Deep Learning for Natural Language Processing

Lecture 1 – Kick-off

Dr. Steffen Eger
Wei Zhao
Niraj Pandey



Natural Language Learning Group (NLLG)
Technische Universität Darmstadt

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Administrative course issues

NLLG Lab

Deep Learning for NLP

Learning Goals

TECHNISCHE UNIVERSITÄT DARMSTADT

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 that are 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.

Learning Goals

TECHNISCHE UNIVERSITÄT DARMSTADT

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 that are 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.

General Information

TECHNISCHE UNIVERSITÄT DARMSTADT

Teaching material:

Lectures, exercises/submissions etc. can be found in Moodle

Resources

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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    - Knowledge is too quickly outdated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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- We provide literature for each topic of our course:
  - Even if you have very limited time, please do read the **mandatory part**
    - we will assume that you know these works
    - The mandatory papers (roughly one per week) will be “prüfungsrelevant”
    - We will ask one/two questions from the mandatory papers in the home exercises
  - We encourage you to read the **optional parts** as well
    - Optional are the references listed in each lecture
- We can give you additional literature hints → feel free to ask
- If you cannot find/access a publication → feel free to ask

Practice Class

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Every Tuesday 15:20 – 17:00, Room S207/167**
- First meeting will be today!

- 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
- Obtain a bonus for your exam

- Practice class is organized by Michael Bugert

Practice Class: Bonus System

TECHNISCHE UNIVERSITÄT DARMSTADT

- By participating in the practice class, you can get a bonus of 0.3 (or 0.4) for the exam – maybe even a full grade

- The following rules apply:
  1. You need to reach 70 out of 100 points for the home exercises
     - ~10 homeworks a 10 points each
  2. You need to participate in a “shared task”
     - Will take place in the 2nd half of the course
  3. You have to pass the exam without the bonus, i.e. the bonus cannot turn a 5 into a 4
  4. The bonus can only be used in the exams in this semester
  5. The bonus can only be used once per student

Practice Class: Shared Task

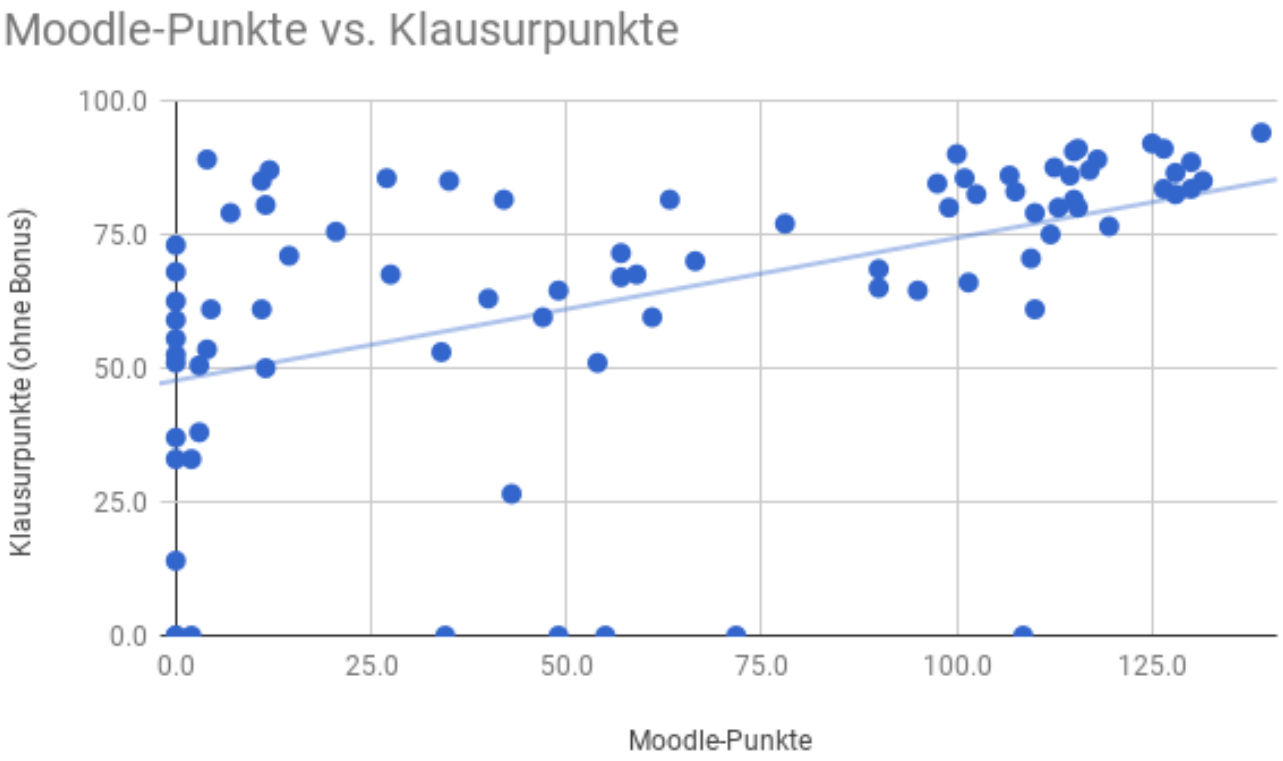
TECHNISCHE UNIVERSITÄT DARMSTADT

- You will work on generating poetry
- You will work in groups of up to 3 people
- You will develop a solution to the problem and then compete with other groups
- Shared task is graded and gives up to 100 points. Grading is based on:
  - A written report about your system (60 points)
  - your ranking compared to other teams -> better system, more points (30 points)
  - A short presentation in the last class (10 points)

Increase your odds!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Exam results and participation in exercises correlate.



NLLG

Programming Framework:

TECHNISCHE UNIVERSITÄT DARMSTADT

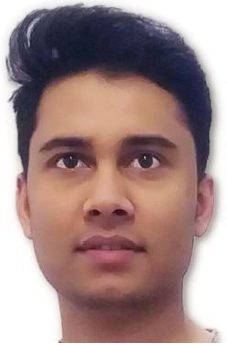
- Python, Numpy
- Tensorflow
- Keras

→ more information this afternoon

Tutors

TECHNISCHE UNIVERSITÄT DARMSTADT

- Niraj Pandey
  - „Sprechstunde“ online via appointment



- Jan-Micha Bodensohn
  - „Sprechstunde“ online via appointment

- If you have questions, contact them via moodle

Participate!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Common practice:
  study the slides in a 24h marathon just before the exam
- 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!

NLLG

Participate!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Common practice:
  study the slides in a 24h marathon just before the exam
- 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 well prepared for the exam and for other courses, thesis, future job without the 24h marathon!**

NLLG

Final Exam

TECHNISCHE UNIVERSITÄT DARMSTADT

Save the date:

Date will be announced

Register via TUCaN, see also:

https://www.informatik.tu-darmstadt.de/de/studierende/studienbuero/pruefungsan-und-abmeldung/

Exam questions will be in German and English.

Answers may be in German or English.

Final Exam: Resources Allowed

TECHNISCHE UNIVERSITÄT DARMSTADT

- Scope: Lecture, mandatory readings, and practice class
- Final grade: Exam (100%) + Bonus System
- You may bring a dictionary to the exam, if neither German nor English is your first language.
- You may bring a non-programmable calculator to the exam.
- No other resources (books, lecture notes etc.) are allowed during the exam.

Questions/Suggestions

TECHNISCHE UNIVERSITÄT DARMSTADT

Any questions, suggestions,…?

1. Post a message in the Moodle forum
2. *If that doesn‘t help:* Write an E-Mail to the tutors or make an appointment with them
3. *If that doesn‘t help also:* Write an email to Michael or me

NLLG

Detailed Syllabus

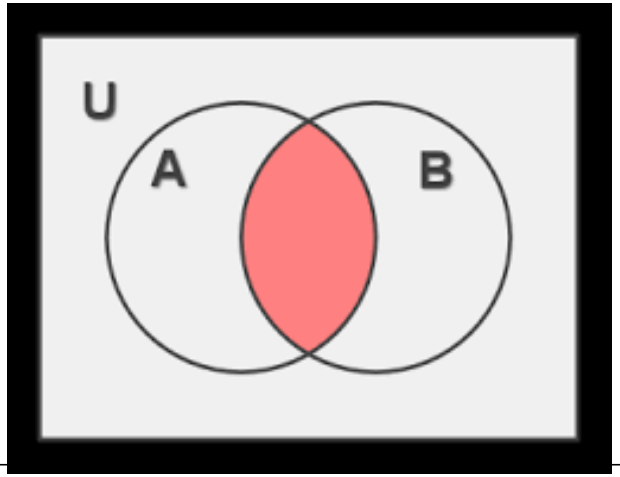
TECHNISCHE UNIVERSITÄT DARMSTADT

1	History+Perceptrons
2	ML background
3	Backpropagation – Learning in deep neural nets
4	Word Embeddings 1 – CBOW and Skip-Gram
5	Dependency Parsing
6	Word Embeddings 2 – Bilingual, Syntactic, Contextualized Embeddings
7	Word Embeddings 3 – Sentence Embeddings
8	Convolutional networks
9	Recurrent neural networks
10	Encoder-Decoder Neural Nets
11	Mock exam (Probe-Klausur)
12	Current Trends
13	Guest lectures
14	Summary and Shared Task award presentations

ICR

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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 itself will then be shorted to about 30min
- And so will the exercise (takes then place in lecture room)
- When we have ICR, there will be no separate exercise slot from 15-17



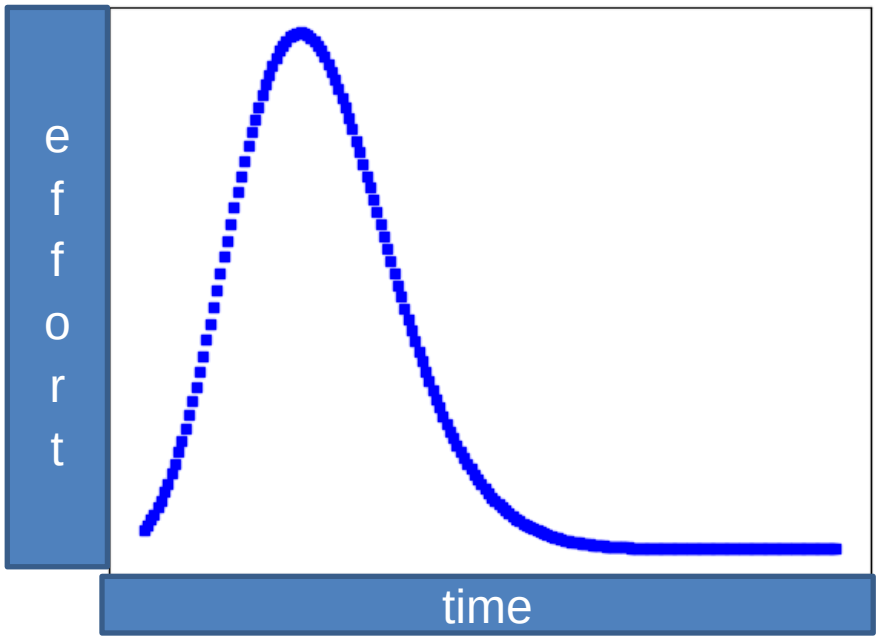
A = Video
B = Lecture

NLLG

Effort Curve

TECHNISCHE UNIVERSITÄT DARMSTADT

Effort curve in the lecture

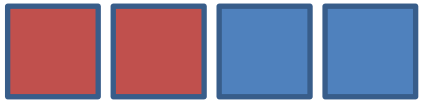


NLLG

Required skills

TECHNISCHE UNIVERSITÄT DARMSTADT

- Math



- Programming

- Linguistics

- Natural Language Processing

Outline

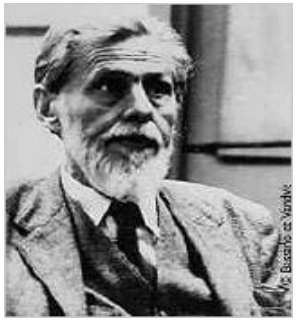
TECHNISCHE UNIVERSITÄT DARMSTADT

Perceptrons

Origins

TECHNISCHE UNIVERSITÄT DARMSTADT

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

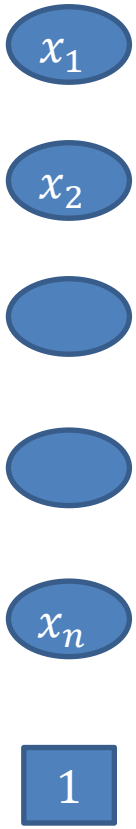


http://www.monizone.de/projects/knn/images/mcculloch_160.jpg

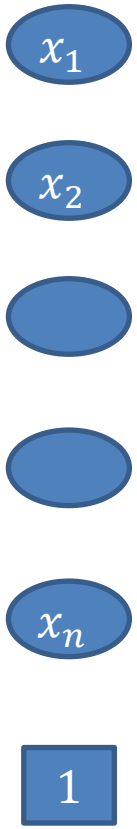
http://www.i-programmer.info/babbages-bag/325-mcculloch-pitts-neural-networks.html

NLLG

Network structure

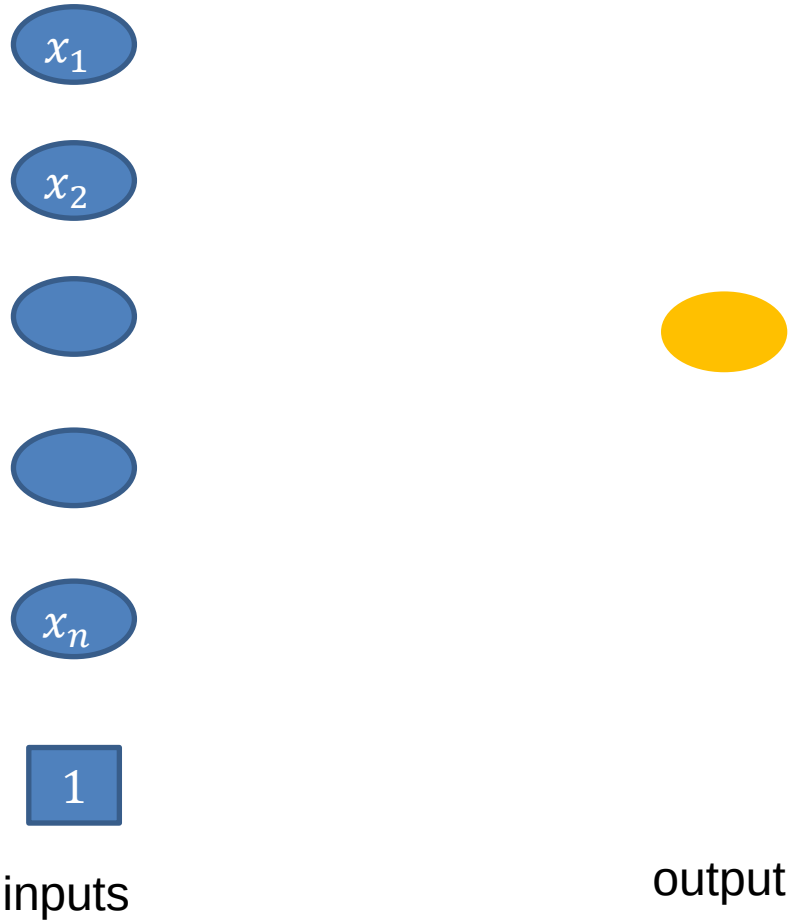


Network structure

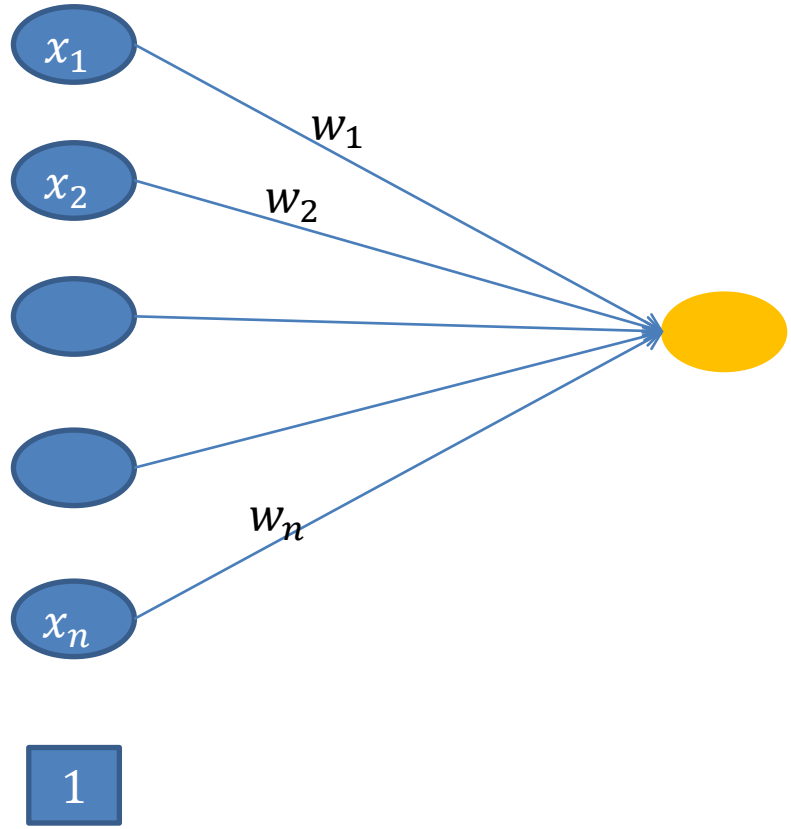


input neurons

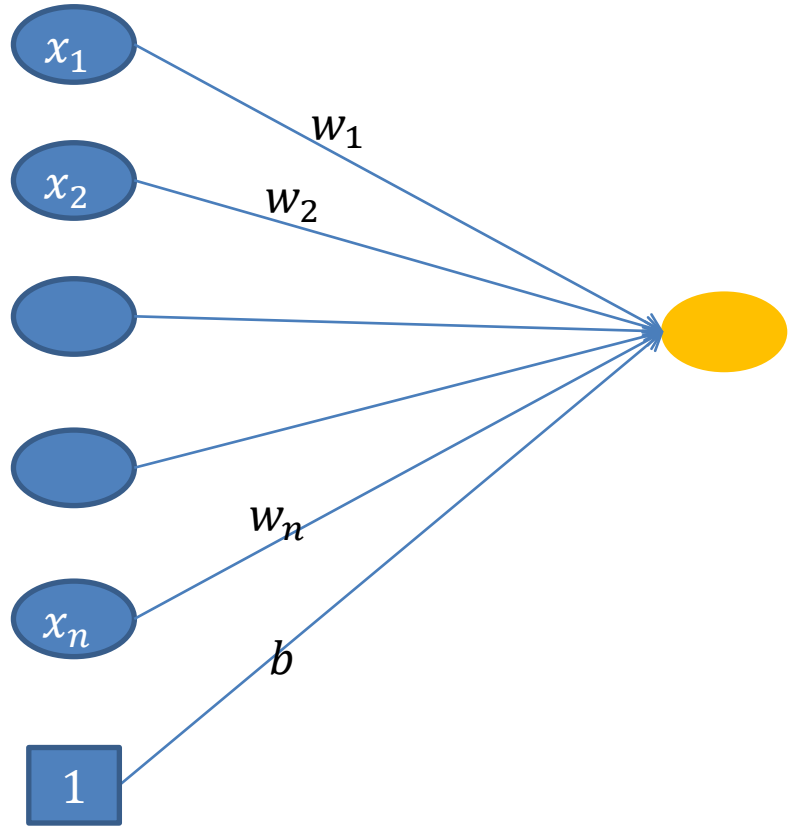
Network structure



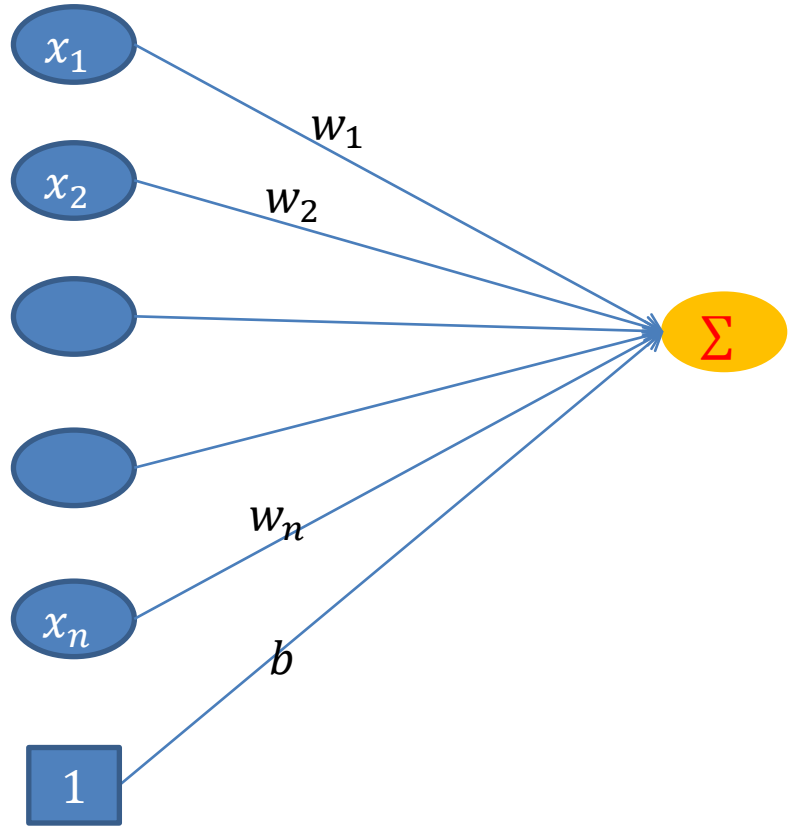
Network structure



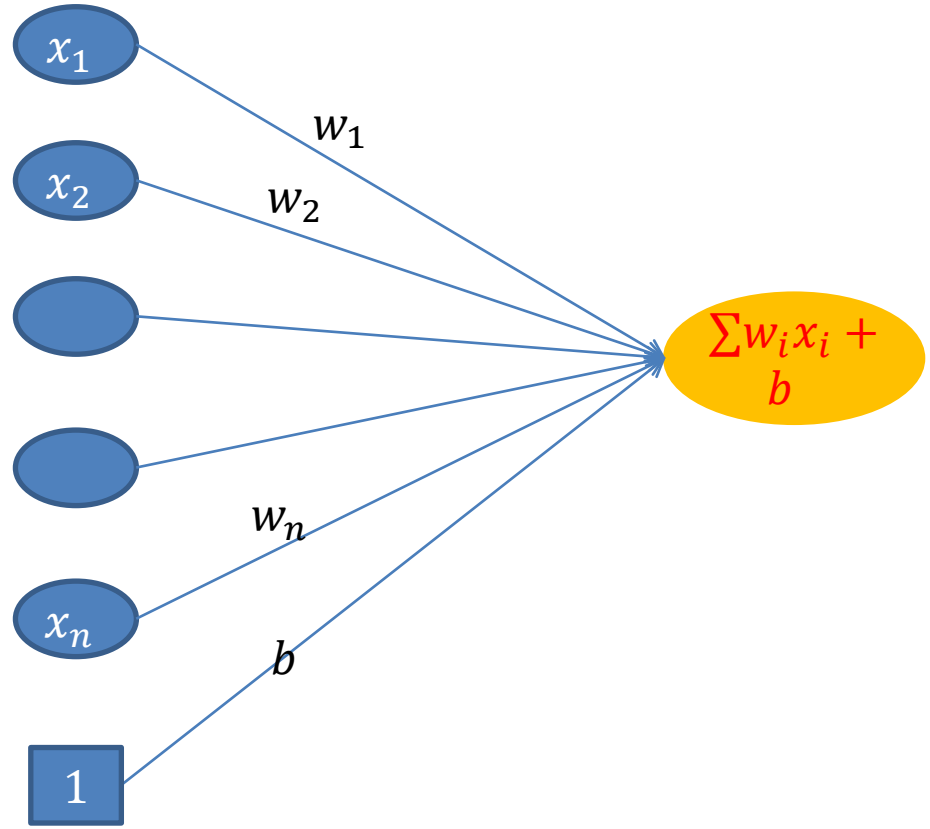
Network structure



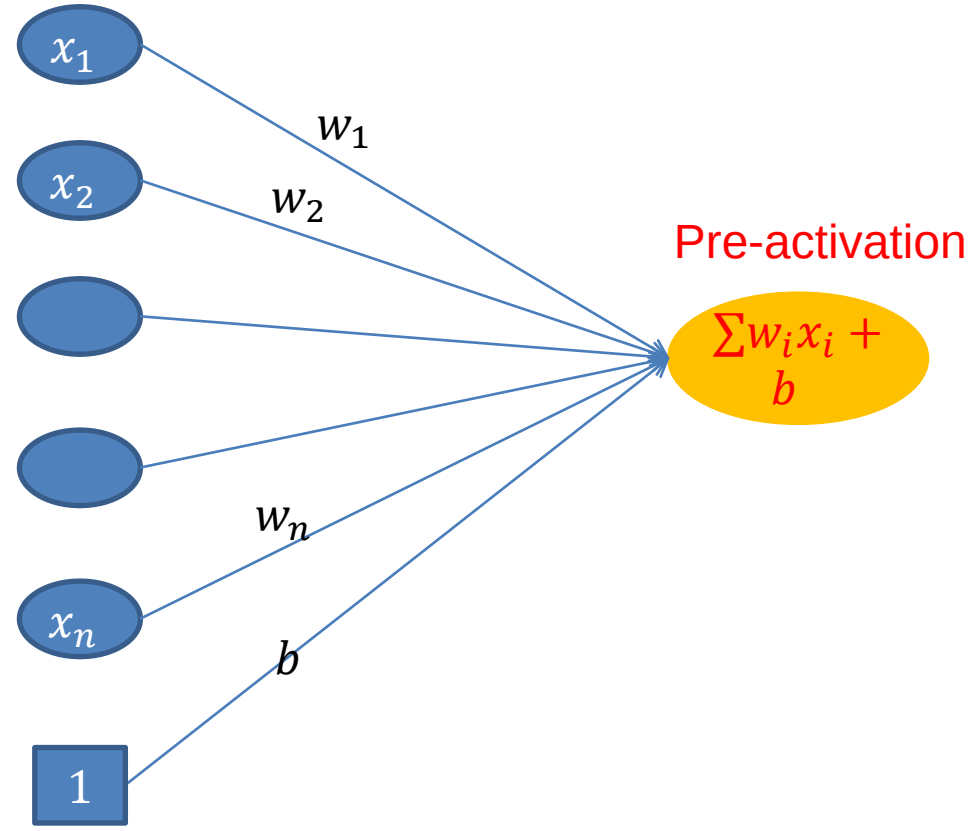
Network structure



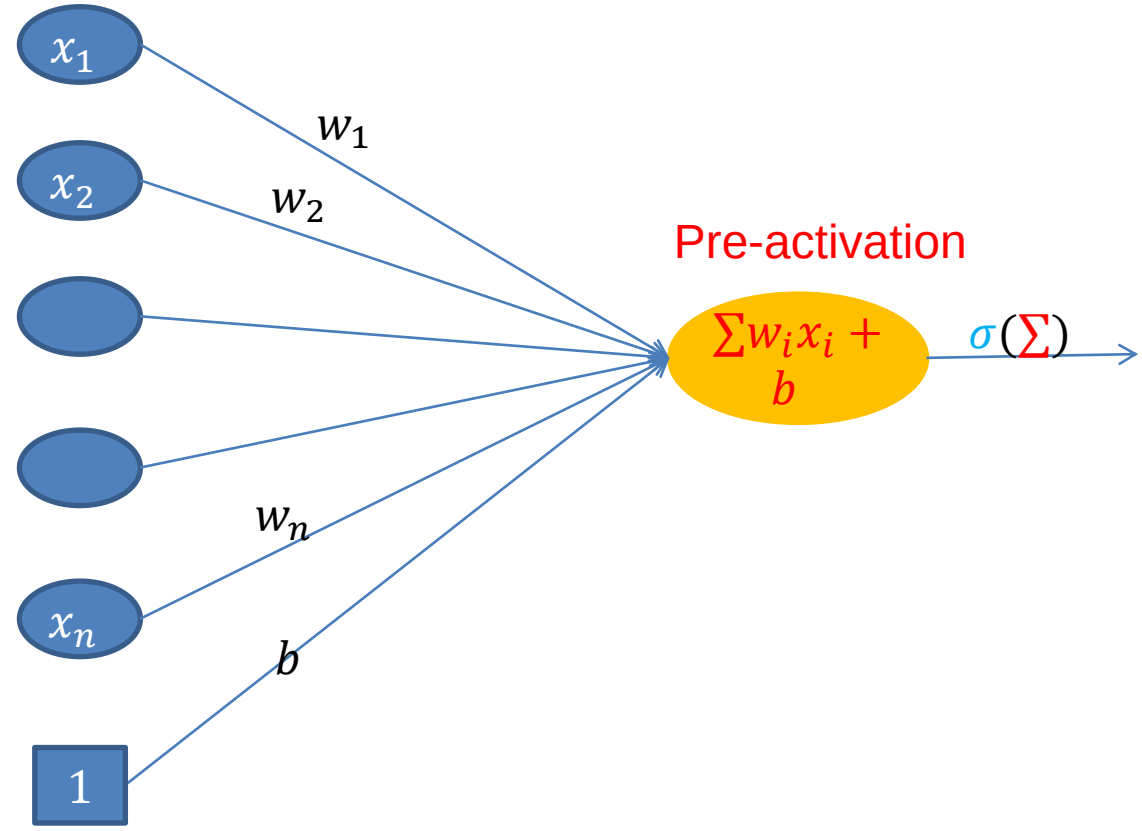
Network structure



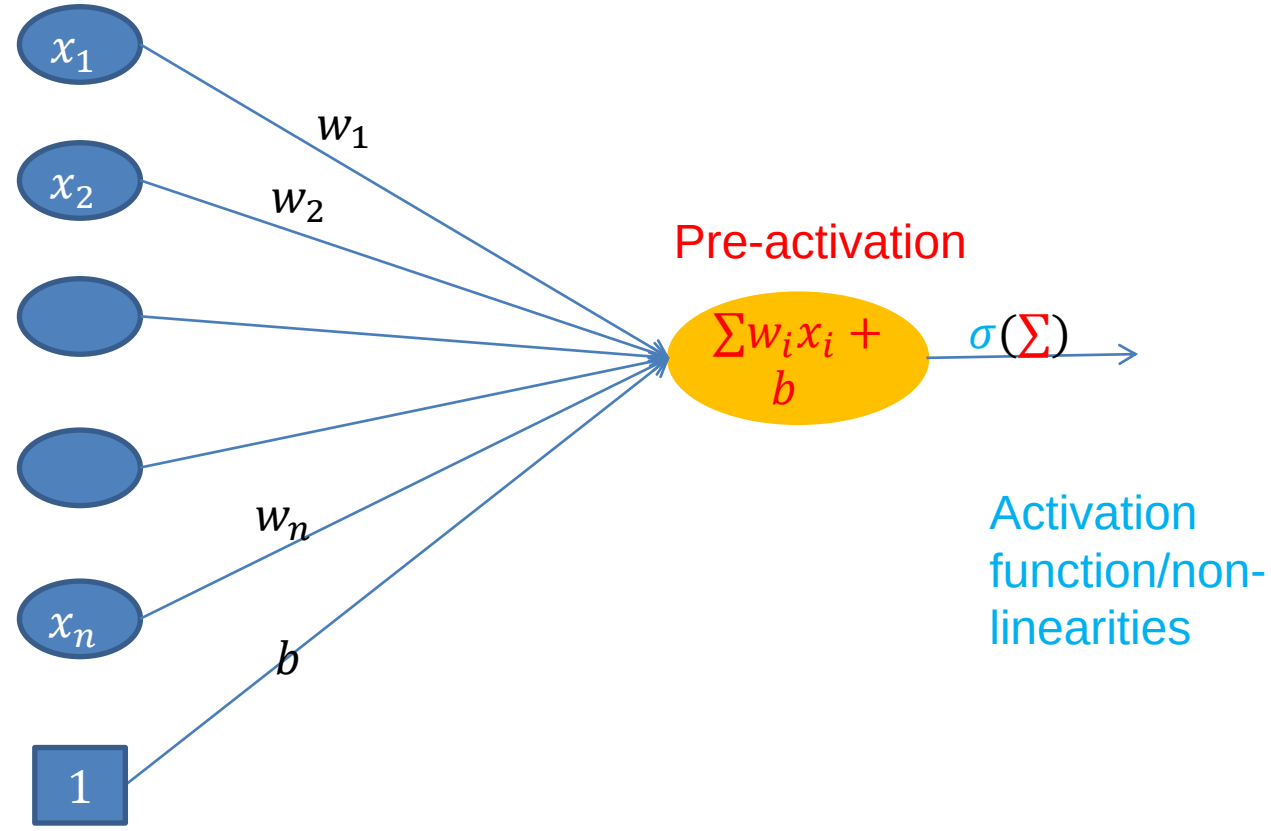
Network structure



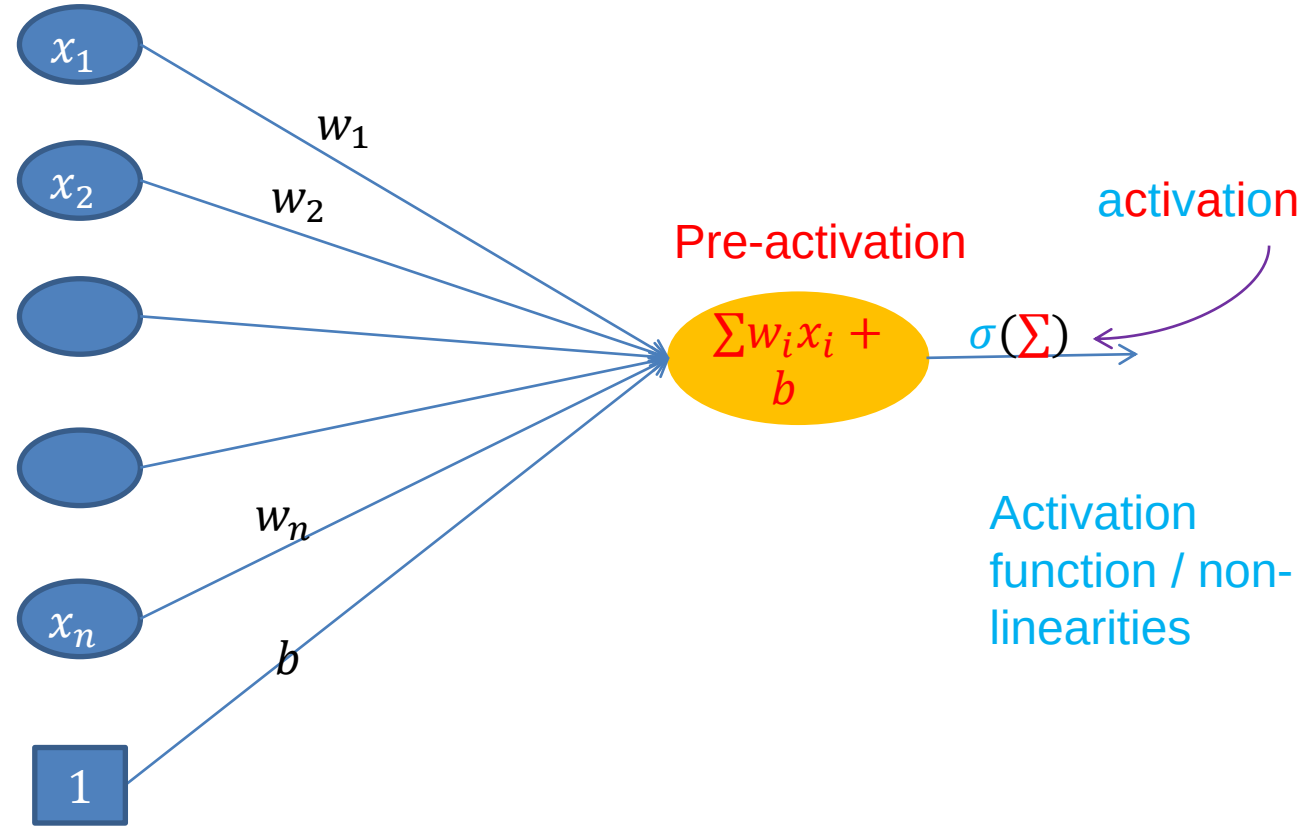
Network structure



Network structure



Network structure



A formal description

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, $b \in \boldsymbol{R}$
  - **Non-linearity** $\sigma : \mathbb{R} \rightarrow \mathbb{R}$
- **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**
  - **Input:** $\mathbf{x} \cdot \mathbf{w} + b = \sum_{i=1}^{n} w_i \cdot x_i \quad + b$
  - **Output:** $\sigma(\mathbf{x} \cdot \mathbf{w} + b)$

A formal description

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, $b \in \boldsymbol{R}$
  - **Non-linearity** $\sigma : \mathbb{R} \rightarrow \mathbb{R}$
- **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**
  - **Input:** $\mathbf{x} \cdot \mathbf{w} + b = \sum_{i=1}^{n} w_i \cdot x_i \quad + b$
  - **Output:** $\sigma(\mathbf{x} \cdot \mathbf{w} + b)$

Dot product
(„Skalarprodukt“)

NLLG

A formal description

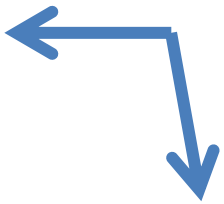
- **Given**
  - **Weight vector** $\mathbf{w} \in \mathbb{R}^n$, ~~$b \in \boldsymbol{R}$~~
  - **Non-linearity** $\sigma : \mathbb{R} \rightarrow \mathbb{R}$
- **Input to network**
  - **Input vector** $\mathbf{x} \in \mathbb{R}^{1 \times n}$, for $n \geq 1$.
  - **And constant** ~~$1 \in \boldsymbol{R}$~~
- **Output unit**
  - **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

TECHNISCHE UNIVERSITÄT DARMSTADT

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

A formal description

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

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

(Training/test) Data

- **Output unit**
  - **Input:** $\tilde{x} \cdot w$
  - **Output:** $\sigma(\tilde{x} \cdot w) = f_w(\tilde{x})$

(Statistical) Model, parametrized by w [w often also denoted as θ]

NLLG

Questions

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

Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Optimization problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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]

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

[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

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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 $\boldsymbol{1 \leq N' \leq N}$, 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?

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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*
    - Then we "randomize" this, arriving at something called *stochastic gradient descent* – or mini-batch learning
- We'll see more on this in lectures 2&3

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Limitations of Perceptrons

Decision surface of a perceptron

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

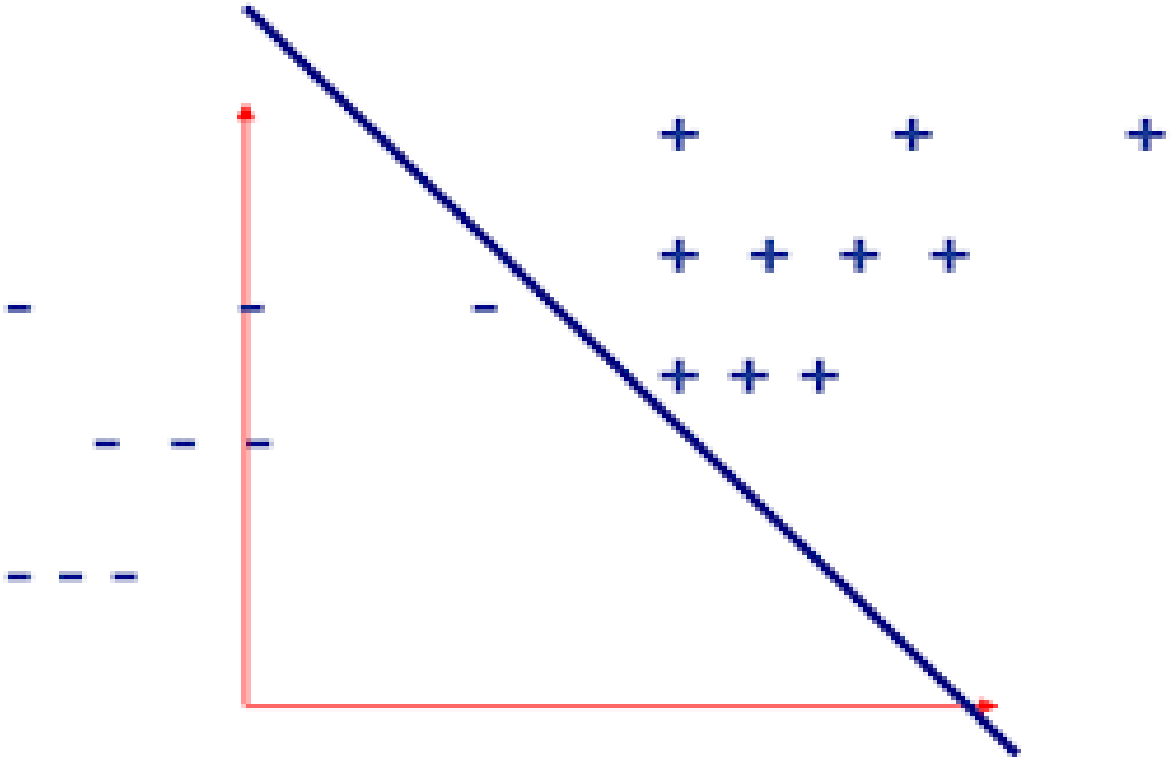
- Consider the *hyperplane* $H_{\boldsymbol{w}}$

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

- In 2-d: $x_1 w_1 + x_2 w_2 + w_{n+1} = 0$ 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 $\widetilde{\boldsymbol{x}} \cdot \boldsymbol{w} > 0$, and every point $\boldsymbol{x} \in X_1$ satisfies $\widetilde{\boldsymbol{x}} \cdot \boldsymbol{w} < 0$. 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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:** When you want to determine if two sets are linearly separable, you can equally ask yourself how to 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)

What a perceptron canNOT do

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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,

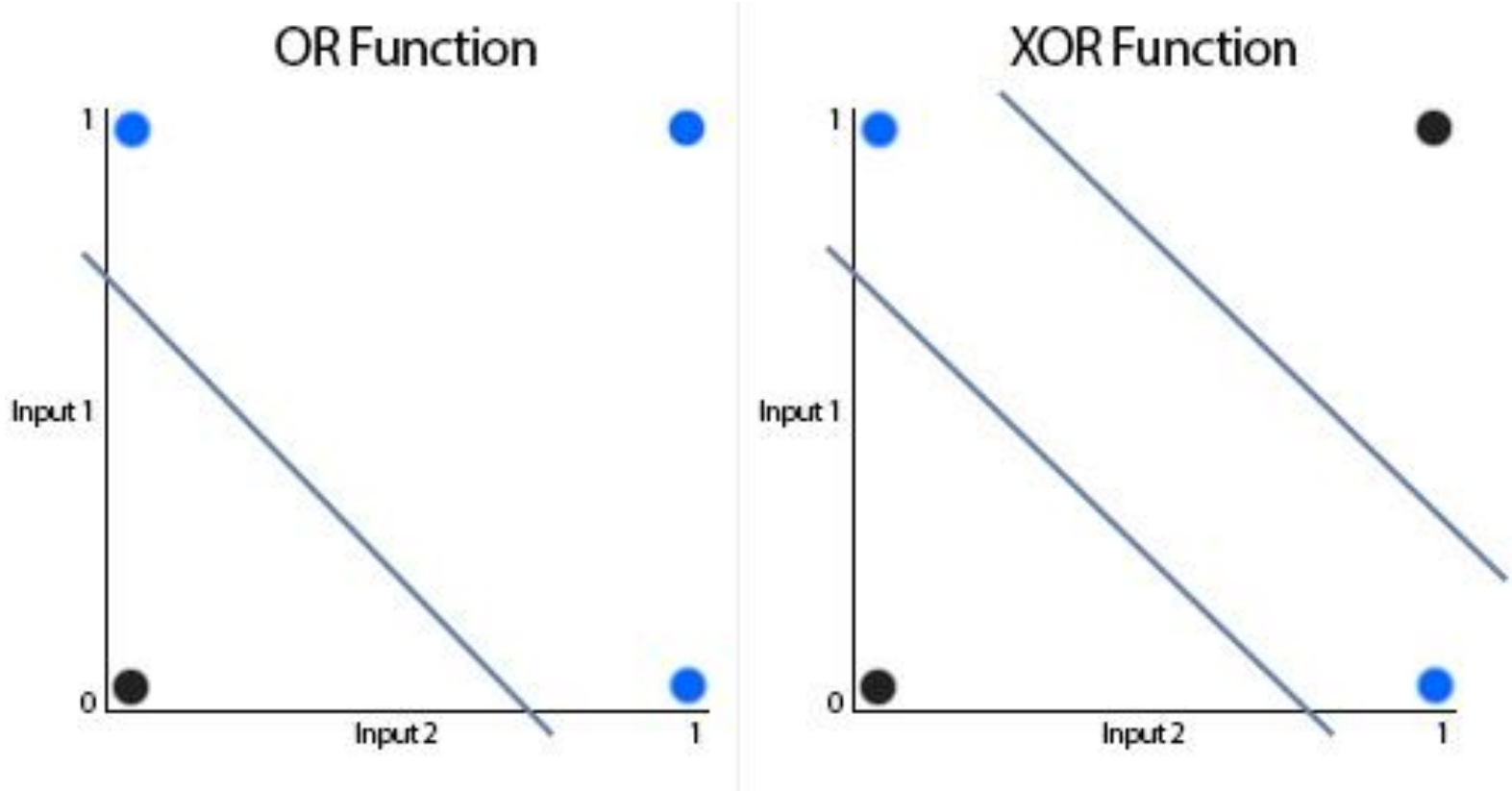
$$w_1 + w_3 + w_2 + w_3 < 0$$

by the last two equations.

This is a contradiction.

NLLG

What a perceptron canNOT do



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

Outline

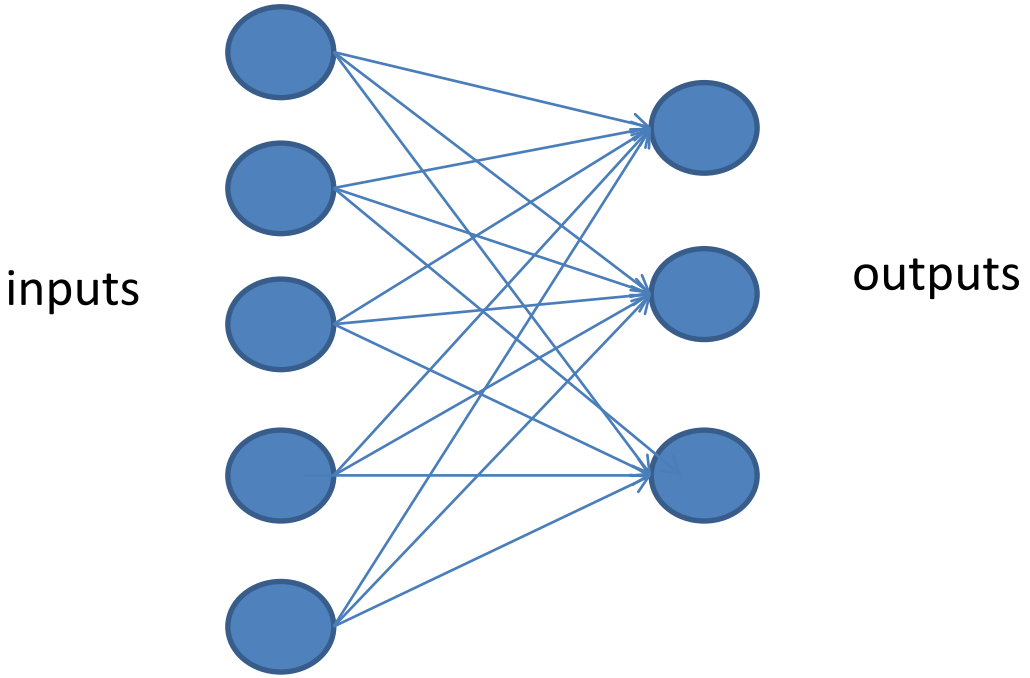
TECHNISCHE UNIVERSITÄT DARMSTADT

Neural Networks Principles

More complex neural networks

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

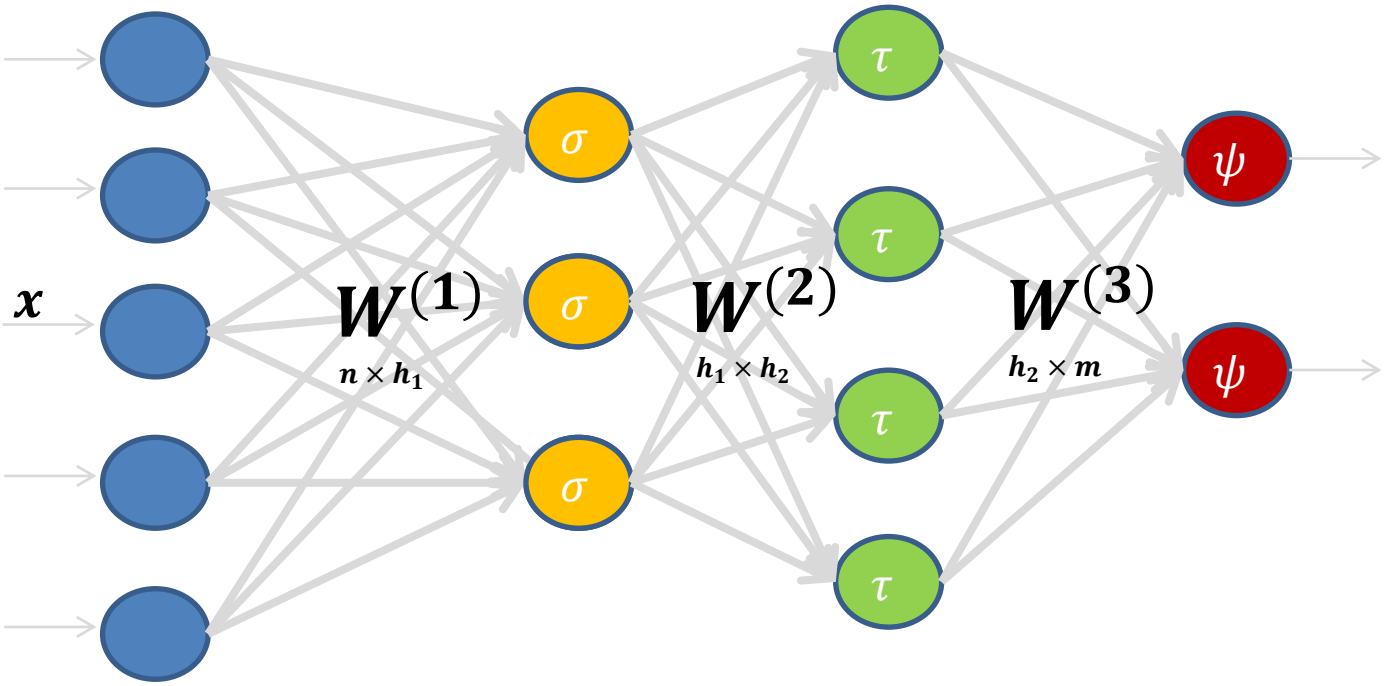
- No additional technical difficulty, can use the previous learning techniques

NLLG

More complex neural networks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multiple **hidden layers/**units



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

NLLG

Mathematically

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1st layer computes:
  - $\boldsymbol{h}_1 = \sigma\left(\boldsymbol{x} \cdot \boldsymbol{W}^{(1)} + \boldsymbol{b}\right)$
  - 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

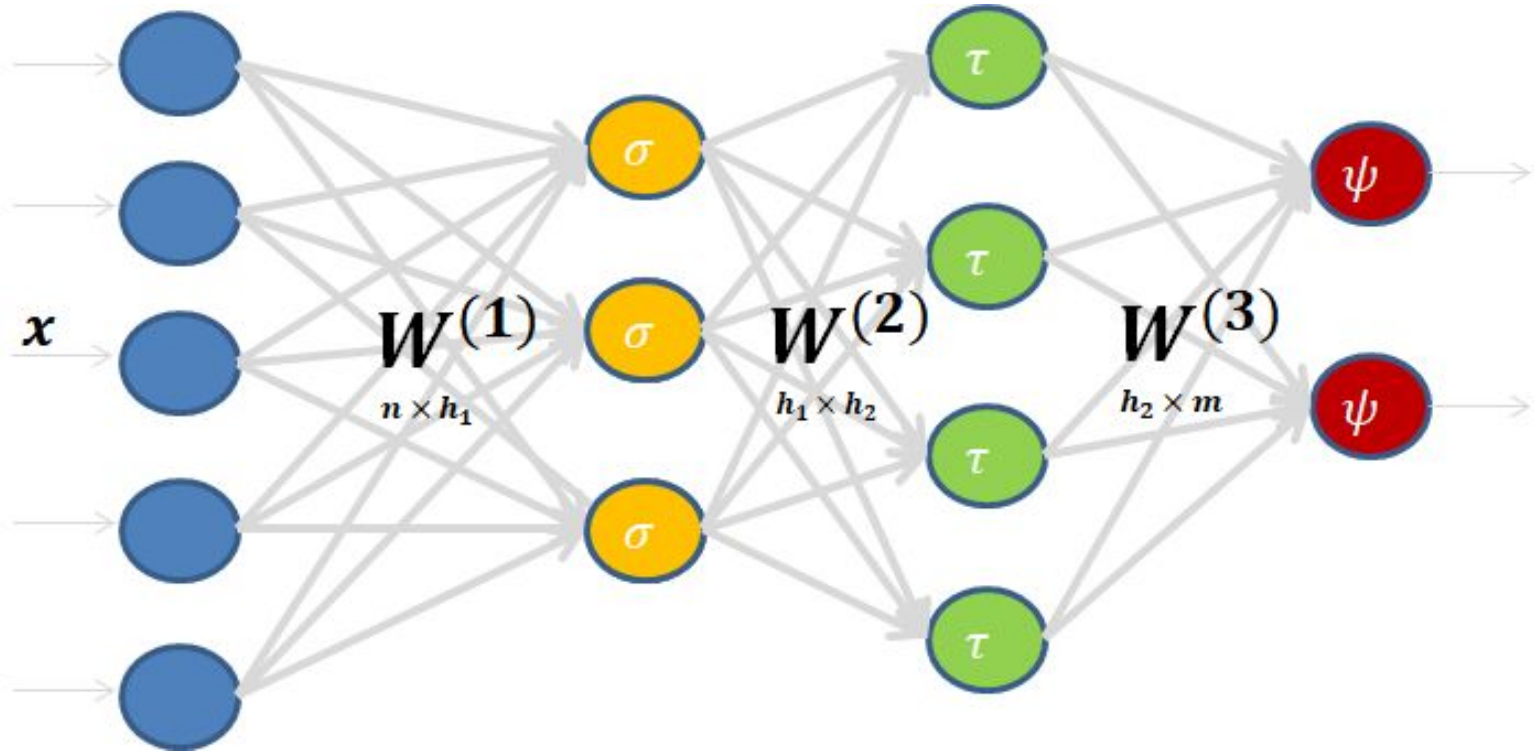
TECHNISCHE UNIVERSITÄT DARMSTADT

- 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
  - $L(\boldsymbol{\theta}) = \sum_{(\boldsymbol{x},\boldsymbol{y})} \quad ||(NN(\boldsymbol{x}; \boldsymbol{\theta}) - \boldsymbol{y})||^2$
  - $(\boldsymbol{x}, \boldsymbol{y})$ is our *training data*
  - $\boldsymbol{\theta}$ are our *parameters* ($= \boldsymbol{W}, \boldsymbol{V}$ above)

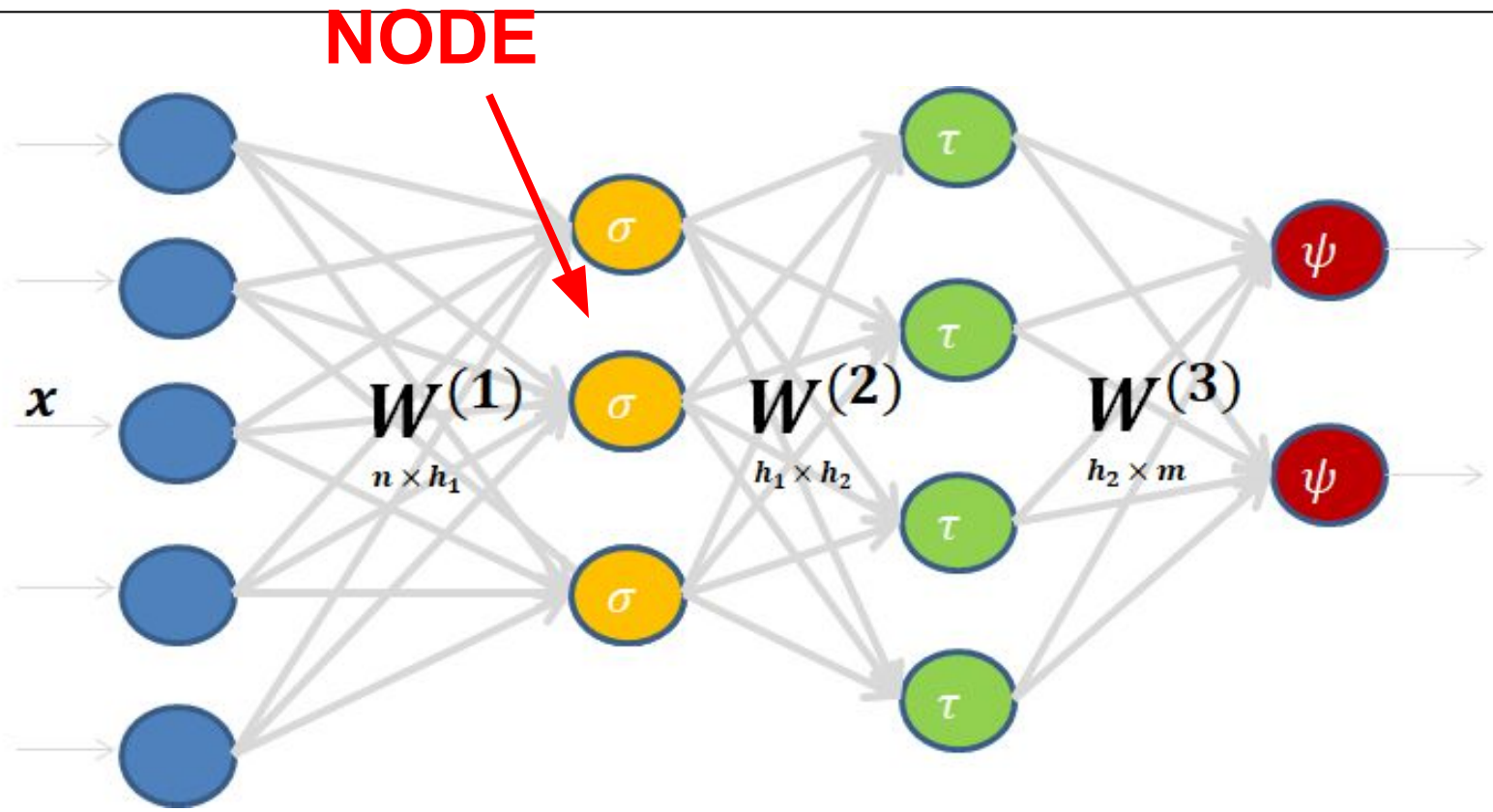
- We want to optimize our parameters $\boldsymbol{\theta}$ such that our loss becomes minimized
  - This is called *learning* or *training*

Deep Networks
Neural Net with at least one hidden layer



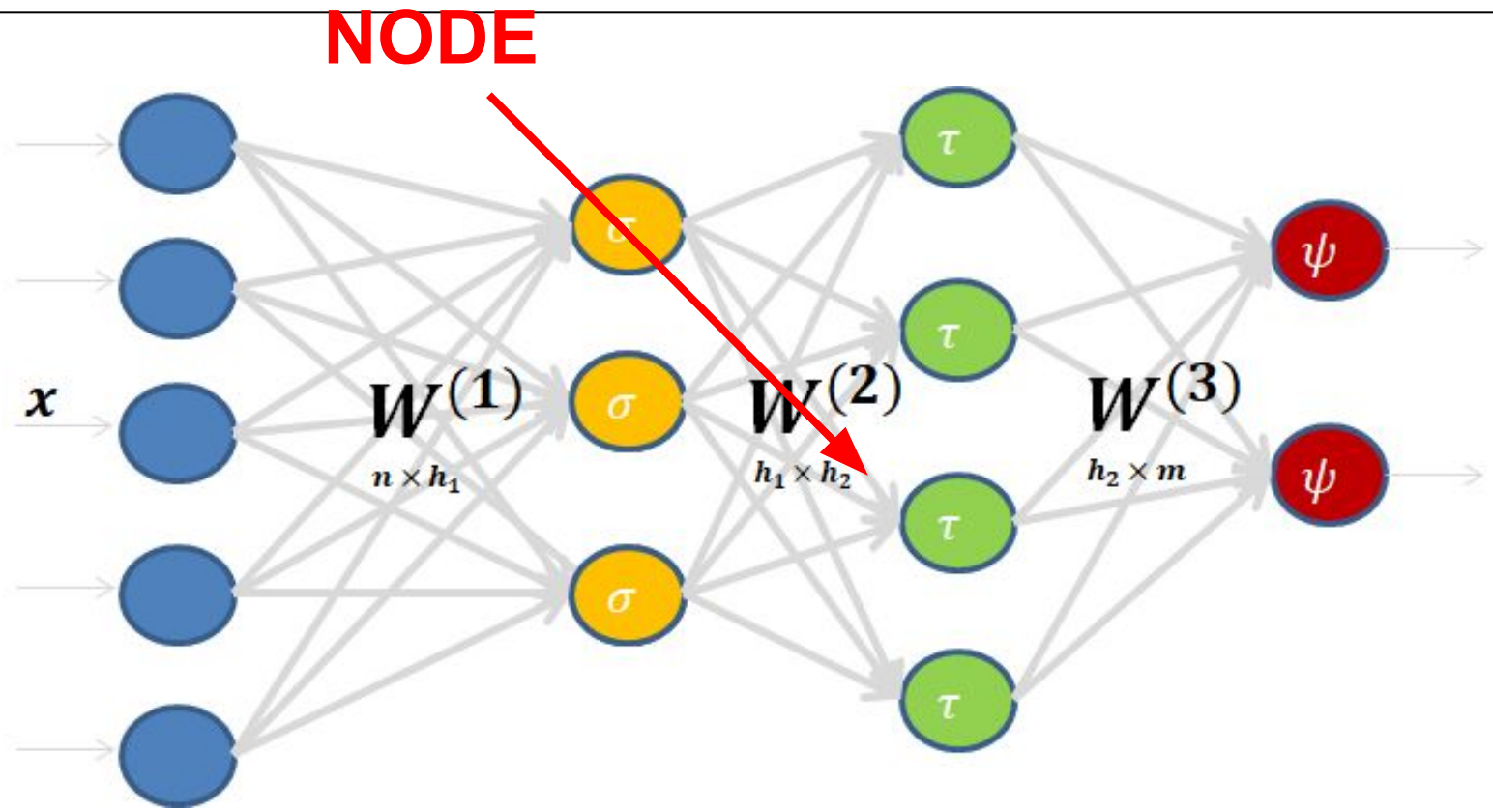
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



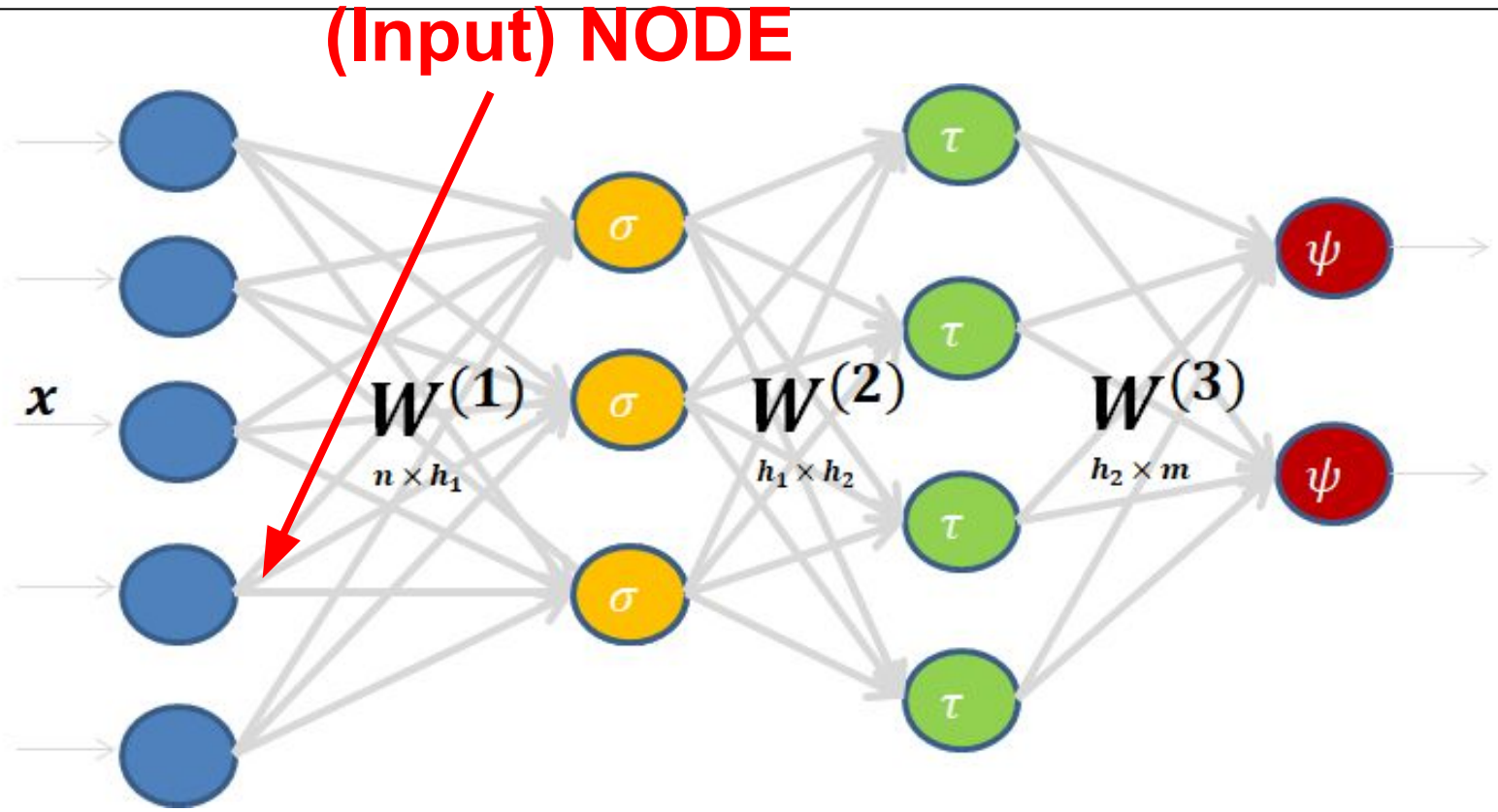
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



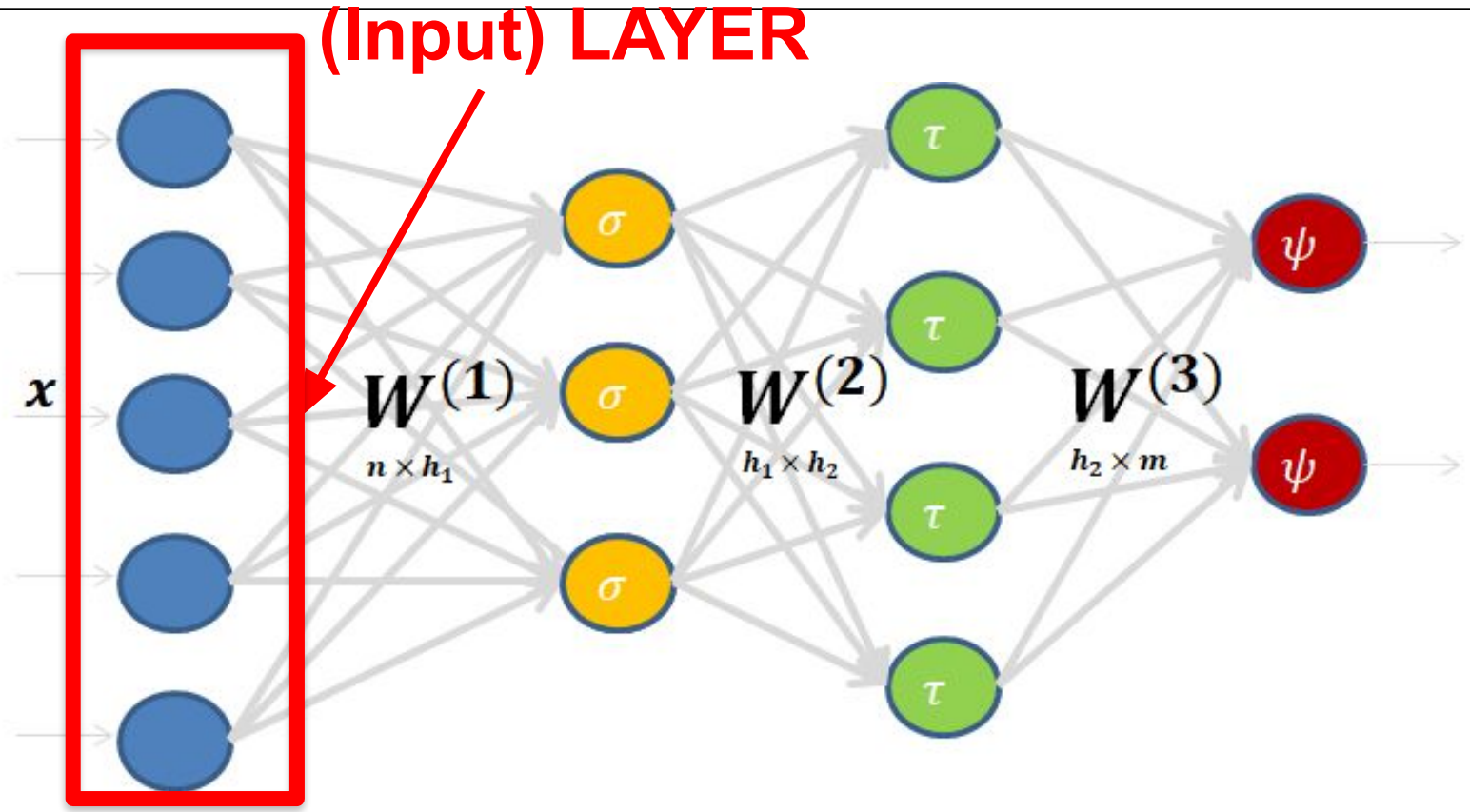
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



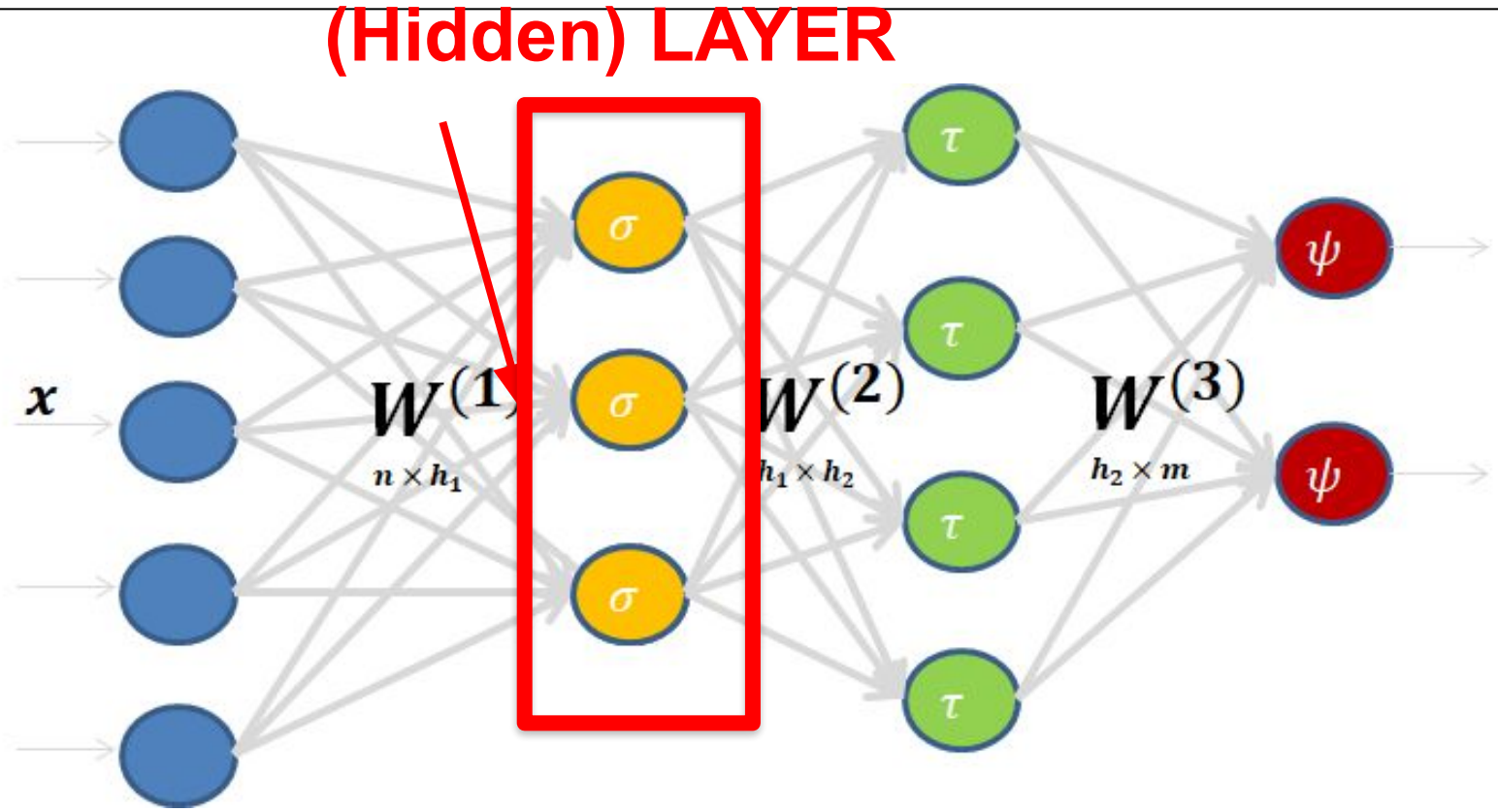
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



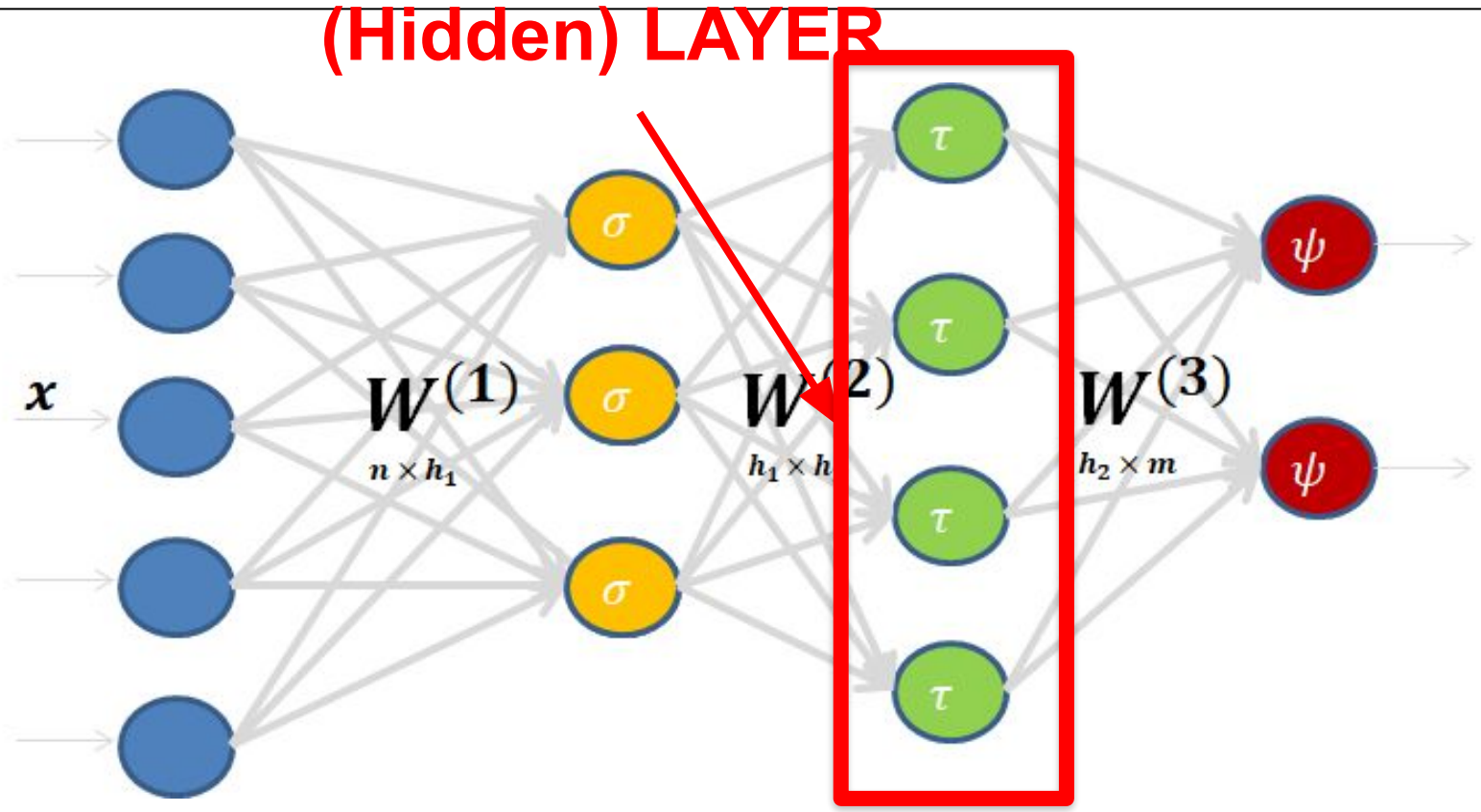
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



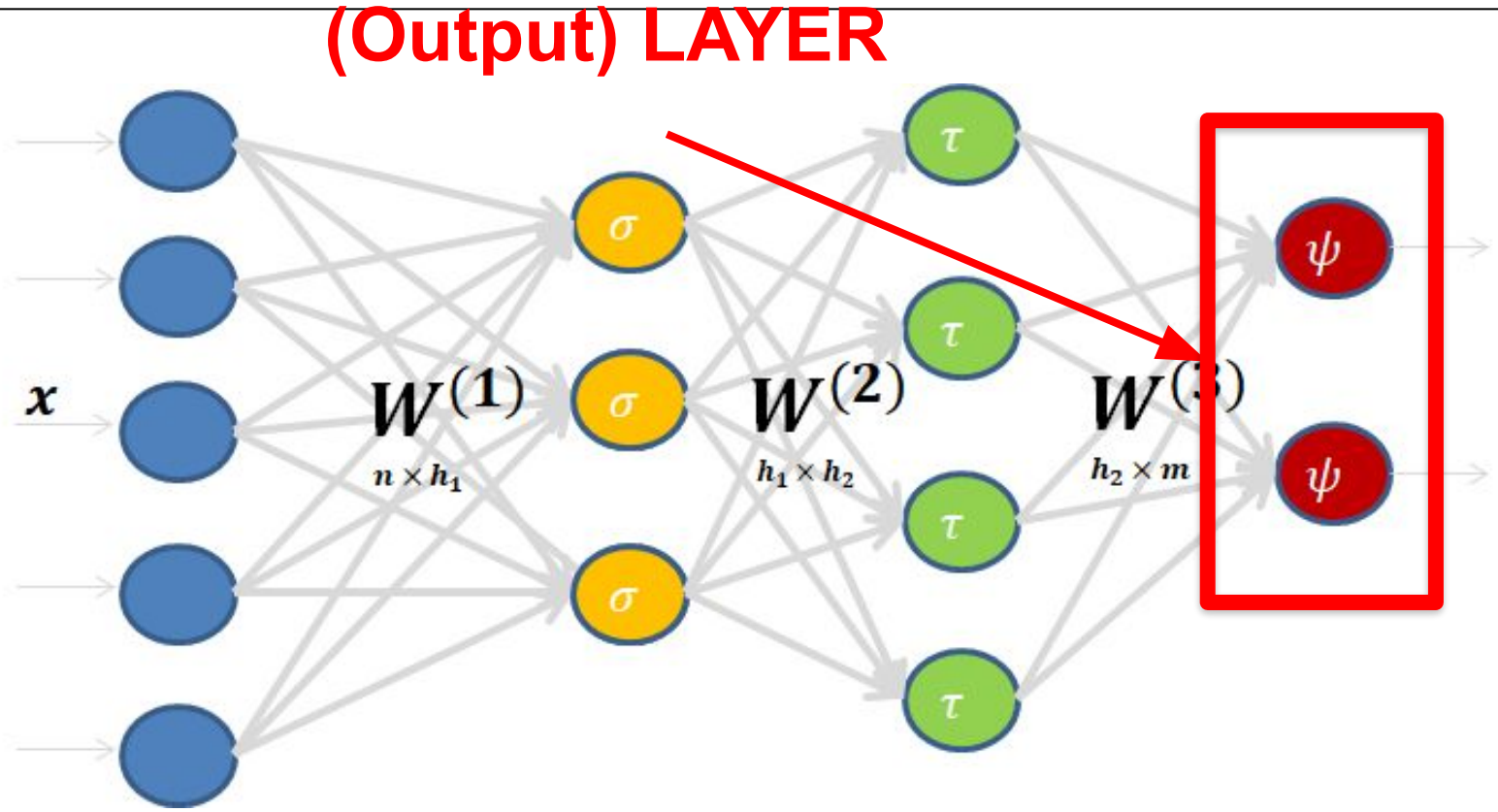
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



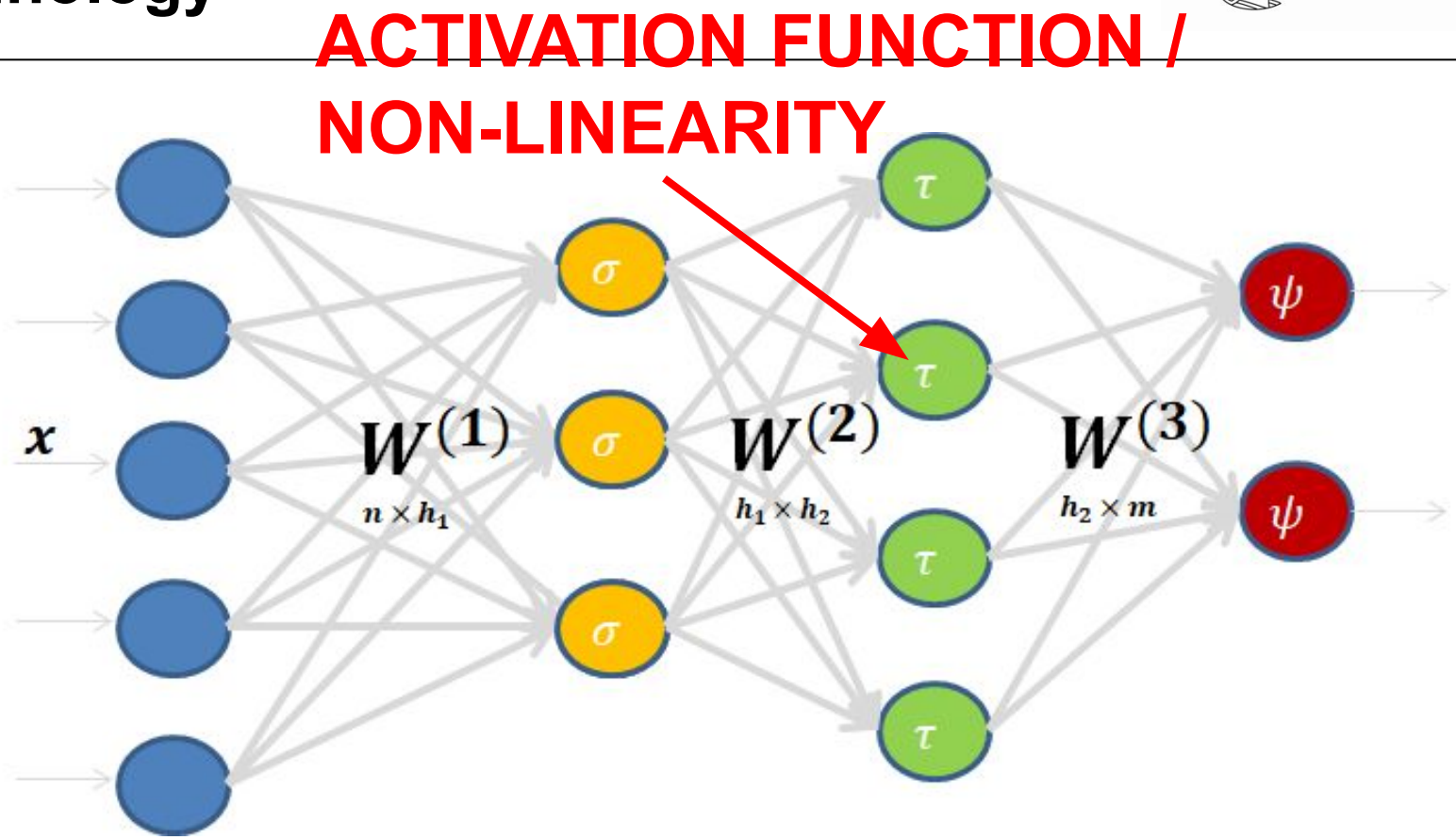
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



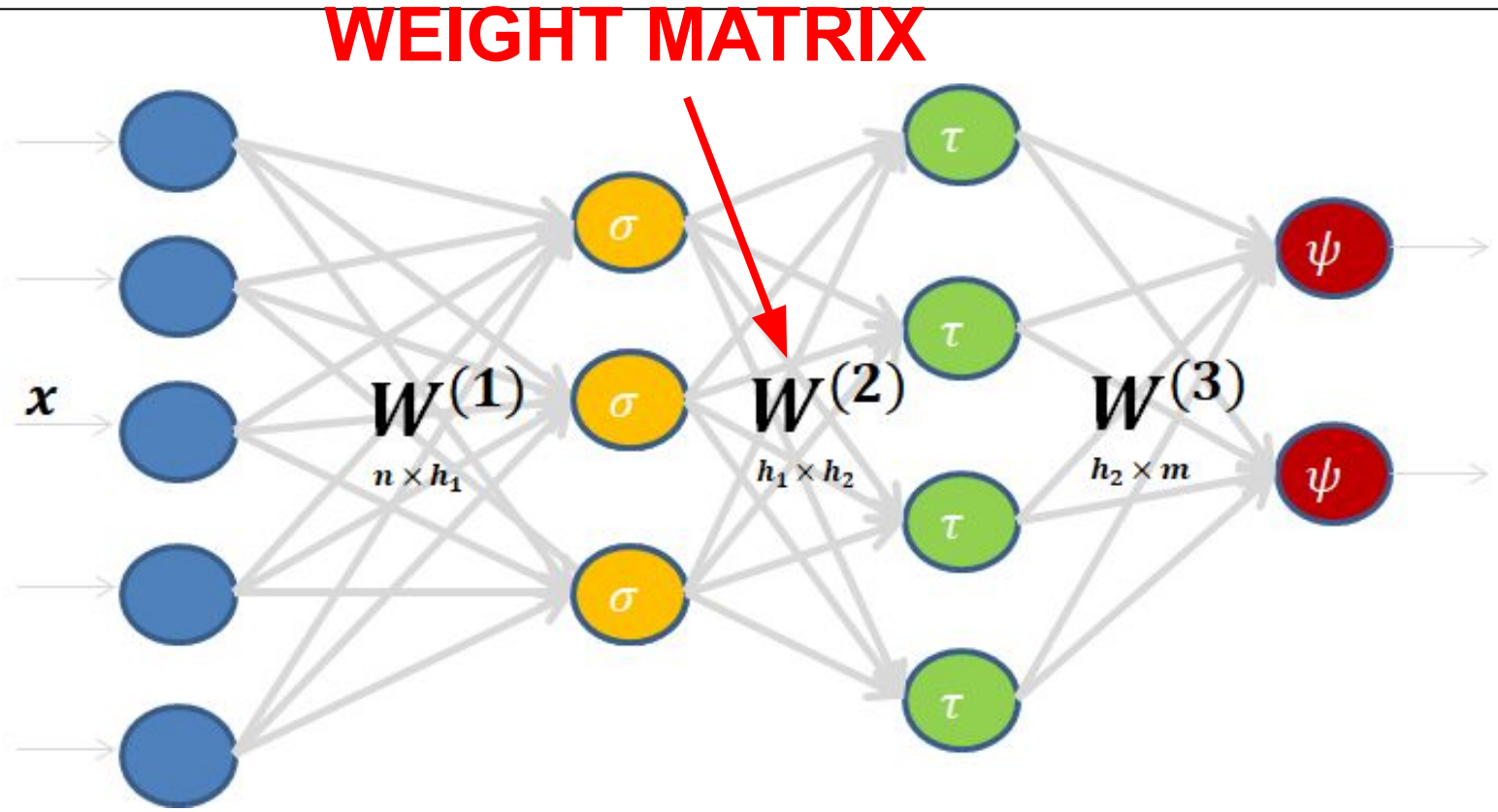
Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks Terminology



Also called “Multi-layer perceptron” or “feedforward network”

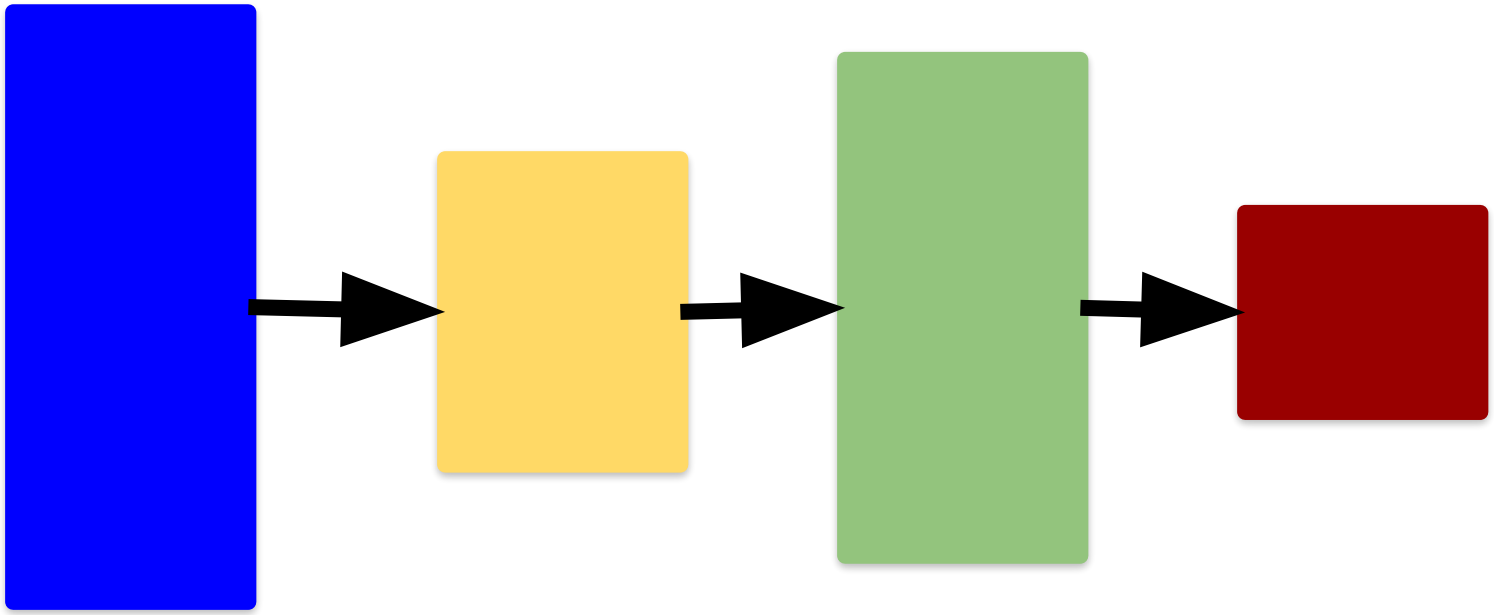
Deep Networks Terminology



Also called “Multi-layer perceptron” or “feedforward network”

Deep Networks
Simplified Visualization

TECHNISCHE UNIVERSITÄT DARMSTADT



Also called “Multi-layer perceptron” or “feedforward network”

NLLG

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Activation functions

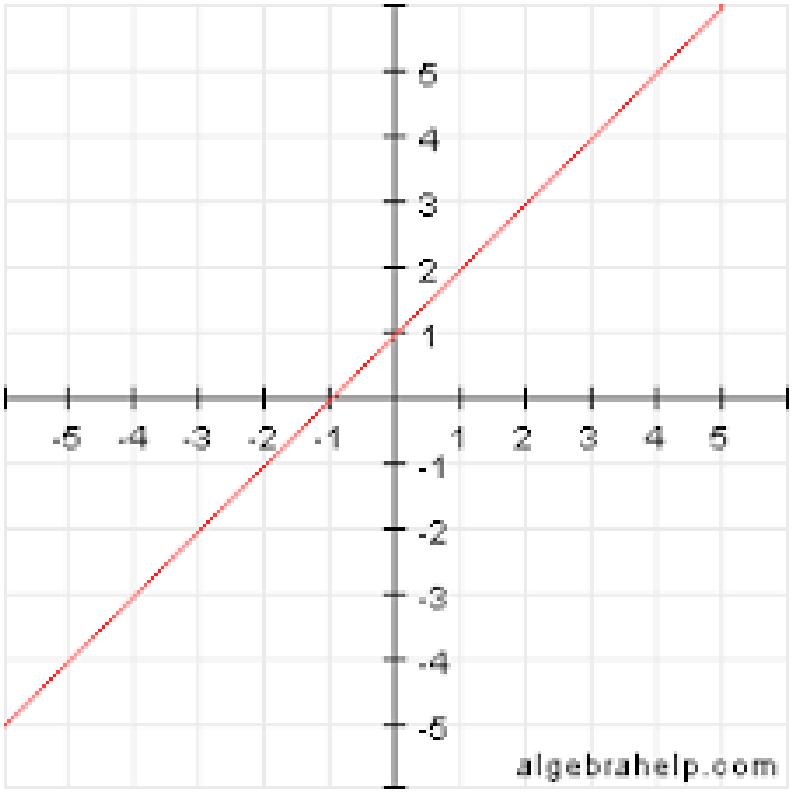
Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

Linear activations

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

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



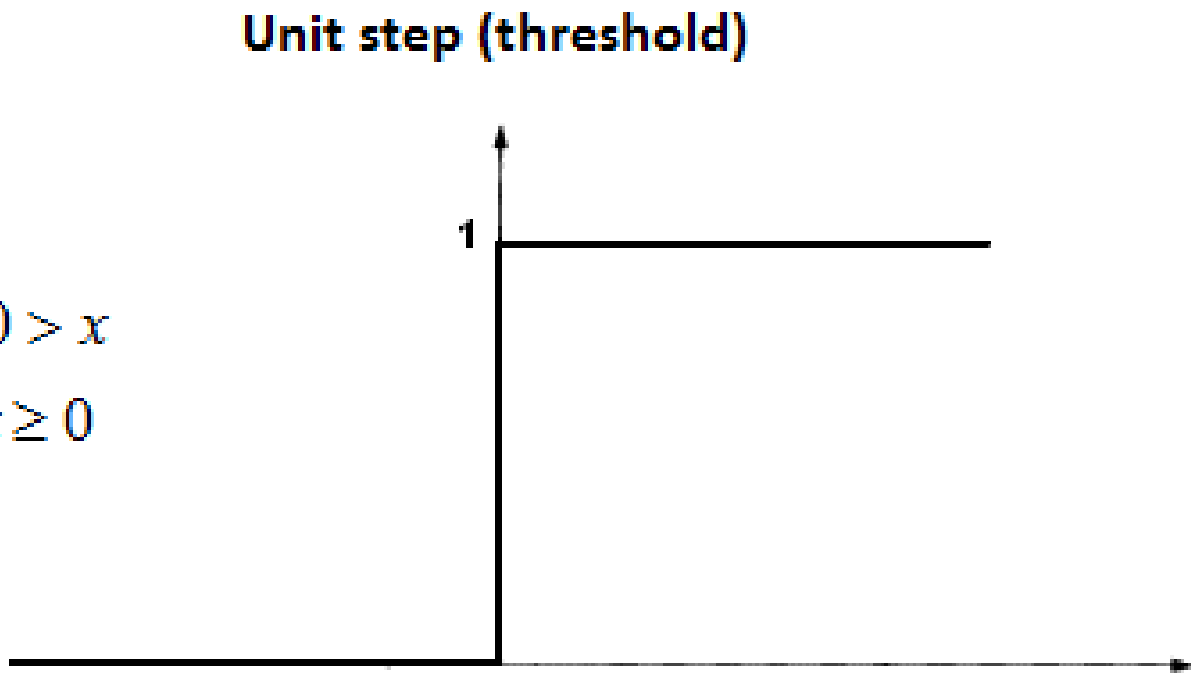
NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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

NLLG

Activation functions σ

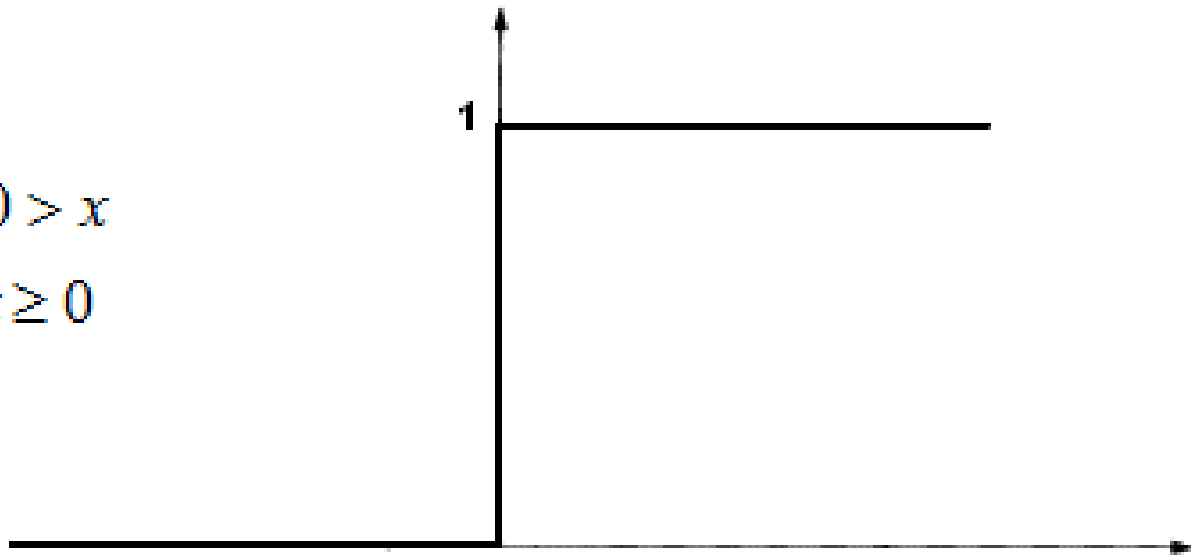
TECHNISCHE UNIVERSITÄT DARMSTADT

Threshold/Step function

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

Unit step (threshold)

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



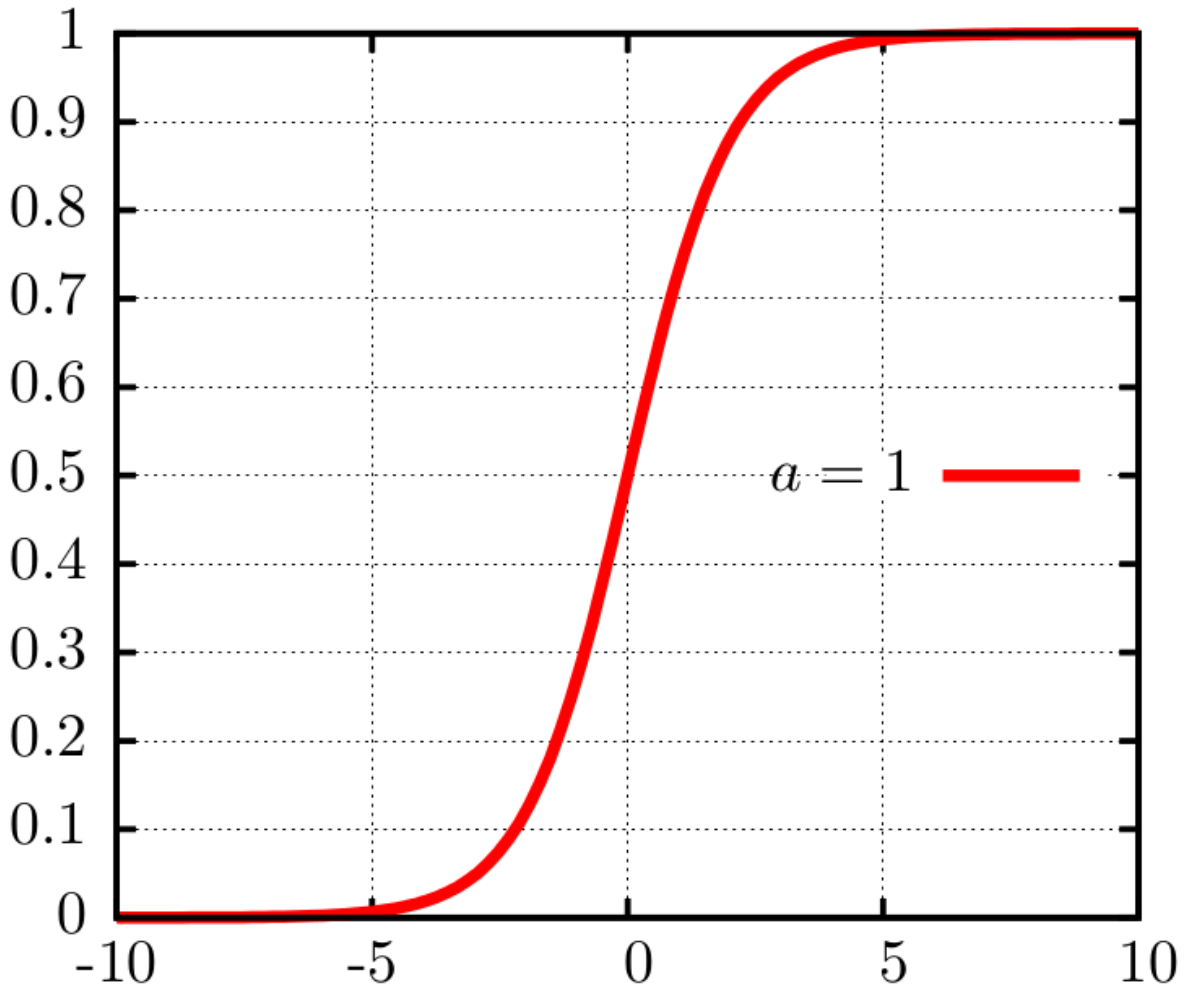
Picture from: http://kylescholz.com/projects/wordnet/, based on representation from WordNet: https://wordnet.princeton.edu

NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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

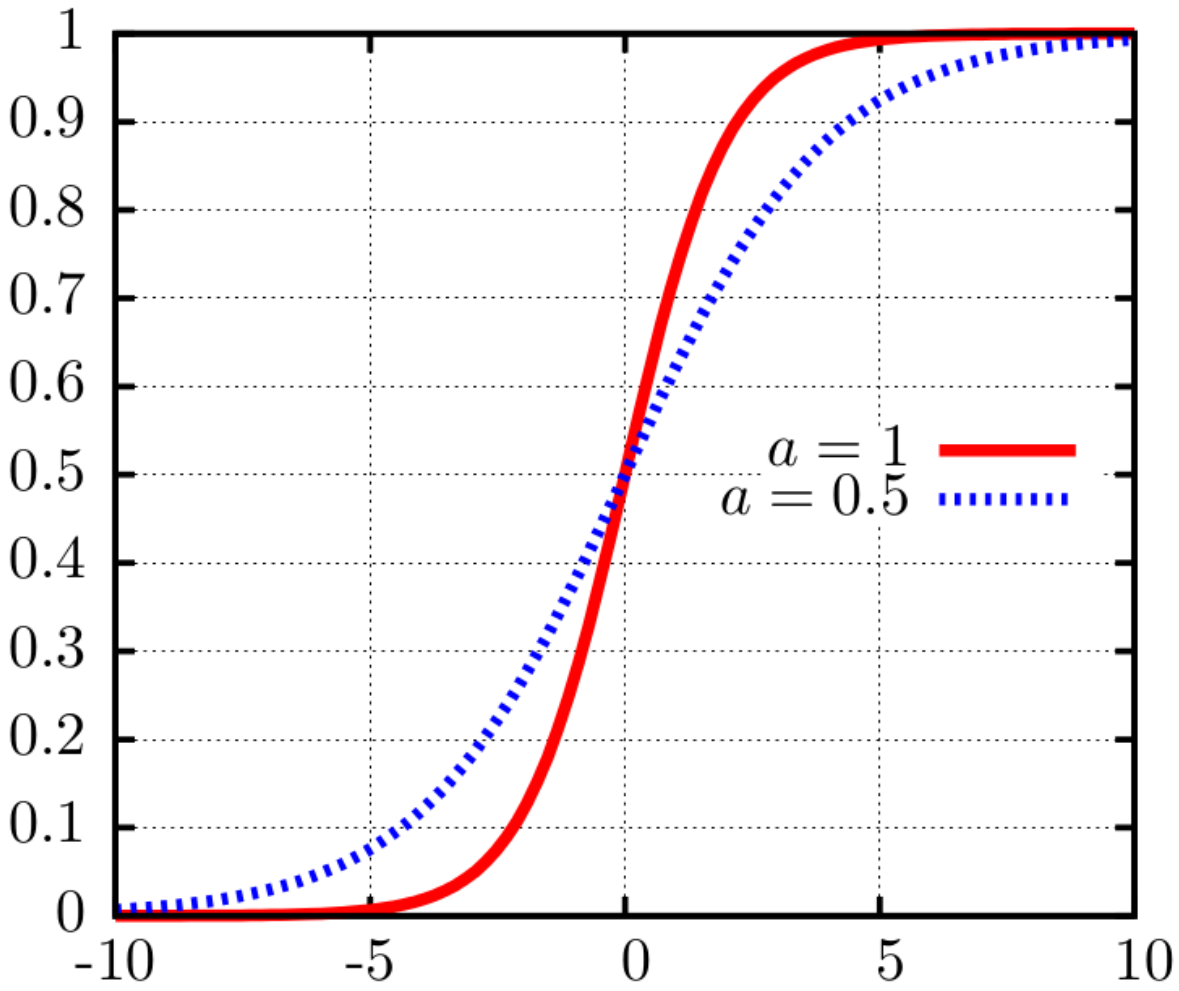


NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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

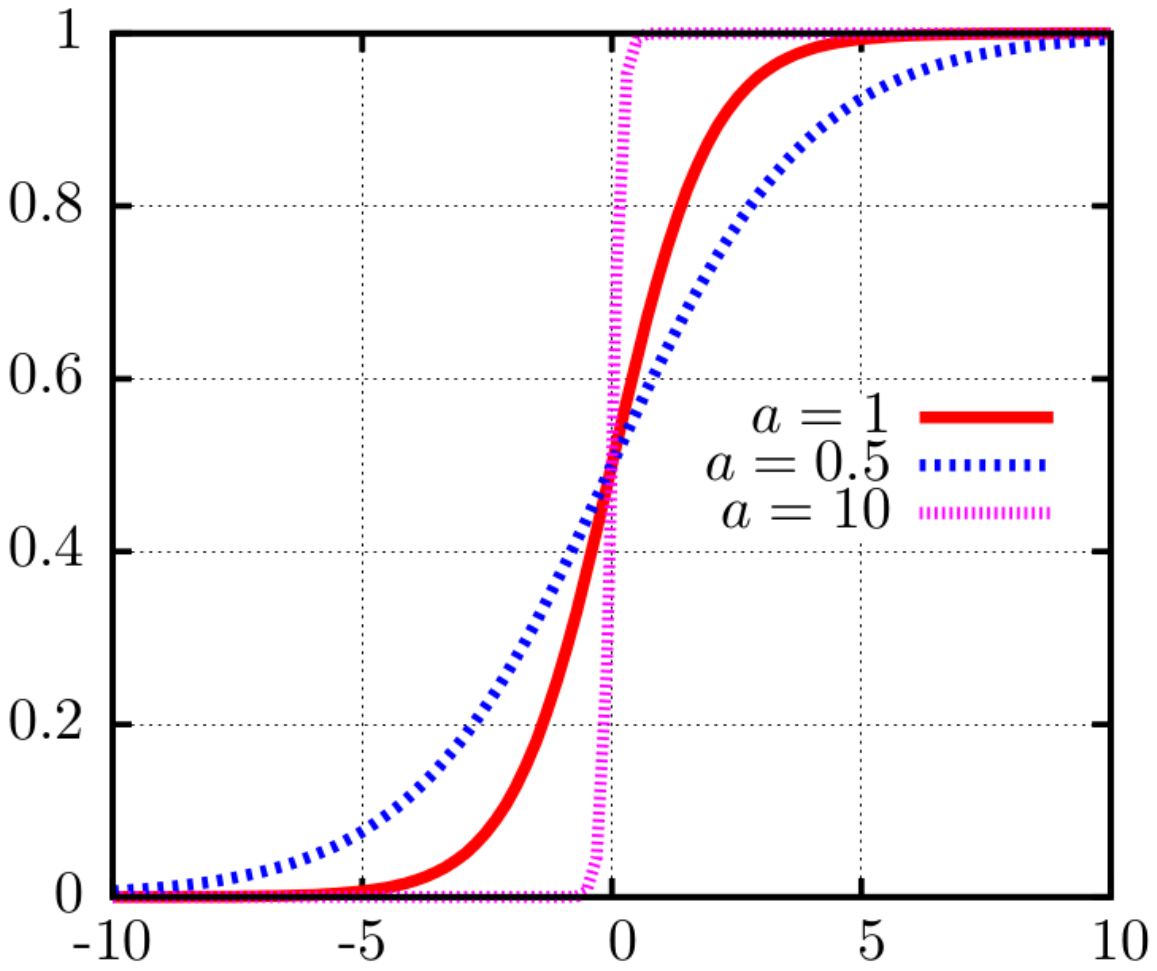


NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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



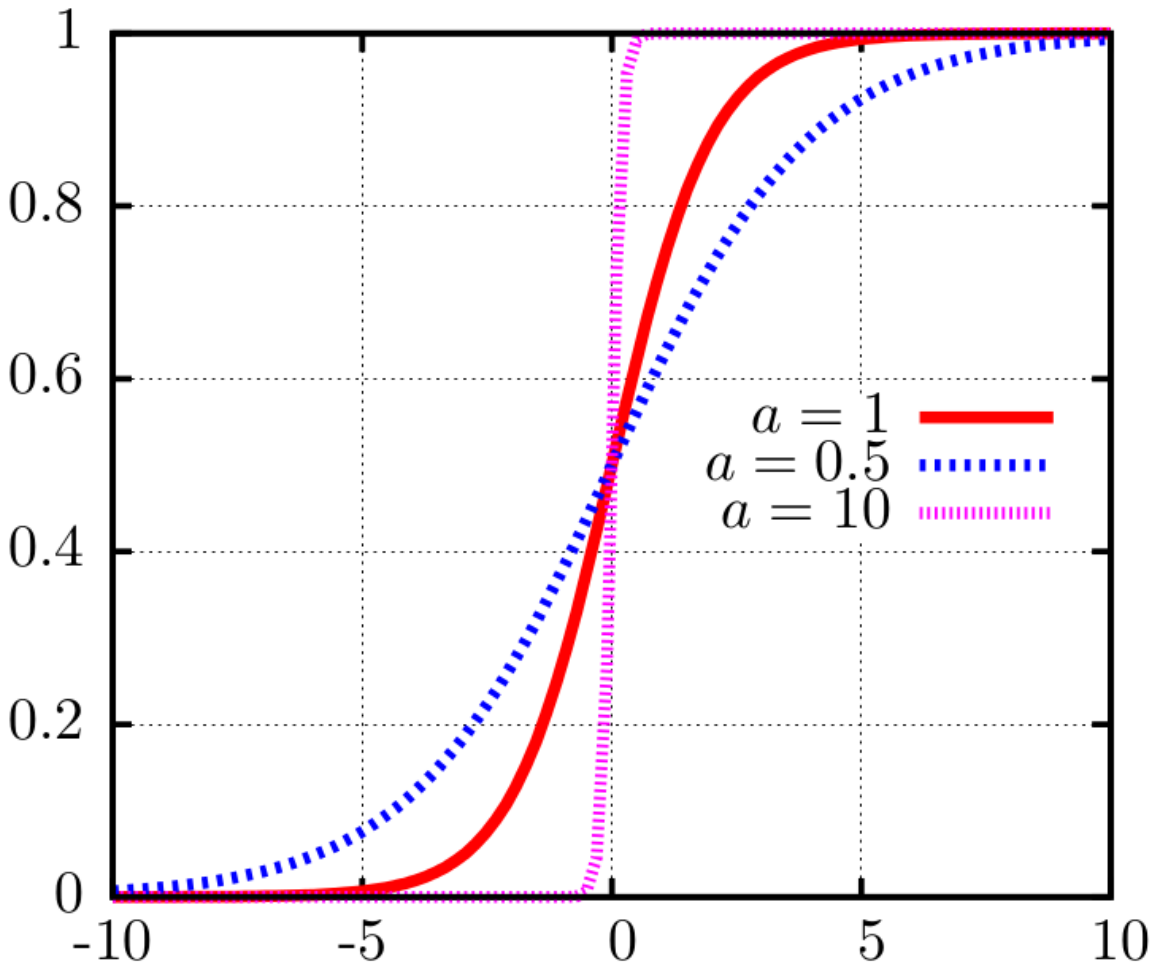
NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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

1. Not zero-centered
2. Kills gradients



NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Not zero-centered: consider $\sigma(\boldsymbol{x} \cdot \boldsymbol{w})$
   - What‘s the derivative/gradient?

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Not zero-centered: consider $\sigma(\boldsymbol{x} \cdot \boldsymbol{w})$
   - What's the derivative/gradient? $\sigma'(\boldsymbol{x} \cdot \boldsymbol{w}) \cdot \boldsymbol{x}$

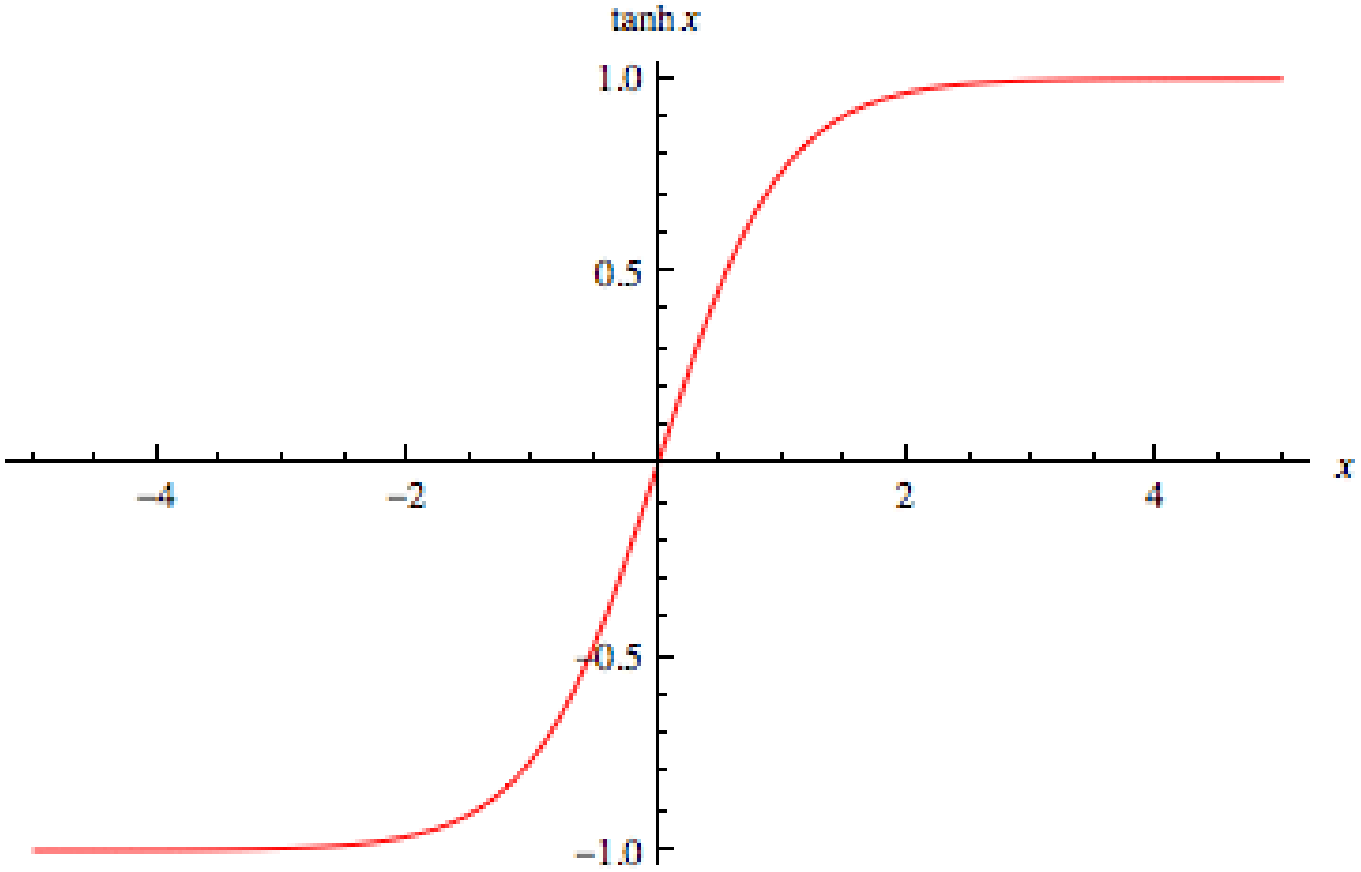
In lecture 3, we'll see that

$$\frac{\partial E}{\partial u_{ik}} = \frac{\partial E}{\partial m_k} \sigma'(z_k) x_i$$

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

Tanh

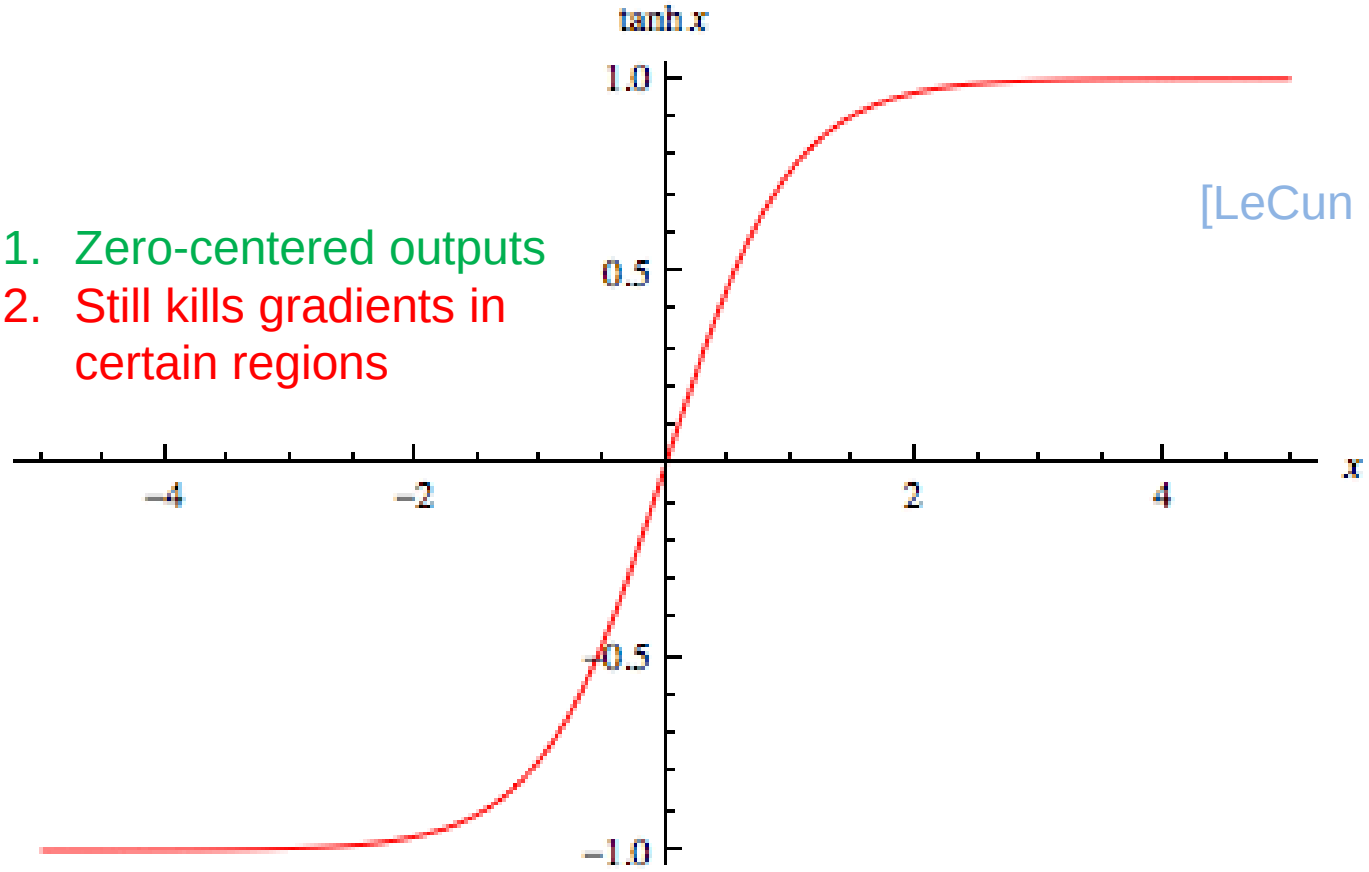


Picture from: : http://mathworld.wolfram.com/images/interactive/TanhReal.gif

NLLG

Activation functions σ

Tanh



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

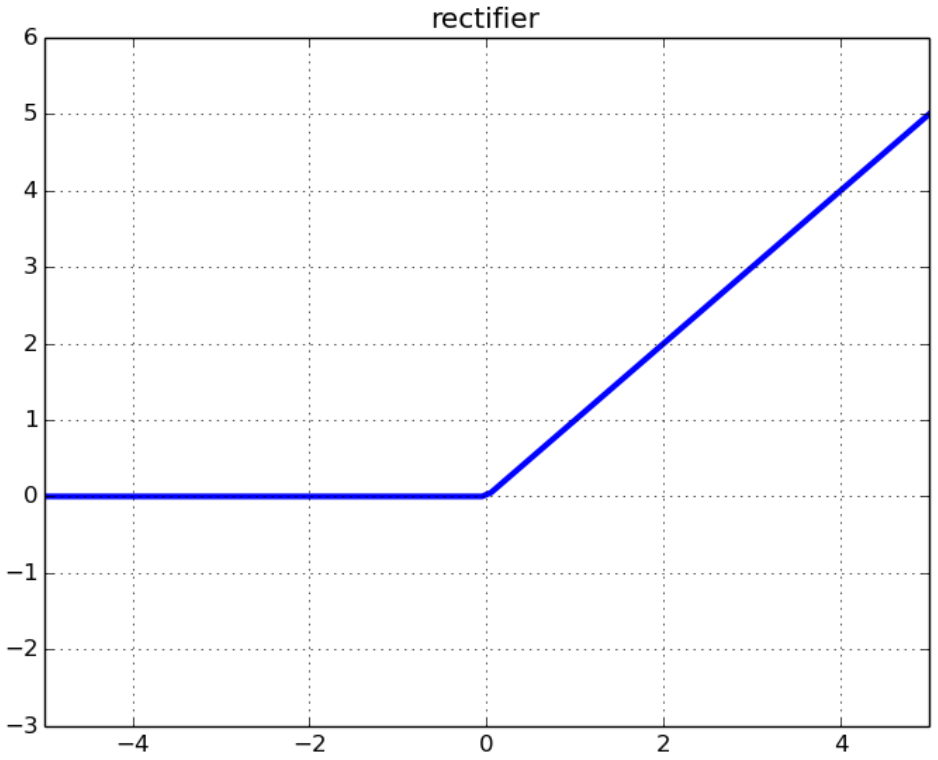
[LeCun et al. 1991]

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

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

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



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

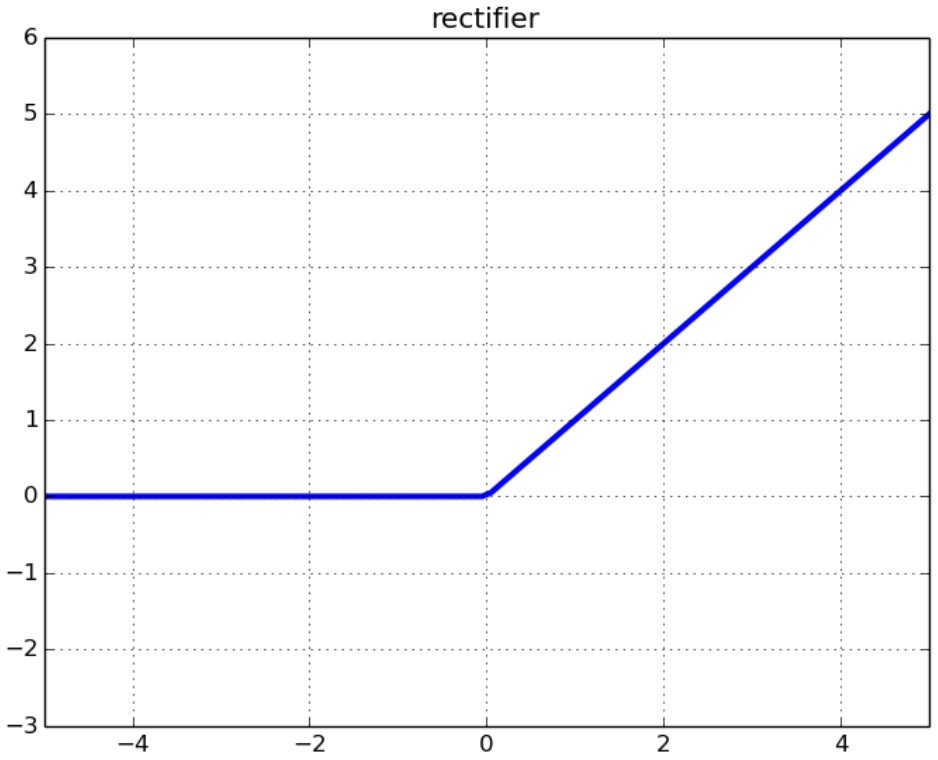
NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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

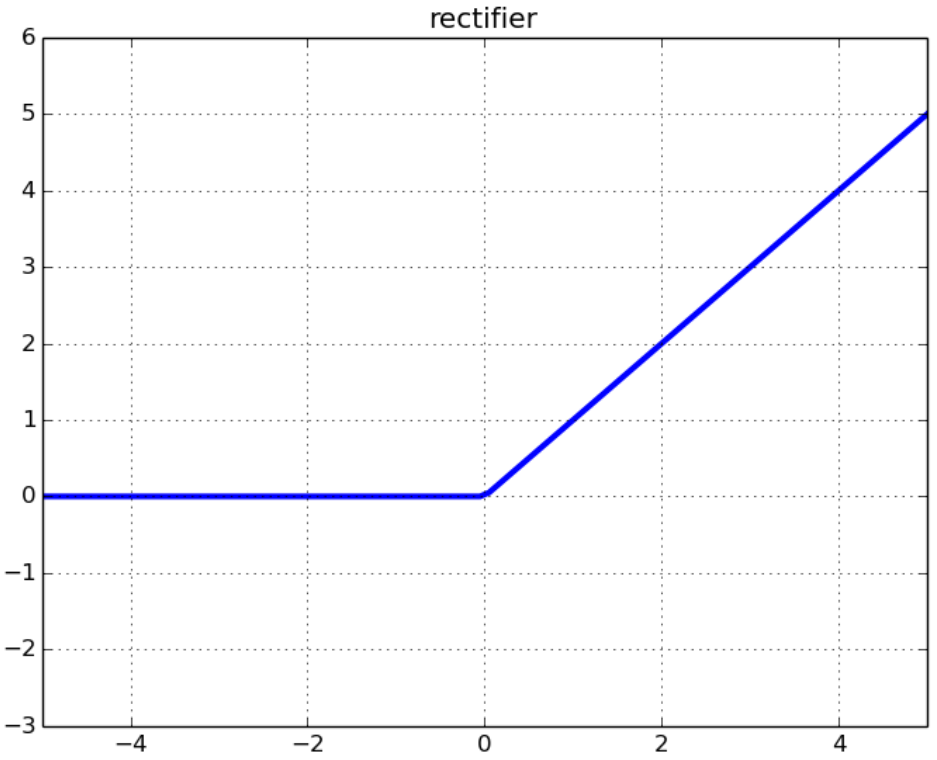
[Krizhevsky et al. 2012]

NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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]

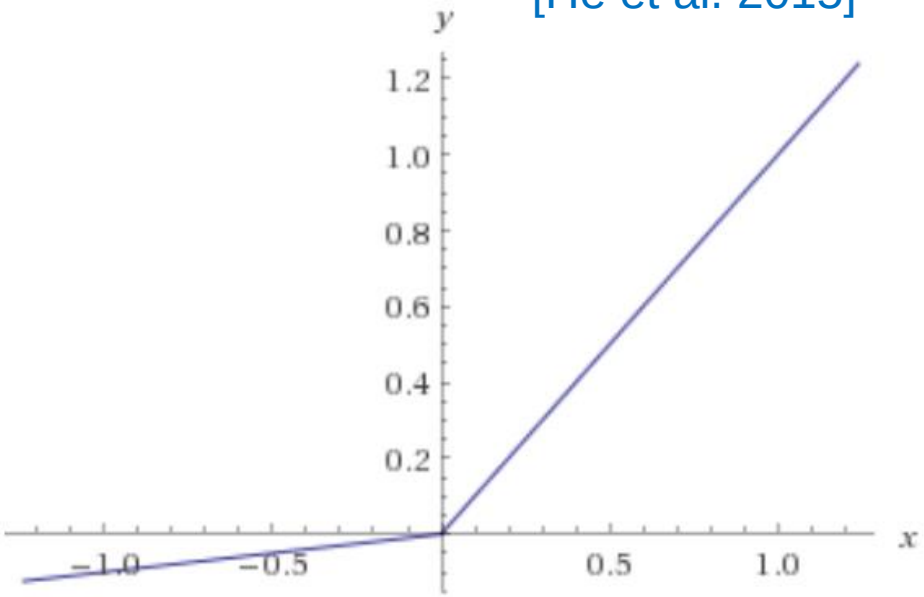
NLLG

Activation functions σ

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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

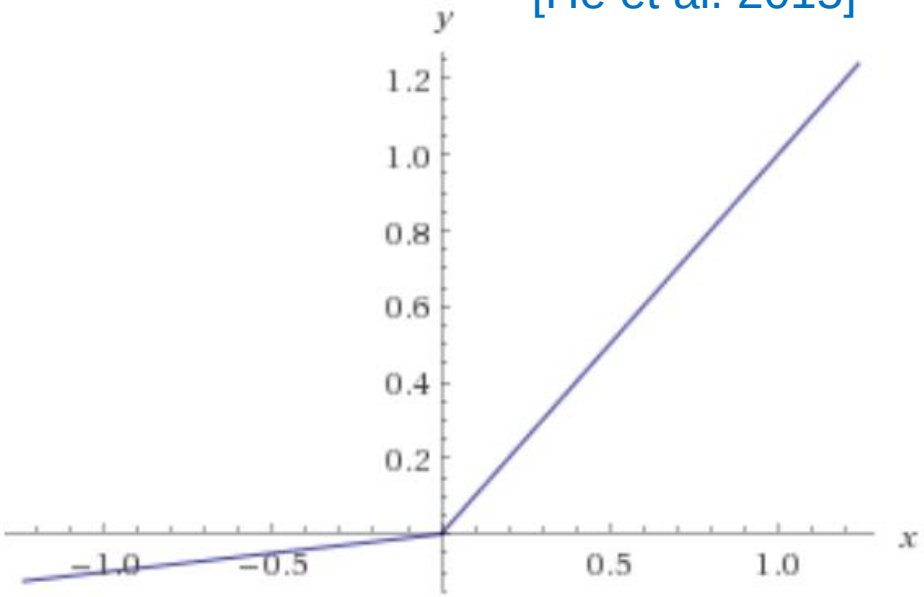
- From http://i.stack.imgur.com/8CGlM.png

NLLG

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 σ

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Maxout "Neuron"**

[Goodfellow et al. 2013]

$$\max(\boldsymbol{w}_1\boldsymbol{x}, \boldsymbol{w}_2\boldsymbol{x})$$

1. Generalizes ReLU and Leaky ReLU
2. Gradients don't die

1. Doubles the number of paramters

Activation functions

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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 / Maxout / ELU
- Don’t use sigmoid or step function

Activation functions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tanh > Sigmoid

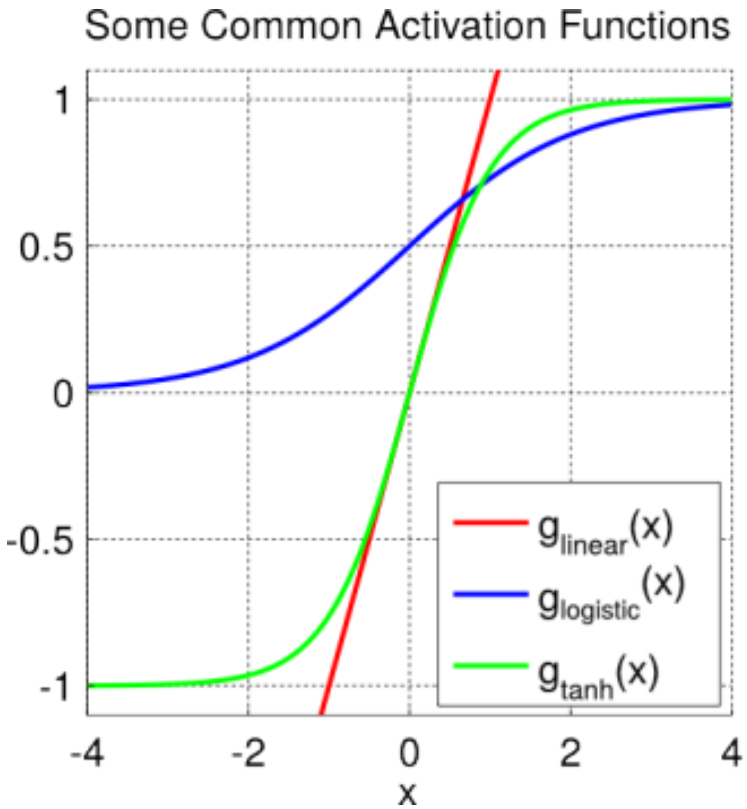
- ReLU is most popular activation function (as of 2015)
  - According to https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

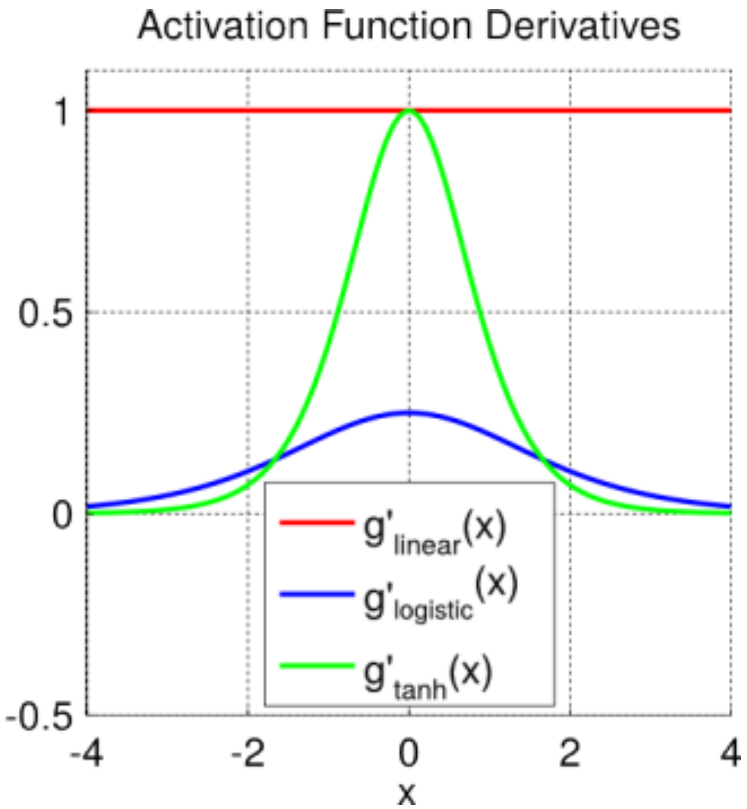
- Others have to stand the test of time

  - Try out Leaky ReLU / Maxout / ELU

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

Activation functions





A special activation function: softmax

- 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

- 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
- The softmax activation function serves that purpose:
  - Given m output units, the softmax activation is

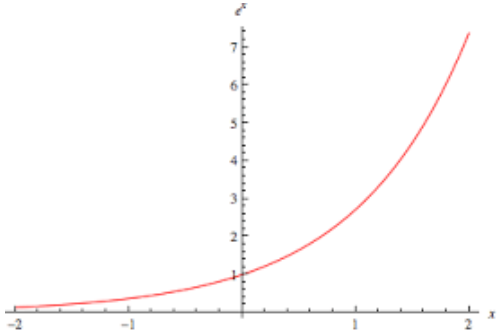
$$y_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$$

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

A special activation function: softmax

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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:

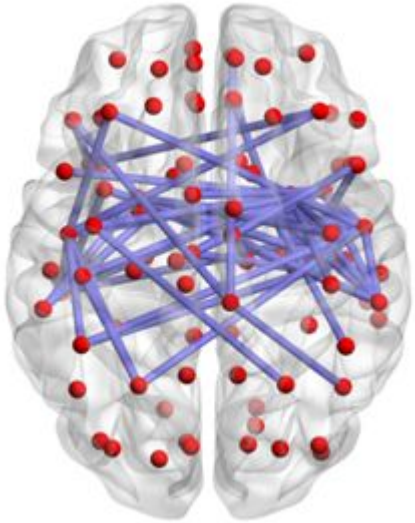
$$\boldsymbol{y} = \mathrm{softmax}(\boldsymbol{z})$$

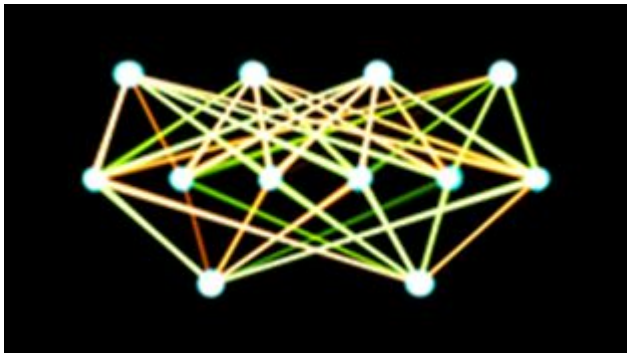
yields a probability distribution, $\boldsymbol{y}$

Deep Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

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

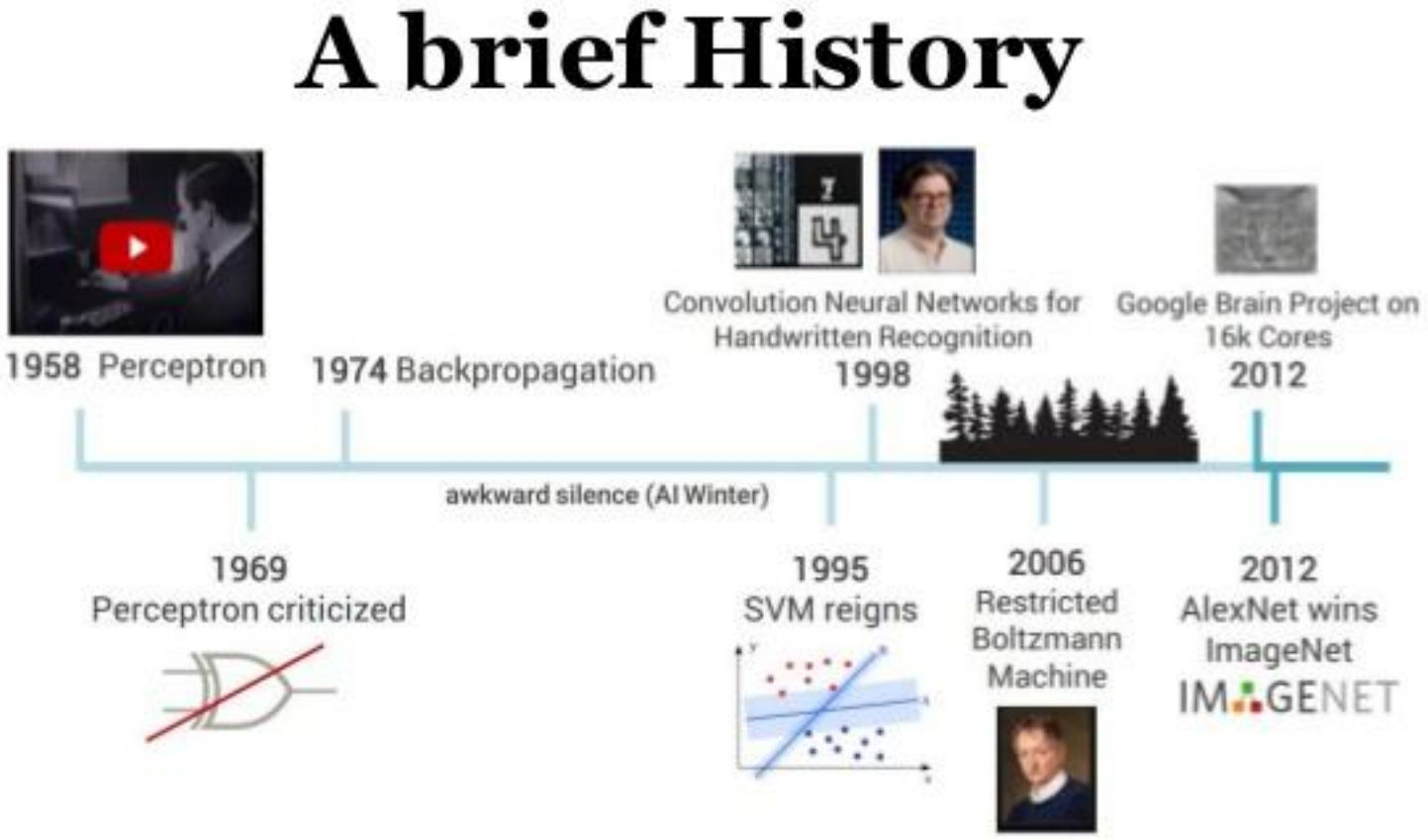




https://upload.wikimedia.org/wikipedia/commons/0/0e/Brain_network.png
https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/Single-layer_feedforward_artificial_neural_network.png/214px-Single-layer_feedforward_artificial_neural_network.png

NLLG

Deep Networks History



Source https://www.slideshare.net/mohamedloey/deep-learning-71352250

Deep Networks History

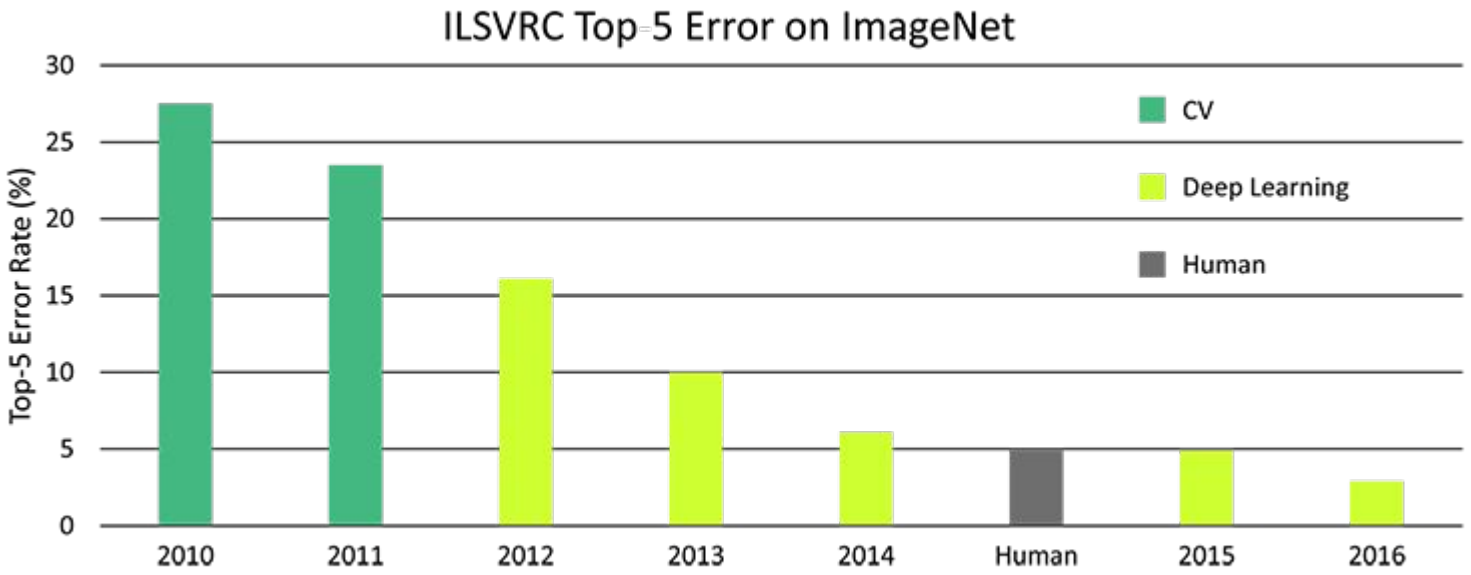
TECHNISCHE UNIVERSITÄT DARMSTADT

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

Deep Networks Successes

TECHNISCHE UNIVERSITÄT DARMSTADT

Image classification



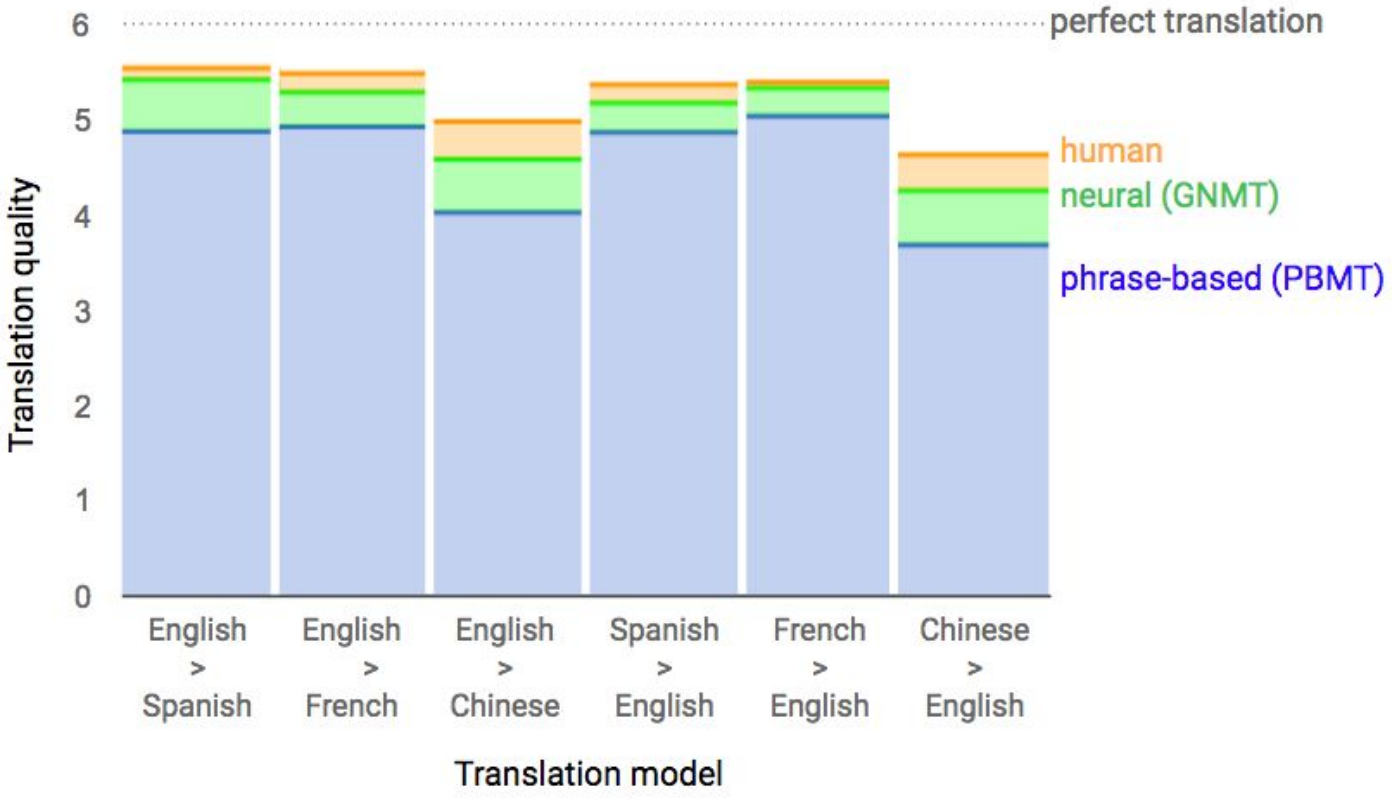
Source https://medium.com/nanonets/how-to-automate-surveillance-easily-with-deep-learning-4eb4fa0cd68d

NLLG

Deep Networks Successes

TECHNISCHE UNIVERSITÄT DARMSTADT

Machine translation

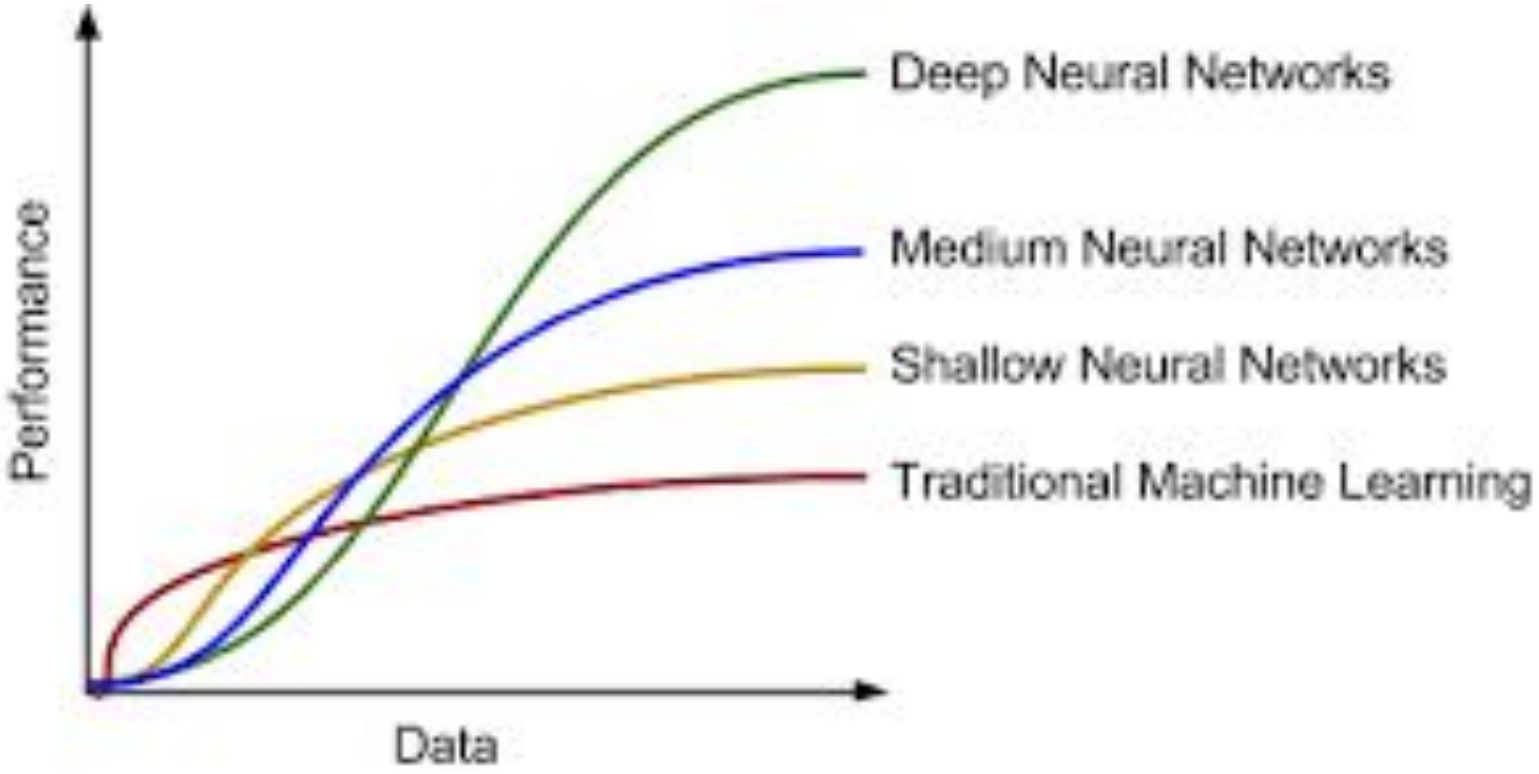


Source https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

NLLG

Deep Networks
When does it work?

TECHNISCHE UNIVERSITÄT DARMSTADT



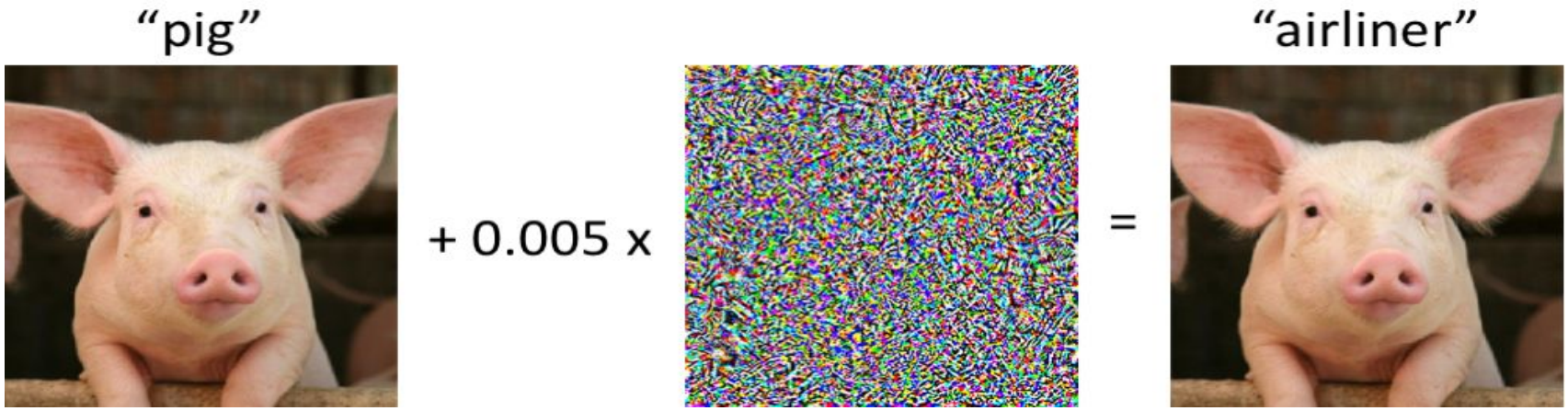
Source https://www.skynettoday.com/editorials/state_of_nmt

NLLG

Deep Networks
What does not work (well yet)?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Small data scenarios
- Domain transfer / generalization / robustness
  - when the test data is different from the training data
  - when the test data is “adversarially” modified
- Deep understanding
  - NN often exploit low-level statistical cues rather than truly understanding the data (in the way humans do)



Source https://medium.com/attentive-ai/fooling-cnns-via-adversarial-examples-877a9e0ee84e

NLLG

Natural Language Processing
Task Types

TECHNISCHE UNIVERSITÄT DARMSTADT

Three types of tasks:

- **(C) Sentence / text classification**
  - I hate this movie → NEGATIVE
- **(T) Sequence tagging**
  - I hate this movie → PRON VERB DET NOUN
- **Seq2Seq**
  - I hate this movie → Diesen Film , ich hasse ihn

NLLG

Deep Learning Architectures vs. Input Representation

TECHNISCHE UNIVERSITÄT DARMSTADT

Two fundamental questions:

- Which neural network architecture to choose?
  - Perceptron
  - MLP
  - RNN (LSTM, GRU)
  - CNN
  - Encoder-Decoder (RNN-, Transformer-based)
- How to represent the input?
  - (Sentence/Word) Embeddings

Deep Learning Architectures vs. Input Representation

Two fundamental questions:

- Which neural network architecture to choose?
  - Perceptron ← → C
  - MLP ← → C
  - RNN (LSTM, GRU) ← → T
  - CNN ← → C
  - Encoder-Decoder (RNN-, Transformer-based) ← → Seq2Seq
- How to represent the input?
  - (Sentence/Word) Embeddings

How to represent the input? Motivation

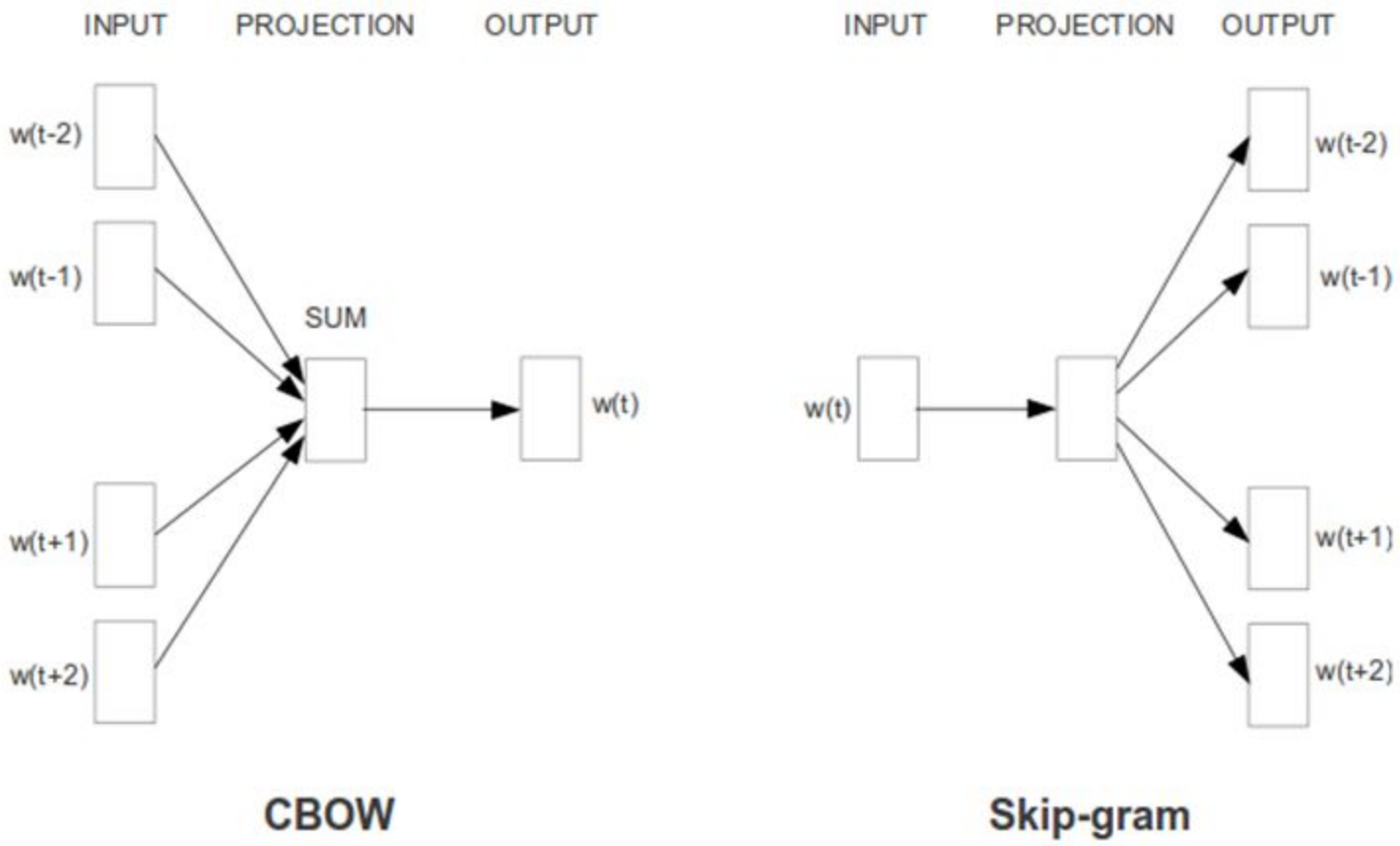
- A neural network takes a vector as an input
- In NLP, the vectors could represent
  - Characters
  - Words
  - Sentences
  - Texts
- The vector representations should encode
  - Semantics
  - Syntax/Morphology/Grammar
- Similar objects should have similar representations

How to represent the input? Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

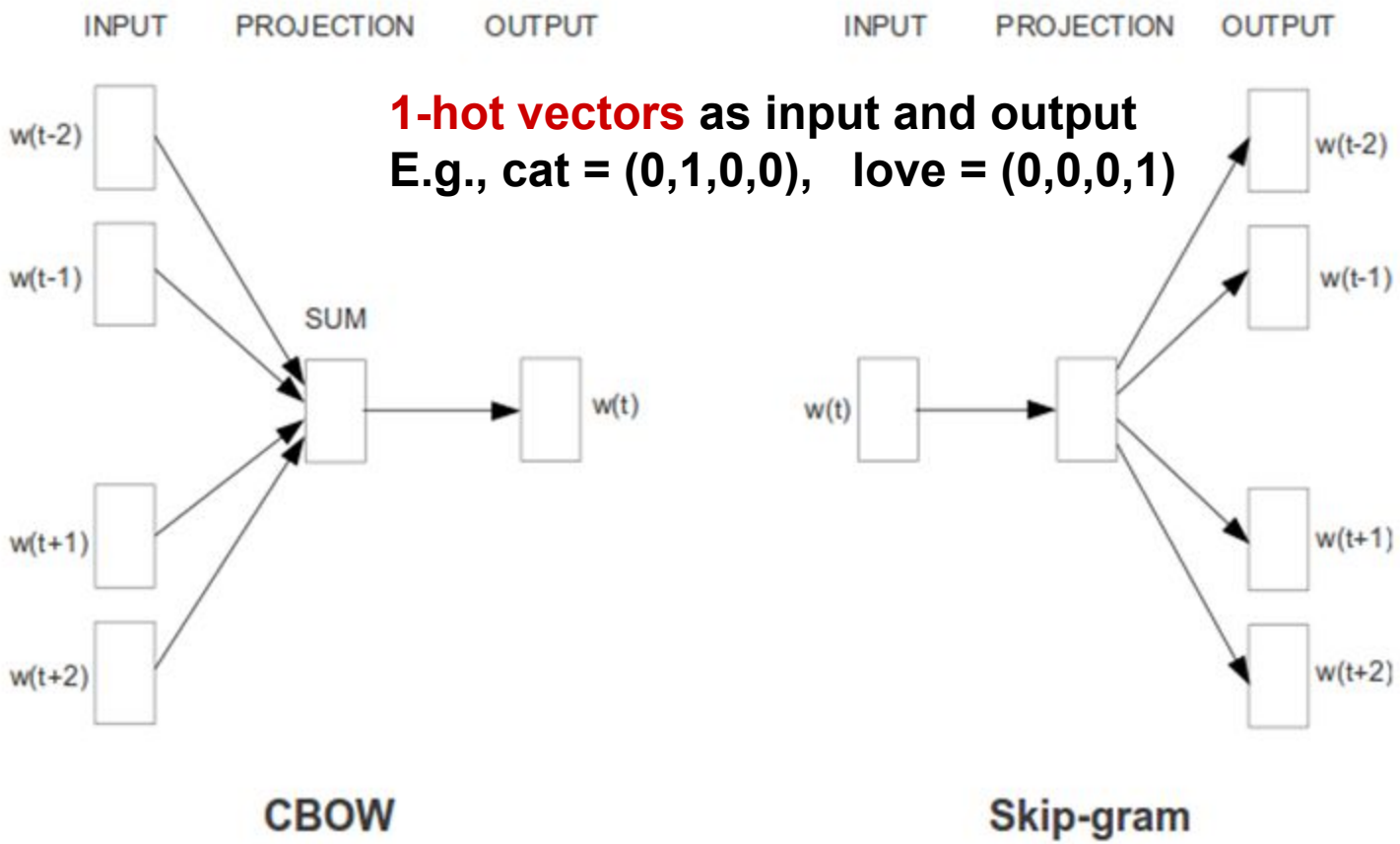
- A neural network takes a vector as an input
- In NLP, the vectors could represent
  - Characters
  - **Words**
  - Sentences
  - Texts
- The vector representations should encode
  - **Semantics**
  - Syntax/Grammar/Morphology
- Similar objects have similar representations

Word2Vec (Mikolov et al. 2013) Visualization



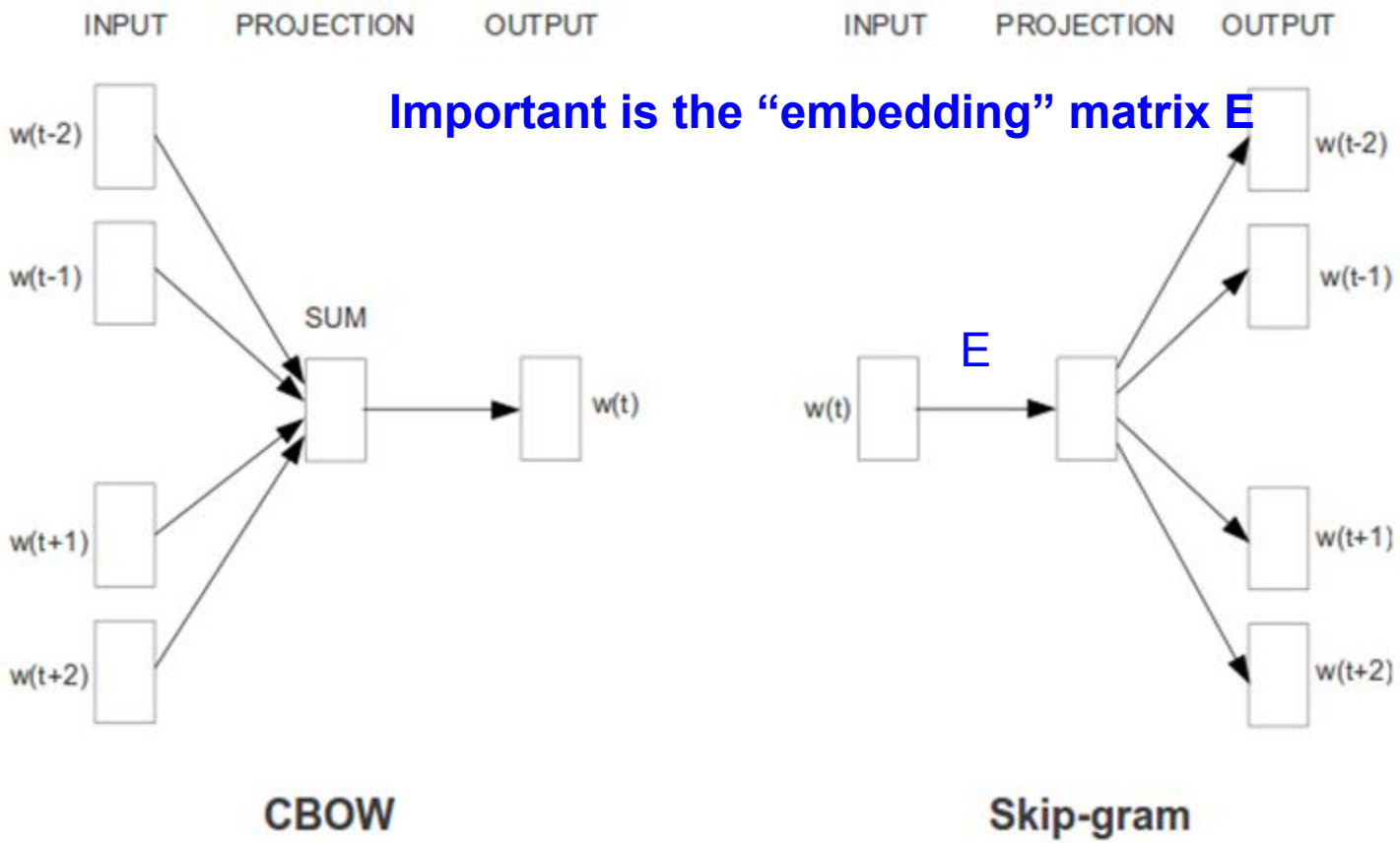
Word2Vec (Mikolov et al. 2013) Visualization

TECHNISCHE UNIVERSITÄT DARMSTADT



1-hot vectors as input and output
E.g., cat = (0,1,0,0), love = (0,0,0,1)

Word2Vec (Mikolov et al. 2013) Visualization



Important is the "embedding" matrix E

Word2Vec (Mikolov et al. 2013)
The embeddings

TECHNISCHE UNIVERSITÄT DARMSTADT

- After training is finished, we only care about the matrix E
- We multiply a word's 1-hot vector with E, to get the embedding of the word

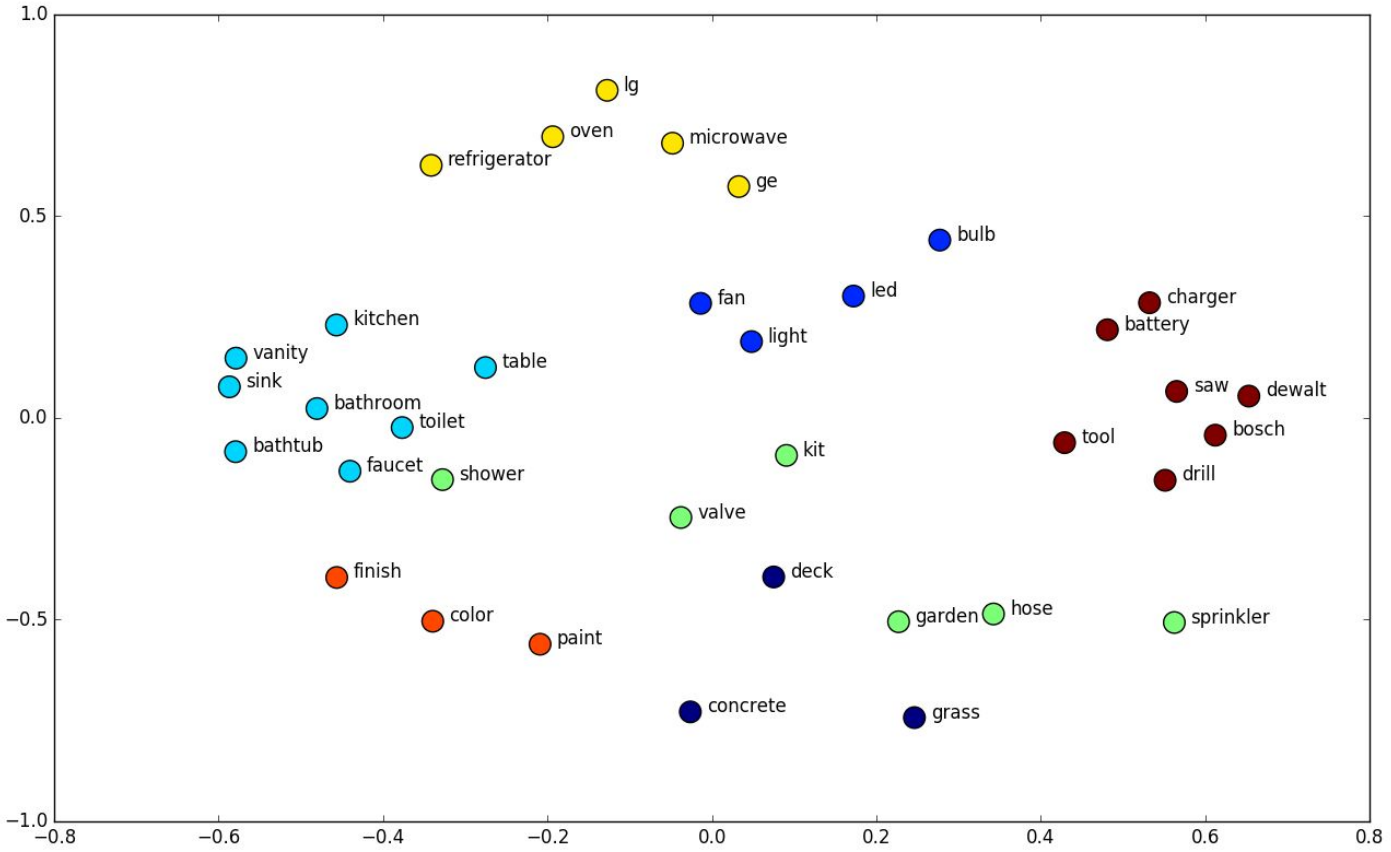
$$\underbrace{\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}}_{\text{One-hot vector}} \times \underbrace{\begin{bmatrix}8 & 2 & 1 & 9\\ 6 & 5 & 4 & 0\\ 7 & 1 & 6 & 2\\ 1 & 3 & 5 & 8\\ 0 & 4 & 9 & 1\end{bmatrix}}_{\text{Embedding Weight Matrix}} = \underbrace{\begin{bmatrix}1 & 3 & 5 & 8\end{bmatrix}}_{\text{Hidden layer output}}$$

Source: https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81?gi=633fe1a189c4

Properties of embeddings

- Observation 1: Semantically similar words are near in the vector space
- Observation 2: Syntax/Morphology properties are (sometimes) also preserved

Properties of embeddings
Semantics

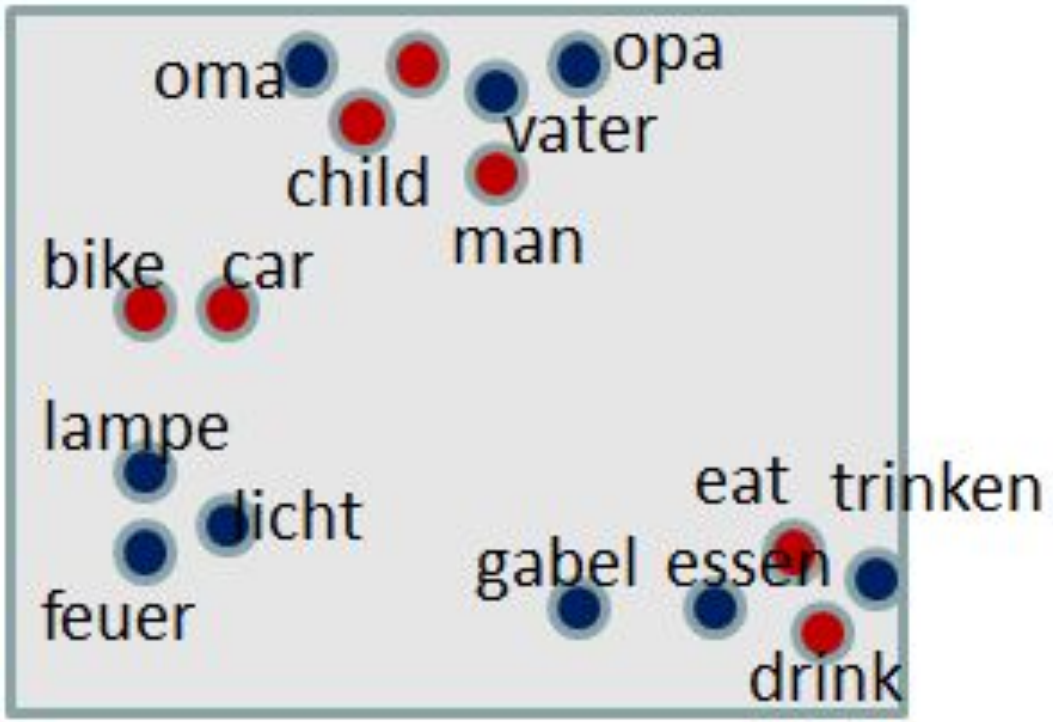


Source: https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Embeddings
Bi- and multi-lingual embeddings

TECHNISCHE UNIVERSITÄT DARMSTADT

Similar words across two or more languages should be close in vector space



Embeddings
Contextualized embeddings

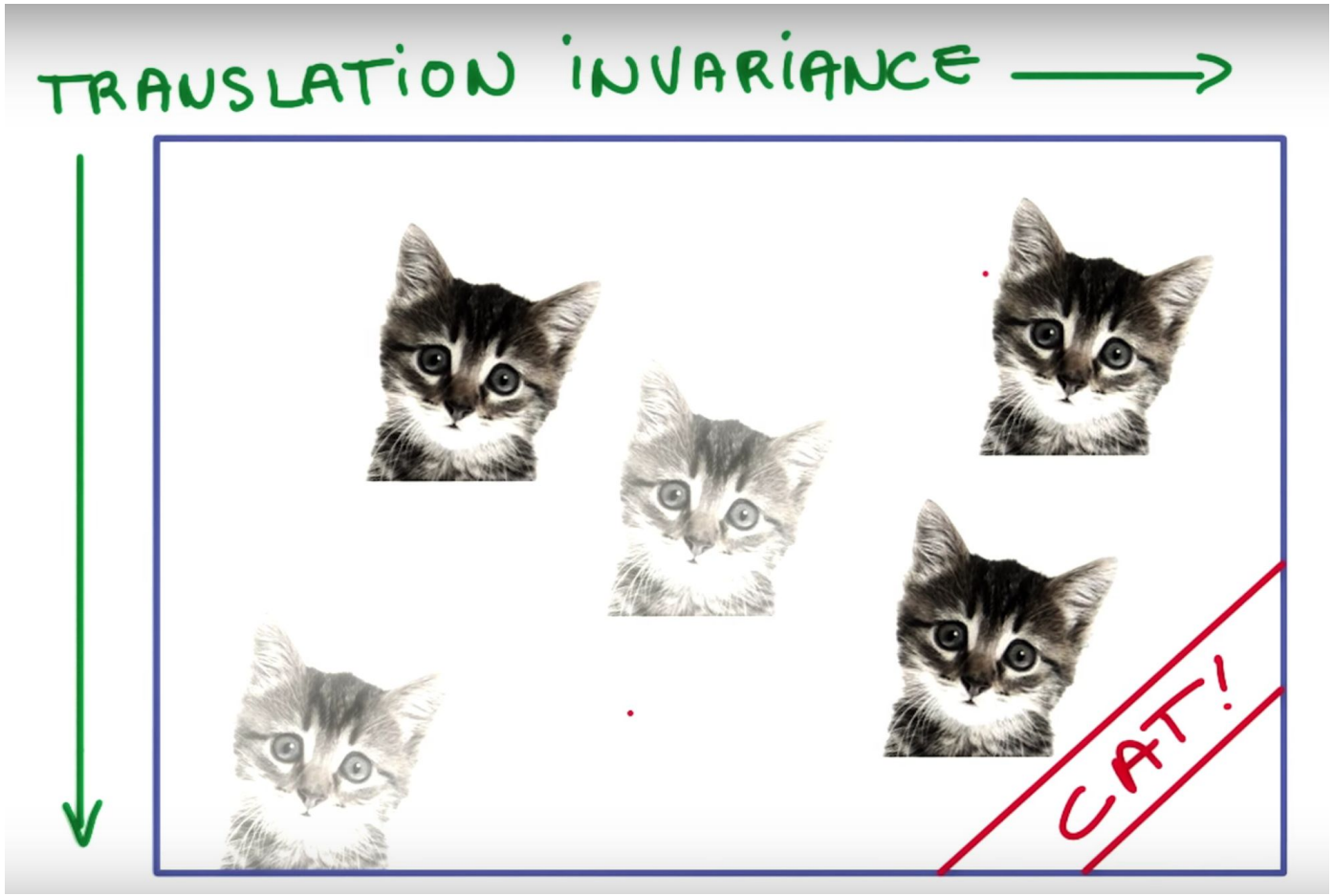
TECHNISCHE UNIVERSITÄT DARMSTADT

- Assign each word a different embedding depending on surrounding context
  - using so-called **Language Models**
- Cf.: “I **can** see you”, “Please give me that **can**”

- Have become extremely popular after 2018
- Have almost completely replaced the previous static embeddings
- Most popular: ELMo, nowadays BERT

Source: https://bensen.ai/elmo-meet-bert-recent-advances-in-natural-language-embeddings/

Convolutional Neural Networks
Translation invariance



Source: https://towardsdatascience.com/what-is-wrong-with-convolutional-neural-networks-75c2ba8fbd6f

Architectures
Recurrent Neural Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

RNNs

- are ideal for modeling **temporal data**
- where a classification should occur **at each time step**
- and the classification depends on all of the previous history (**infinite context**)

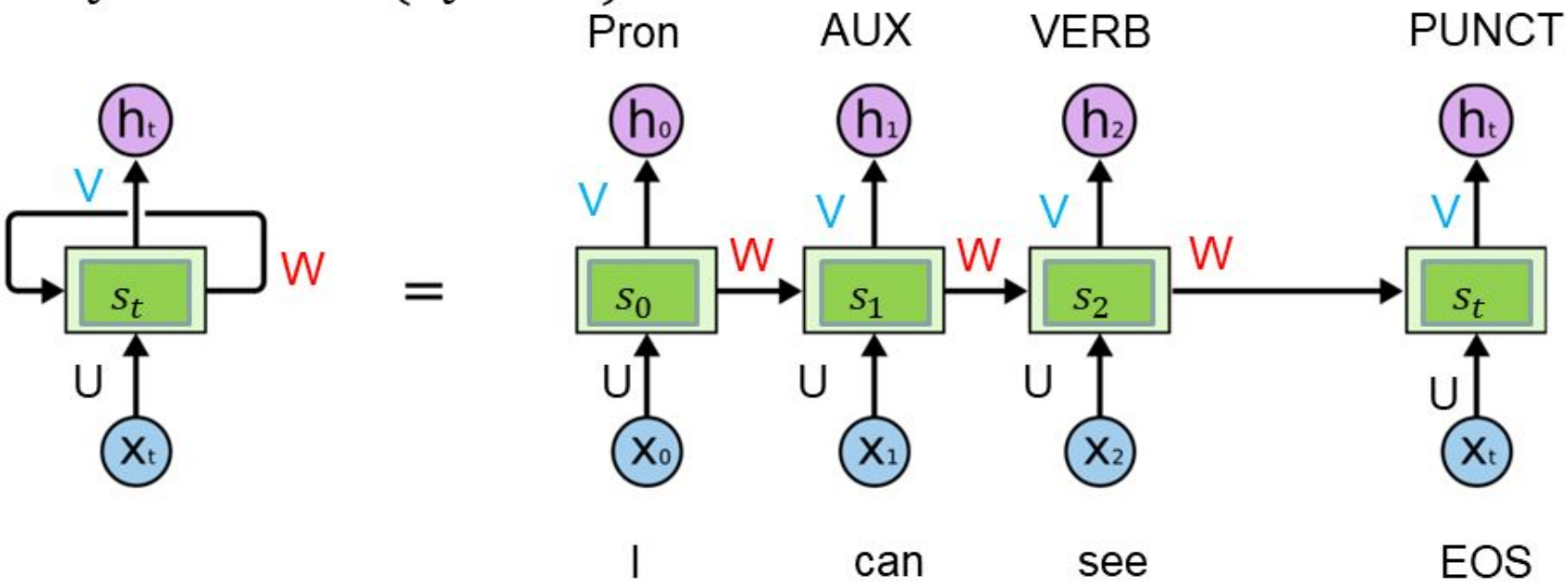
- are standard workhorses in NLP
  - because any text is a sequence of words/tokens
  - and many tasks are instances of sequential classification (POS, NER, ...)

Architectures
Recurrent Neural Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

Math: $\boldsymbol{s}_t = f(\boldsymbol{x}_t\boldsymbol{U} + \boldsymbol{s}_{t-1}\boldsymbol{W} + \boldsymbol{b})$

$\boldsymbol{h}_t = \mathrm{softmax}(\boldsymbol{s}_t\boldsymbol{V} + \boldsymbol{c})$



Architectures
Encoder-Decoder

TECHNISCHE UNIVERSITÄT DARMSTADT

Encoder-Decoder Neural Nets

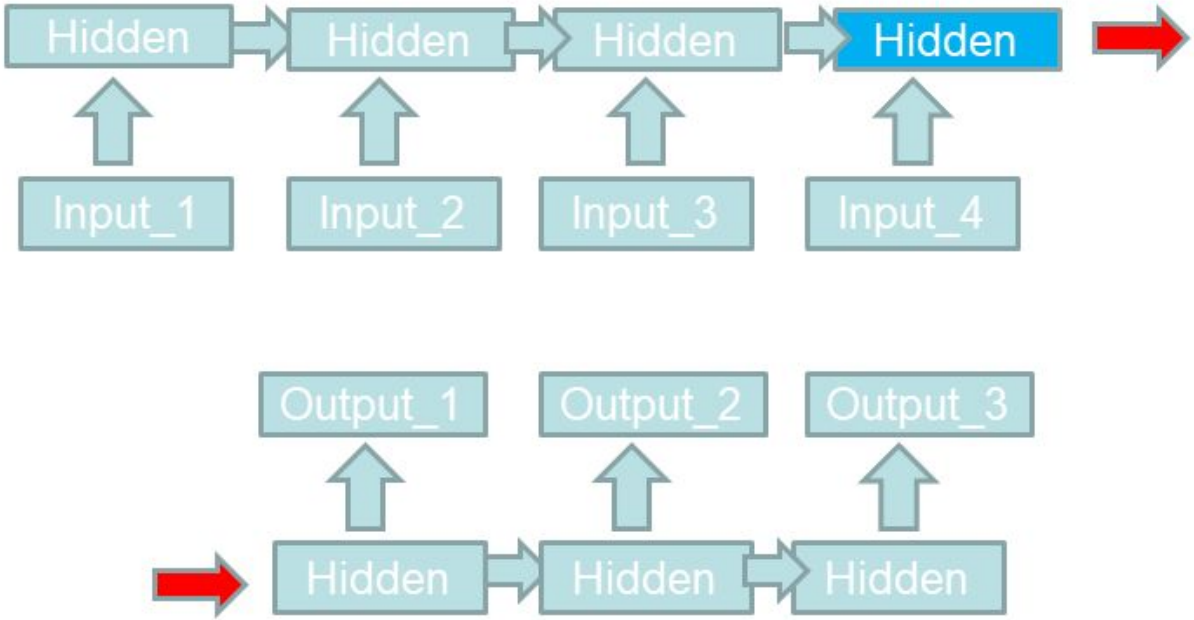
- are ideal for mapping an input sequence to an output sequence
- input and output can be of different lengths
- and the order of elements in both can be very different
- typically, they consist of two RNNs

- they are ideal for machine translation

Architectures
Encoder-Decoder

TECHNISCHE UNIVERSITÄT DARMSTADT

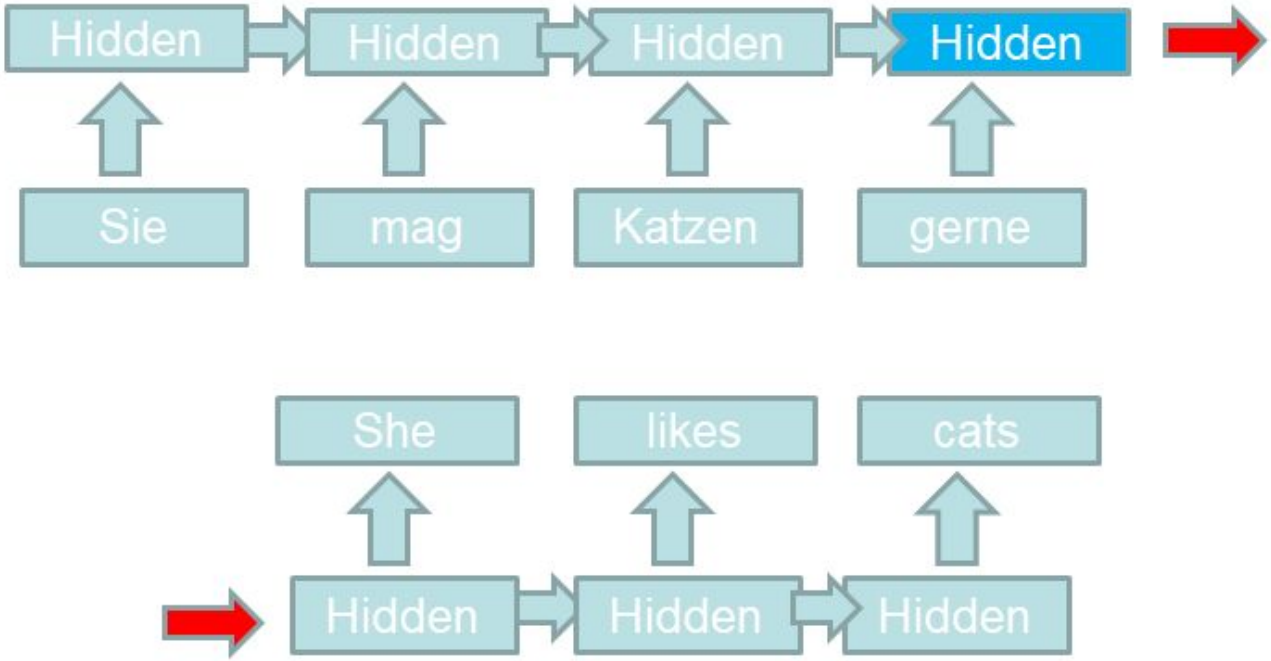
- Encoder-Decoder Models: We stack two RNNs together
- The last hidden layer in the input is taken as representation of the input



Architectures
Encoder-Decoder

TECHNISCHE UNIVERSITÄT DARMSTADT

- Encoder-Decoder Models are typically emloyed in Machine Translation
- E.g. Translate a German sentence into an English sentence



NLLG

Recent extensions
Multi-task learning

TECHNISCHE UNIVERSITÄT DARMSTADT

MTL: Learn several tasks jointly

- Like humans
- Spill-over effects
- May be beneficial when tasks are related, and when there's little data for the **main** task

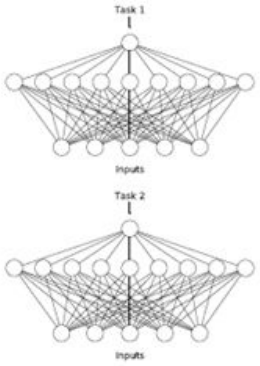


Figure: Single-task learning (inspired by Caruana (1993))

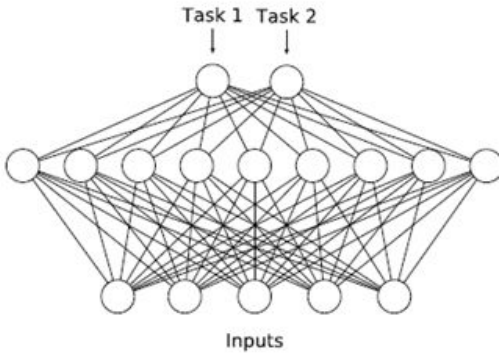


Figure: Multi-task learning (inspired by Caruana (1993))

NLLG

Recent extensions
Generative Adversarial Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

GANs: Two networks trying to outsmart each other

- Competition: “police” vs. “counterfeiters” (dt. Währungsfälscher)
- Huge success in computer vision
  - One network generates an image
  - The other determines whether it is “fake” or not

Source: Karras et al. 2018, Progressive growing of gans for improved quality stability and variation

NLLG

Recent advances
Language Modeling

TECHNISCHE UNIVERSITÄT DARMSTADT

LM: predict next word given history

- Can be used to generate text
- Has rapidly advanced with the advent of DL
  - Better, faster models (Transformer)
  - Can be trained on much more data (web-scale)
  - Has led to human-like performances in MT and general text generation

- See: https://talktotransformer.com/

DL4NLP 2020

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr. Steffen Eger
Wei Zhao

NLLG

Natural Language Learning Group

NLLG

Who are we?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Natural Language Learning Group (NLLG)
  - Aka “NLP for the Humanities”
  - https://www.informatik.tu-darmstadt.de/aiphes/aiphes/irg_position/index.en.jsp
- Research interests:
  - Cross-linguality
  - Cross-temporality
  - Automatic Poetry generation
  - Evaluation

Who are we?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Natural Language Learning Group (NLLG)
  - Aka “NLP for the Humanities”
- Courses:
  - Deep Learning and Digital Humanities (WS)
    - https://github.com/SteffenEger/dldh
  - Meta-Science (SS)
    - https://github.com/SteffenEger/MetaScience
  - DL4NLP (SS)
  - FOLT (WS)

Who are we?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Natural Language Learning Group (NLLG)
  - Aka “NLP for the Humanities”
- Theses available!
- Hiwi positions also available

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).