Deep Learning for Natural Language Processing

Universität Bielefeld

Lecture 1 – Kick-off

(Prof.) Dr. Steffen Eger

seminar.dldh@gmail.com

CITEC NLLG

Outline

Universität Bielefeld

Administrative course issues

NLLG, CITEC

Deep Learning for NLP

Learning Goals

Universität Bielefeld

After completing this course, you are able to

- explain the basic concepts of neural networks and deep learning
- explain the concept of word embeddings, train word embeddings and use them for solving NLP problems
- understand and describe neural network architectures used to tackle classical NLP problems such as text/sentence classification and sequence tagging
- implement neural networks for NLP problems using existing libraries in Python

CITEC NLLG

Learning Goals

Universität Bielefeld

After completing this course, you are able to

- explain the basic concepts of neural networks and deep learning
- explain the concept of word embeddings, train word embeddings and use them for solving NLP problems.
- understand and describe neural network architectures used to tackle classical NLP problems such as text/sentence classification and sequence tagging.
- implement neural networks for NLP problems using existing libraries in Python.

CITEC NLLG

General Information

Universität Bielefeld

Teaching material:

Lectures, exercises/submissions etc. can be found in “LernraumPlus” and on my github

https://github.com/SteffenEger/dl4nlp

CITEC NLLG

Resources

- This lecture is mainly based on the latest papers from (top) NLP conferences
  - Deep Learning is changing too quickly to base it on text books
    - Knowledge is too quickly outdated
- The lecture is in English

Useful Additional Resources

- Stanford Lecture by Richard Socher : Deep Learning for Natural Language Processing cs224d – look it up on youtube

- Stanford Lecture by Andrej Karpathy, cs231n – look it up on youtube

- Ian Goodfellow, Yoshua Bengio, and Aaron Courville:

  Deep Learning, MIT Press

Recommended Readings

Universität Bielefeld

- We provide literature for the topics of the course:
  - You should at least be aware of the literature relevant to your topic (see below)
  - We encourage you to read other referenced work as well
- We can give you additional literature hints → feel free to ask
- If you cannot find/access a publication → feel free to ask

Practice/Seminar Class

Universität Bielefeld

- **Every Tuesday 08:15 – 09:45, Zoom**
  - organized by me and Jonas Belouadi
- The exercises will give you some practical experience and hands-on training of what you learned
  - You will learn to program neural nets, something that we don't do/teach in the lecture
- You will choose a topic to work on in the practice class

CITEC NLLG

Grading

Universität Bielefeld

- In the tutorial, you will choose a topic to work on
- You can form groups of 1 to N students
  - N depends on students interested in participation
- In the last third of the tutorial, you will present your on-going work
- Finally, you write a report (8-10 pages) on your work
- Grading: 0.8 x term paper + 0.2 x presentation

CITEC NLLG

Programming Framework:

Universität Bielefeld

- Python, Numpy
- Tensorflow
- Keras

→ more information next Tuesday

Tutors

Universität Bielefeld

- Jonas Belouadi
  - „Sprechstunde“ on demand

- If you have questions, you can contact us via LernraumPlus

Sprechstunde

- Sprechstunde every Thursday from 15:30-17:00
  - Can discuss your ongoing projects or other DL4NLP questions
  - Please contact me at least one day early to be sure I am available
  - steffen.eger@uni-bielefeld.de

Participate!

Universität Bielefeld

- We want to encourage you to participate in this course
  - Attend lecture and practice class whenever possible!
  - Discuss with others in the forum!
  - Think beyond what we communicate in class!
  - Try out something new!
  - Read!
  - Ask!

CITEC NLLG

Participate!

- We want to encourage you to participate in this course
  - Attend lecture and practice class whenever possible!
  - Discuss with others in the forum!
  - Think beyond what we communicate in class!
  - Try out something new!
  - Read!
  - Ask!

- **Why? You get a much deeper understanding, are better prepared for a thesis, your project+term paper and future job!**

Questions/Suggestions

Any questions, suggestions,…?

1. Post a message in the LernraumPlus forum
2. *If that doesn‘t help:* Write an email to me or Jonas Belouadi (emails will be announced or can be found on the net)

2633

Oliver Lieske 12119

Medientechnik, remote

Outline

Administrative course issues

NLLG, CITEC

Deep Learning for NLP

Who am I? – Steffen Eger

- PhD in economics (background in math, NLP)
- PostDocs 2014-2018
- Group leader 2018-2022
- Stand-In Professor 2022-2023

Who am I? – Steffen Eger

Universität Bielefeld

- Research interests:
  - Evaluation Metrics for Text Generation
  - NLP & Social Sciences
    - Biases
    - Solidarity
  - Digital Humanities
    - Language Change
    - Poetry Generation, etc.

Final Thesis @ NLLG (Natural Language Learning Group)

Universität Bielefeld

Completed & Ongoing Master Theses (selection):

- Explainability /Efficiency for Evaluation Metrics
- Cross-lingual Cross-temporal Text Summarization
- Social Solidarity with Muslims on Twitter

Completed & Ongoing Bachelor Theses (selection):

- Reproducibility for Evaluation Metrics
- Abstract-to-title generation
- Syntactic Language Change
- Negativity in Science

Interested in a thesis at NLLG? Contact me (Steffen Eger) directly!
Multiple open theses (DE or EN) on different topics.

What about CITEC?

- CITEC = Center for Cognitive Interaction Technology
- Groups:
  - Maschinelles Lernen
  - Linguistik (Klinische, Experimentelle Neurolinguistik)
  - Kognitive Systeme und soziale Interaktion
  - Theoretische Computerlinguistik
  - Semantische Datenbanken (**Philipp Cimiano**)
    - Text Mining
    - Semantics
    - Information Retrieval
    - Ontologies

Outline

Administrative course issues

NLLG, CITEC: profile and projects

Deep Learning for NLP

Scope of the Lecture

Universität Bielefeld

Deep learning for NLP:

- Exploring the use of deep learning techniques for natural language processing (NLP) tasks

Deep learning	+	NLP
Machine learning method		Analysis of language

CITEC NLLG

Scope of the Lecture

Universität Bielefeld

Deep learning for NLP:

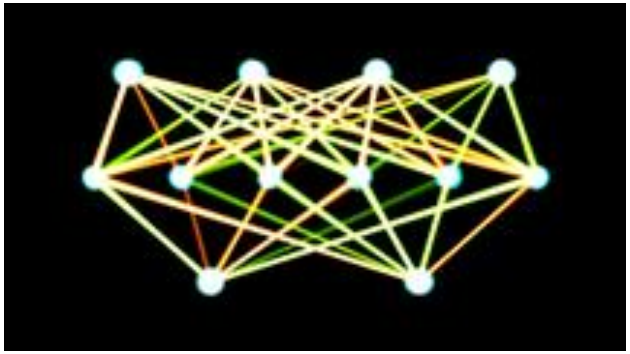
- Exploring the use of deep learning techniques for natural language processing (NLP) tasks

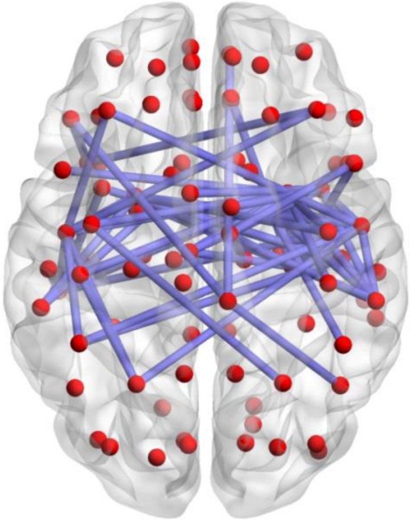
Deep learning	+	NLP
Machine learning method		Analysis of language

CITEC NLLG

Deep Learning (aka Neural Networks)

- Subfield of machine learning
- Neural networks: a brain-inspired metaphor for a computational model





https://upload.wikimedia.org/wikipedia/commons/0/0e/Brain_network.png
https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/Single-layer_feedforward_artificial_neural_network.png/214px-Single-layer_feedforward_artificial_neural_network.png

History of Deep Learning

- Early booming (1950s – early 1960s)
  - Rosenblatt (1958) , Perceptron
  - Widrow and Hoff (1960, 1962)
    - Learning rule based on gradient descent
- Setback (mid 1960s late 1970s)
  - Serious problems with perceptron model (Minsky’s book 1969)
  - Can’t even represent simple functions
- Renewed enthusiasm (1980s)
  - New techniques (Backpropagation for “deep nets”)

Deep Learning

Universität Bielefeld

- Out-of-fashion - again (1990s to mid 2000s)
  - Other techniques were considered superior and more understandable
    - Support Vector Machines, Integer linear programming, etc.
  - Played virtually no role in top NLP conferences

- Since mid 2000: huge progress for “deep learning”
  - Hinton and Salakhutdinov (2006): one can actually train deep nets
  - Since ~2013: Veritable hype in NLP:
    - Word embeddings, work of Mikolov et al.
    - 2018: BERT

Deep Learning

Universität Bielefeld

- Since mid 2000: huge progress for “deep learning”
- Why?
  - More data
  - Faster computers, better hardware (GPU)
  - Unsupervised pre-training
  - Better optimization techniques, better understanding of the approaches, …

The Big Guys in Deep Learning

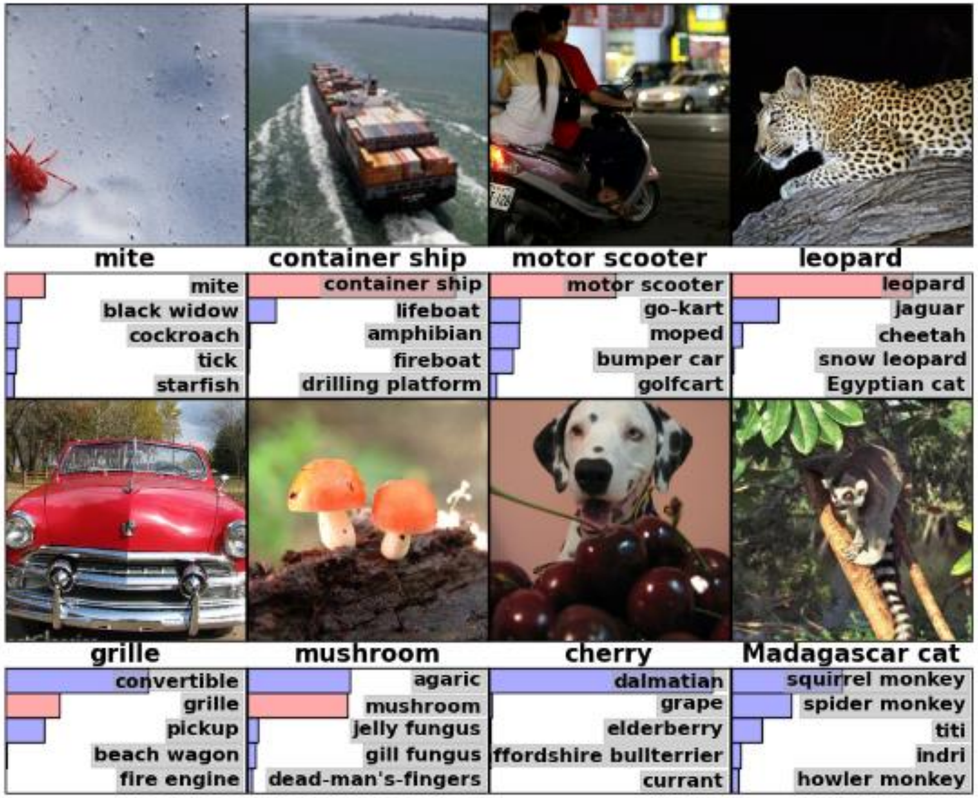
- In 2019, Hinton, LeCun, and Bengio won the Turing award in computer science
- Schmidhuber, another big guy in the field (LSTMs), didn‘t

DL Example: Object Recognition

Universität Bielefeld

- *ImageNetClassification With Deep Convolutional Neural Networks* Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, NIPS 2012

Figure 4:

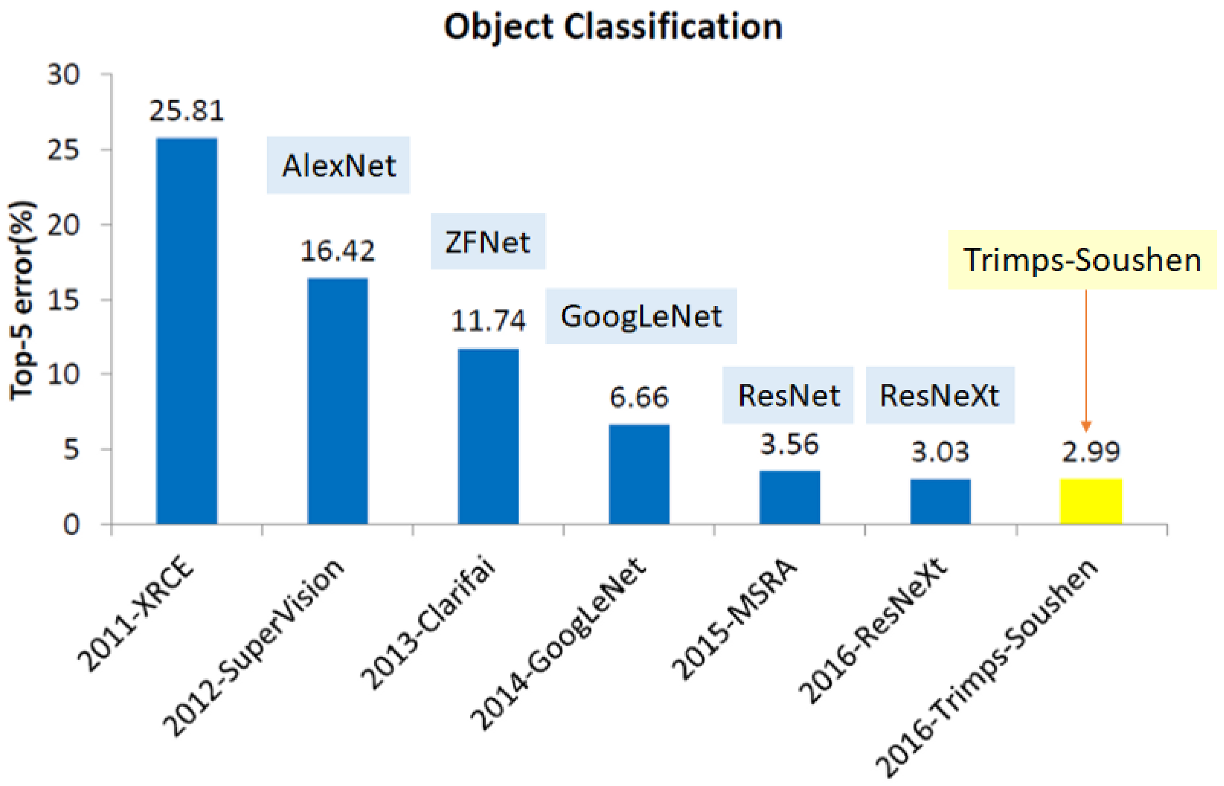


CITEC NLLG

DL Example: Object Recognition

Universität Bielefeld

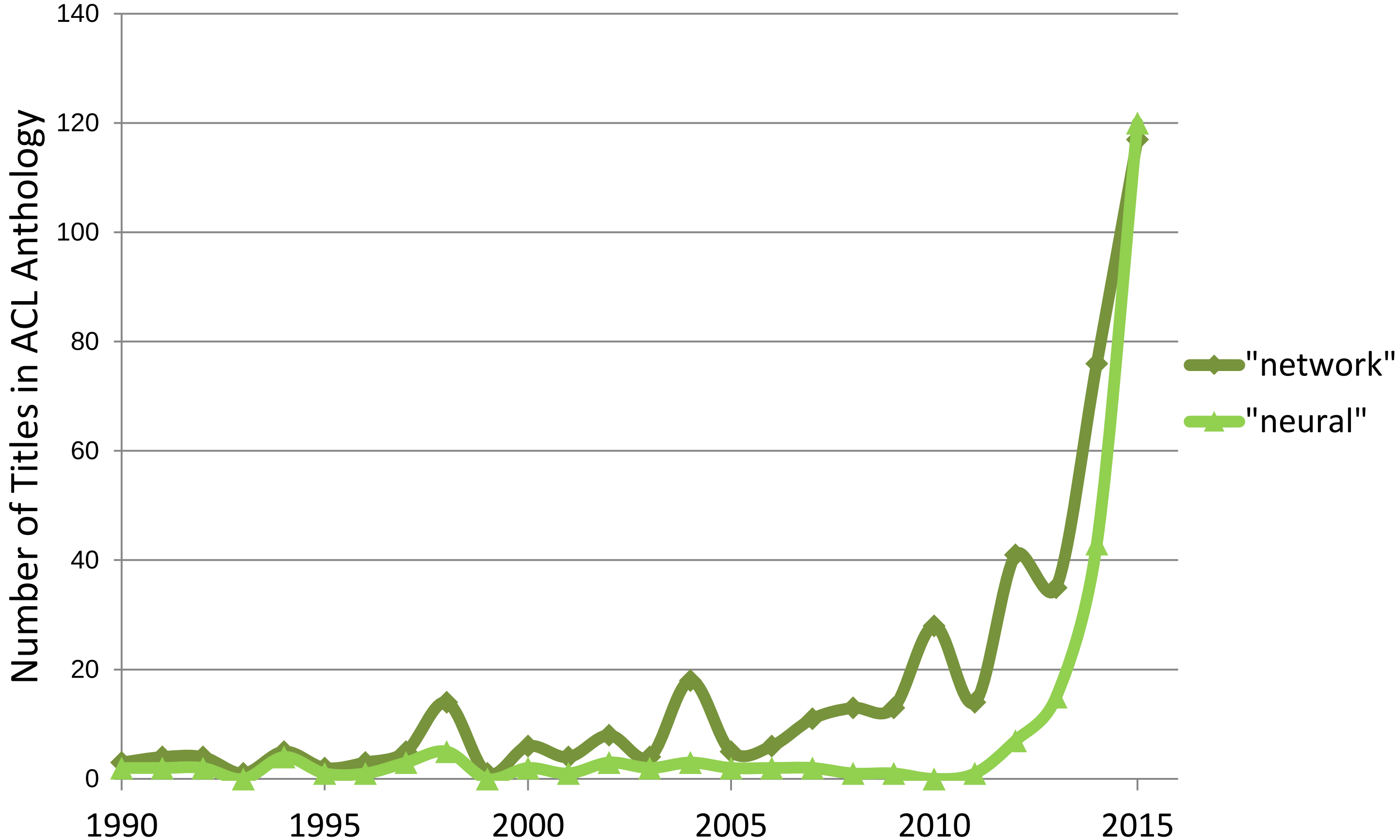
- Error rates:



And in natural language processing?

Universität Bielefeld

- Occurrences of “neural” and “network” in titles in the ACL Anthology

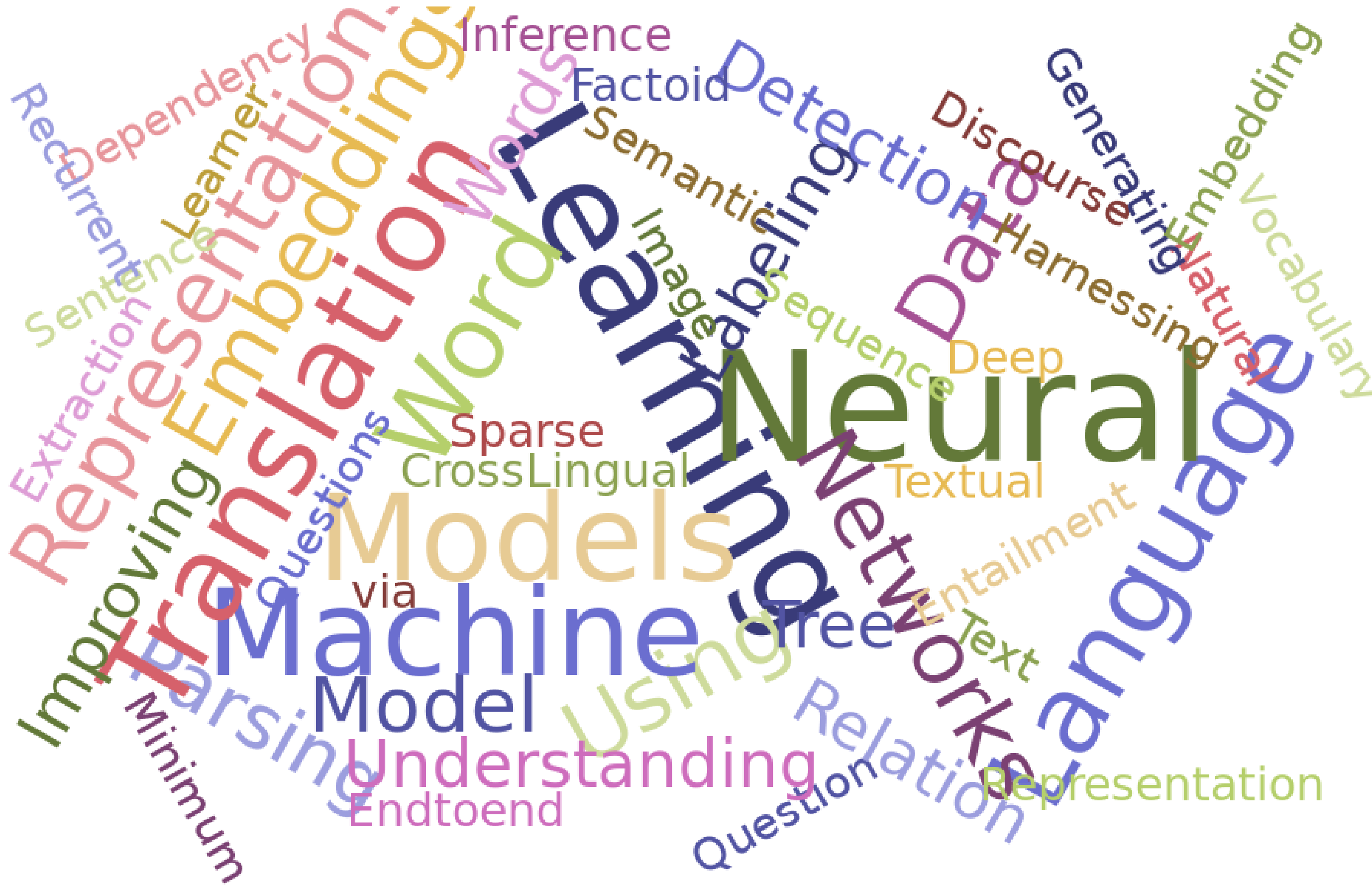


CITEC NLLG

And in natural language processing?

Universität Bielefeld

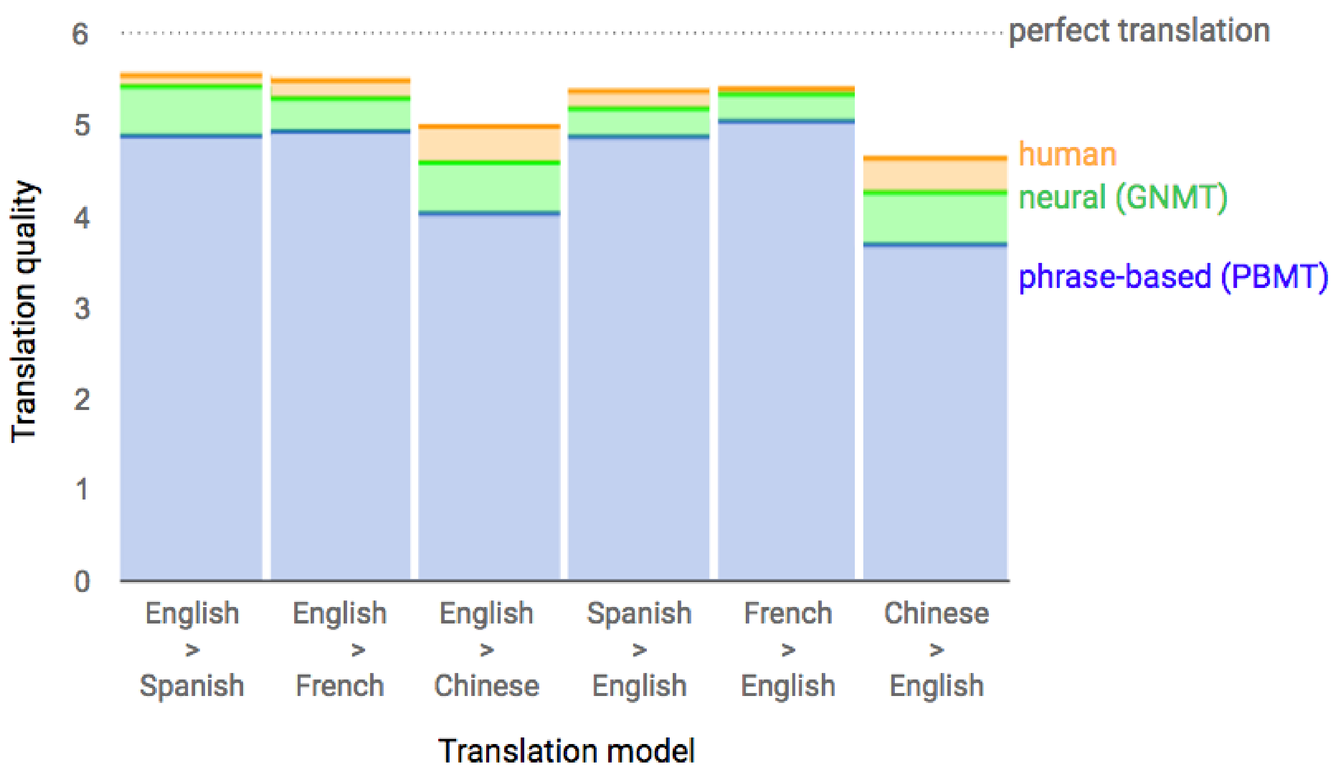
- Title words of accepted ACL long papers 2016



CITEC NLLG

DL Example: Machine translation

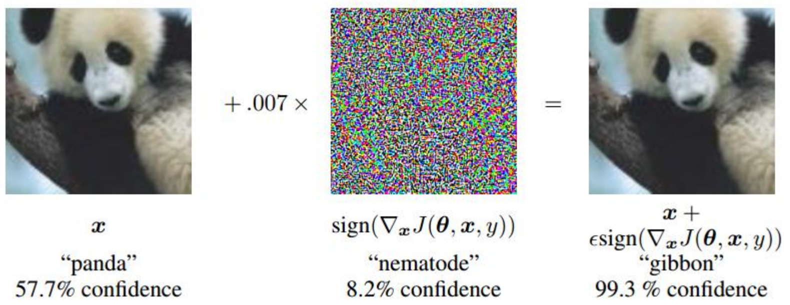
- Accuracy:



Not everything works!

- Depending on characteristics of your task, traditional approaches may still be preferable or superior
  - Neural nets seem to particularly excel on “difficult” tasks, maybe less on simple tasks **(task difficulty)**
  - Other factors such as **training data size** may play a crucial role

- Neural Networks are also prone to fooling



Scope of the Lecture

Universität Bielefeld

Deep learning for NLP:

- Exploring the use of deep learning techniques for natural language processing tasks

Deep learning	+	NLP
Machine learning method		Analysis of language

CITEC NLLG

Low-level NLP problems/tasks

Universität Bielefeld

Examples

- Sequence tagging
  - POS-tagging

Time flies like an arrow. Fruit flies like a banana.

Low-level NLP problems/tasks

Universität Bielefeld

Examples

- Sequence tagging
  - POS-tagging

NN	VBZ	IN	DT	NN
Time	flies	like	an	arrow.

NN	NN	VB	DT	NN
Fruit	flies	like	a	banana.

Low-level NLP problems/tasks

Universität Bielefeld

Examples

- Sequence tagging
  - POS-tagging
  - Named entity recognition

NN	VBZ	IN	DT	NN
Time	flies	like	an	arrow.

NN	NN	VB	DT	NN
Fruit	flies	like	a	banana.

Named Entity?

CITEC NLLG

Low-level NLP problems/tasks

Examples

- Sequence tagging
  - POS-tagging
  - Named entity recognition
- Structure prediction
  - Chunking/Parsing

Low-level NLP problems/tasks

Universität Bielefeld

Examples

- Sequence tagging
  - POS-tagging
  - Named entity recognition
- Structure prediction
  - Chunking/Parsing
- Semantics
  - Word sense disambiguation

Time flies like an arrow.

Fruit flies like a banana.

CITEC NLLG

High-level NLP tasks

Universität Bielefeld

- Text Classification
  - spam, sentiment, author, plagiarism, natural language identification
- Natural language understanding
  - metaphor analysis, argumentation mining, question-answering
- Natural language generation
  - summarization, tutoring systems, chat bots, exercise generation
- Multilinguality
  - machine translation, cross-lingual information retrieval
- Writing Assistance
  - spell checking, grammar checking, style checking, auto-completion
- …

Example: Named Entity Recognition

Universität Bielefeld

Output: B-Person (BIO-Scheme)

German chancelor $\boxed{\text{Angela}}$ Merkel said

CITEC NLLG

Traditional feature engineering

Universität Bielefeld

Output: B-Person

uppercase	1
isNoun	1
previousWord DET	0
previousWord uppercased	0
following word N	1
following word uppercased	1
...	...

German chancelor Angela Merkel said

CITEC NLLG

Traditional feature engineering

Output: I-Person

uppercase	1
isNoun	1
previousWord DET	0
previousWord uppercased	1
following word N	0
following word uppercased	0
...	...

German chancellor Angela $\boxed{\text{Merkel}}$ said yesterday

Feature representation

Universität Bielefeld

- Word specific features

previous word is minister

previous word is chancellor

previous word is president

previous word is company

previous word is product

following word is says

following word is declares

following word is claims

CITEC NLLG

Feature representation

Universität Bielefeld

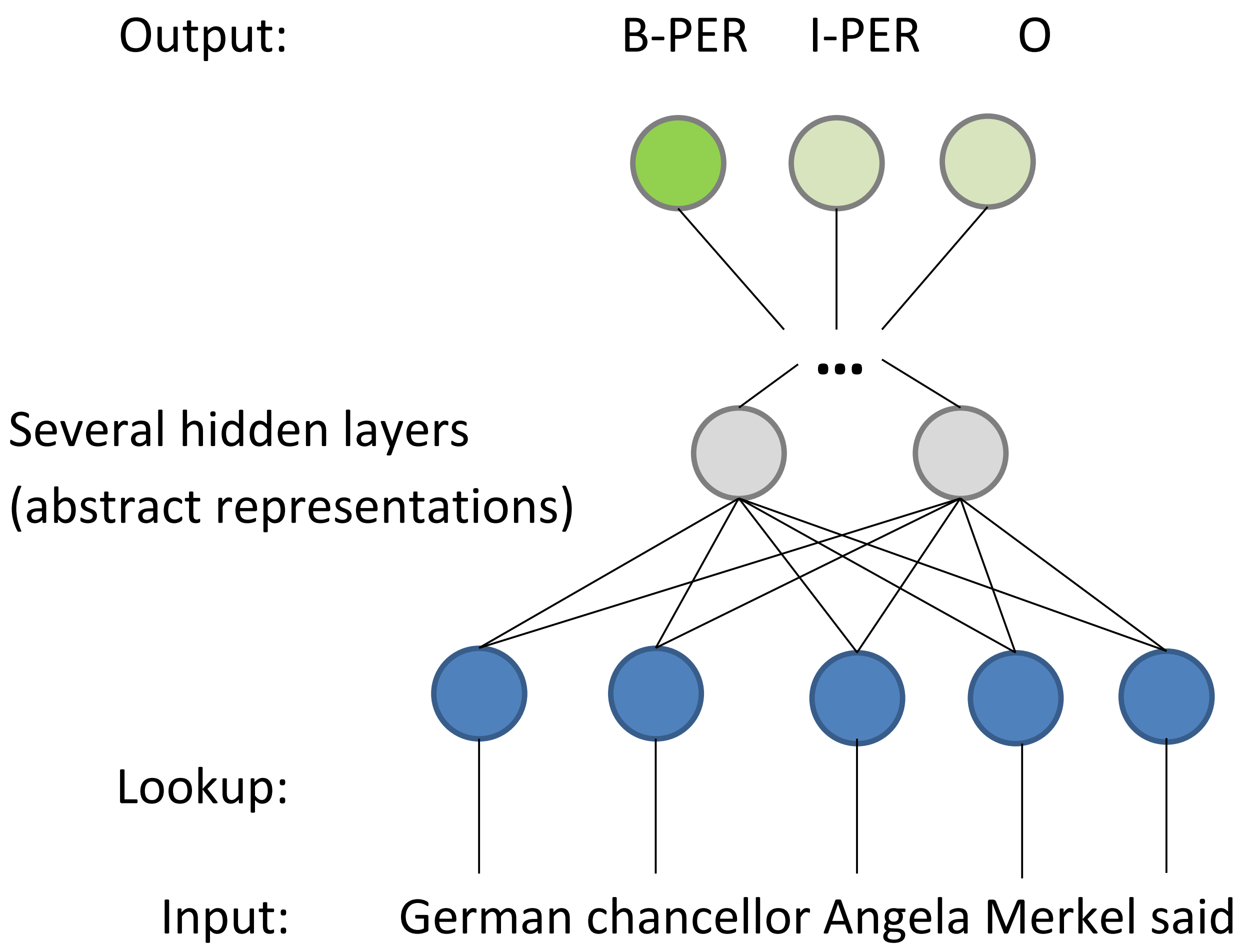
- Word specific features

previous word is minister
previous word is chancellor
previous word is president
previous word is company
previous word is product
following word is says
following word is declares
following word is claims

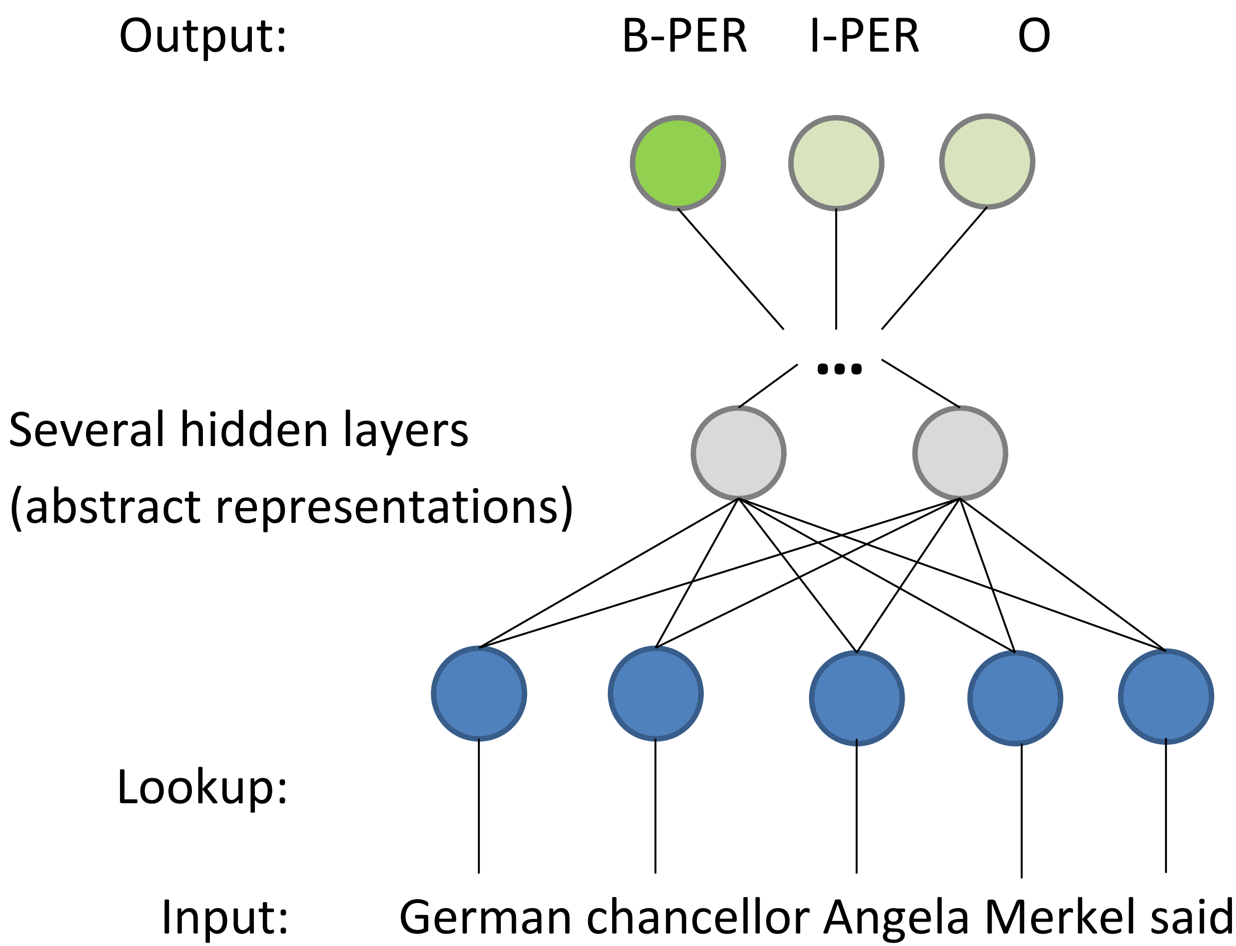
- Problems:
  - Feature engineering
  - Similarity of words not taken into account

CITEC NLLG

DL for NLP


Output:
B-PER
I-PER
O
Several hidden layers
(abstract representations)
Lookup:
Input:
German chancellor Angela Merkel said

DL for NLP



Architecture of neural networks: Lecture 1/7-

Mapping words into vector representations: Lecture 3-6

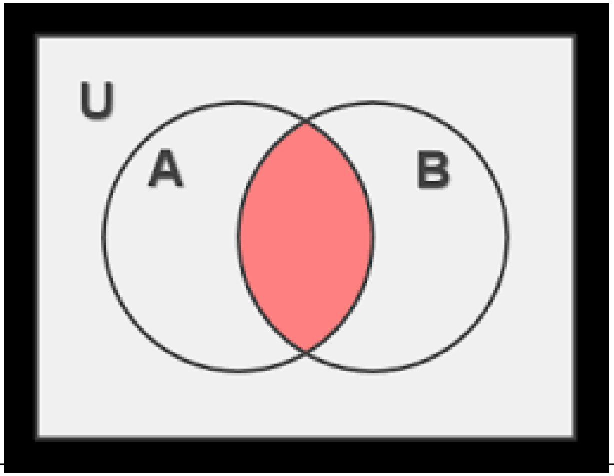
Detailed Syllabus

Universität Bielefeld

1	07.04	Introduction+Perceptrons (In-Class)
2	14.04	ML background (Inverted Class Room --- **ICR**)
3	21.04	Backpropagation – Learning in deep neural nets (In-Class)
4	28.04	Word Embeddings 1 – CBOW and Skip-Gram (**ICR**)
5	05.05	Dependency Parsing (**ICR**)
6	12.05	Word Embeddings 2 – Bilingual and Syntactic Embeddings (Online)
7	19.05	Word Embeddings 3 – Contextualized Embeddings (In-Class)
8	02.06	Convolutional networks (**ICR**)
9	09.06	Recurrent neural networks (In-Class)
10	23.06	Encoder-Decoder Neural Nets (**ICR**)
11	30.06	Evaluation Metrics (In-Class)
12	07.07	Efficiency, Explainability, Adversarial Attacks (Online)
13	14.07	Guest lectures (Online)

ICR

- You’ll watch a video at home **before** the lecture (typically 100-120min)
  - Video and lecture will overlap, but the lecture may cover other aspects as well
- The lecture + QA itself will then be shorted to about 30-45min



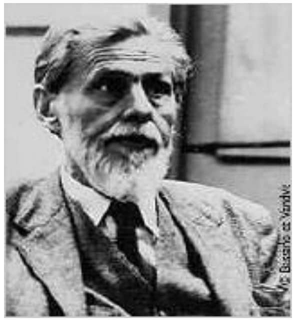
A = Video
B = Lecture

Outline

Perceptrons

Origins

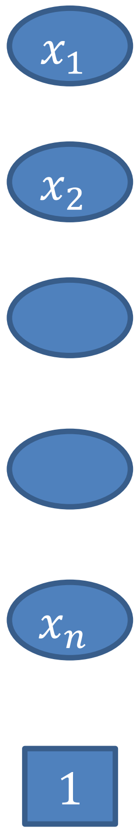
- ‘Simplest’ form of a neural network
- Introduced by McCulloch and Pitts (1943) and Frank Rosenblatt (1958)



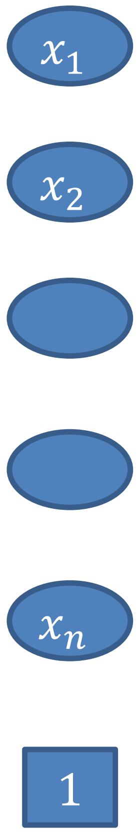
http://www.monizone.de/projects/knn/images/mcculloch_160.jpg

http://www.i-programmer.info/babbages-bag/325-mcculloch-pitts-neural-networks.html

Network structure

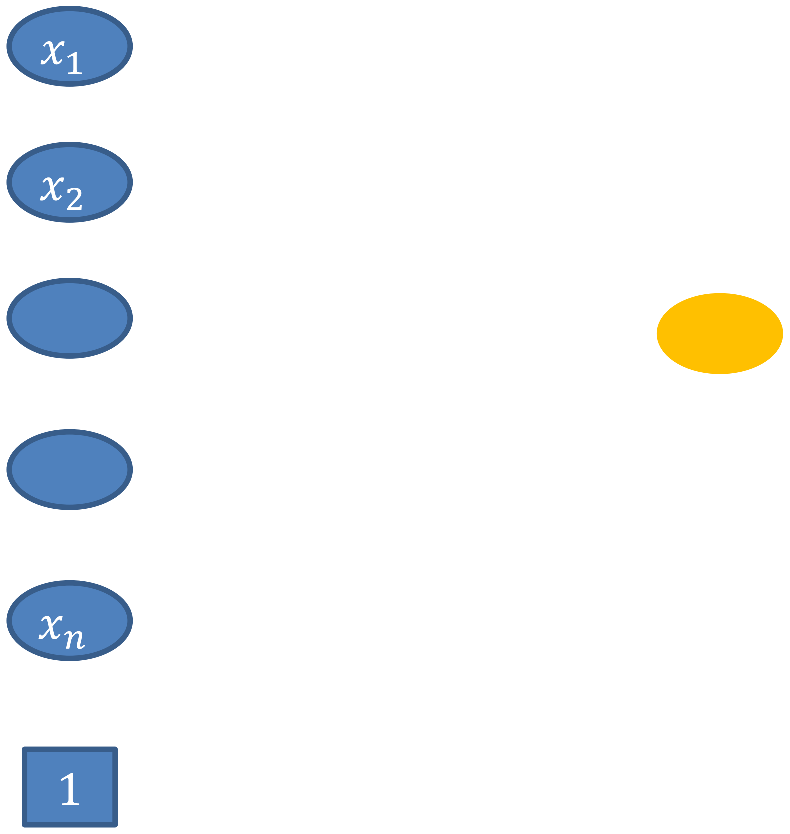


Network structure



input neurons

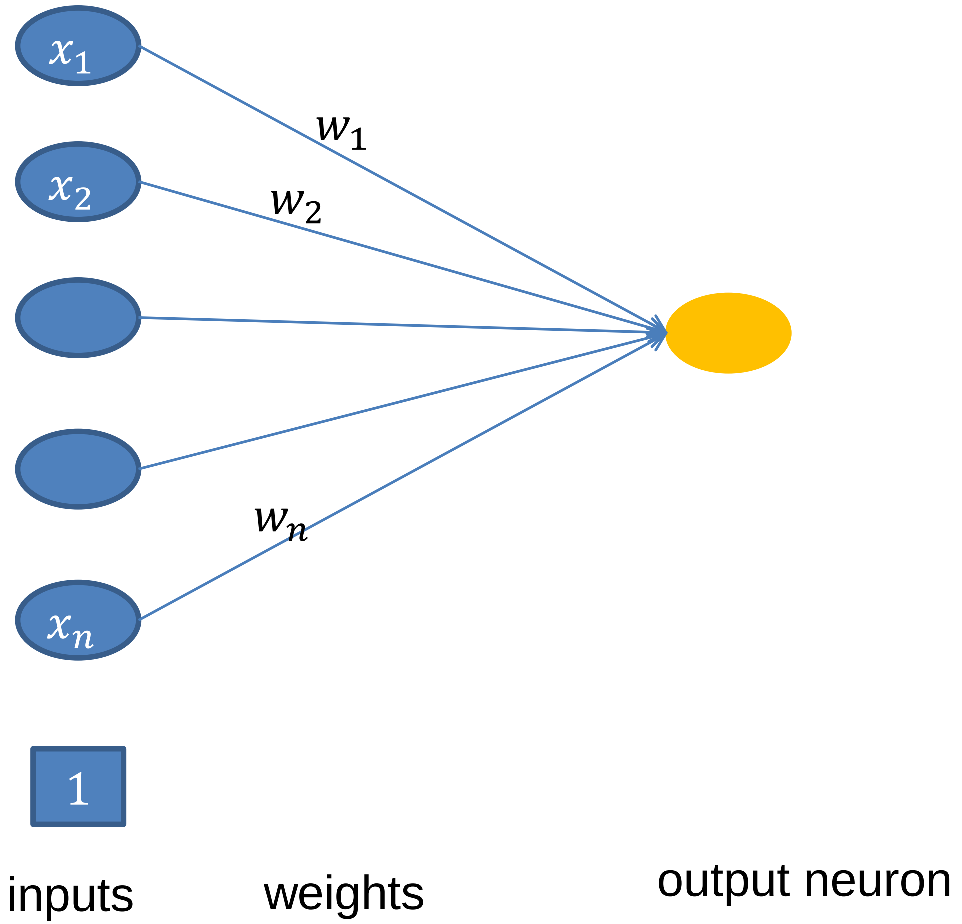
Network structure



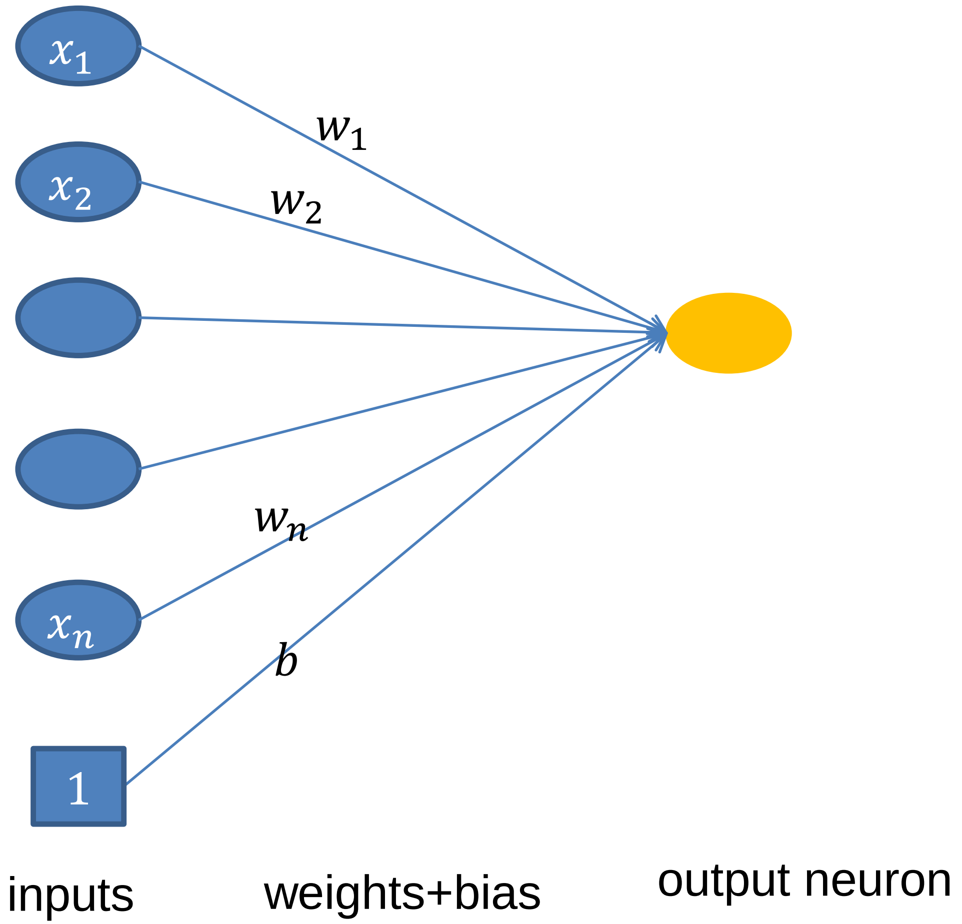
inputs

output neuron

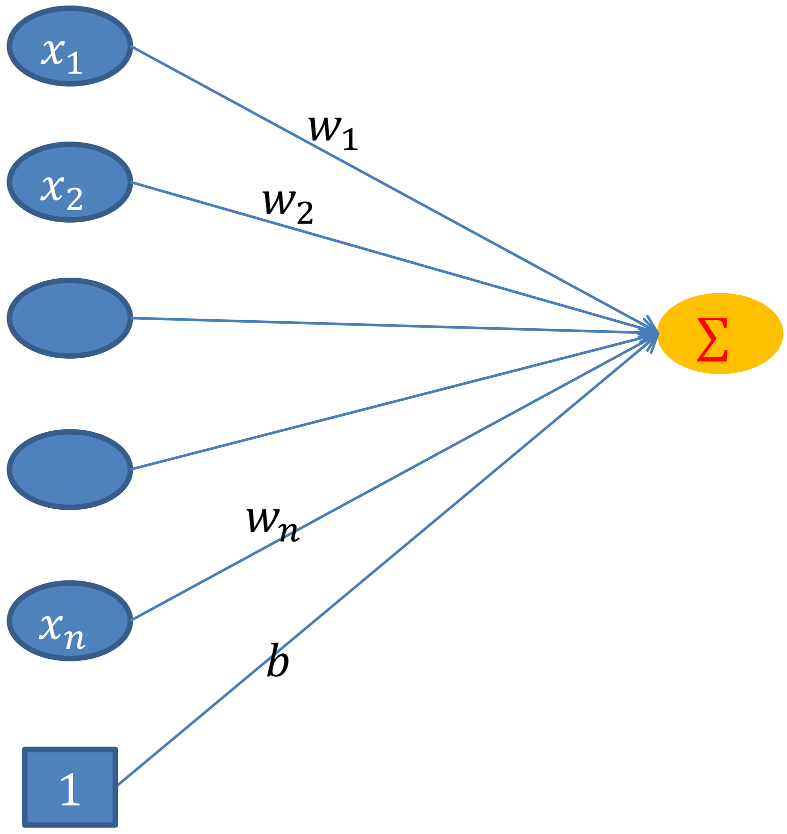
Network structure



Network structure

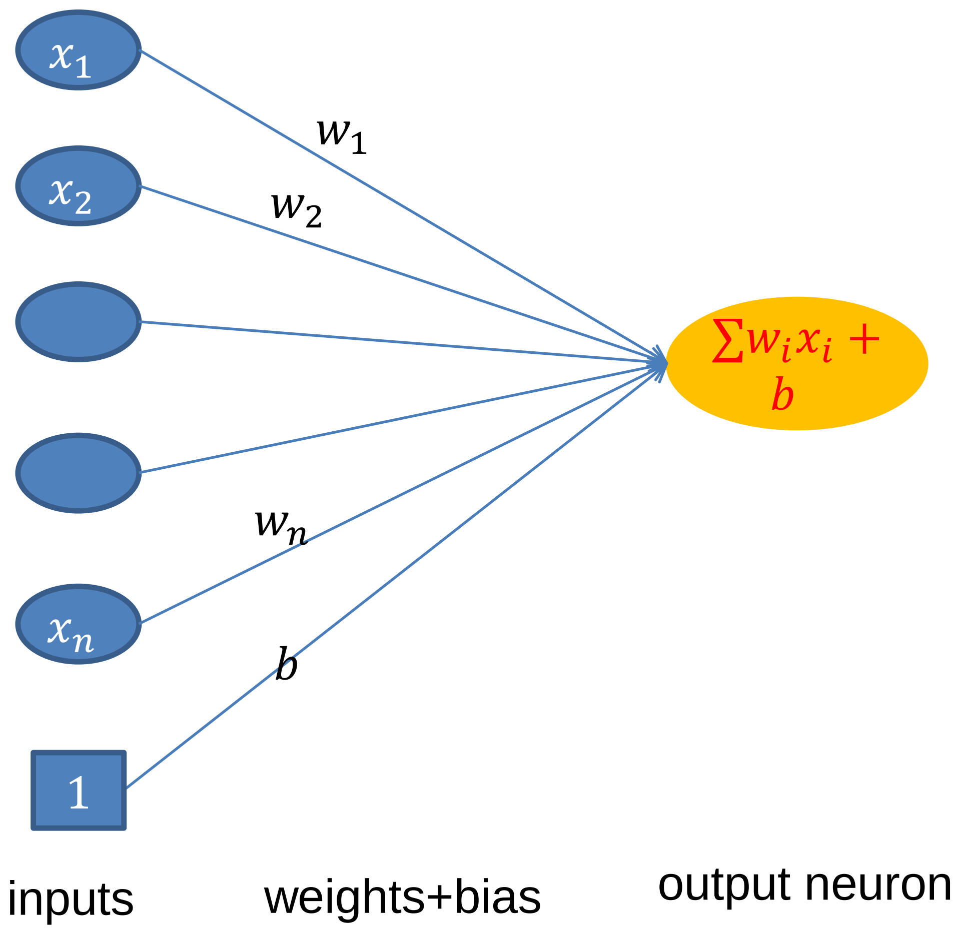


Network structure

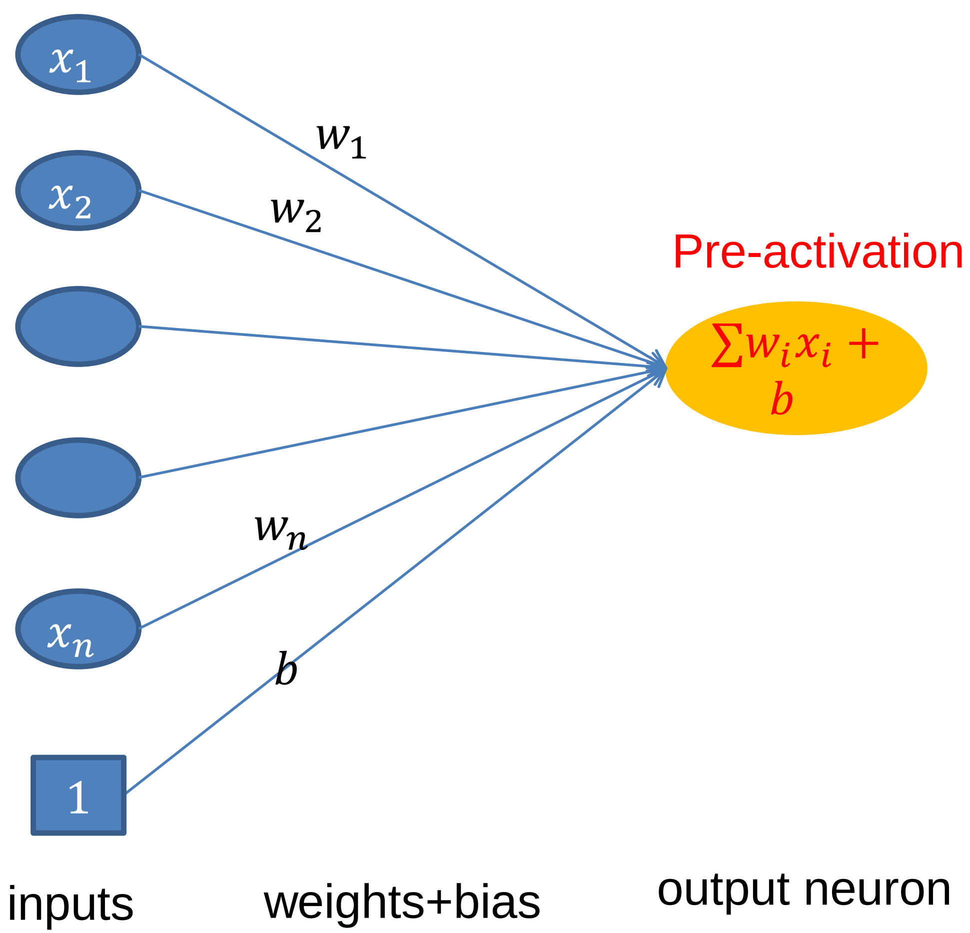


inputs weights+bias output neuron

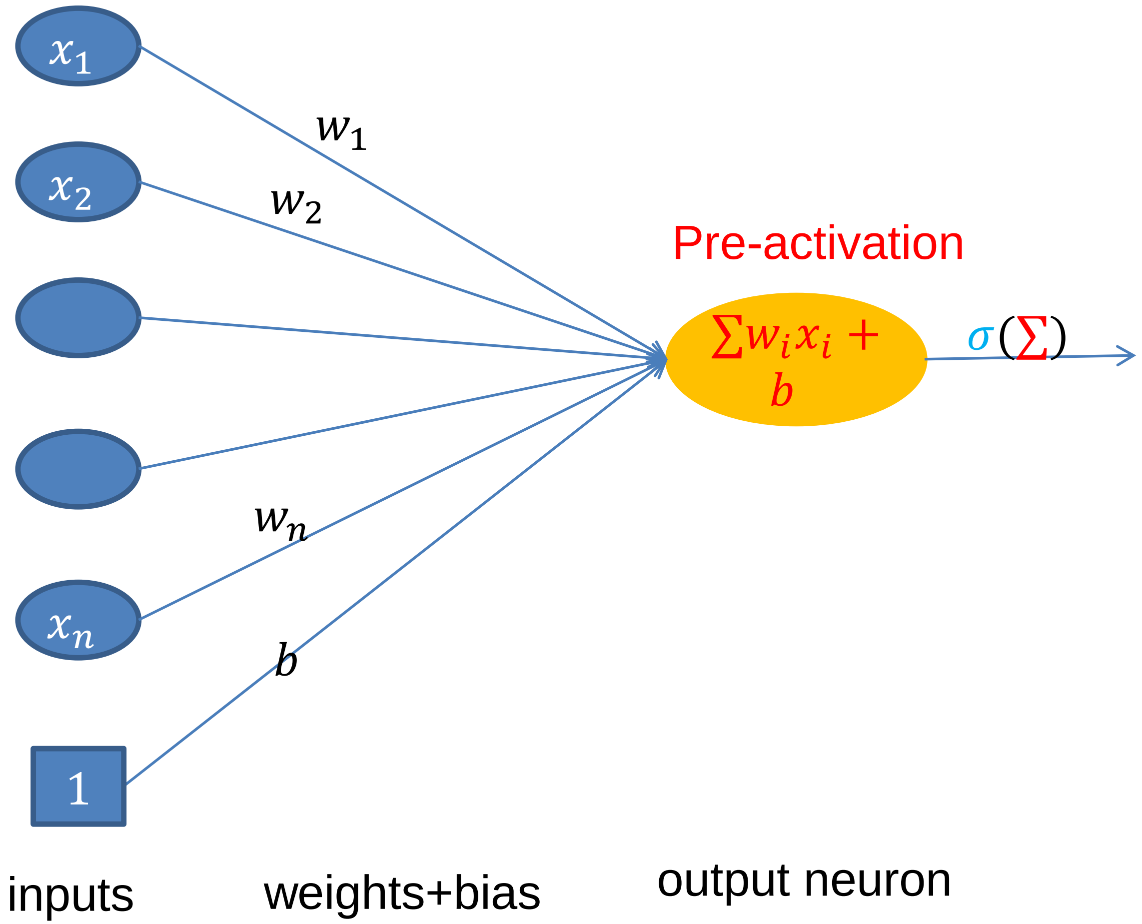
Network structure



Network structure

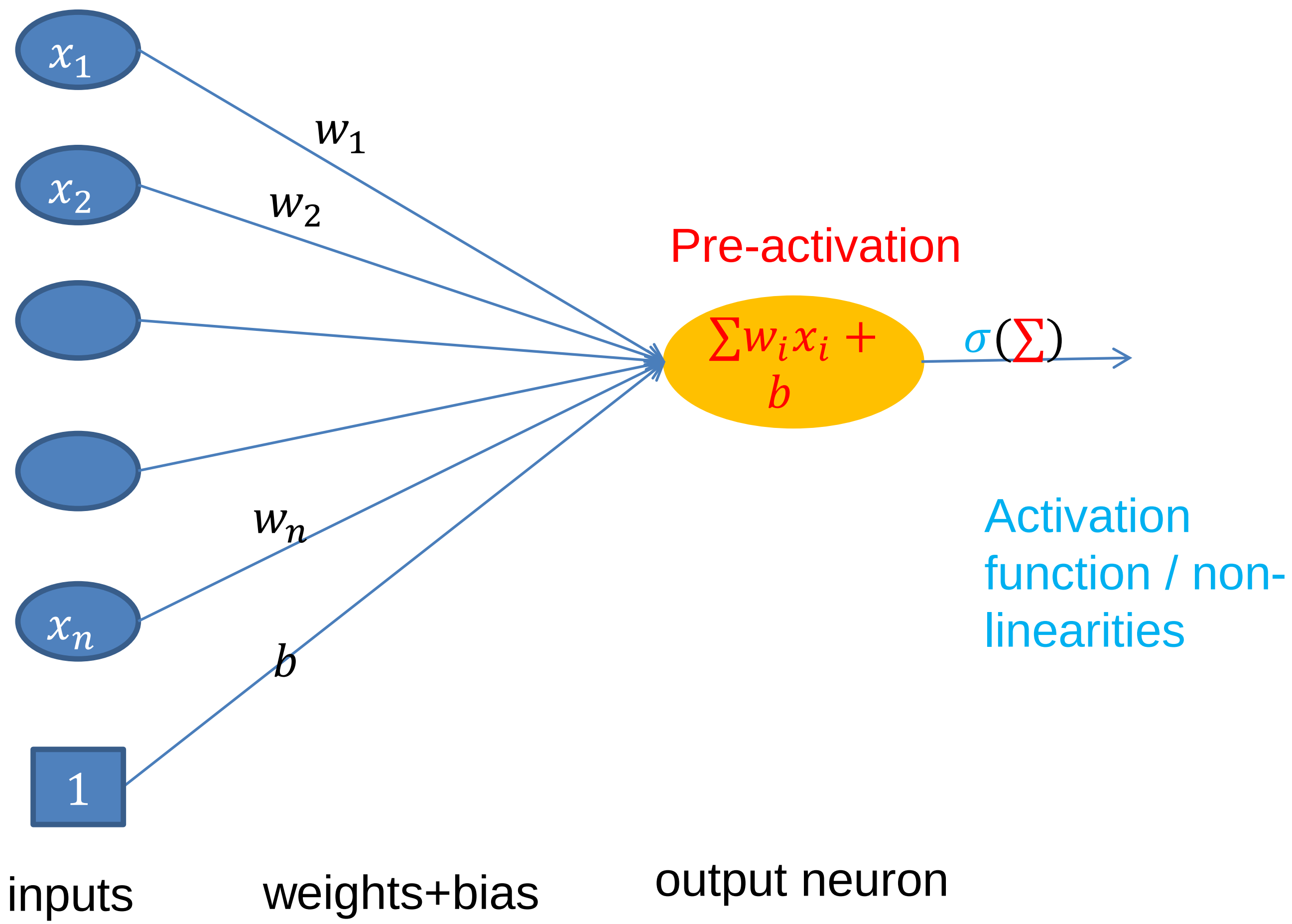


Network structure



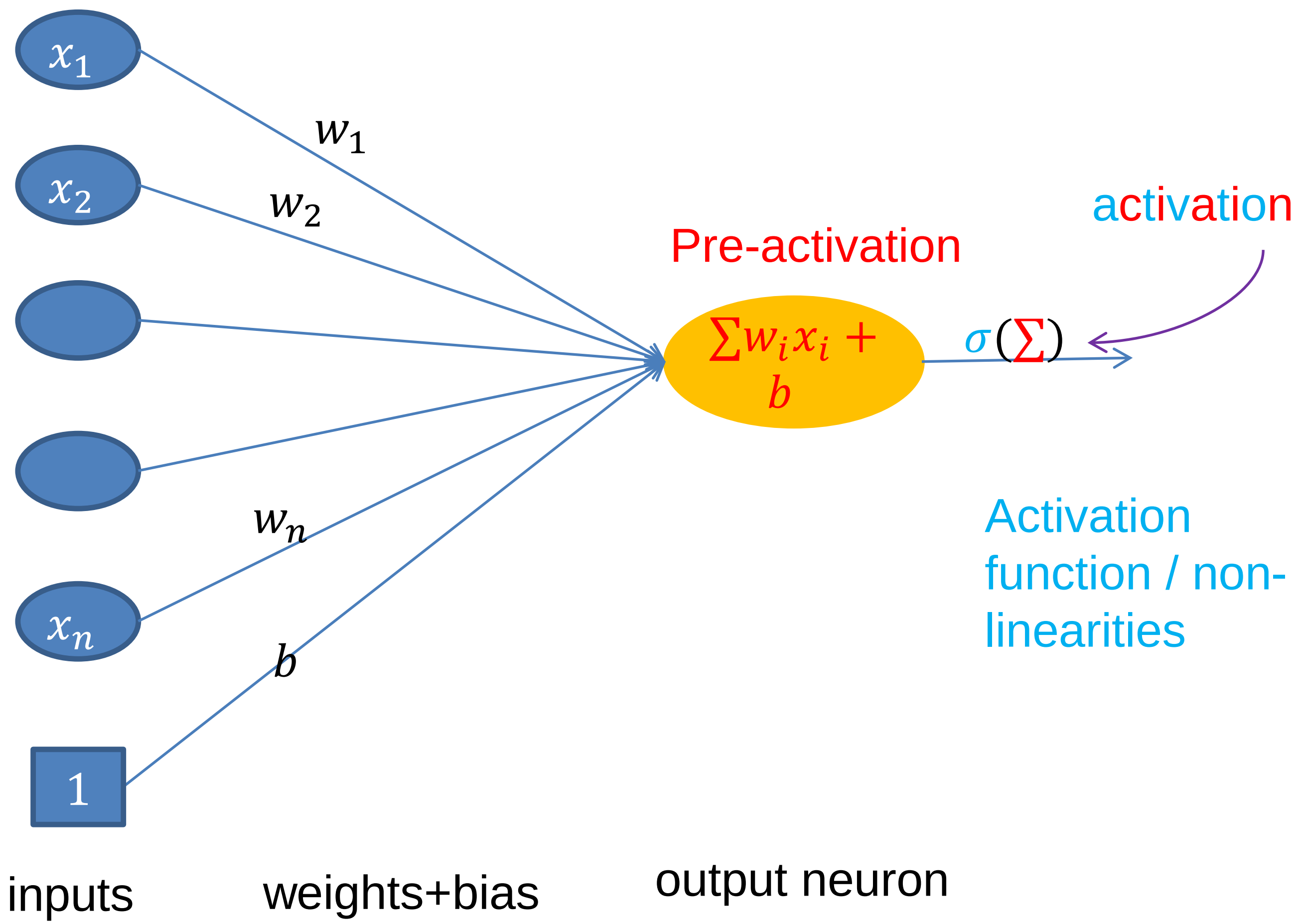
Network structure

Universität Bielefeld



CITEC NLLG

Network structure



A formal description

Universität Bielefeld

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, $b \in \boldsymbol{R}$
  - **Non-linearity** $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$
- **Input to network**
  - **Input vector** $\mathbf{x} \in \mathbb{R}^{1 \times n}$, for $n \geq 1$.
  - **And constant** $1 \in \boldsymbol{R}$
- **Output unit**
  - **Pre-activation:** $\mathbf{x} \cdot \mathbf{w} + b = \sum_{i=1}^{n} w_i \cdot x_i \quad + b$
  - **Activation:** $\sigma(\mathbf{x} \cdot \mathbf{w} + b)$

A formal description

Universität Bielefeld

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, $b \in \boldsymbol{R}$
  - **Non-linearity** $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$
- **Input to network**
  - **Input vector** $\mathbf{x} \in \mathbb{R}^{1 \times n}$, for $n \geq 1$.
  - **And constant** $1 \in \boldsymbol{R}$
- **Output unit**
  - **Pre-activation:** $\mathbf{x} \cdot \mathbf{w} + b = \sum_{i=1}^{n} w_i \cdot x_i \quad + b$
  - **Activation:** $\sigma(\mathbf{x} \cdot \mathbf{w} + b)$

Dot product
(„Skalarprodukt“)

CITEC NLLG

A formal description

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, $\cancel{b \in \boldsymbol{R}}$
  - **Non-linearity** $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$
- **Input to network**
  - **Input vector** $\mathbf{x} \in \mathbb{R}^{1 \times n}$, for $n \geq 1$.
  - **And constant** $\cancel{1 \in \boldsymbol{R}}$
- **Output unit**
  - **Input:** $\mathbf{x} \cdot \mathbf{w} + b = \sum_{i=1}^{n} w_i \cdot x_i \quad + b$
  - **Output:** $\sigma(\mathbf{x} \cdot \mathbf{w} + b)$

We‘ll put the constant input „1“ into ***x*** for simplicity

A formal description

Universität Bielefeld

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^{n+1}$
  - **Non-linearity** $\sigma : \mathbb{R} \to \mathbb{R}$
- **Input to network**
  - **Input vector** $\tilde{\mathbf{x}} = (\mathbf{x} \quad 1) \in \mathbb{R}^{n+1}$
  - **And constant** $\cancel{1 \in \boldsymbol{R}}$
- **Output unit**
  - **Input:** $\tilde{\boldsymbol{x}} \cdot \boldsymbol{w}$
  - **Output:** $\boldsymbol{\sigma}(\tilde{\boldsymbol{x}} \cdot \boldsymbol{w})$

Notice that dimensionality is n+1 now

CITEC NLLG

A formal description

Universität Bielefeld

- **Given**
  - **Weight vector** $\boxed{\mathbf{w} \in \mathbb{R}^{n+1}}$
  - **Non-linearity** $\sigma : \mathbb{R} \to \mathbb{R}$

Parameter (vector): we „learn"/"estimate" this

- **Input to network**
  - **Input vector** $\boxed{\tilde{\mathbf{x}} = (\mathbf{x} \quad 1) \in \mathbb{R}^{n+1}}$

(Training/test) Data

- **Output unit**
  - **Input:** $\tilde{\boldsymbol{x}} \cdot \boldsymbol{w}$
  - **Output:** $\boxed{\boldsymbol{\sigma}(\tilde{\boldsymbol{x}} \cdot \boldsymbol{w}) = \boldsymbol{f}_{\boldsymbol{w}}(\tilde{\boldsymbol{x}})}$

(Statistical) Model, parametrized by $\boldsymbol{w}$ [$\boldsymbol{w}$ often also denoted as $\boldsymbol{\theta}$]

Questions

Universität Bielefeld

- **Why do we need a bias unit?**

CITEC NLLG

Questions

Universität Bielefeld

- **Why do we need a bias unit? A: To increase the “capacity” of our statistical model**

CITEC NLLG

Optimization problem

- Perceptron is the function $f : \mathbb{R}^{n+1} \to \mathbb{R}$ where $f(\tilde{\mathbf{x}}; \mathbf{w}) = \sigma(\tilde{\mathbf{x}} \cdot \mathbf{w})$
- We consider $\mathbf{w}$ as the parameters which we want to optimize
  - Scenario: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ is our *training data*
    - We want to solve:[1]

$$\min_{\mathbf{w} \in \mathbb{R}^{n+1}} \sum_{j=1}^{N} \left(f(\mathbf{x}_j; \mathbf{w}) - y_j \right)^2$$

[1] Note that in ML, we actually want to optimize our parameters such that performance is good on data that follows the same distribution as our training data

We write here $f(\boldsymbol{x}; \boldsymbol{w})$ rather than $f_{\boldsymbol{w}}(\boldsymbol{x})$ as before.

A perceptron learning algorithm

- **Given**:
  - Training data $T = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_N, y_N)\}$
  - Learning rate α
  - Initial parameter vector $\boldsymbol{w}$
- While stopping criterion not met
  - Choose a random sample $\boldsymbol{T}'$ of size $\boldsymbol{N}'$, for $\mathbf{1} \leq \boldsymbol{N}' \leq \boldsymbol{N}$, from $\boldsymbol{T}$
  - Update:
    $$\mathbf{w}' \leftarrow \mathbf{w} - \alpha \sum_{(\mathbf{x},y)\in T'} (\sigma(\mathbf{x}\cdot\mathbf{w}) - y)\,\sigma'(\mathbf{x}\cdot\mathbf{w})\mathbf{x}$$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w}'$

How do we arrive at this?

Universität Bielefeld

It’s a bit like optimization in school

- Take first **derivative** (aka **gradient**), set it to zero
- Except that we’re in a multi-dimensional space, rather than in 1-d
- And that exact solutions for $\boldsymbol{w}$ don’t exist
  - Instead, we look at the gradient and plug it into a general optimization technique called *gradient descent*

We’ll see more on this in lectures 2&3

Outline

Universität Bielefeld

Limitations of Perceptrons

Decision surface of a perceptron

Universität Bielefeld

- Perceptron is the function $f : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ where $f(\tilde{\mathbf{x}}; \mathbf{w}) = \sigma(\tilde{\mathbf{x}} \cdot \mathbf{w})$
- Perceptron uses threshold function

$$\sigma(z) = \begin{cases} 0 & \text{if } z < 0, \\ 1 & \text{if } z \geq 0 \end{cases}$$

Decision surface of a perceptron

- Perceptron is the function $f : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ where $f(\tilde{\mathbf{x}}; \mathbf{w}) = \sigma(\tilde{\mathbf{x}} \cdot \mathbf{w})$
- Perceptron uses threshold function

$$\sigma(z) = \begin{cases} 0 & \text{if } z < 0, \\ 1 & \text{if } z \geq 0 \end{cases}$$

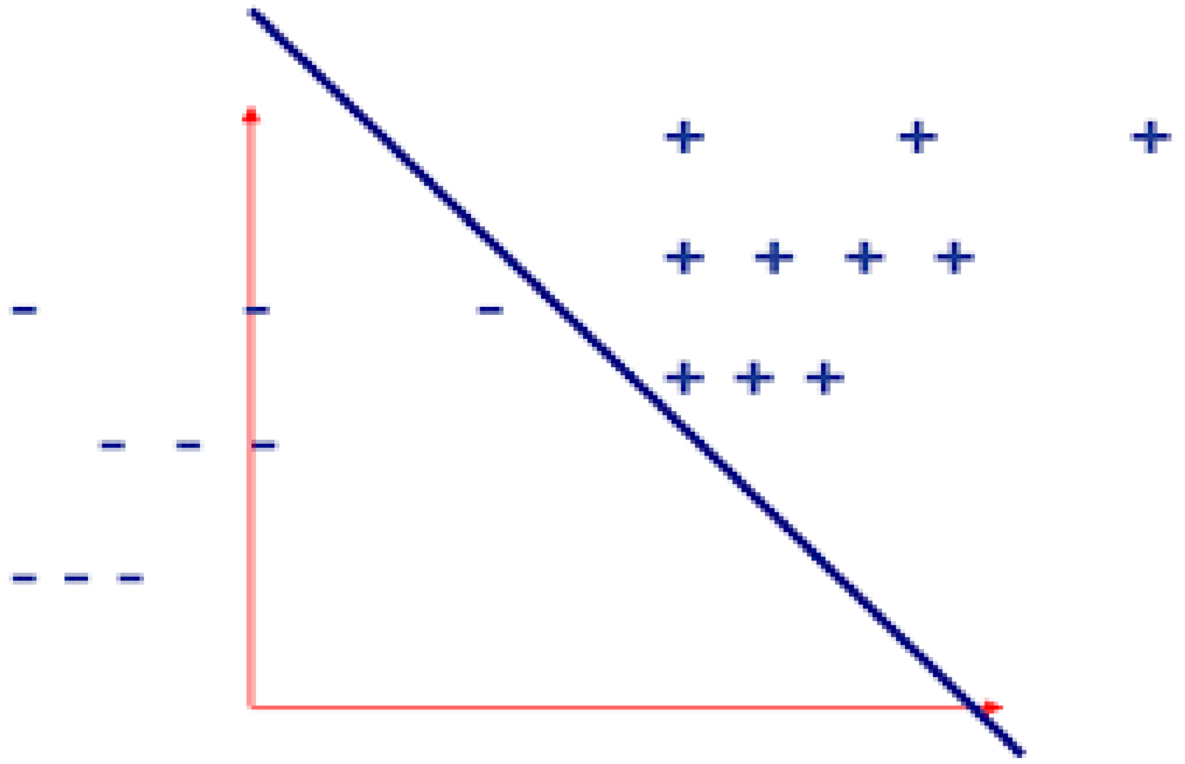
- Consider the *hyperplane* $H_{\boldsymbol{w}}$

$$\{(x_1, \ldots, x_n) \,|\, \tilde{\mathbf{x}} \cdot \mathbf{w} = x_1 w_1 + \cdots + x_n w_n + w_{n+1} = 0\}$$

- In 2-d: $x_1 w_1 + x_2 w_2 + w_{n+1} = 0 \text{ iff } x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_3}{w_2}$
- $y = az + b$ is the equation of a straight line

Decision surface

- Decision surface of threshold function perceptron



From http://marcuswhybrow.com/lecturenotes/build/html/images/euclidean-feature-space.png

What a perceptron can do

- **Linear separability** Let X_0 and X_1 be two sets of points in an n-dimensional Euclidean space. Then X_0 and X_1 are *linearly separable* if there exists n+1 real numbers $w_1, \ldots, w_n, w_{n+1}$ such that every $\boldsymbol{x} \in X_0$ satisfies $\tilde{\boldsymbol{x}} \cdot \boldsymbol{w} > 0$, and every point $\boldsymbol{x} \in X_1$ satisfies $\tilde{\boldsymbol{x}} \cdot \boldsymbol{w} < 0$. See https://en.wikipedia.org/wiki/Linear_separability

- **Theorem** If data is linearly separable, then the (original) perceptron learning algorithm converges after a finite amount of time and classifies all training data examples correctly.

What a perceptron can do

Universität Bielefeld

- **Example** (The OR problem) Let $X_0 = \{(0,1),(1,0),(1,1)\}$ and $X_1 = \{(0,0)\}$. Both sets are linearly separable. We may choose (among others) $\boldsymbol{w} = (1, 1, -0.5)$.

- **Hint:**
  - two sets are linearly separable:
  - set the weight vector $\boldsymbol{w}$ of a perceptron such that it classifies each instance in the two sets "correctly" (the instances in X_0 are thought of having class label 0, and elements in X_1 have labels 1, or vice versa)

CITEC NLLG

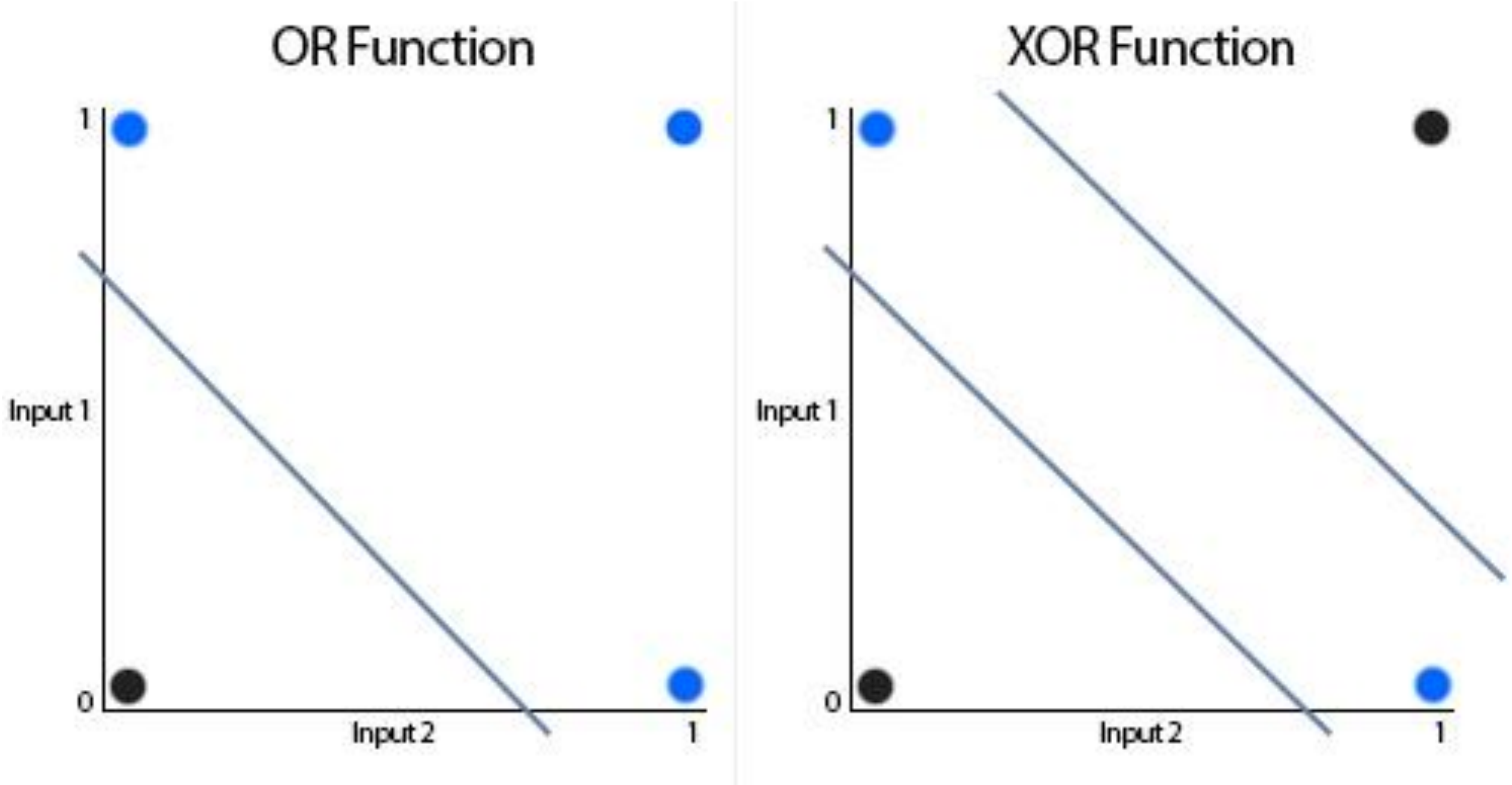
What a perceptron **can**NOT **do**

Universität Bielefeld

- **Theorem** The XOR problem is not linearly separable

- **Proof** We have $X_0 = \{(1,1),(0,0)\}$ and $X_1 = \{(1,0),(0,1)\}$. If X_0, X_1 were linearly separable, there were w_1, w_2, w_3 as in the definition. Then
$$w_1 + w_2 + w_3 > 0, \qquad w_3 > 0,$$
$$w_1 + w_3 < 0, \qquad w_2 + w_3 < 0,$$
or vice versa. These equations cannot be satisfied. For example,
$$w_1 + w_2 + w_3 + w_3 > 0$$
by the first equations. However,
$$w_1 + w_3 + w_2 + w_3 < 0$$
by the last two equations.
This is a contradiction.

What a perceptron canNOT do



From http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

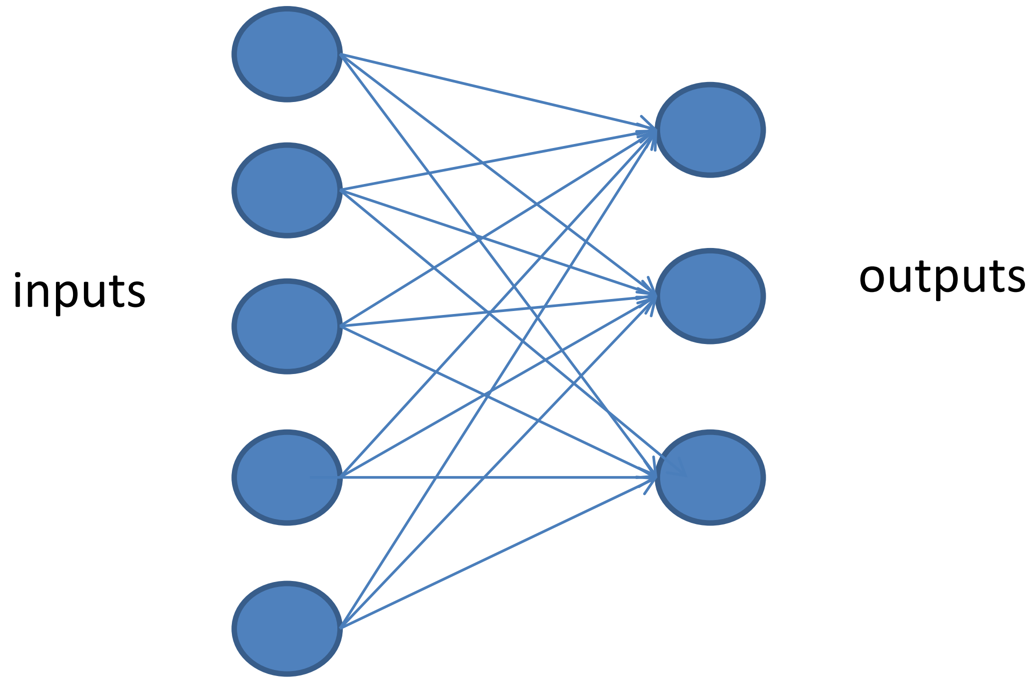
Outline

Multi-layer perceptrons

More complex neural networks

Universität Bielefeld

- Several output neurons instead of a single output neuron (**we still call it "perceptron"**)



Now, rather than a dot product $\boldsymbol{x} \cdot \boldsymbol{w}$, we have a vector-matrix multiplication:

$$\boldsymbol{x} \cdot \boldsymbol{W}$$

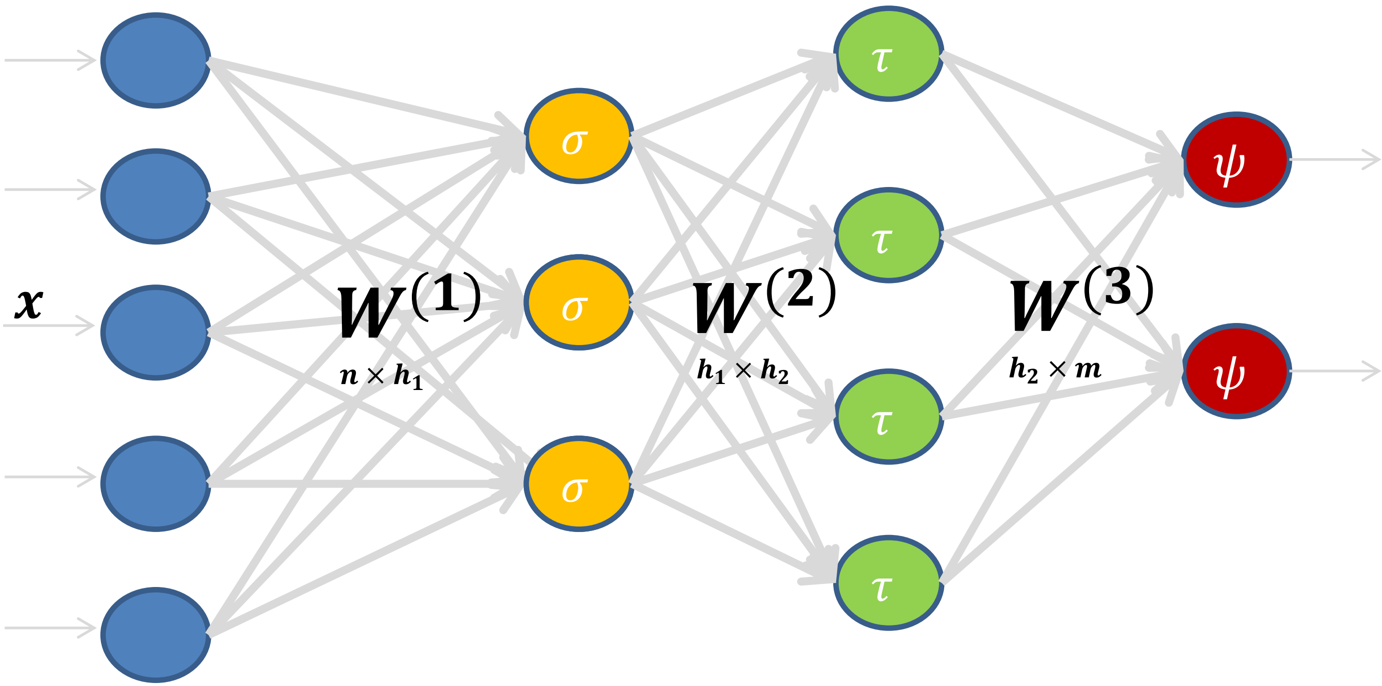
where $\boldsymbol{W}$ holds the weight vectors $\boldsymbol{w}_j$ for each output neuron j

- No additional technical difficulty, can use the previous learning techniques

More complex neural networks

Universität Bielefeld

- Multiple **hidden layers/**units

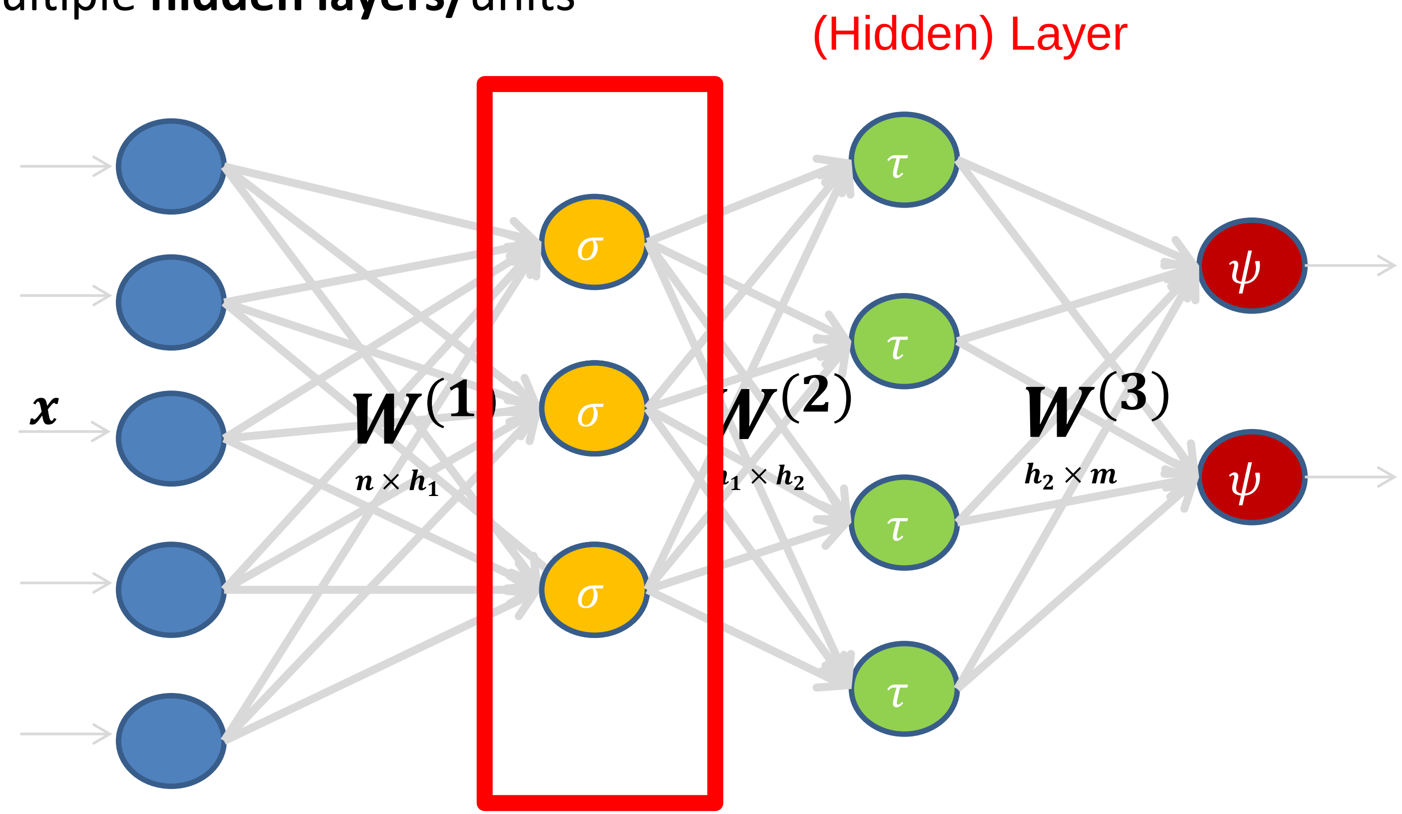


- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

CITEC NLLG

More complex neural networks

- Multiple **hidden layers/**units

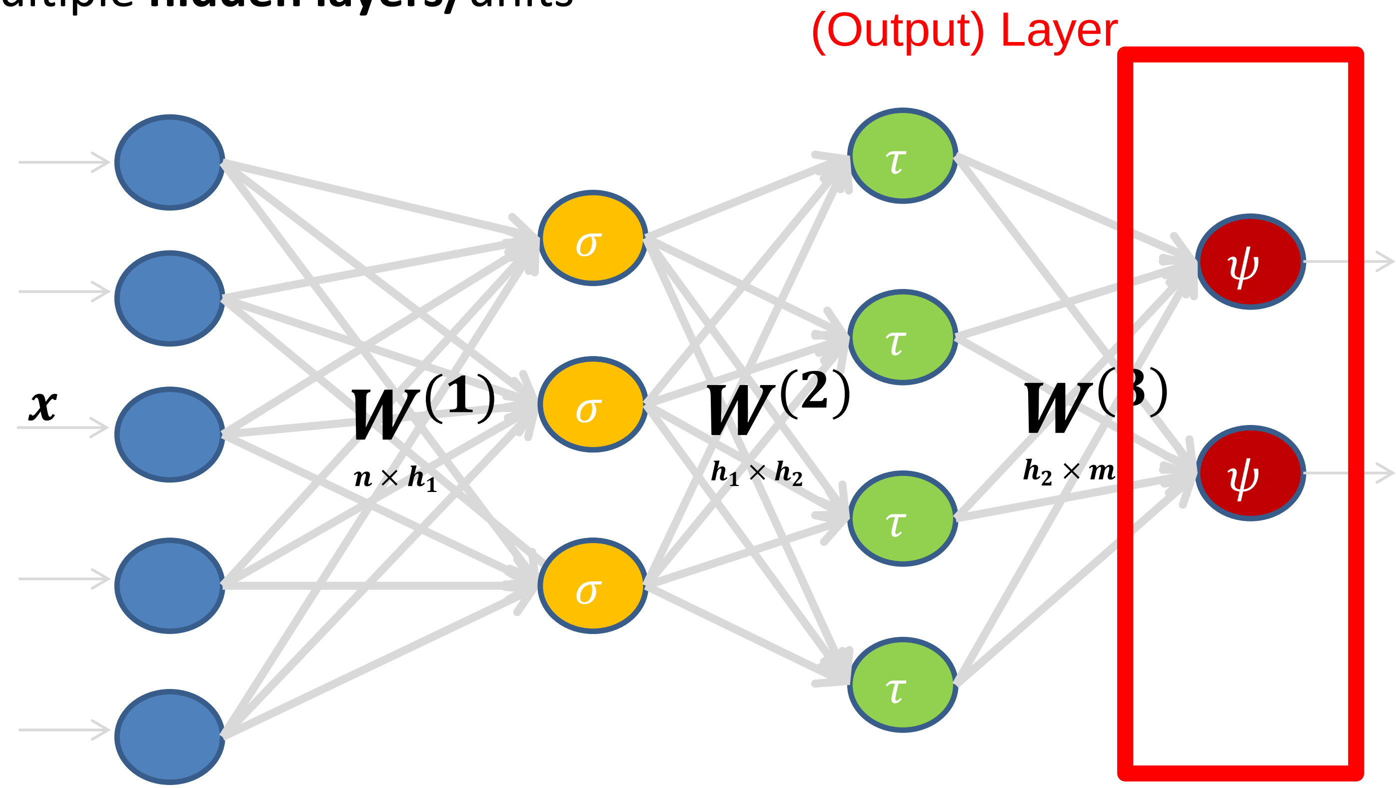


- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

More complex neural networks

Universität Bielefeld

- Multiple **hidden layers/**units

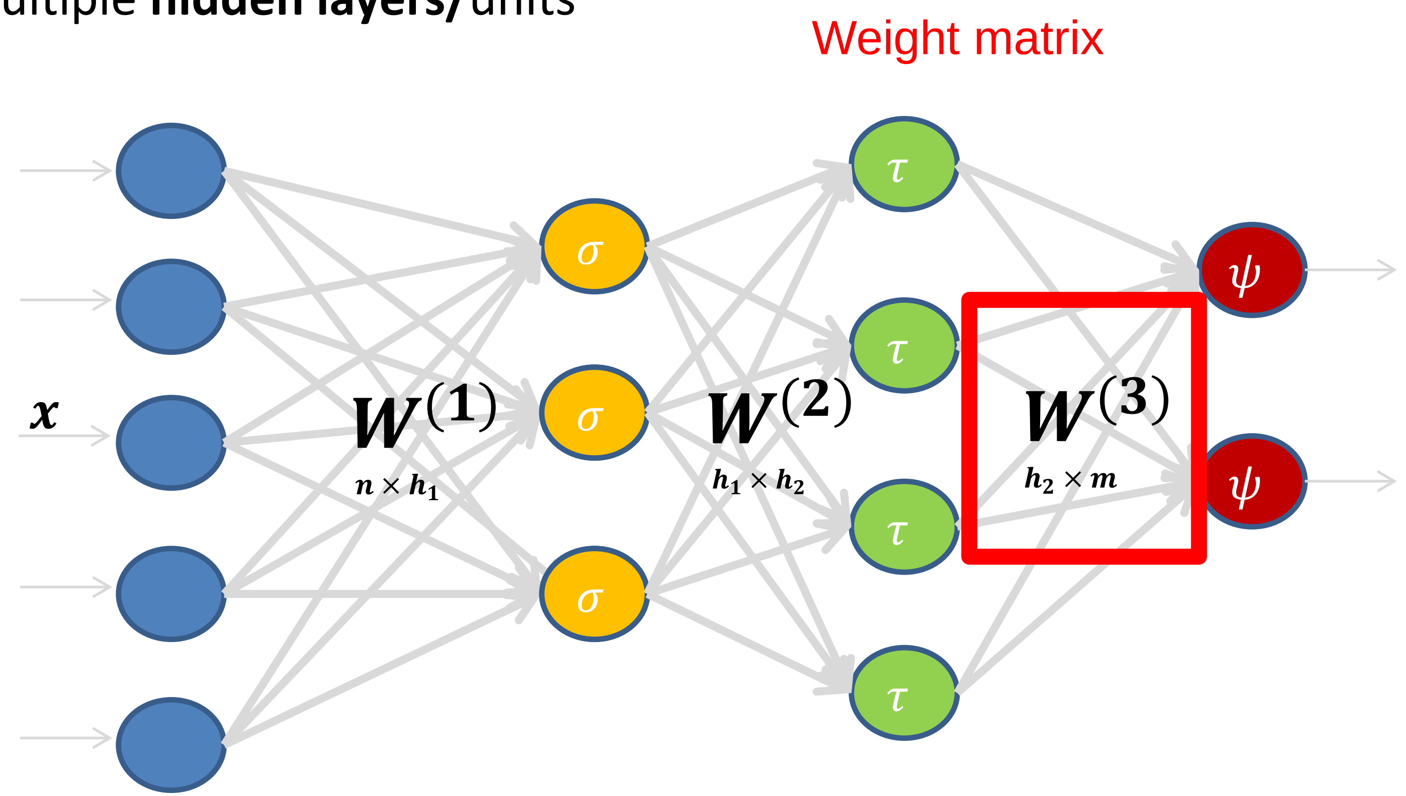


- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

More complex neural networks

Universität Bielefeld

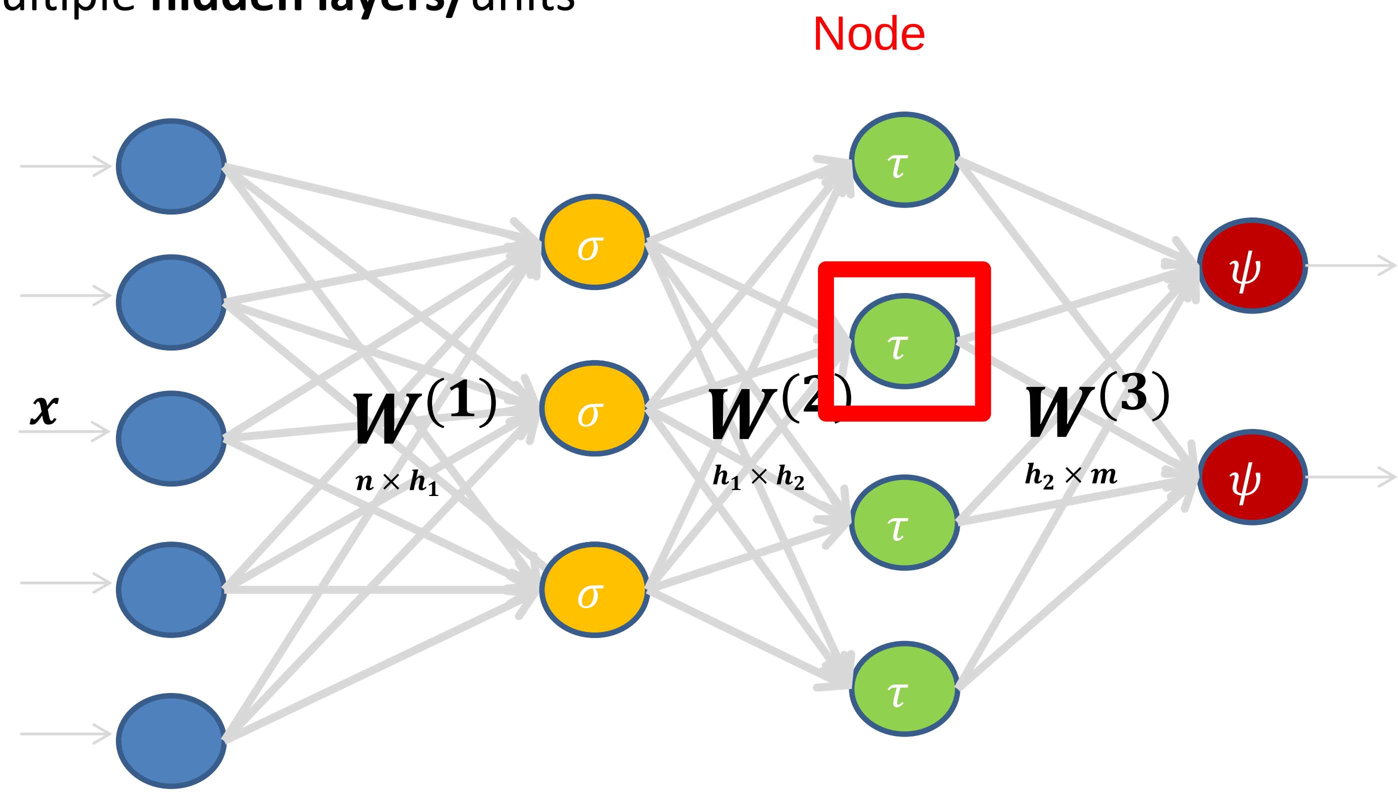
- Multiple **hidden layers/**units



- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

More complex neural networks

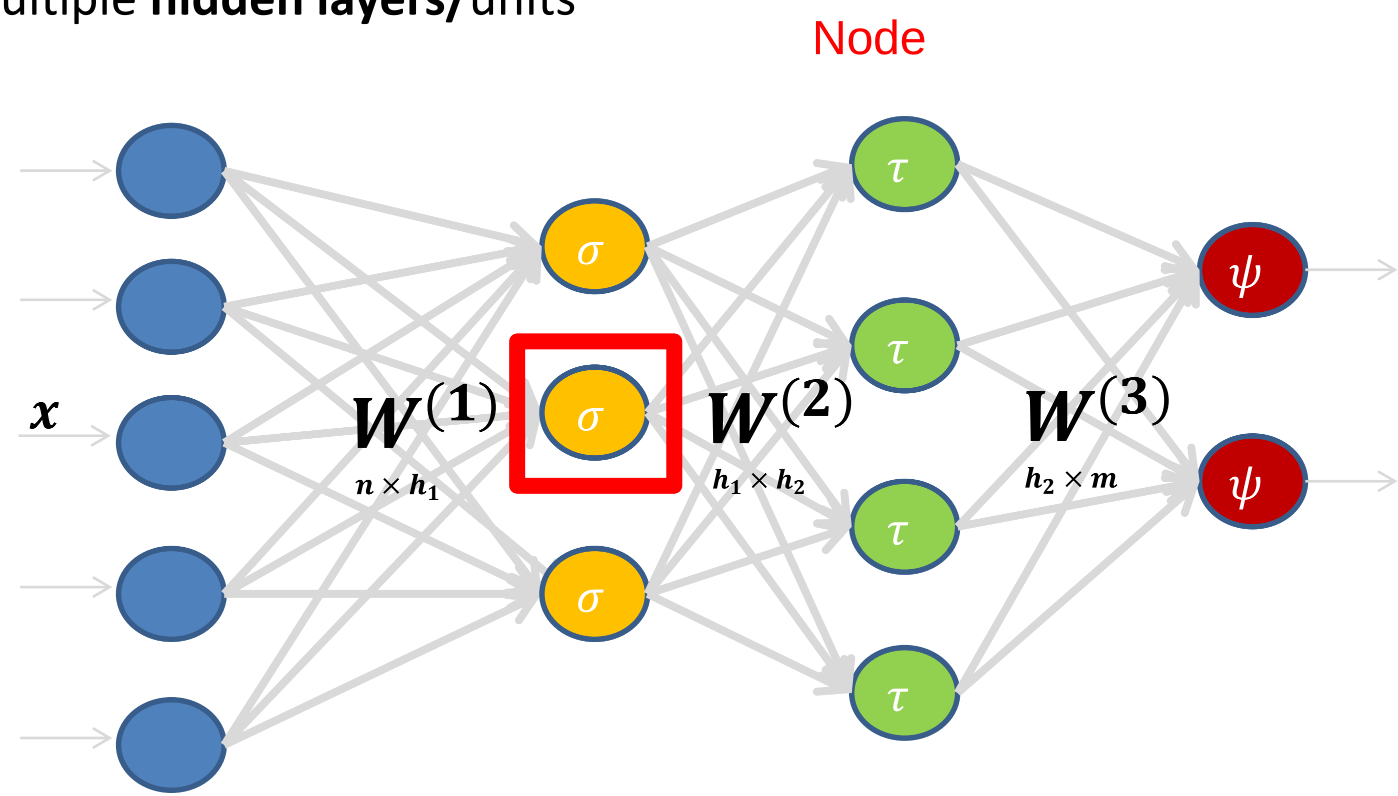
- Multiple **hidden layers/**units



- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

More complex neural networks

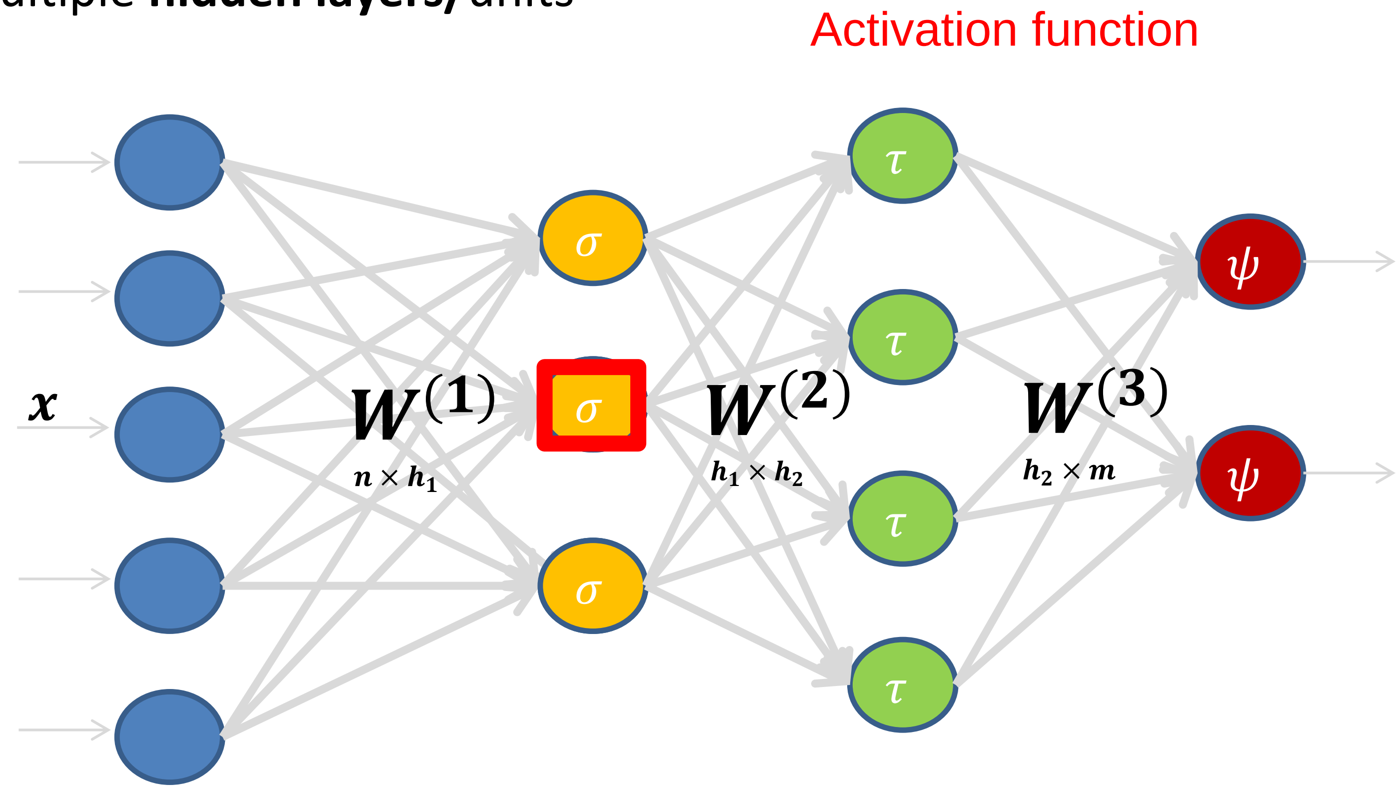
- Multiple **hidden layers/**units



- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

More complex neural networks

- Multiple **hidden layers/**units



- This is called Multi-Layer-Perceptron (MLP). More difficult in terms of optimization

Mathematically

- 1st layer computes:
  - $\boldsymbol{h}_1 = \sigma\left(\boldsymbol{x} \cdot \boldsymbol{W}^{(\mathbf{1})} + \boldsymbol{b}\right)$
  - σ is applied element-wise, $\boldsymbol{h}_1$ is a vector

- 2nd layer computes:
  - $\boldsymbol{h}_2 = \tau(\boldsymbol{h}_1 \cdot \boldsymbol{W}^{(2)} + \boldsymbol{c})$

- etc.

High-level view of neural networks

- Neural networks are mathematical functions of the form (ignoring bias terms)
  - $\mathrm{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{V}) = g(f(\boldsymbol{x} \cdot \boldsymbol{W}) \cdot \boldsymbol{V})$
  - $\boldsymbol{x}$ is the given input, g, f are also given, $\boldsymbol{W}, \boldsymbol{V}$ is what we want to *learn*

- We define a *loss function* of the form
  - $L(\boldsymbol{\theta}) = \sum_{(\boldsymbol{x},\boldsymbol{y})} ||(NN(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{y})||^2$
  - $(\boldsymbol{x}, \boldsymbol{y})$ is our *training data*
  - $\boldsymbol{\theta}$ are our *parameters* ($= \boldsymbol{W}, \boldsymbol{V}$ above)

- We want to optimize our parameters $\boldsymbol{\theta}$ such that our loss becomes minimized
  - This is called *learning* or *training*

Outline

Universität Bielefeld

Activation functions

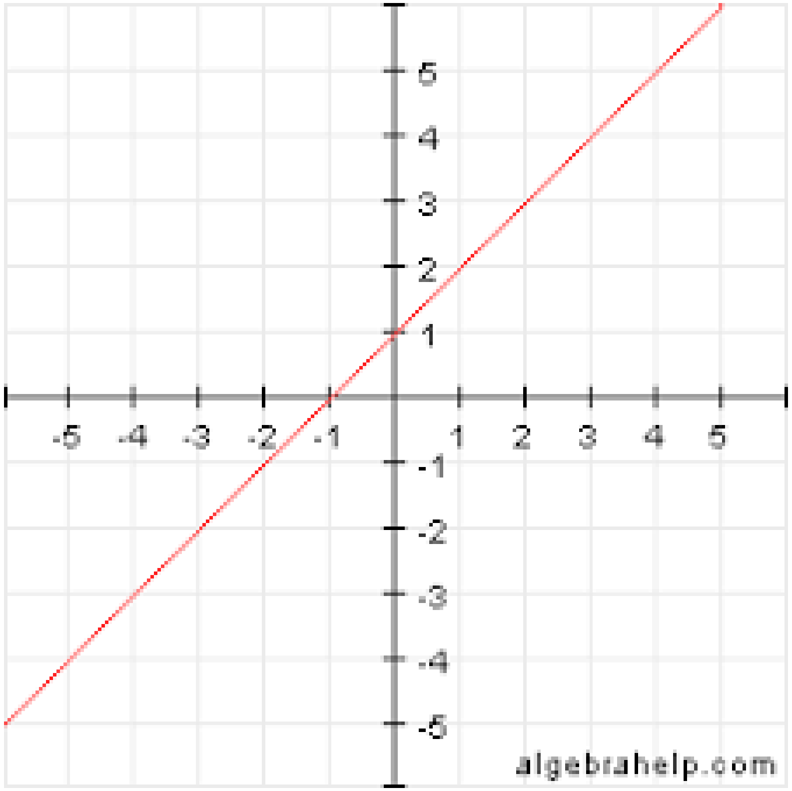
CITEC NLLG

Activation functions σ

Linear activations

Suppose all your neurons have linear activation functions …
What's your net computing?

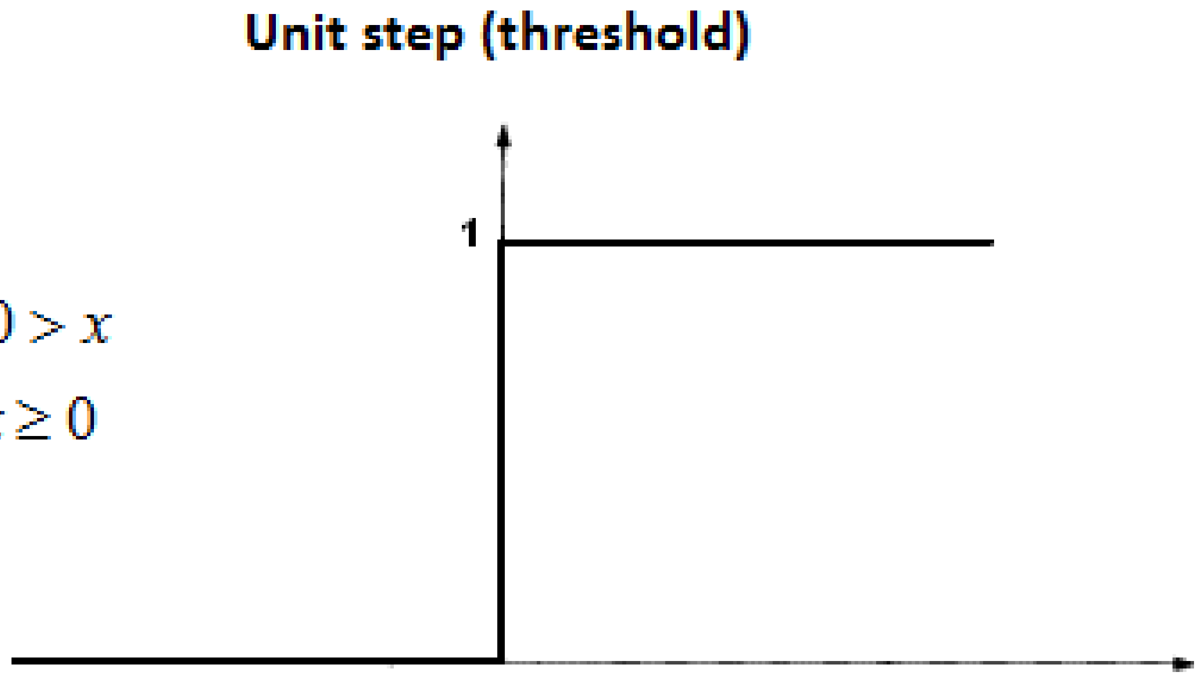
$$\sigma(x) = ax + b$$



Activation functions σ

Threshold/Step function

$$\sigma(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



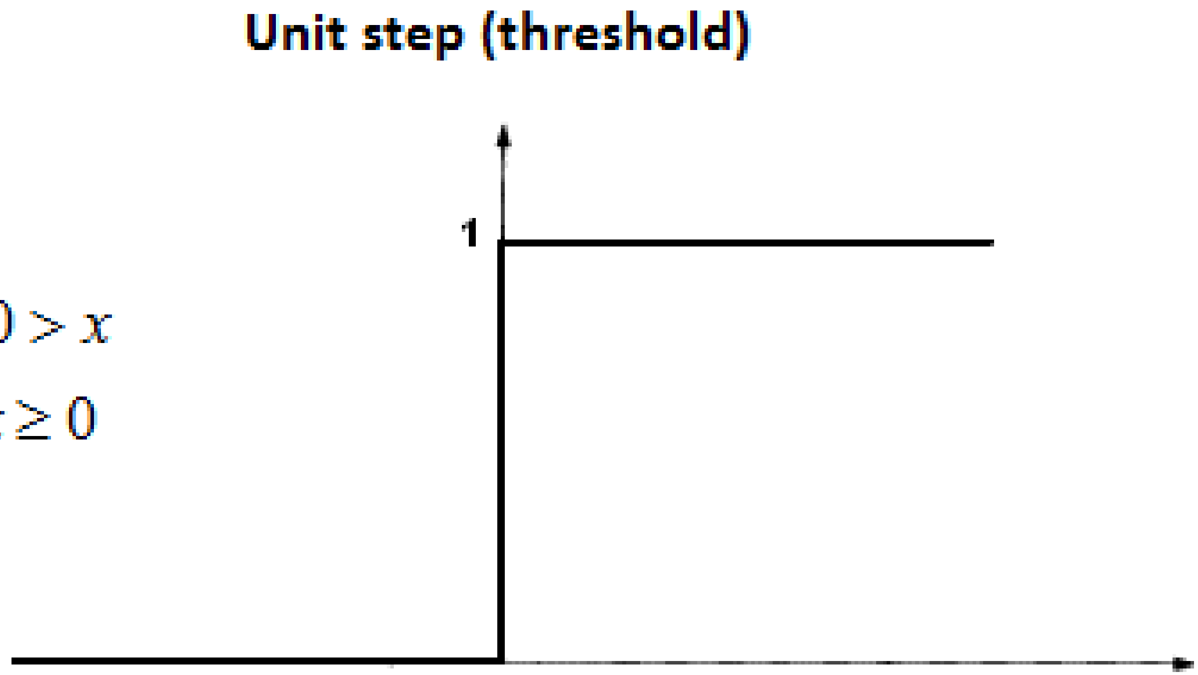
Picture from: http://kylescholz.com/projects/wordnet/, based on representation from WordNet: https://wordnet.princeton.edu

Activation functions σ

Threshold/Step function

1. Historically first specification
2. Not everywhere differentiable
3. Derivative is zero

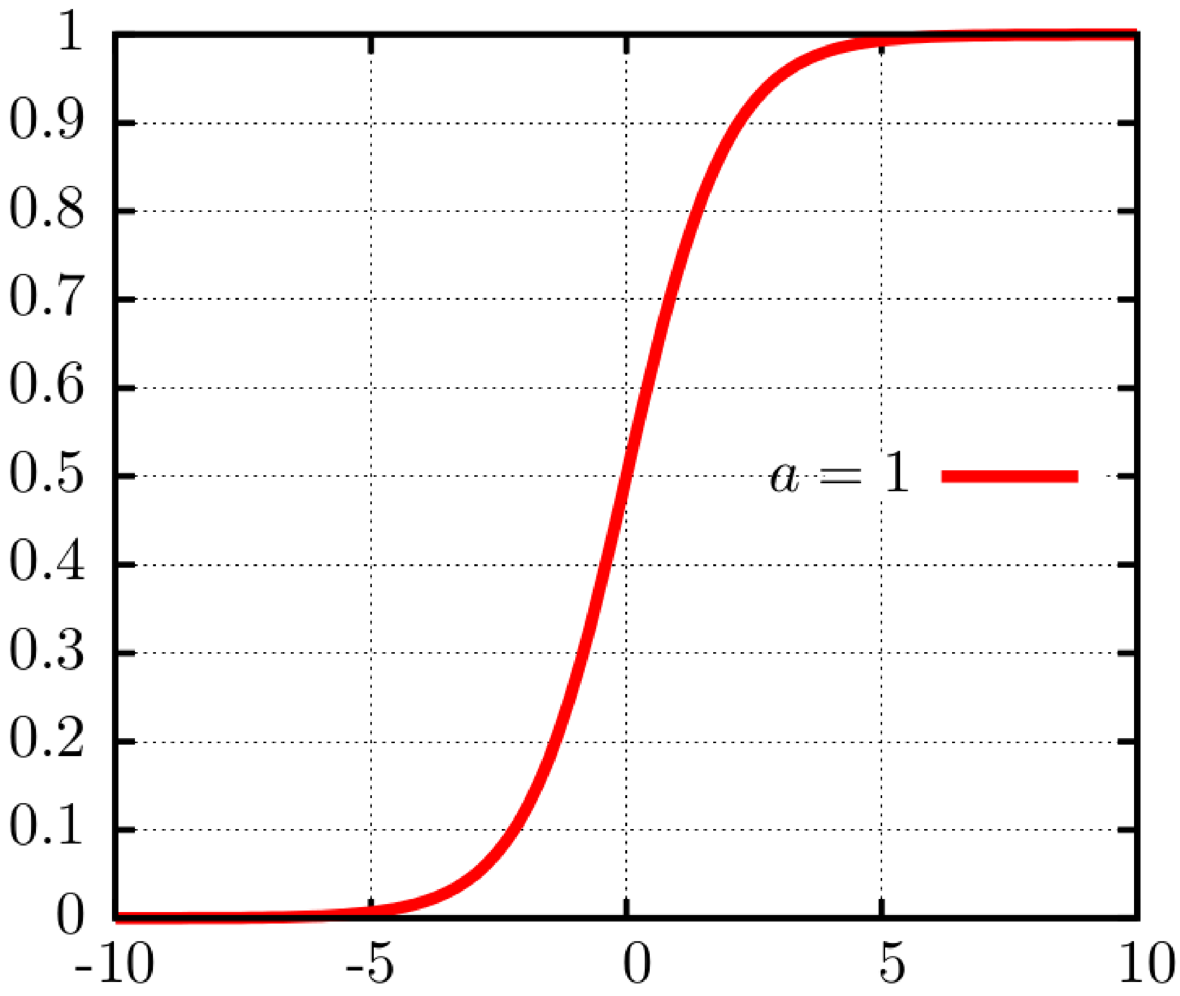
$$\sigma(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



Picture from: http://kylescholz.com/projects/wordnet/, based on representation from WordNet: https://wordnet.princeton.edu

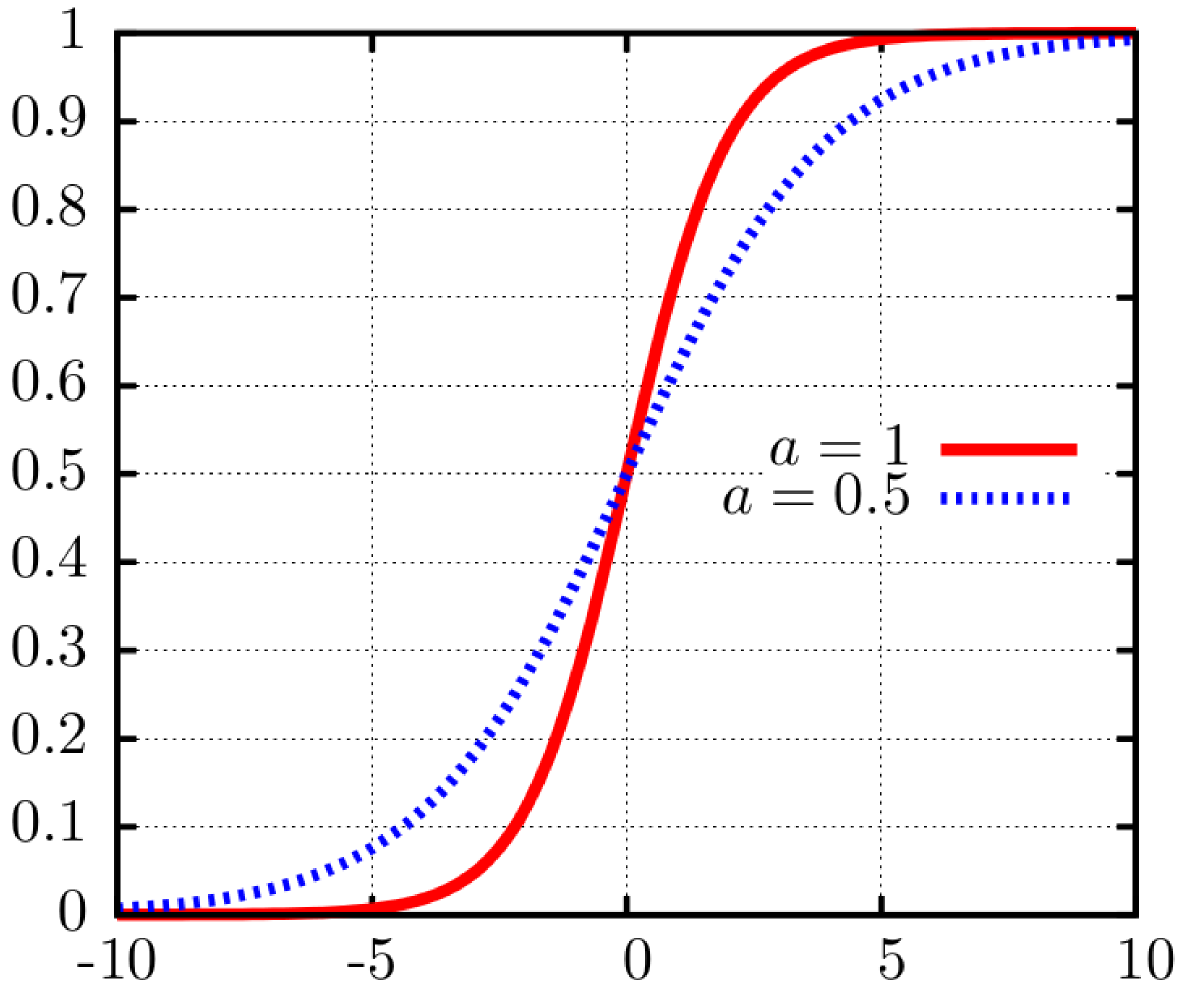
Activation functions σ

- **Sigmoid (logistic)**
- $\sigma(x) = \frac{1}{1+\exp(-ax)}$



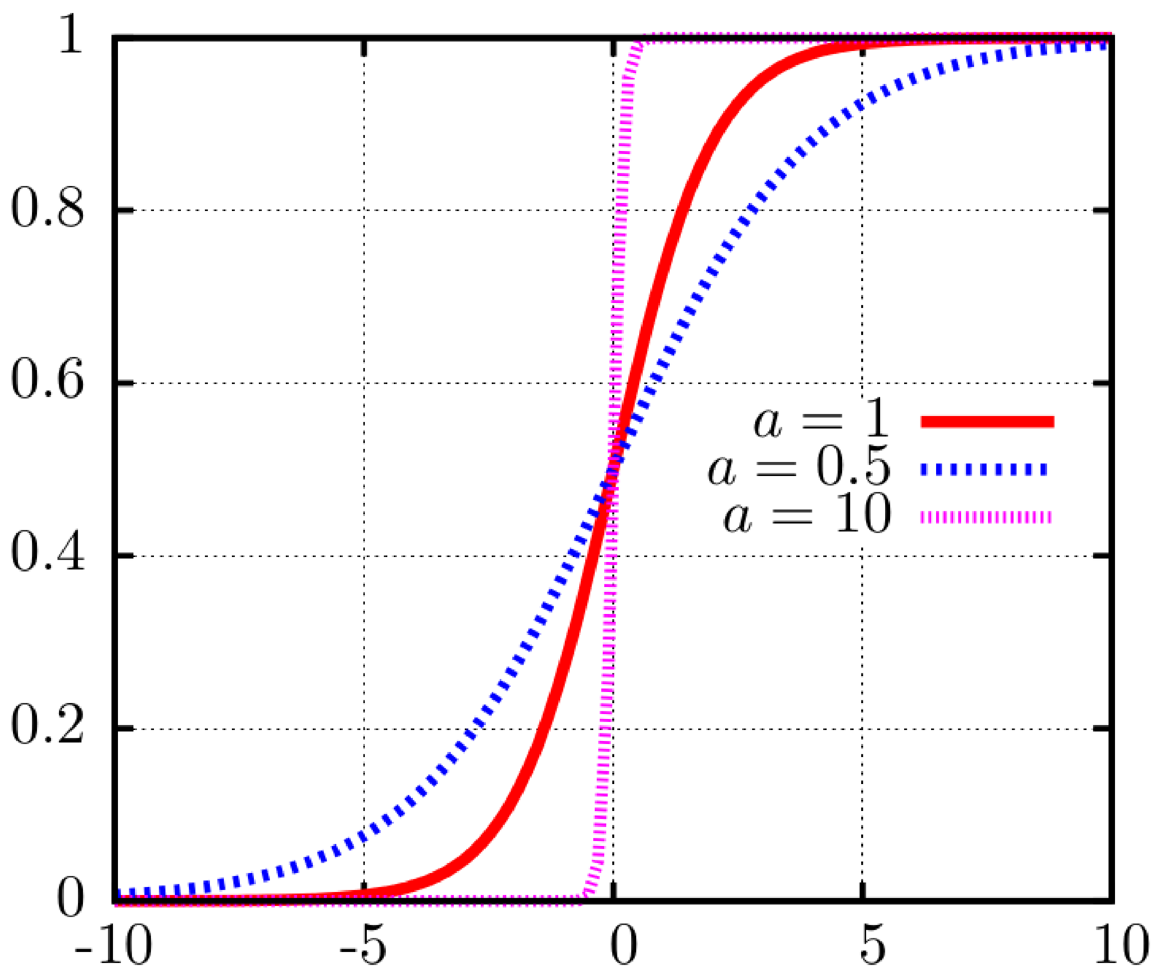
Activation functions σ

- **Sigmoid** (logistic)
- $\sigma(x) = \frac{1}{1+\exp(-ax)}$



Activation functions $\boldsymbol{\sigma}$

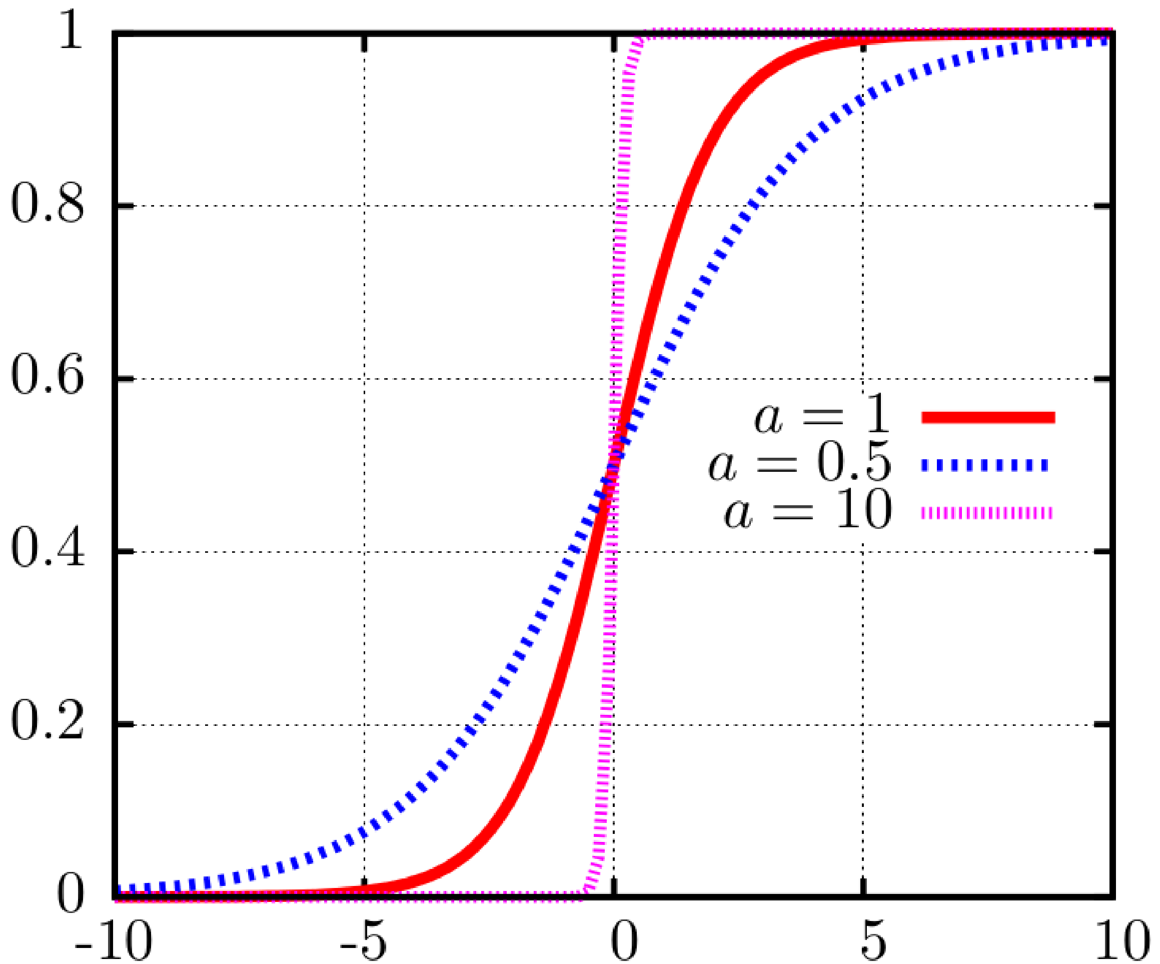
- **Sigmoid (logistic)**
- $\boldsymbol{\sigma}(x) = \frac{1}{1+\exp(-ax)}$



Activation functions σ

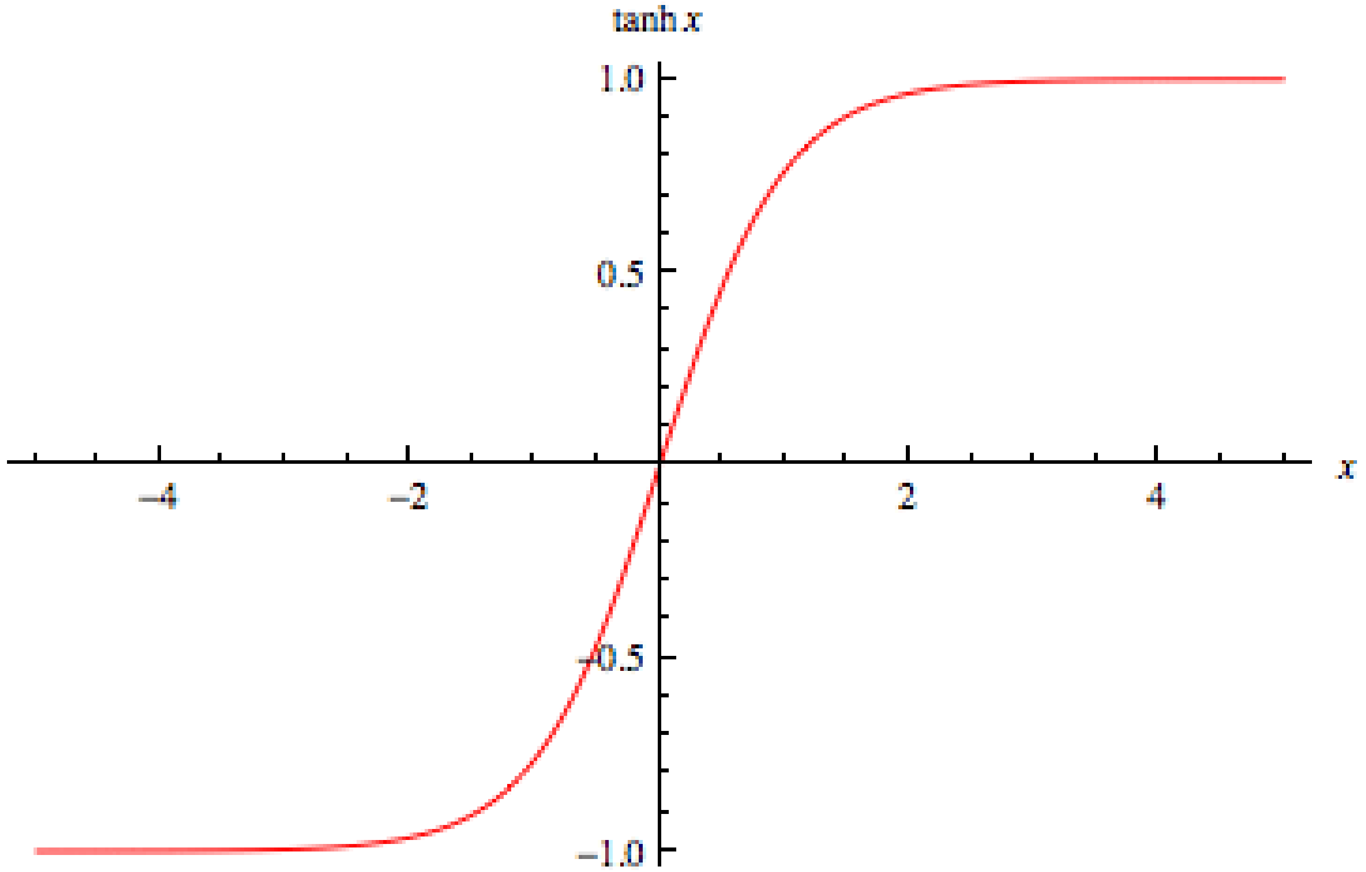
- **Sigmoid (logistic)**
- $\sigma(x) = \frac{1}{1+\exp(-ax)}$

1. Not zero-centered
2. Kills gradients



Activation functions σ

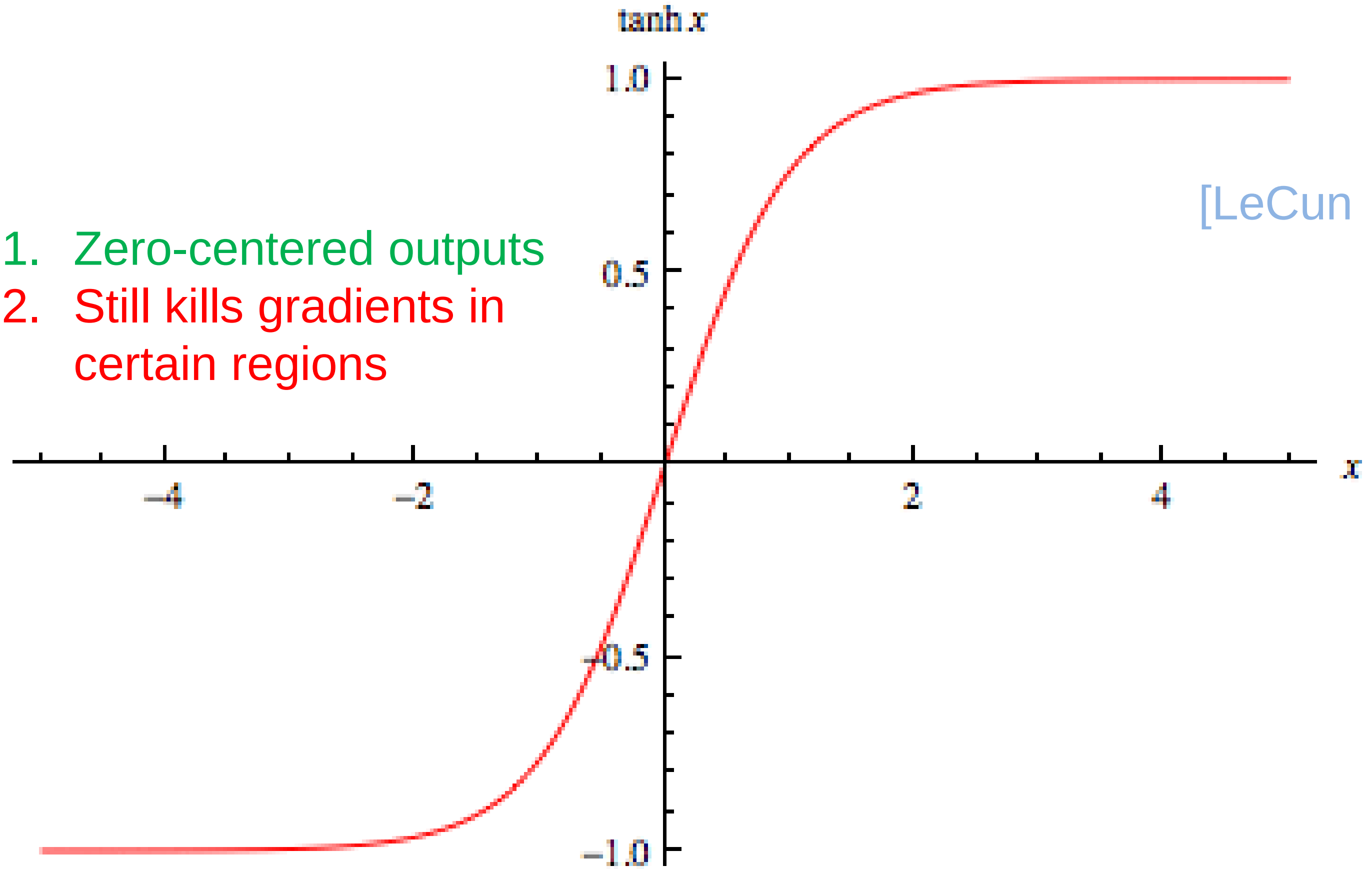
Tanh



Picture from: : http://mathworld.wolfram.com/images/interactive/TanhReal.gif

Activation functions σ

Tanh



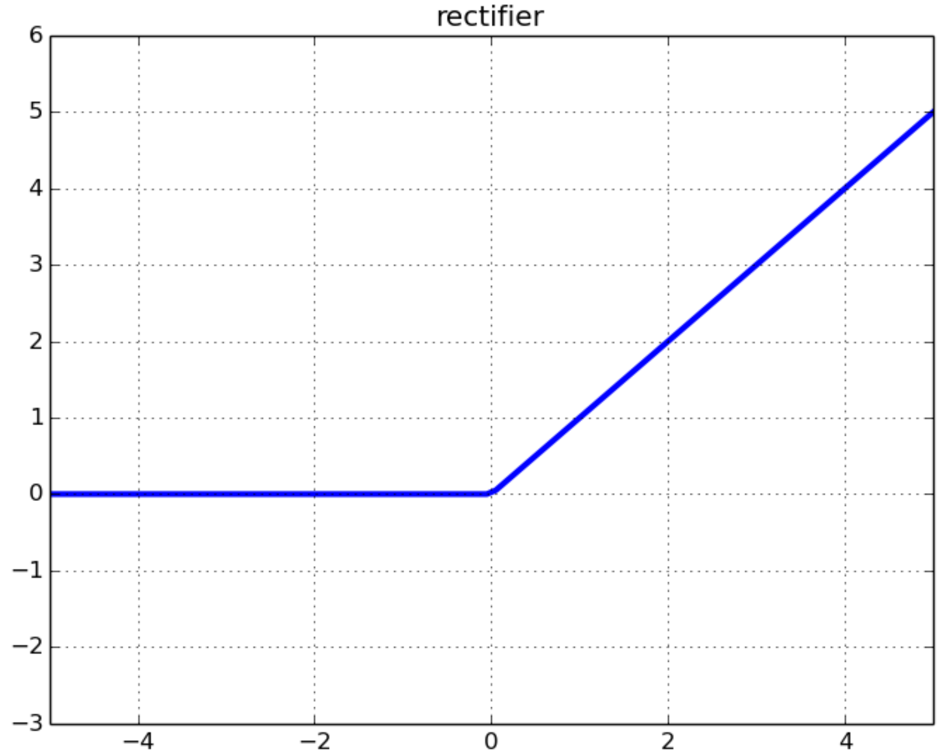
1. Zero-centered outputs
2. Still kills gradients in certain regions

[LeCun et al. 1991]

Picture from: : http://mathworld.wolfram.com/images/interactive/TanhReal.gif

Activation functions σ

- **ReLU (rectified linear unit)**
- $\mathbf{relu}(x) = \max(0, x)$



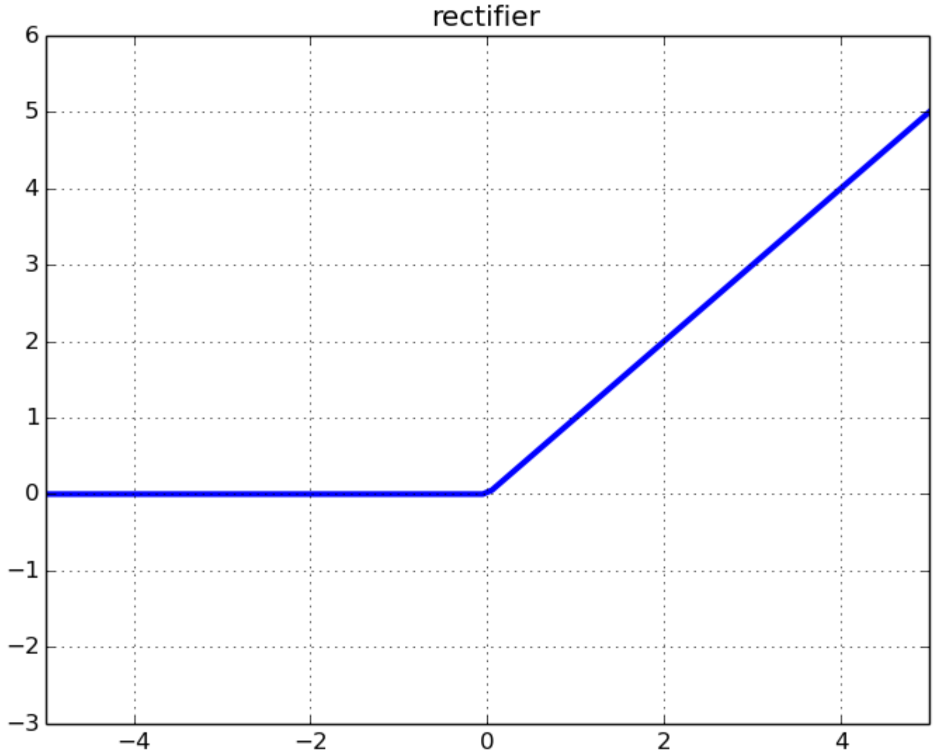
- **From http://i.stack.imgur.com/8CGlM.png**

Activation functions σ

Universität Bielefeld

- **ReLU (rectified linear unit)**
  - $\mathbf{relu}(x) = \max(0, x)$

1. Gradients don‘t die in +region
2. Computationally efficient
3. Experimentally: Convergence is faster



- From http://i.stack.imgur.com/8CGlM.png

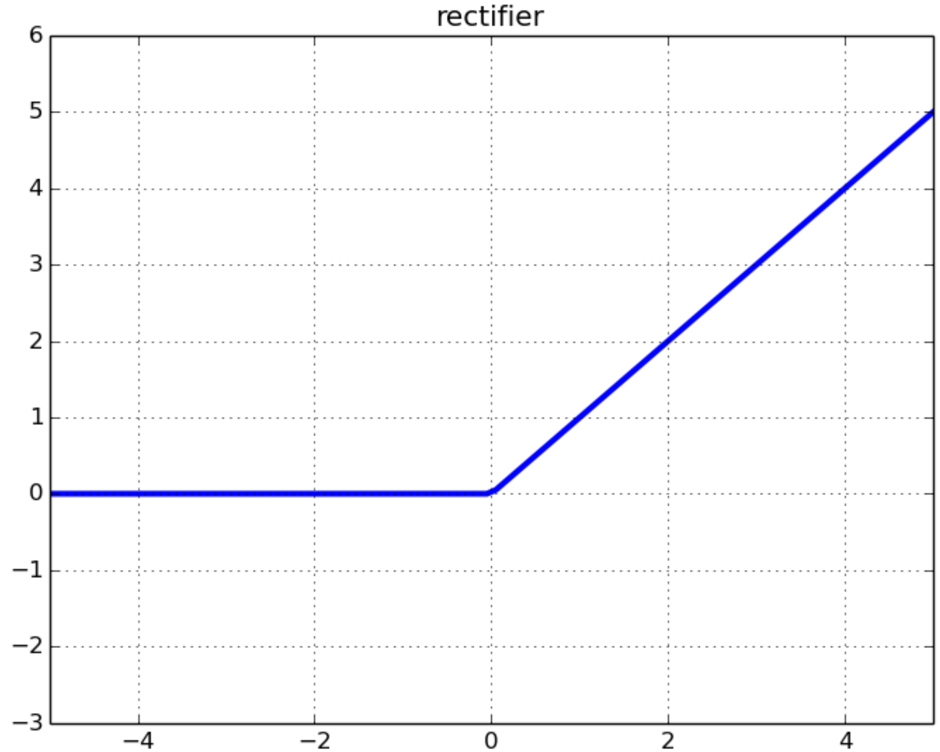
[Krizhevsky et al. 2012]

CITEC NLLG

Activation functions σ

- **ReLU (rectified linear unit)**
  - $\mathbf{relu}(x) = \max(0, x)$

1. Gradients don‘t die in +region
2. Computationally efficient
3. Experimentally: Convergence is faster



1. Kills gradients in –region
2. Not zero centered

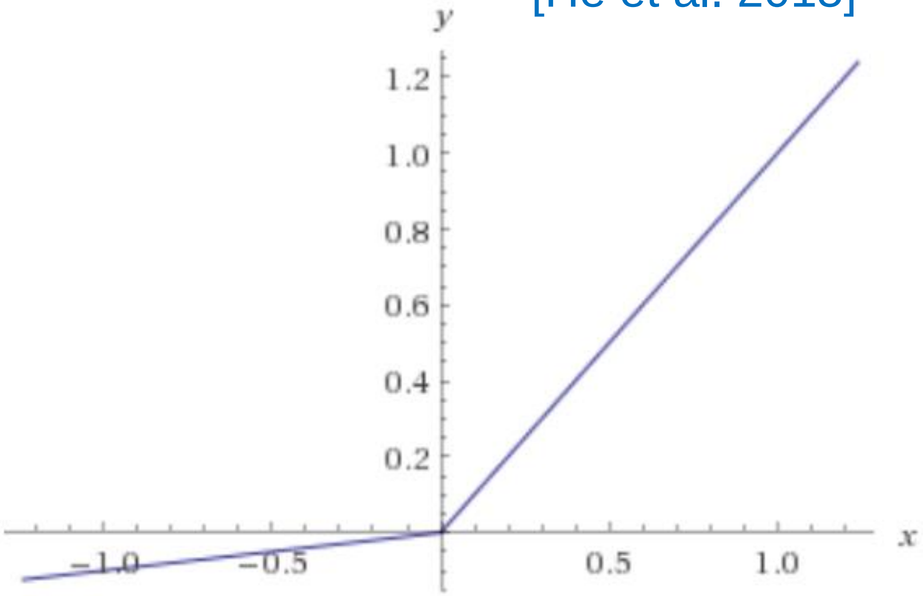
- **From http://i.stack.imgur.com/8CGlM.png**

[Krizhevsky et al. 2012]

Activation functions σ

- **Leaky ReLU (rectified linear unit)**
  - $\mathbf{relu}(x) = \max(0.01x, x)$

[Maas et al. 2013]
[He et al. 2015]



1. Gradients don‘t die in +region
2. Computationally efficient
3. Experimentally: Convergence is faster
4. Gradients don‘t die in -region

1. Not zero centered

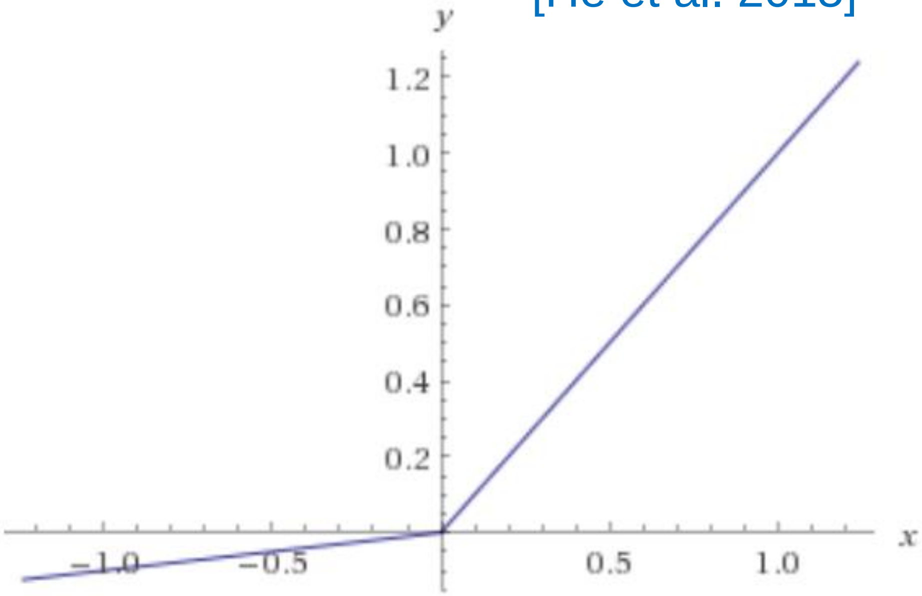
- From http://i.stack.imgur.com/8CGlM.png

Activation functions σ

- **Leaky ReLU (rectified linear unit)**
- $\mathbf{relu}(x) = \max(0.01x, x)$

[Maas et al. 2013]
[He et al. 2015]

1. Gradients don‘t die in +region
2. Computationally efficient
3. Experimentally: Convergence is faster
4. Gradients don‘t die in -region



1. Not zero centered

Paramteric ReLU
$\sigma(x) = \max(\alpha x, x)$

- From http://i.stack.imgur.com/8CGlM.png

Activation functions

- Tanh > Sigmoid
- ReLU is most popular activation function (as of 2015)
  - According to https://en.wikipedia.org/wiki/Rectifier_(neural_networks)
- Others have to stand the test of time
  - Try out Leaky ReLU / ELU
- Don’t use sigmoid or step function

Activation functions

- Tanh > Sigmoid

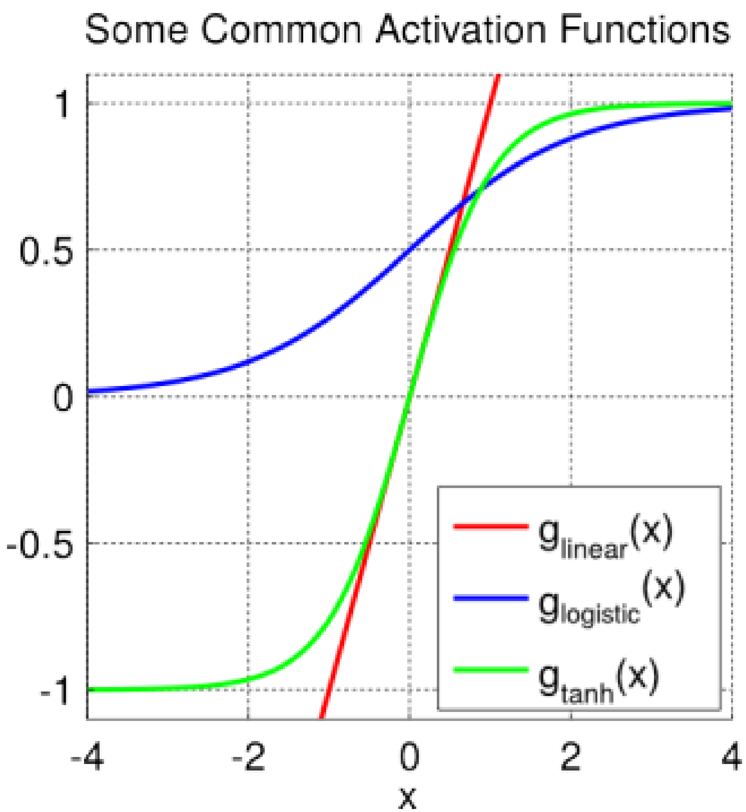
- ReLU is most popular activation function (as of 2015)
  - According to https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

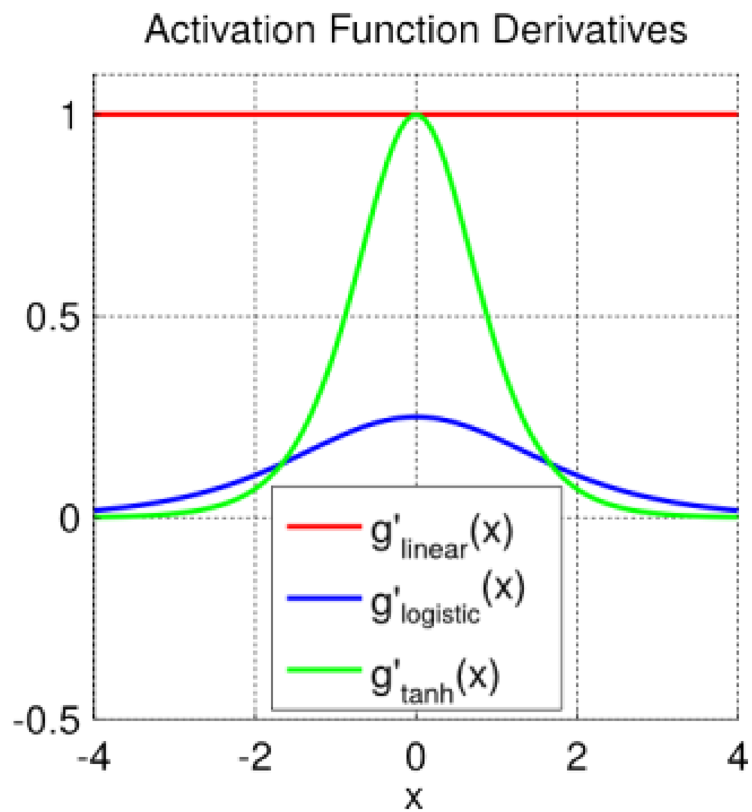
- Others have to stand the test of time

  - Try out Leaky ReLU / ELU

- Don’t use sigmoid or step function
- Activation functions are a hyperparameter!

Activation functions





A special activation function: softmax

Universität Bielefeld

- Particularly when we have multiple output classes („Positive“, „Negative“, „Neutral“),
- we‘d often like our outputs to represent a *probability distribution* over these classes
  - i.e., final outputs should sum to 1 and be non-negative

A special activation function: softmax

The softmax activation function serves that purpose:

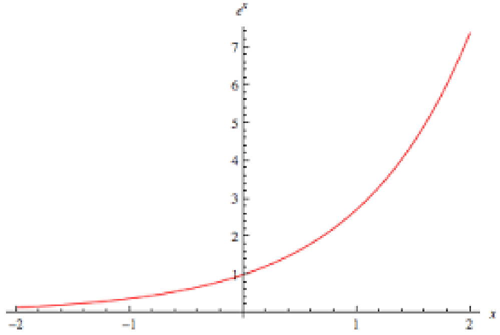
- Given m output units, the softmax activation is

$$y_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$$

- for all $\mathrm{j} = 1, \ldots, m$. Here, $z_1, \ldots, z_m$ are the pre-activations

A special activation function: softmax

- Is $y_j \geq 0$ for all j and does $\sum_j y_j = 1$?



- Note that softmax is a *global* activation function while all the other discussed previously were *local*:
  - Output of softmax for unit j depends on pre-activation of all other units j'

A special activation function: softmax

Universität Bielefeld

- Softmax challenges the interpretation of individual neurons with individual activation functions
- Instead, it supports a view of a function acting on a layer of several neurons, i.e., a vector
- I.e., let $\mathbf{z} = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ be the pre-activations in some layer
- For a standard activation function, we have as activation $\boldsymbol{y}$

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) = [\sigma(z_1) \cdots \sigma(z_n)]$$

- Softmax cannot be applied element-wise, however:

$$\boldsymbol{y} = \mathrm{softmax}(\boldsymbol{z})$$

yields a probability distribution, $\boldsymbol{y}$

To conclude

- We started with organizational stuff
- Then looked at the history of deep learning
- … and problems in NLP (these are also historically changing!)
- Afterwards, we presented the perceptron,
  - Derived a learning algorithm – you can directly go implement it
- Finally, we looked at some basics of advanced deep learning

References

- Goldberg, Yoav. "A primer on neural network models for natural language processing." *Journal of Artificial Intelligence Research* 57 (2016): 345-420.
- Bengio, Yoshua, Ian J. Goodfellow, and Aaron Courville. "Deep learning." *Nature* 521 (2015): 436-444.
- Rosenblatt, Frank. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.
- Minsky, Marvin L. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.
- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *science* 313.5786 (2006): 504-507.
- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
- Huang, Xuedong, James Baker, and Raj Reddy. "A historical perspective of speech recognition." *Communications of the ACM* 57.1 (2014): 94-103.
- Mohamed, Abdel-rahman, Dong Yu, and Li Deng. "Investigation of full-sequence training of deep belief networks for speech recognition." *Interspeech*. Vol. 10. 2010.
- Schnober, Carsten, et al. "Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks." *Proceedings of COLING* (2016).
- Kuncoro, Adhiguna, et al. "Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser." *arXiv preprint arXiv:1609.07561* (2016).