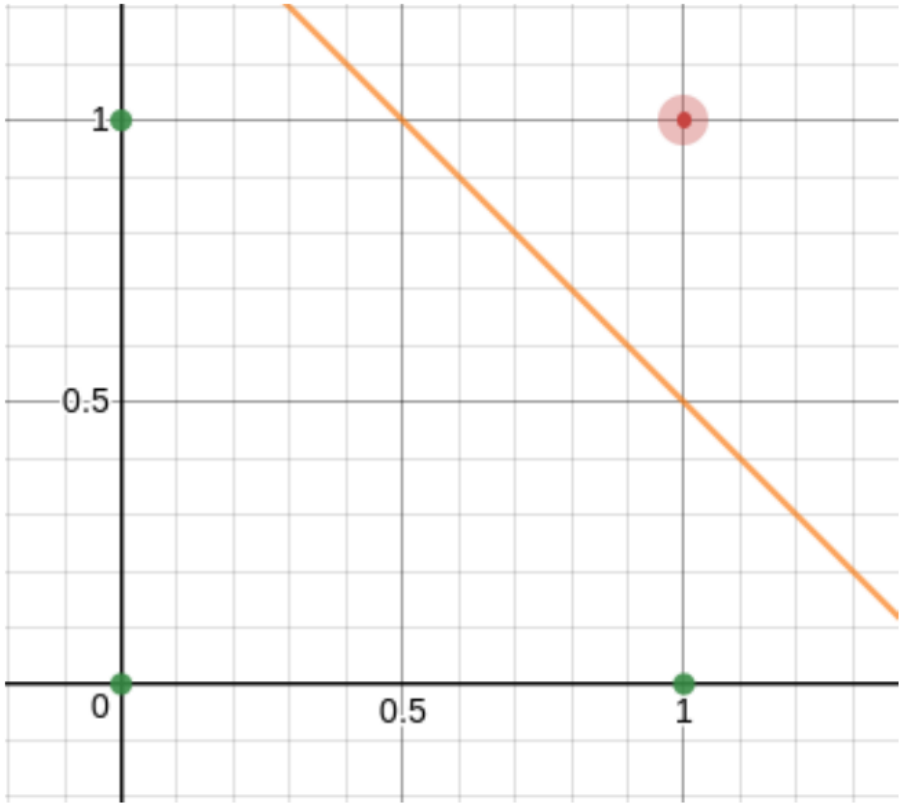
(Images from Michael Staniek)

$$w \cdot x + b = (0.5, 0.5) \cdot (0, 0) + -0.7 = 0 + 0 - 0.7 = -0.7 \Rightarrow -1$$

Solving the AND problem with the perceptron

x	y	$w \cdot x + b$
$\begin{pmatrix}0\\0\end{pmatrix}$	-1	-0.7
$\begin{pmatrix}0\\1\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\0\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\1\end{pmatrix}$	1	0.3

(Perceptron predictions with $w = (0.5, 0.5)$ and $b = -0.7$)



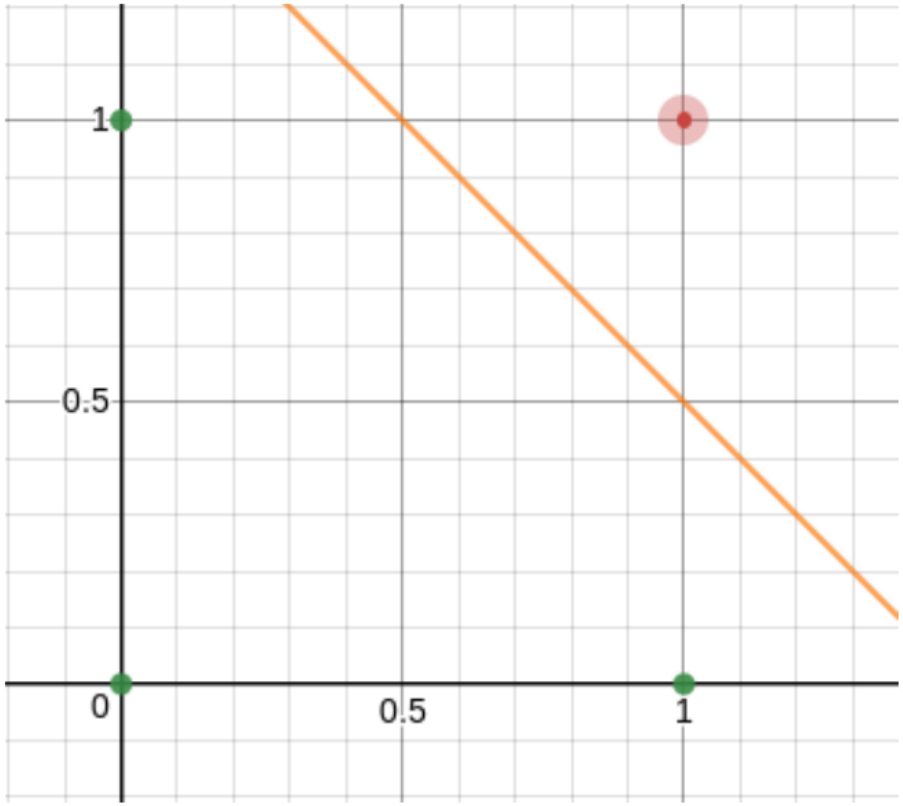
(Images from Michael Staniek)

$$w \cdot x + b = (0.5, 0.5) \cdot (0, 0) + -0.7 = 0 + 0 - 0.7 = -0.7 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (0, 1) + -0.7 = 0 + 0.5 - 0.7 = -0.2 \Rightarrow -1$$

Solving the AND problem with the perceptron

x	y	$w \cdot x + b$
$\begin{pmatrix}0\\0\end{pmatrix}$	-1	-0.7
$\begin{pmatrix}0\\1\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\0\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\1\end{pmatrix}$	1	0.3

(Perceptron predictions with $w = (0.5, 0.5)$ and $b = -0.7$)



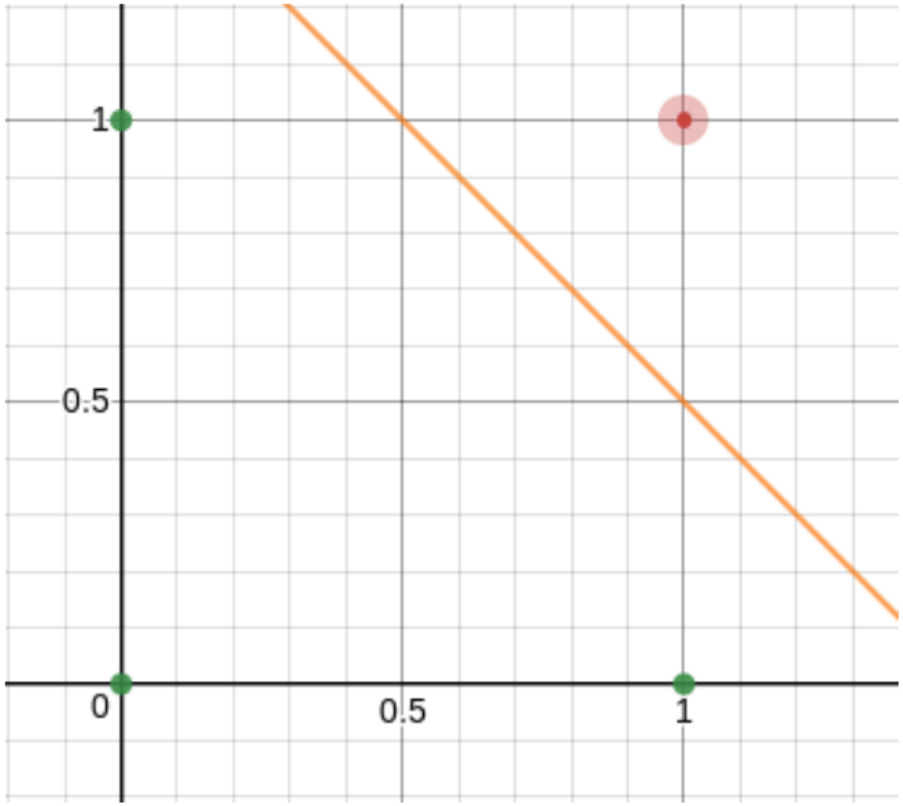
(Images from Michael Staniek)

$$w \cdot x + b = (0.5, 0.5) \cdot (0, 0) + -0.7 = 0 + 0 - 0.7 = -0.7 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (0, 1) + -0.7 = 0 + 0.5 - 0.7 = -0.2 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (1, 0) + -0.7 = 0.5 + 0 - 0.7 = -0.2 \Rightarrow -1$$

Solving the AND problem with the perceptron

x	y	$w \cdot x + b$
$\begin{pmatrix}0\\0\end{pmatrix}$	-1	-0.7
$\begin{pmatrix}0\\1\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\0\end{pmatrix}$	-1	-0.2
$\begin{pmatrix}1\\1\end{pmatrix}$	1	0.3

(Perceptron predictions with $w = (0.5, 0.5)$ and $b = -0.7$)



(Images from Michael Staniek)

$$w \cdot x + b = (0.5, 0.5) \cdot (0, 0) + -0.7 = 0 + 0 - 0.7 = -0.7 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (0, 1) + -0.7 = 0 + 0.5 - 0.7 = -0.2 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (1, 0) + -0.7 = 0.5 + 0 - 0.7 = -0.2 \Rightarrow -1$$
$$w \cdot x + b = (0.5, 0.5) \cdot (1, 1) + -0.7 = 0.5 + 0.5 - 0.7 = 0.3 \Rightarrow 1$$

Training and testing the perceptron

Algorithm 5 PERCEPTRONTRAIN(**D**, *MaxIter*)

$w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \ldots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \ldots, w_D, b$

(From Hal Daumé III (2017): A course in machine learning `http://ciml.info`)

Training and testing the perceptron

Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}$, $MaxIter$)

$w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \ldots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \ldots, w_D, b$

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \ldots, w_D, b, \hat{x}$)

$a \leftarrow \sum_{d=1}^{D} w_d \hat{x}_d + b$ // compute activation for the test example
return SIGN(a)

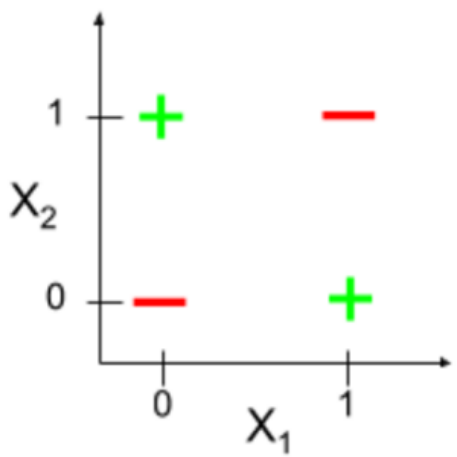
(From Hal Daumé III (2017): A course in machine learning `http://ciml.info`)

Perceptron – Sum-up

- Advantages
  - simple algorithm
  - online learning $\Rightarrow$ considers one input at a time
  - error-driven $\Rightarrow$ update only if there is an error
- Disadvantages
  - linearly separable problems only
    (cannot solve the XOR problem)

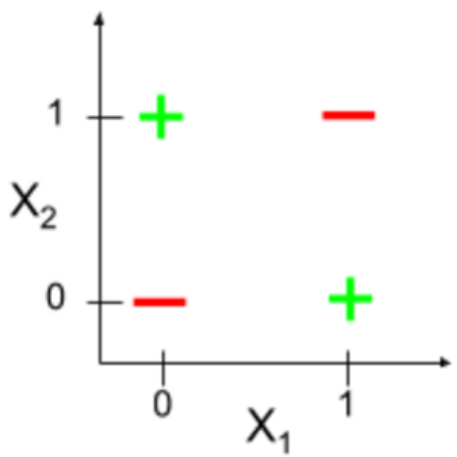
Perceptron – Sum-up

- Advantages
  - simple algorithm
  - online learning ⇒ considers one input at a time
  - error-driven ⇒ update only if there is an error
- Disadvantages
  - linearly separable problems only
    (cannot solve the XOR problem)



Perceptron – Sum-up

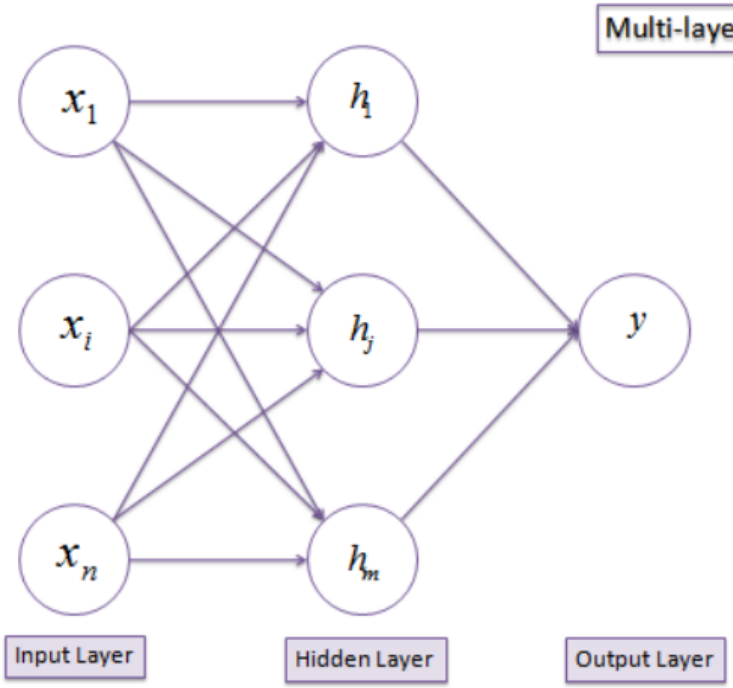
- Advantages
  - simple algorithm
  - online learning ⇒ considers one input at a time
  - error-driven ⇒ update only if there is an error
- Disadvantages
  - linearly separable problems only
    (cannot solve the XOR problem)

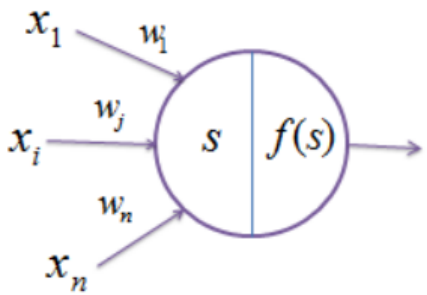


Solution: Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP)

- Stack multiple perceptrons on top of each other ⇒ MLP
  - each node in a layer is connected to every node in the previous layer
  - outputs of previous layer are inputs of next layer
  - output of every neuron is transformed with a nonlinear activation function (e.g. sigmoid, tanh, ReLu, ...)





Summation

$$s = \sum w \cdot x$$

Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$

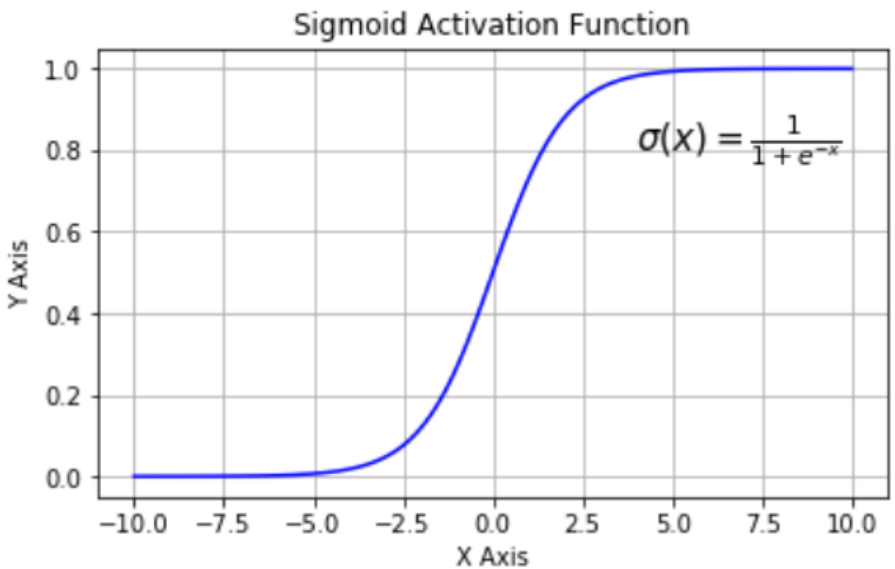
(Images from Michael Staniek)

Non-linear activation functions: Sigmoid

Sigmoid or Logistic Activation Function

- takes a real-valued number and squashes it into a range between 0 and 1
- used to transform scores into probabilities
- range: 0, 1

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

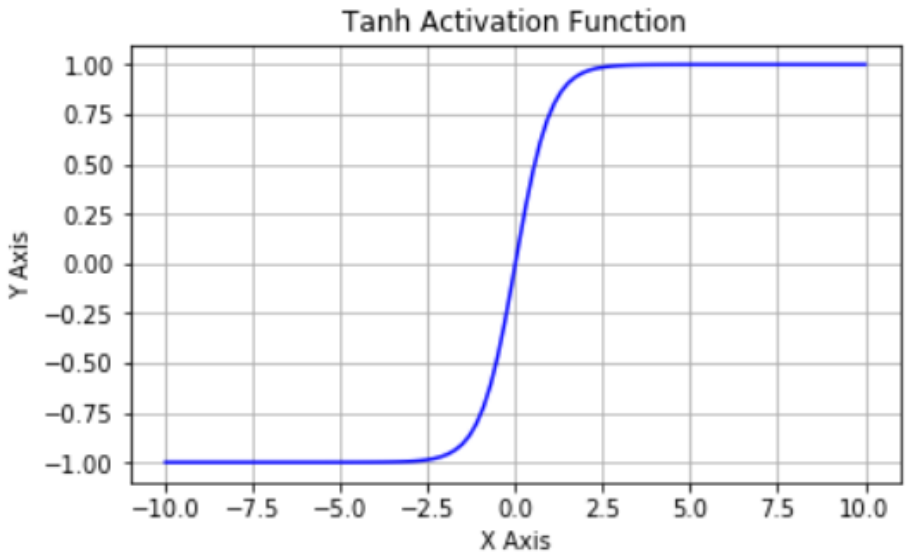


Non-linear activation functions: Tanh and ReLU

Tanh Function

- range: -1, 1

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

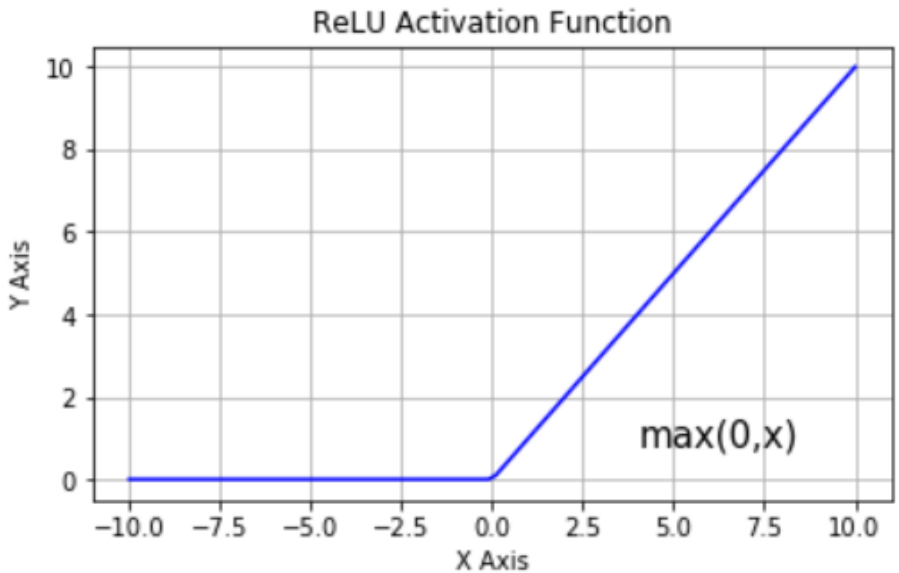


Non-linear activation functions: Tanh and ReLU

Rectified Linear Unit (ReLU) Function

- if input $x < 0$ then output $= 0$
- if input $x > 0$ then output $= x$

$$f(x) = max(0, x)$$



Multi-Layer Perceptron – Sum-up

- MLP: type of feedforward neural network
  (also called dense neural network or vanilla neural network)
  - has at least 3 layers (input, hidden, output)
  - each node is a neuron that uses a non-linear activation function
  - MLPs are usually trained in a supervised fashion using backpropagation
- Can learn non-linear decision boundaries

Multi-Layer Perceptron – Sum-up

- MLP: type of feedforward neural network
  (also called dense neural network or vanilla neural network)
  - has at least 3 layers (input, hidden, output)
  - each node is a neuron that uses a non-linear activation function
  - MLPs are usually trained in a supervised fashion using backpropagation
- Can learn non-linear decision boundaries

- Word2vec architecture
  - simple neural network with 1 hidden layer

Word2vec

Mikolov, Chen, Corrado & Dean (2013): Efficient Estimation of Word Representations in Vector Space

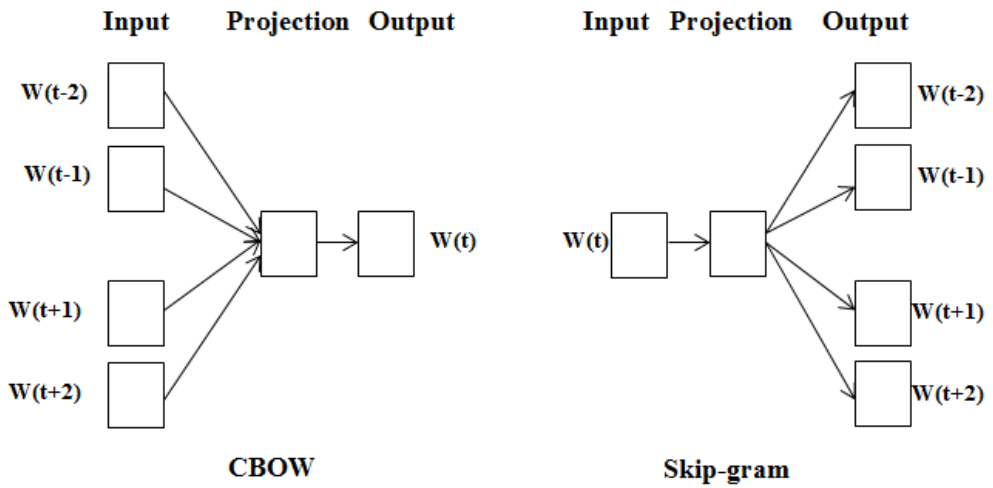
- Comes in two flavors:
  - Continuous Bag Of Words (CBOW)
  - Skipgram
- Given a corpus,
  - iterate over all words in the corpus and either
    - use context words to predict current word (CBOW), or
    - use current word to predict context words (Skipgram)

Word2vec

Mikolov, Chen, Corrado & Dean (2013): Efficient Estimation of Word Representations in Vector Space

- Comes in two flavors:
  - Continuous Bag Of Words (CBOW)
  - Skipgram

- Given a corpus,
  - iterate over all words in the corpus and either
    - use context words to predict current word (CBOW), or
    - use current word to predict context words (Skipgram)



Word2vec – Intuition

Predict the next word...

Deine blauen _____

Word2vec – Intuition

Predict the next word...

Deine blauen Augen

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so _____

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so sentimental

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so sentimental

Die Labour-Hochburgen im Norden haben mehrheitlich für den Brexit _____

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so sentimental

Die Labour-Hochburgen im Norden haben mehrheitlich für den Brexit votiert

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so sentimental

Die Labour-Hochburgen im Norden haben mehrheitlich für den Brexit votiert

Kaum zu _____

Word2vec – Intuition

Predict the next word...

Deine blauen Augen machen mich so sentimental

Die Labour-Hochburgen im Norden haben mehrheitlich für den Brexit votiert

Kaum zu glauben

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

- center word at position $t = 5$
- window size $= 2$ (2 words to the left and to the right)

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

- center word at position $t = 5$
- window size $= 2$ (2 words to the left and to the right)

1. Iterate over all words in the corpus

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

- center word at position $t = 5$
- window size $= 2$ (2 words to the left and to the right)

1. Iterate over all words in the corpus
2. For each word at position t, consider w_t as the center word and all words in context of size n around w_t as the context

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

- center word at position $t = 5$
- window size $= 2$ (2 words to the left and to the right)

1. Iterate over all words in the corpus
2. For each word at position t, consider w_t as the center word and all words in context of size n around w_t as the context
3. For each word in the vocabulary, compute the probability that this word is a context word of w_t
   $\Rightarrow$ How likely is it to find w_v in the context of w_t?

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

- center word at position $t = 5$
- window size $= 2$ (2 words to the left and to the right)

1. Iterate over all words in the corpus
2. For each word at position t, consider w_t as the center word and all words in context of size n around w_t as the context
3. For each word in the vocabulary, compute the probability that this word is a context word of w_t
   $\Rightarrow$ How likely is it to find w_v in the context of w_t?
4. Update parameters of the model to maximise this probability

Skipgram – Intuition

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

Given the center word Maus, which words in the vocabulary are more likely to occur in near context?

Vocab:

klein	lila
grau	Eiskunstlauf
Käse	schwimmen
Katze	Europa

Skipgram – Intuition

Corpus

Die kleine graue Maus frißt den leckeren Käse

Train samples

Skipgram – Intuition

Corpus								**Train samples**	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)

Skipgram – Intuition

Corpus								Train samples	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse		

Skipgram – Intuition

Corpus

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
Die	kleine	graue	Maus	frißt	den	leckeren	Käse

Train samples

(Die, kleine)
(Die, graue)
(kleine, Die)
(kleine, graue)
(kleine, Maus)

Skipgram – Intuition

Corpus								Train samples	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(kleine,	Die)
								(kleine,	graue)
								(kleine,	Maus)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse		

Skipgram – Intuition

Corpus								Train samples	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(kleine,	Die)
								(kleine,	graue)
								(kleine,	Maus)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(graue,	Die)
								(graue,	kleine)
								(graue,	Maus)
								(graue,	frißt)

Skipgram – Intuition

Corpus

Die	kleine	graue	Maus	frißt	den	leckeren	Käse
Die	kleine	graue	Maus	frißt	den	leckeren	Käse
Die	kleine	graue	Maus	frißt	den	leckeren	Käse
Die	kleine	graue	Maus	frißt	den	leckeren	Käse

Train samples

(Die, kleine)
(Die, graue)
(kleine, Die)
(kleine, graue)
(kleine, Maus)
(graue, Die)
(graue, kleine)
(graue, Maus)
(graue, frißt)

Skipgram – Intuition

Corpus								Train samples	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(kleine,	Die)
								(kleine,	graue)
								(kleine,	Maus)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(graue,	Die)
								(graue,	kleine)
								(graue,	Maus)
								(graue,	frißt)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Maus,	kleine)
								(Maus,	graue)
								(Maus,	frißt)
								(Maus,	den)

- Input: word pairs found in the corpus

Skipgram – Intuition

Corpus								Train samples	
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Die,	kleine)
								(Die,	graue)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(kleine,	Die)
								(kleine,	graue)
								(kleine,	Maus)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(graue,	Die)
								(graue,	kleine)
								(graue,	Maus)
								(graue,	frißt)
Die	kleine	graue	Maus	frißt	den	leckeren	Käse	(Maus,	kleine)
								(Maus,	graue)
								(Maus,	frißt)
								(Maus,	den)

- Input: word pairs found in the corpus
  More instances of (Maus, klein) or (Maus, frißt) than (Maus, Museum)... ⇒ higher output probabilities

(http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model)