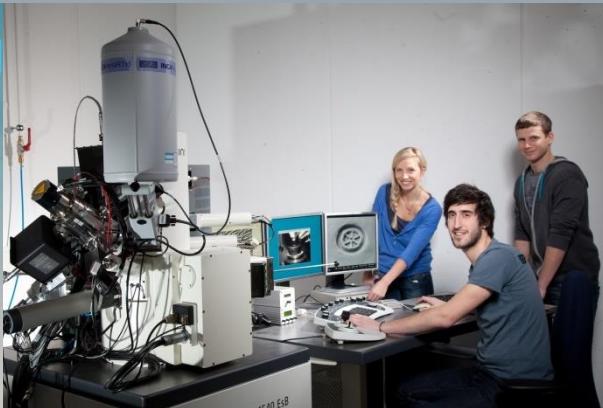
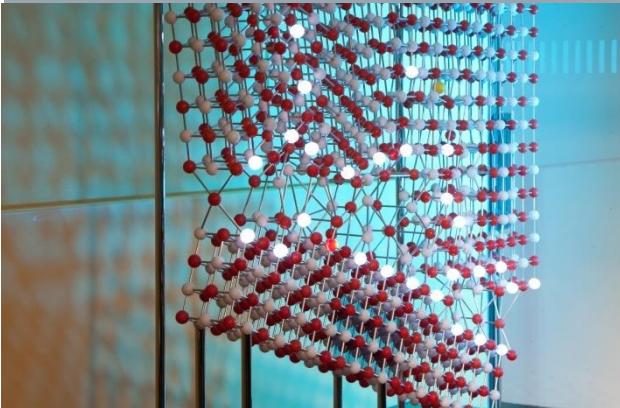


# Machine Learning for Engineers

## Organizational Information



Bilder: TF / Malter

# Course Organizers

**Prof. Dr. Bjoern M. Eskofier**

Machine Learning and Data Analytics Lab  
University Erlangen-Nürnberg

**Prof. Dr. Jörg Franke**

Institute for Factory Automation and Production Systems  
University Erlangen-Nürnberg

**Prof. Dr. Nico Hanenkamp**

Institute for Resource and Energy Efficient Production Systems  
University Erlangen-Nürnberg

# Course Details

**5 ECTS**

**4 SWS**

## Written Exam

- Exam is in English. The questions are based on understanding the methods and ability to implement them.
- Two exercises should be successfully completed.
- Exam date will be announced for all universities.

# Course Structure

## Lectures

1. Introduction to Machine Learning (today)
2. Linear Regression
3. Support Vector Machines
4. Dimensionality Reduction
5. Deep Neural Networks

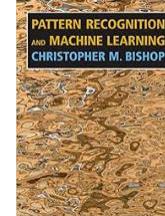
## Exercises

Two real-world industrial applications

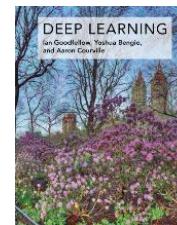
- Exercise 1: Energy prediction using Support Vector Machines
- Exercise 2: Image classification using Convolutional Neural Networks

# Lecture Material

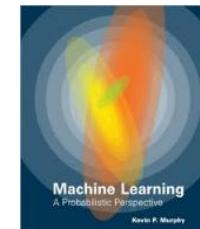
**Pattern Recognition and Machine Learning**  
by C. Bishop, 2007



**Deep Learning**  
by A. Courville, I. Goodfellow, Y. Bengio, 2015



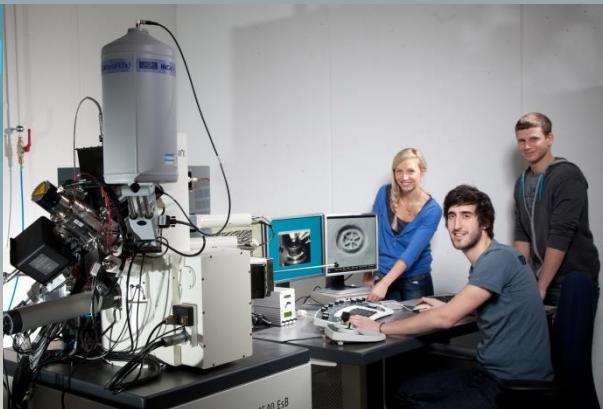
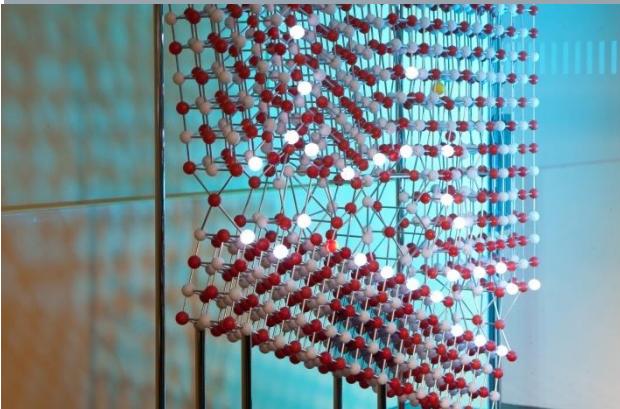
**Machine Learning: A Probabilistic Perspective**  
by K. Murphy, 2012





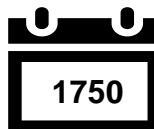
# Introduction to Machine Learning

## Motivation

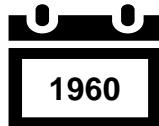
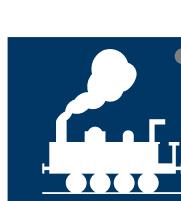


Bilder: TF / Malter

# Four Industrial Revolutions

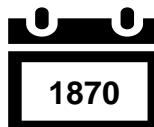
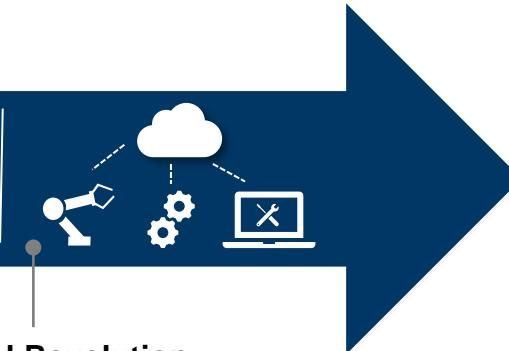


**1. Industrial Revolution**  
Steam machines allow the Industrialization

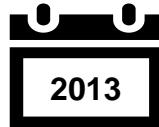


## 3. Industrial Revolution

Electronics and IT enable automation-driven rationalization as well as varied series production



**2. Industrial Revolution**  
Mass division of labor with the help of electrical energy



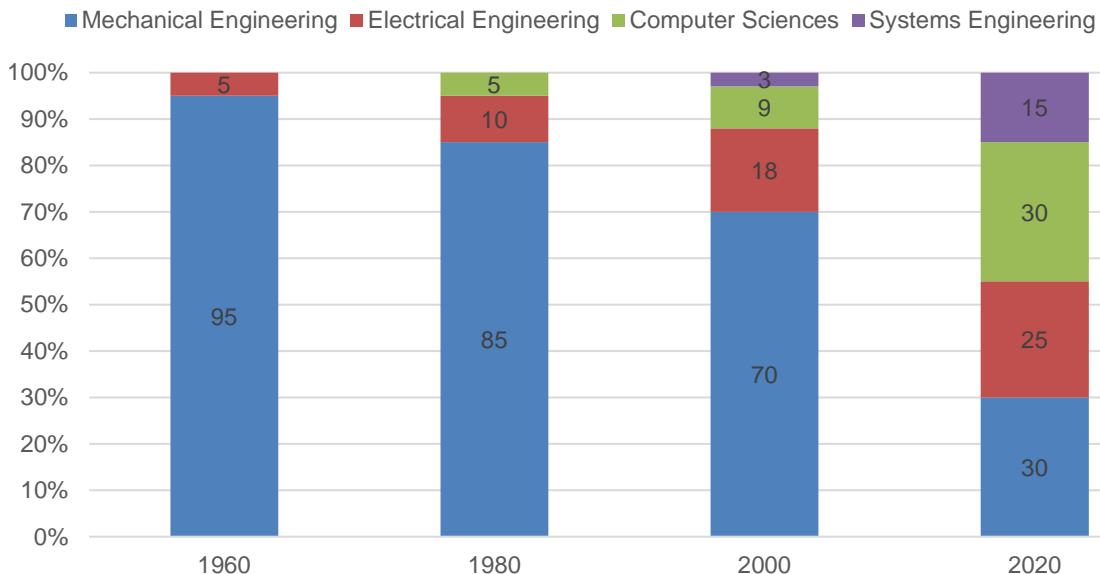
## 4. Industrial Revolution

Industrial production is interlinked with information and communication technology  
In April 2013, the national platform Industry 4.0 was founded at the Hannover Messe

Source: Industrie 4.0 in Produktion, Automatisierung und Logistik, T. Bauernhansl, M. Hompel, B. Vogel-Heuser, 2014

# Core of the Fourth Industrial Revolution is Digitalization

## Percentage of Engineering in Mechanical and Plant Engineering



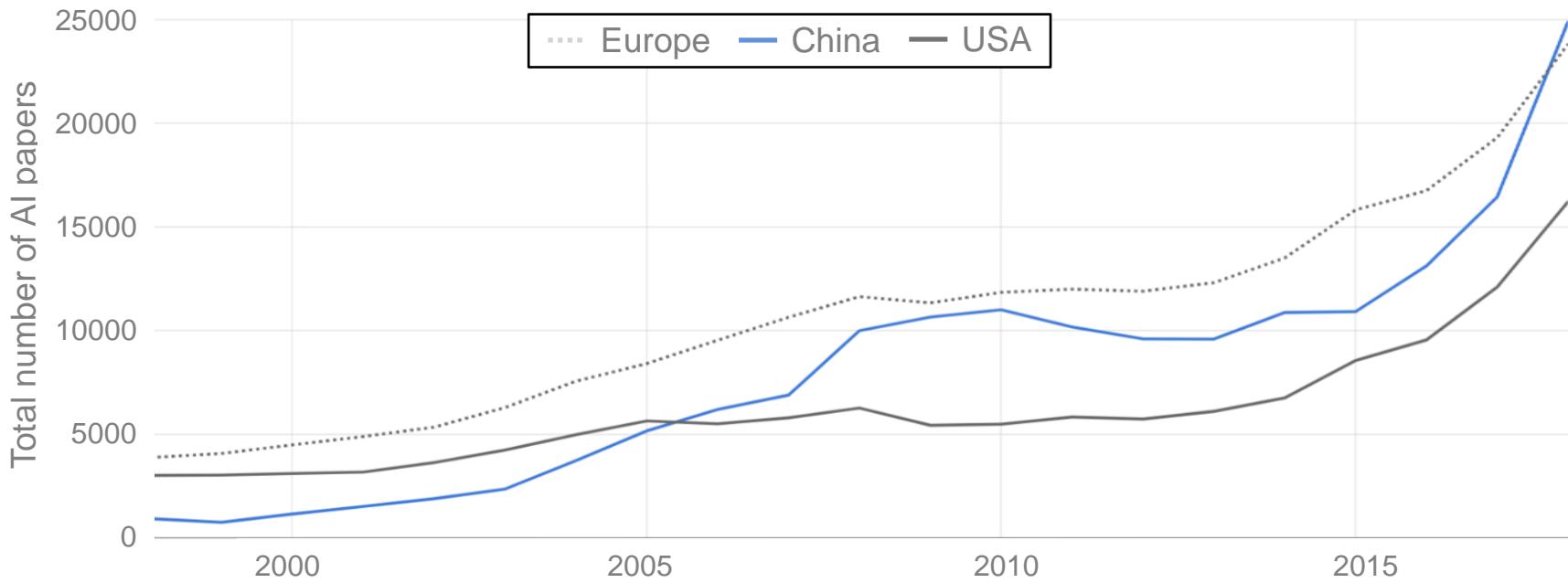
- Mechanical and plant engineering was dominated by mechanical engineering in 1960s
- Share of mechanical engineering has been steadily decreasing until today
- Impact of computer sciences has been increasing since 1980s

→ Mechanical and plant engineering is composed of multiple fields and became an interdisciplinary area unlike 1960s

Source: Automatisierung 4.0, T. Schmertosch, M. Krabbes, 2018

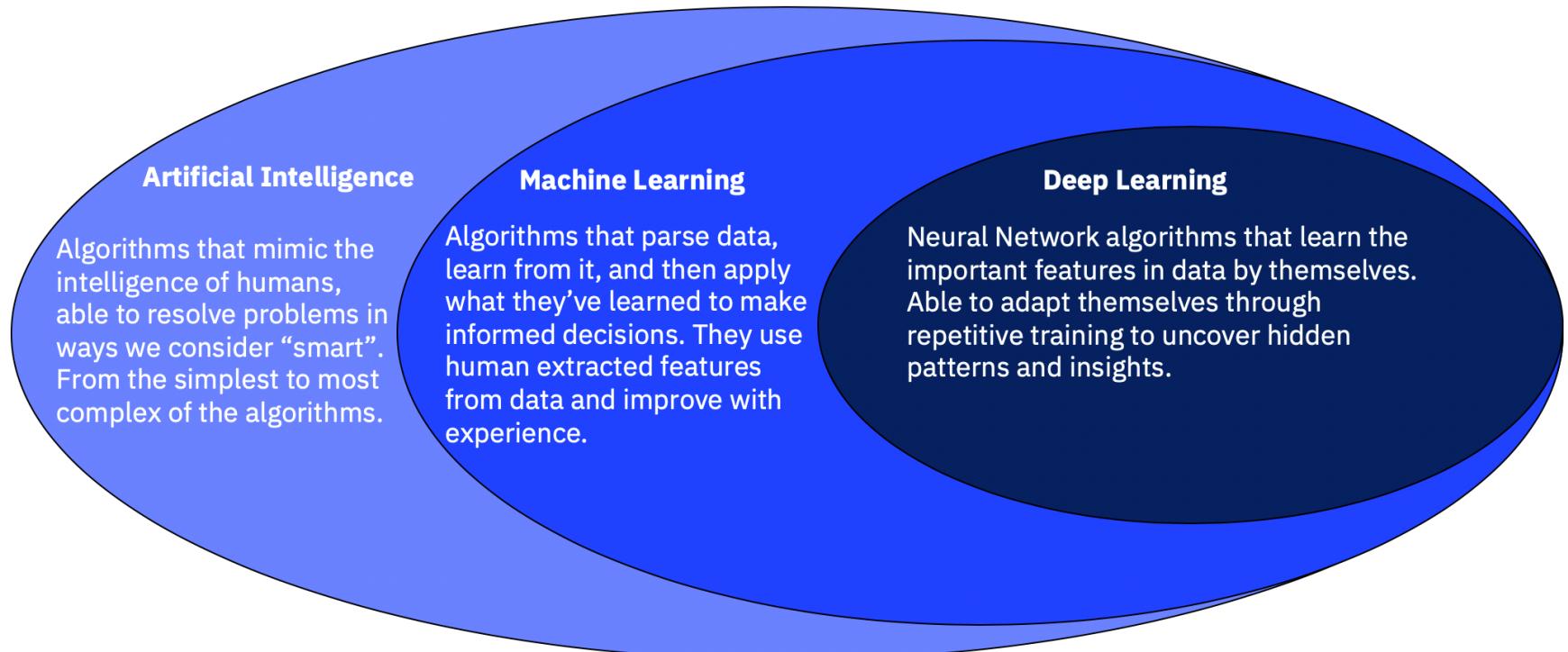
# What drives that progress in Computer Science?

Annual number of AI papers



Source: Left: Scopus, 2019

# Artificial Intelligence vs. Machine Learning



Source: <https://www.ibm.com/blogs>

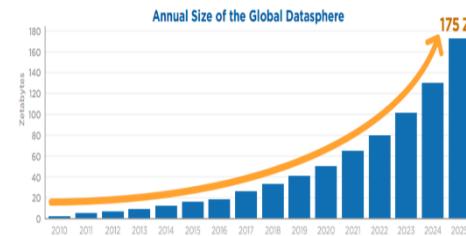
# Driving factors for the Advancement of AI

There are three important reasons for these advancements!

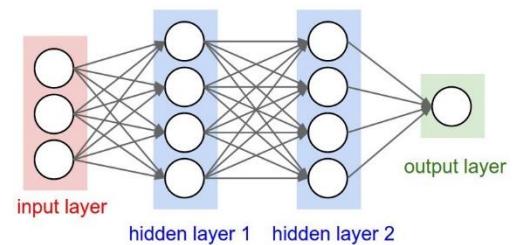
Increase of the computational power



Increase of the amount of available data



Development of new algorithms



Sources:

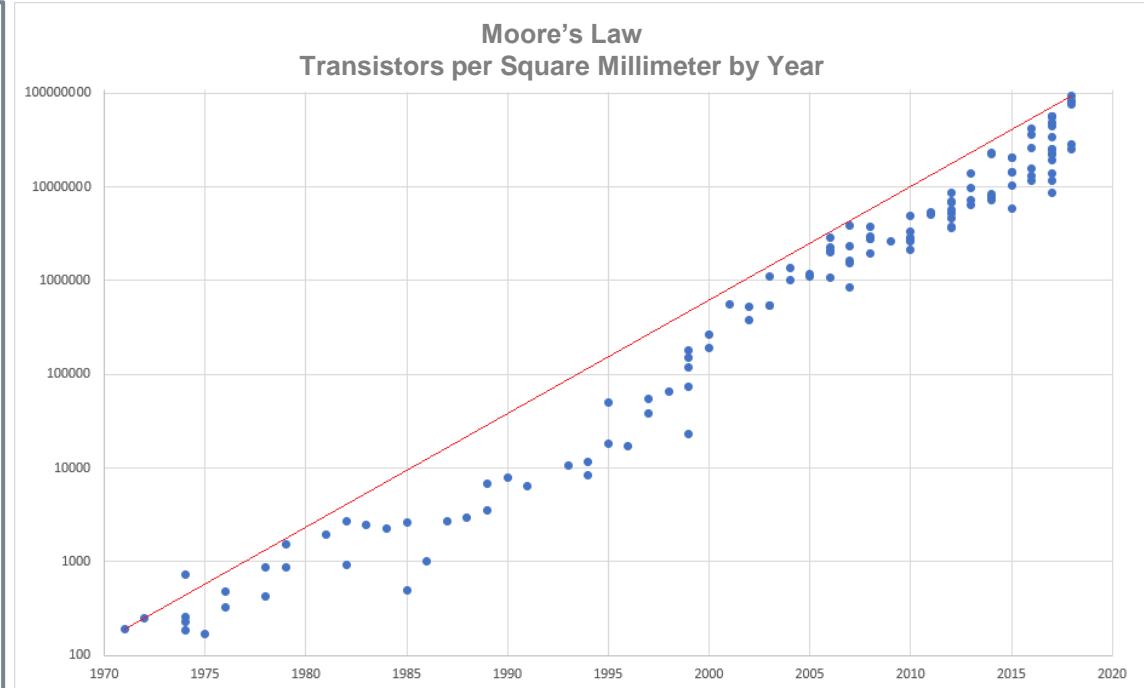
Left Image: <https://datafloq.com>

Middle Image:<https://europeansting.com>

Right Image: <https://medium.com>

# Increase of Computational Power

- Transistors are the building blocks of CPUs
- Numbers of transistors are correlated with computational power
- Number of transistors has been increasing, hence the computational power
- This phenomenon is stated in Moore's law



Source: <https://medium.com>

# Computational Power of today's supercomputers

## TOP500 Supercomputers (2020)

June 2020	System	Specs	Site	Country	Cores	Rmax Pflop/s	Power
1	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,288,072	415.5	28.3
2	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
3	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.4
4	Sunway TaihuLight	Shenwei SW26010 (260C, 1.45GHz) Custom Interconnect	NSCC in Wuxi	China	10,649,600	93	15.4
5	Tianhe-2A	Intel Ivy Bridge (12C, 2.2GHz)	NSCC Guangzhou	China	4,981,760	61.4	18.5

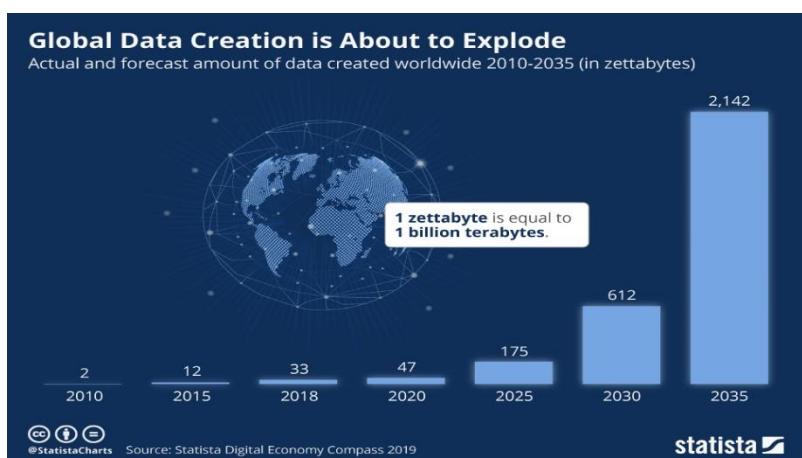
The best german supercomputer:

13	SuperMUC-NG	ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path	Leibniz RZ	Germany	305,856	19	-
----	-------------	---	------------	---------	---------	----	---

Source: <https://www.top500.org>

# Increase in the Amount of Available Data

The amount of created data increased from two zettabytes in 2010 to 47 zettabytes in 2020

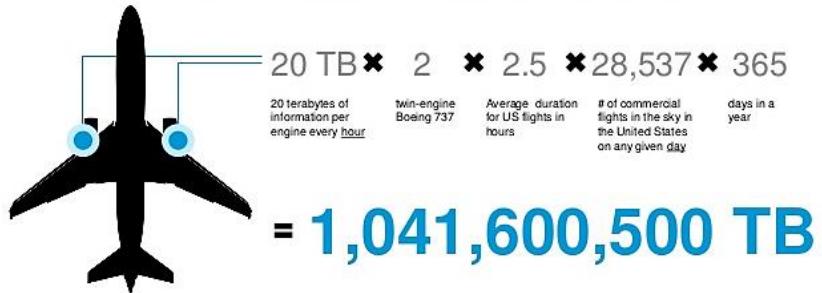


<https://www.statista.com>

According to General Electric (GE), each of its aircraft engines produces around twenty terabyte of data per hour

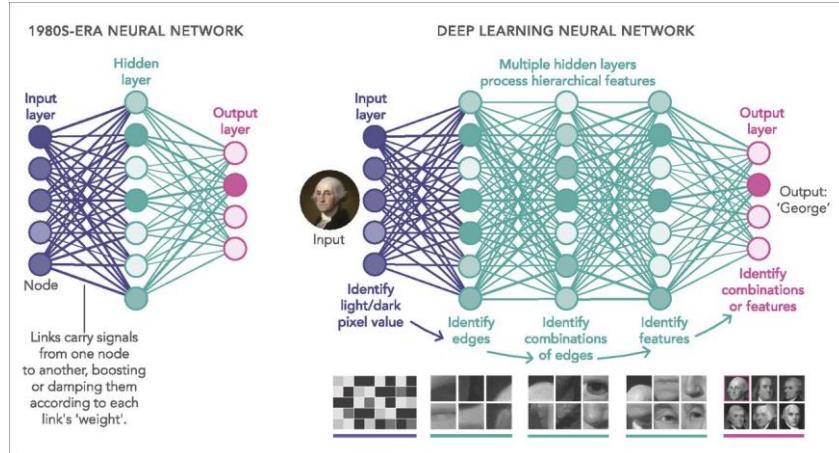
## A real world example:

Sensor data collected from US commercial jet engines during 1 year

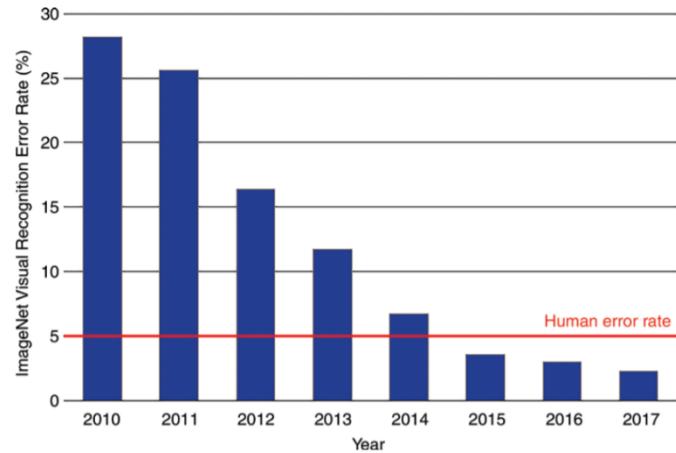


<https://www.forbes.com>

# Development of New Algorithms



- In 2006, Geoffrey Hinton et al. published a paper showing how to train a deep neural network
- They branded this technique “Deep Learning.”



- Increasing performance of visual object recognition
- In 2015 machine learning algorithms exceed the human capability

Source: A Roadmap for Foundational Research on Artificial Intelligence in Medical Imaging, Curtis P. Langlotz et al., 2018

# Industrial Examples – Real Environments

## KUKA Bottleflip Challenge



- Challenge called «Bottleflipping»
- Robot trained itself over night

Source: <https://www.youtube.com/watch?v=HkCTRcN8TB0&t>

## WAYMO driverless driving



- Autonomous driving
- A traditional human job is carried out by machine learning algorithms

Source: <https://www.youtube.com/watch?v=aaOB-ErYq6Y&t>

# Game Examples – Defined Rules

## AlphaGo and Go



- Go is considered as the most challenging classical game
- In October 2015, AlphaGo won against the European Champion

## AlphaStar and Starcraft 2



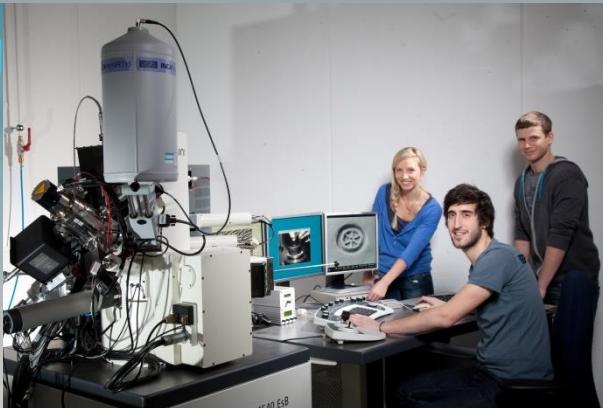
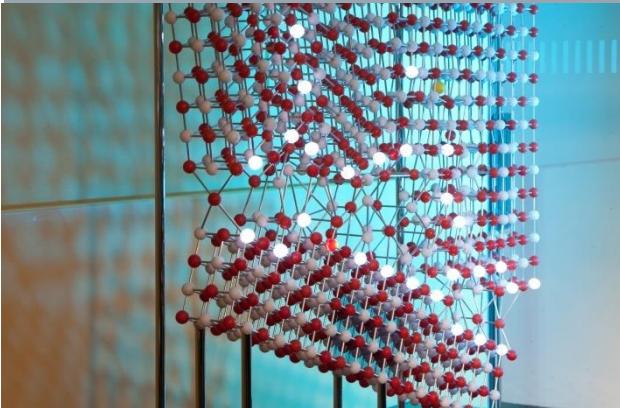
- StarCraft II is one of the most popular real-time strategy video games
- AlphaStar is the first AI that reached the top league

Source: <https://deepmind.com>



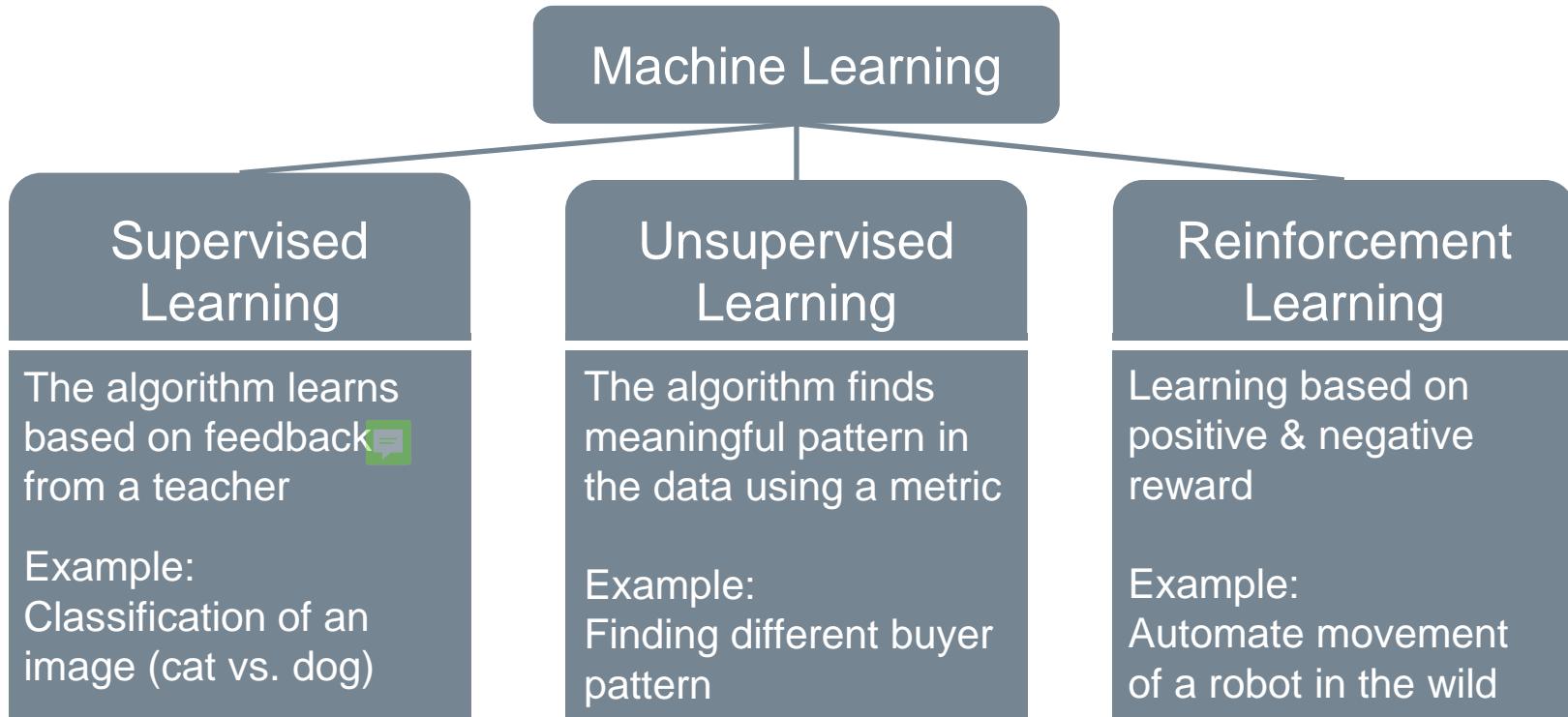
# Introduction to Machine Learning

## Machine Learning Types



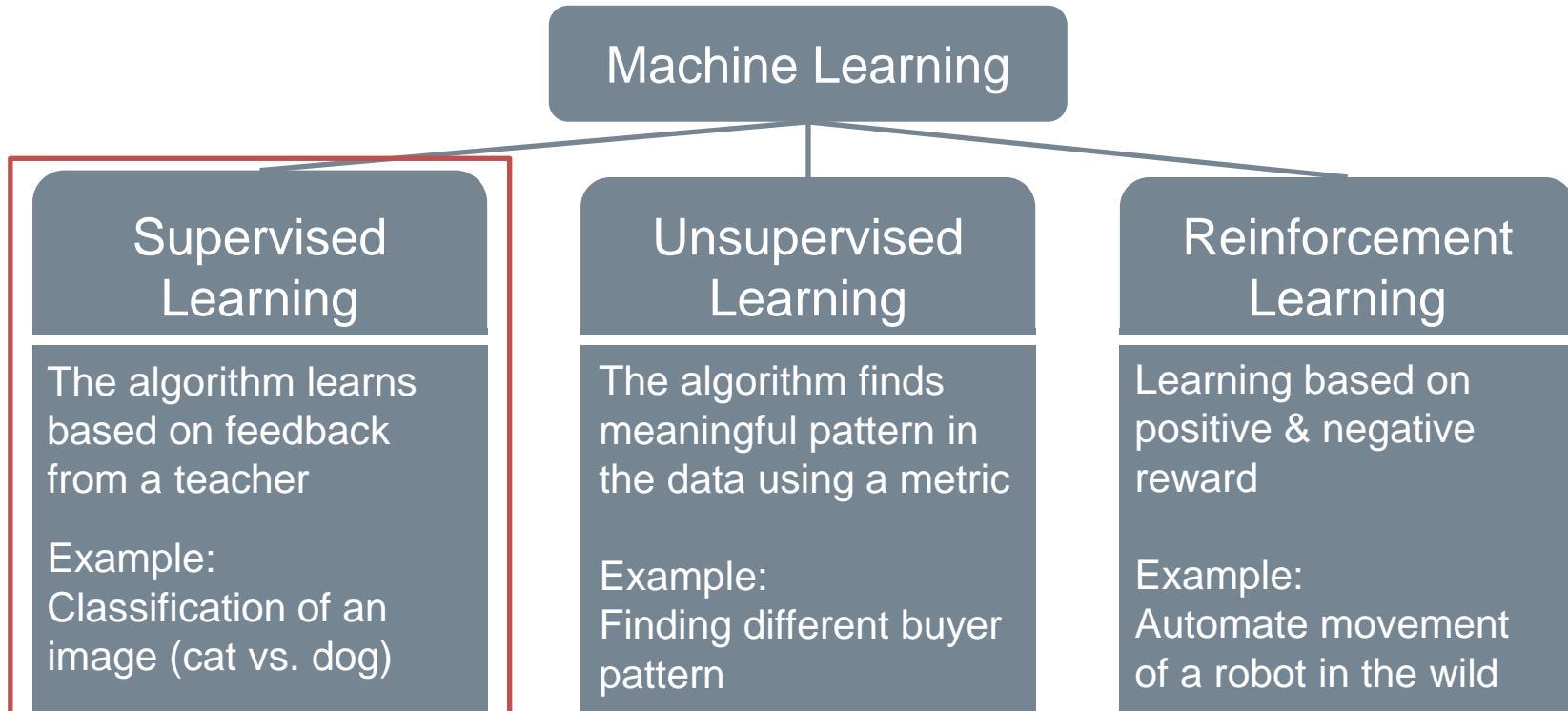
Bilder: TF / Malter

# Machine Learning Categories



Source: Business Intelligence and its relationship with the Big Data, Data Analytics and Data Science, Armando Arroyo, 2017

# Machine Learning Categories



Source: Business Intelligence and its relationship with the Big Data, Data Analytics and Data Science, Armando Arroyo, 2017

# Supervised Learning

The agent  observes some example **Input (Features)** and **Output (Label) pairs** and learns a **function that maps input to output**.

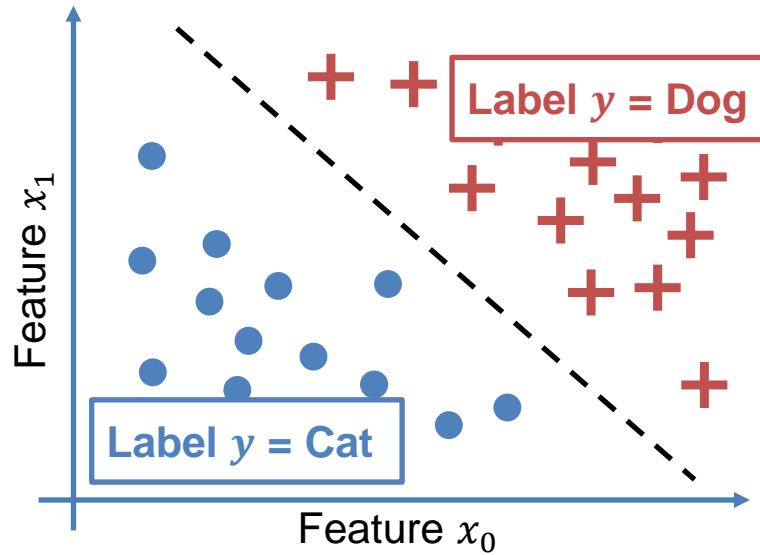
Key Aspects:

- Learning is **explicit**
- Learning using **direct feedback**
- Data with **labeled output**

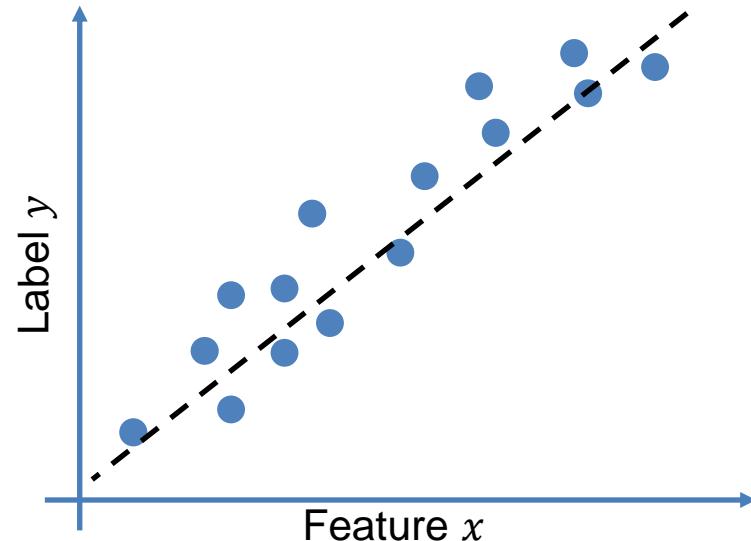
→ Resolves classification and regression problems

# Supervised Learning Problems

## Classification



## Regression



Source: <https://www.kdnuggets.com>

# Regression

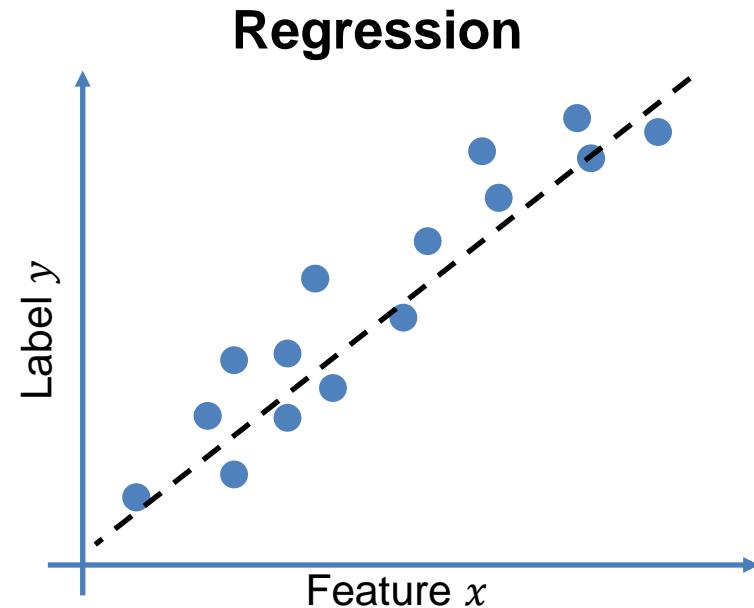
Regression is used to predict  
a **continuous value**



Training is based on a set of  
input – output pairs (**samples**)  
 $\mathcal{D} = \{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\}$

Sample :  $(x_i, y_i)$

$x_i \in \mathbb{R}^m$  is the **feature vector** of sample  $i$   
 $y_i \in \mathbb{R}$  is the **label value** of sample  $i$



# Regression

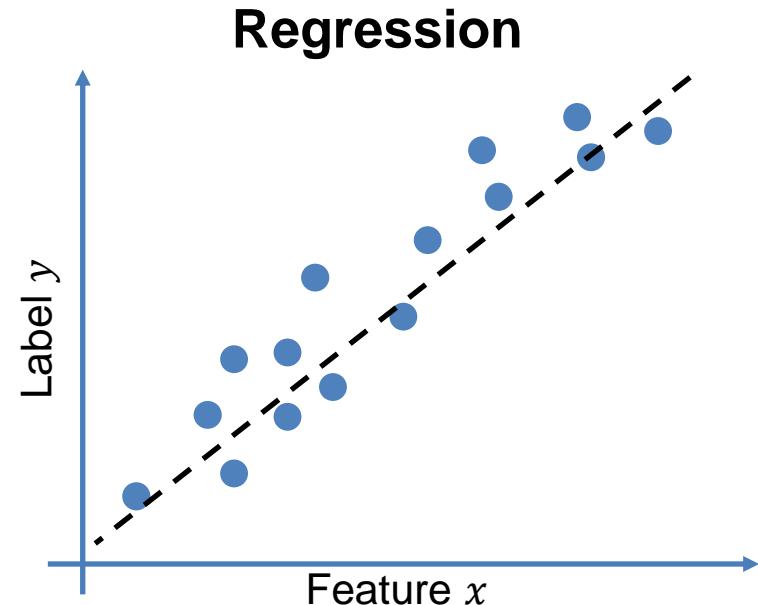
## Goal:

Find a relationship (function), which expresses the input and output the best!

That means, we **fit a regression model**  $f$  to all samples:

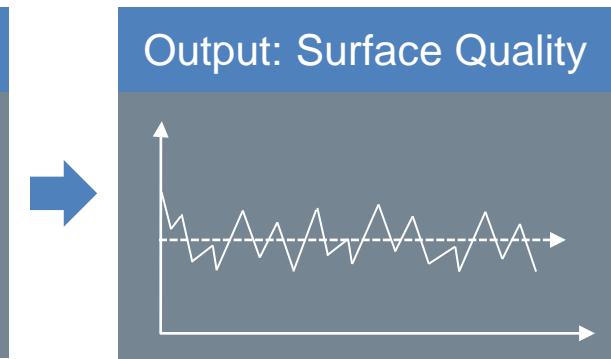
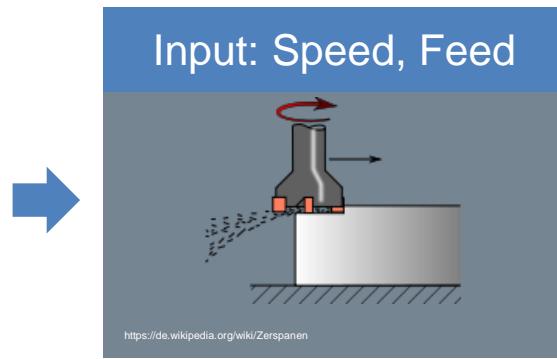
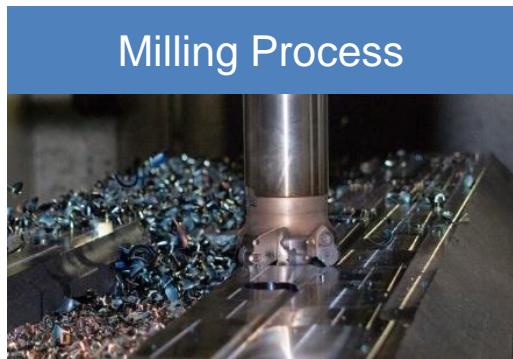
$$f(x_i) = y_i \quad , \forall (x_i, y_i) \in \mathcal{D}$$

In this case  $f$  is a **linear regression model!** (Black Line)



# Example: Predicting Surface Quality

In this example, we consider a **milling** process. We take **speed** and **feed** as input features and the value of **surface roughness** is predicted



## Objective:

- Prediction of the surface quality based on the production parameters

## Realization:

- Speed and Feed data is gathered and the surface roughness measured for some trials
- By using regression algorithms surface quality is predicted

# Example: Prediction of Remaining Useful Life (RUL)

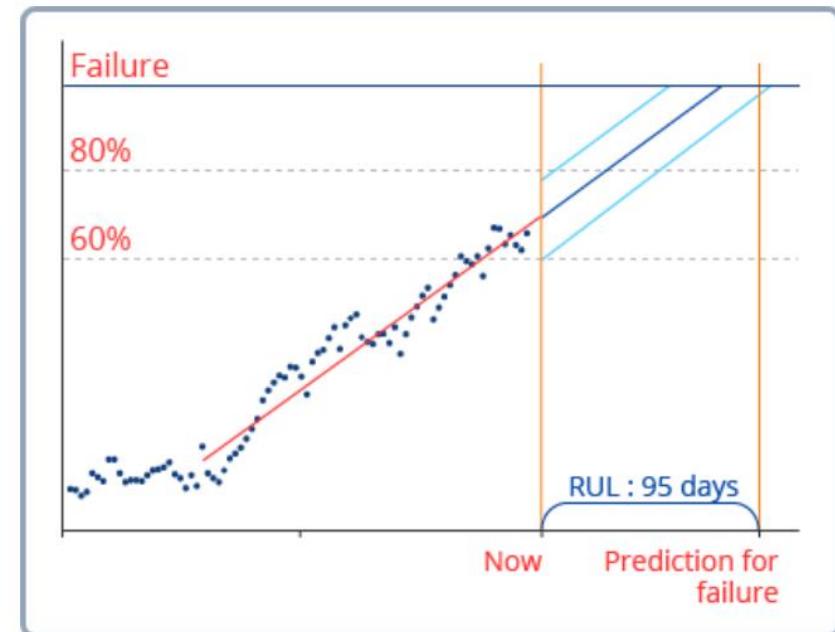
“Health Monitoring and Prediction (HMP)” system (developed by BISTel)

## Objective:

- Machine components **degrade over time**
- This causes malfunctions in the production line
- Replace component **before** the malfunction!

## Realization:

- Use data of malfunctioning systems
- Fit a regression model to the data and predict the RUL
- Use the model on active systems and apply necessary maintenance



Source: <https://www.bistel.com/>

# Classification

Classification is used to predict the **class** of the input

## Important:

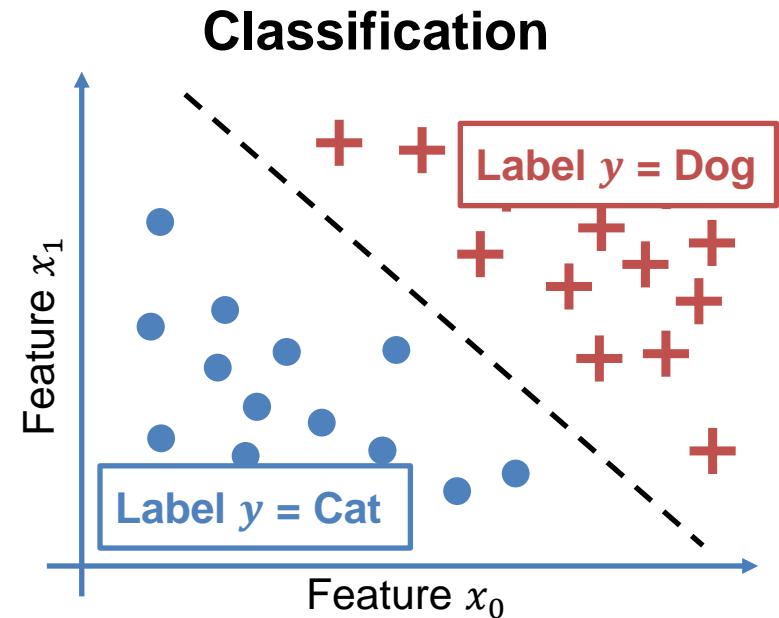
The output belongs to only one class

Sample :  $s_i = (\vec{x}_i, \vec{y}_i)$

$\vec{y}_i \in L$  is the **label** of sample  $i$

In this example:  $L = \{\text{Cat}, \text{Dog}\}$

Source: <https://scikit-learn.org/>



# Classification

## Goal:

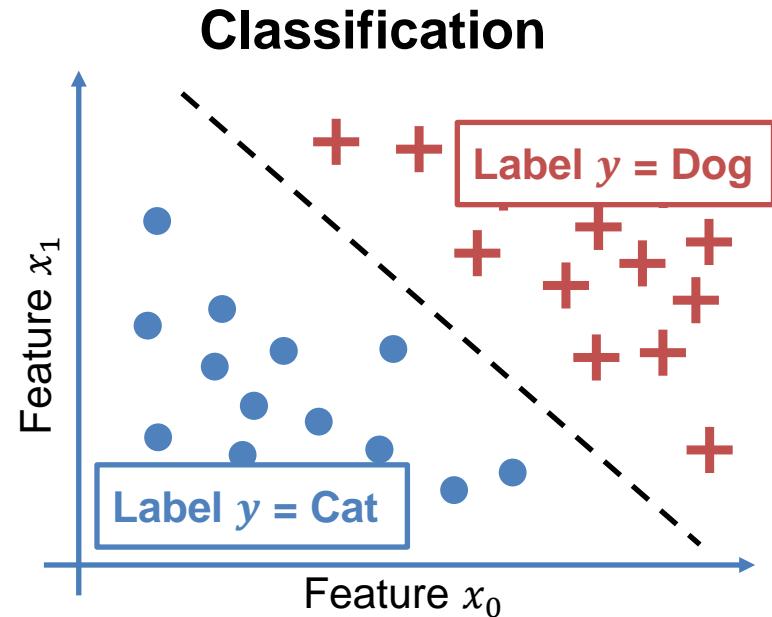
Find a way to divide the input into the output classes!

That means, we **find a decision boundary**  $f$  for all samples:

$$f(x_i) = y_i \quad , \forall (x_i, y_i) \in \mathcal{D}$$

In this case  $f$  is a **linear decision boundary!** (Black Line)

Source: <https://scikit-learn.org/>



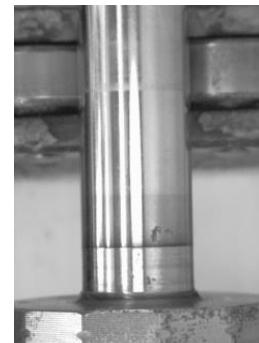
# Example: Foreign object detection

## Objective:

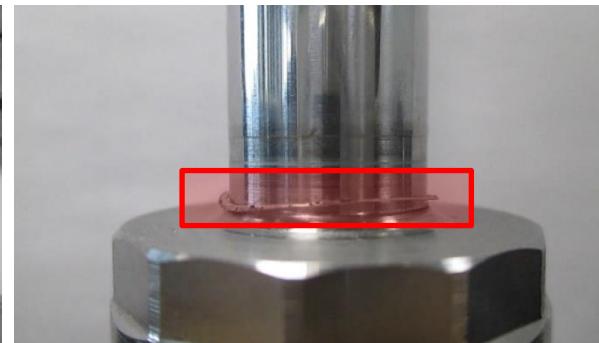
- Foreign object detection on a production part
- After a production step a chip can remain on the piston rod
- Quality assurance: **Only parts without chip are allowed** for the next production step

## Realization:

- A camera system records 3,000 pictures
- All images are labeled by a human
- A machine learning classification algorithm was trained to **differentiate between chip or no chip situations**



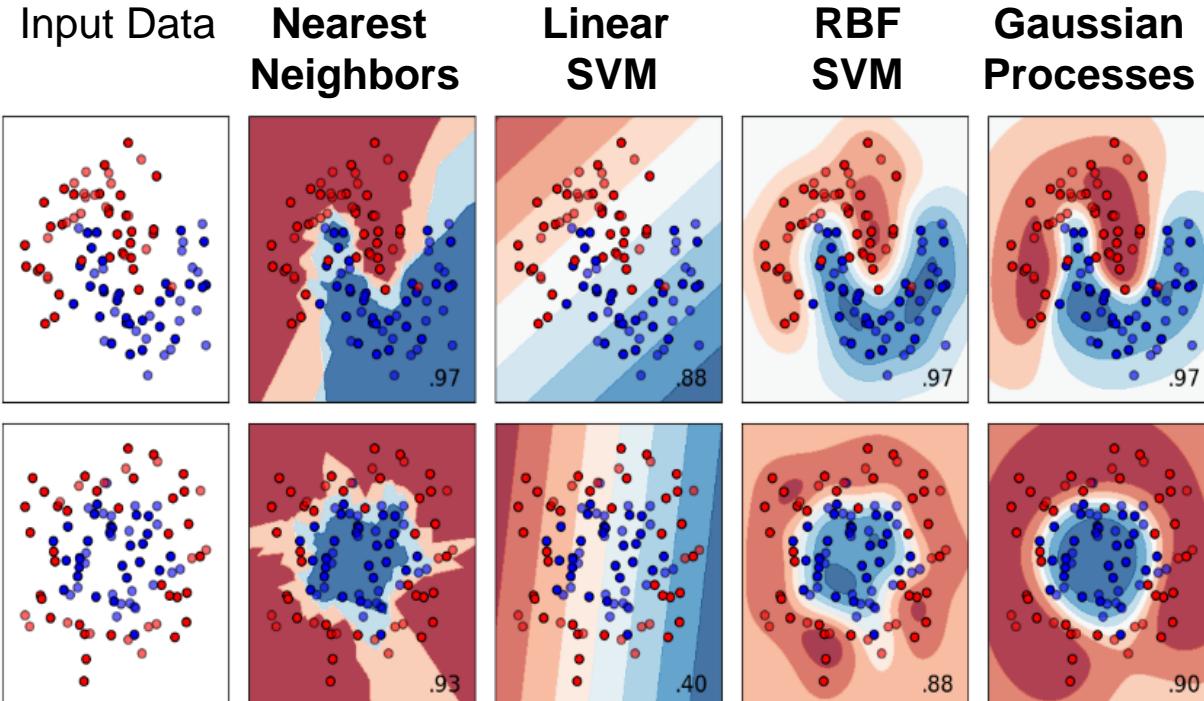
Piston rod  
without chip



Piston rod with chip

Source: Implementation and potentials of a machine vision system in a series production using deep learning and low-cost hardware, Hubert Würschinger et al., 2020

# Classification algorithms



The lower right value shows the classification accuracy

Source: <https://scikit-learn.org/>

Different algorithms use different ways to classify data

Therefore:  
The algorithms **perform different** on the datasets

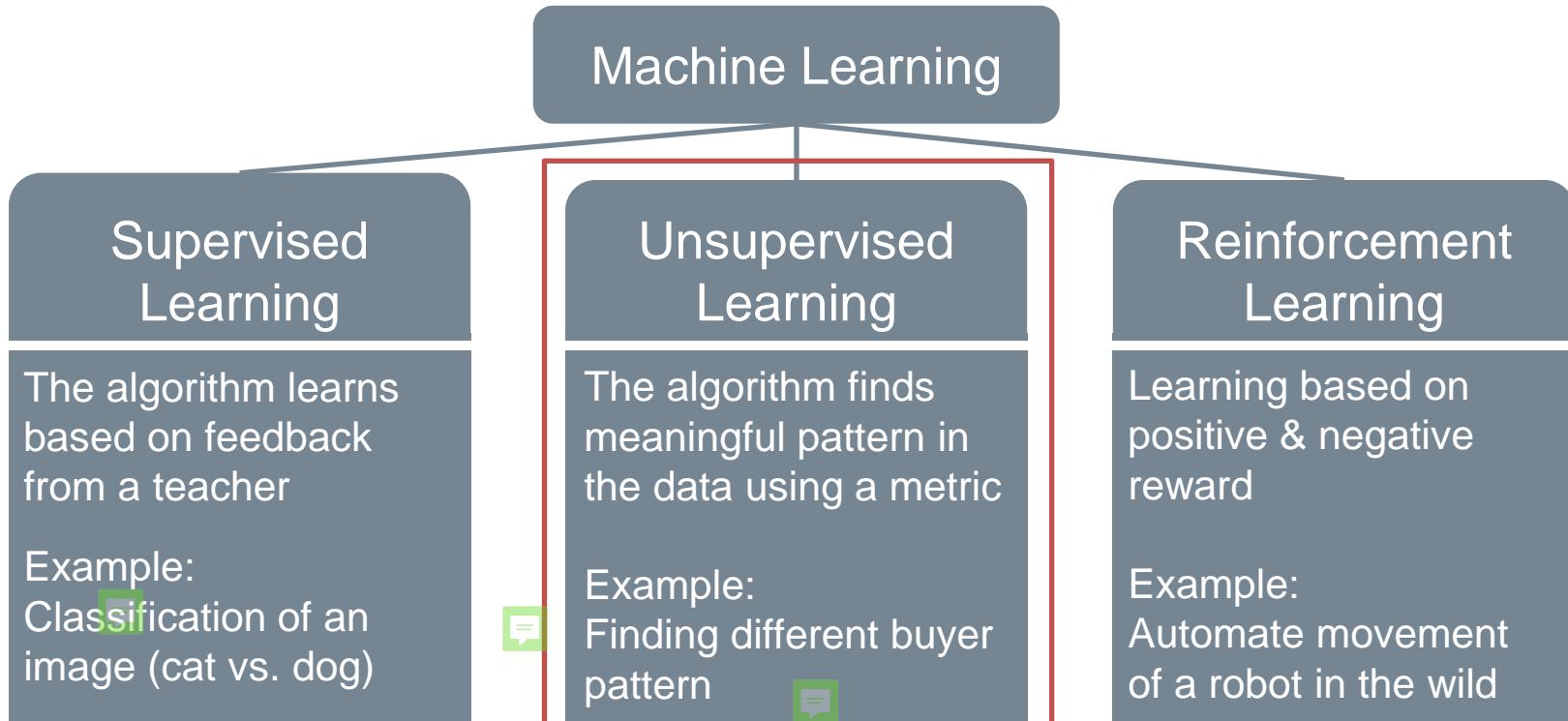
**Example:**  
Linear SVM's has bad results in the second dataset  
**Reason:** No linear way to distinguish data

# Classification vs. Regression

Regression	Classification
<ul style="list-style-type: none"><li>The output are <b>continuous or real values</b> </li></ul>	<ul style="list-style-type: none"><li>The output variable must be a <b>discrete value (class)</b> </li></ul>
<ul style="list-style-type: none"><li>We try to fit a <b>regression model</b>, which can predict the output more accurately</li></ul>	<ul style="list-style-type: none"><li>We try to find a <b>decision boundary</b>, which can divide the dataset into different classes.</li></ul>
<ul style="list-style-type: none"><li>Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, Stock market prediction etc.</li></ul>	<ul style="list-style-type: none"><li>Classification Algorithms can be used to solve classification problems such as Hand-written digits(<b>MNIST</b>), Speech Recognition, Identification of cancer cells, Defected or Undefected solar cells etc.</li></ul>

Source: <https://scikit-learn.org/>

# Machine Learning Categories



Source: Business Intelligence and its relationship with the Big Data, Data Analytics and Data Science, Armando Arroyo, 2017

# Unsupervised Learning

Unsupervised learning observes some example Input (Features) – **No Labels!** - and finds patterns based on a metric

Key Aspects:

- Learning is **implicit**
- Learning using **indirect feedback**
- Methods are **self-organizing**

Resolves **clustering** and **dimensionality reduction** problems

# Example for Clustering

## Iris Flower data set

Contains 50 samples of the flowers **Iris setosa**, **Iris virginica** and **Iris versicolor**

### Measured parameters:

petal length, petal width, sepals length, sepals width



Iris setosa



Iris versicolor



Iris virginica

Source: Fisher's Iris data set

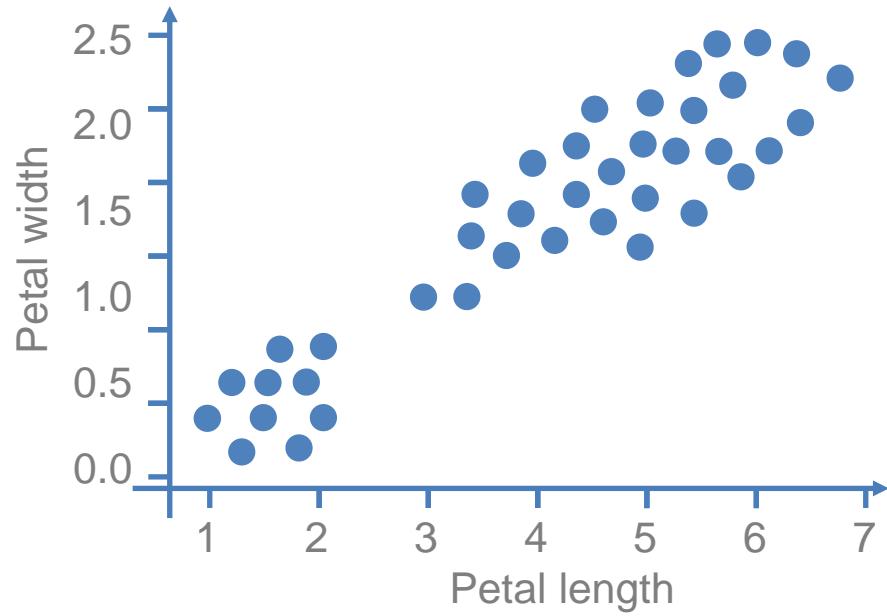
# Clustering

**Goal:** Identify similar samples and assign them the same label

Mostly used for data analysis,  
data exploration, and/or  
data preprocessing

Data should be:

- Homogeneous in the cluster  
(Intra-cluster distance) 
- Heterogenous between the cluster  
(Inter-cluster distance)



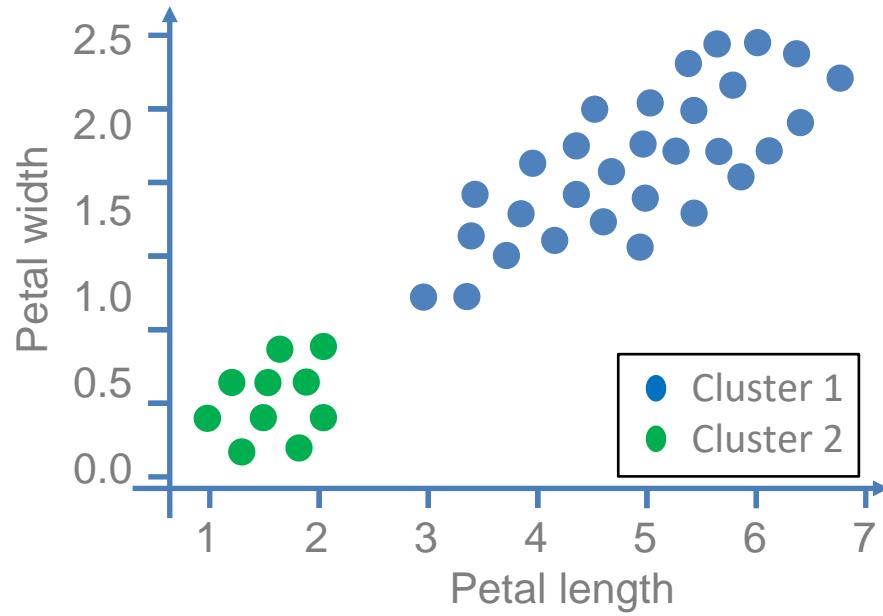
# Clustering

**Goal:** Identify similar samples and assign them the same label

Mostly used for data analysis,  
data exploration, and/or  
data preprocessing

Data should be:

- Homogeneous in the cluster  
(Intra-cluster distance)
- Heterogenous between the cluster  
(Inter-cluster distance)

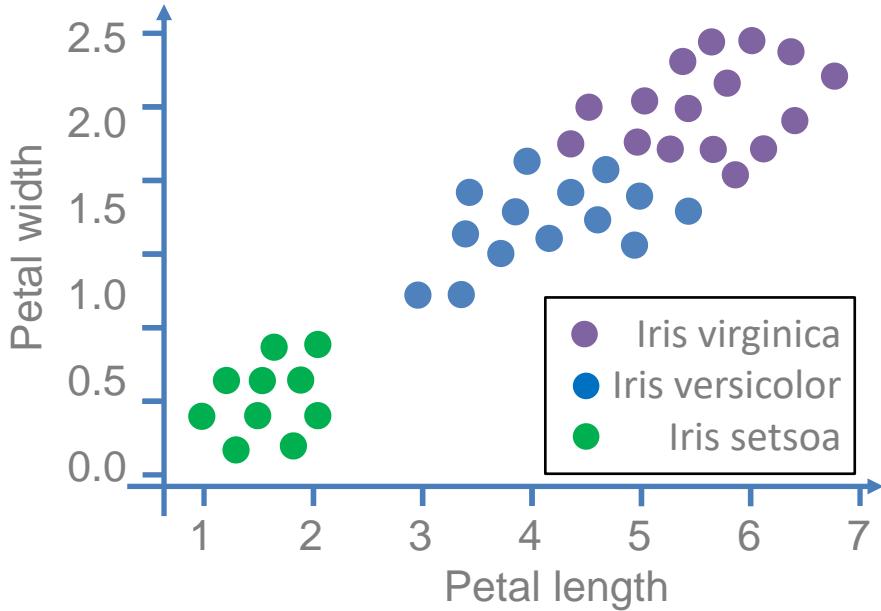


## Clustering vs. Classification

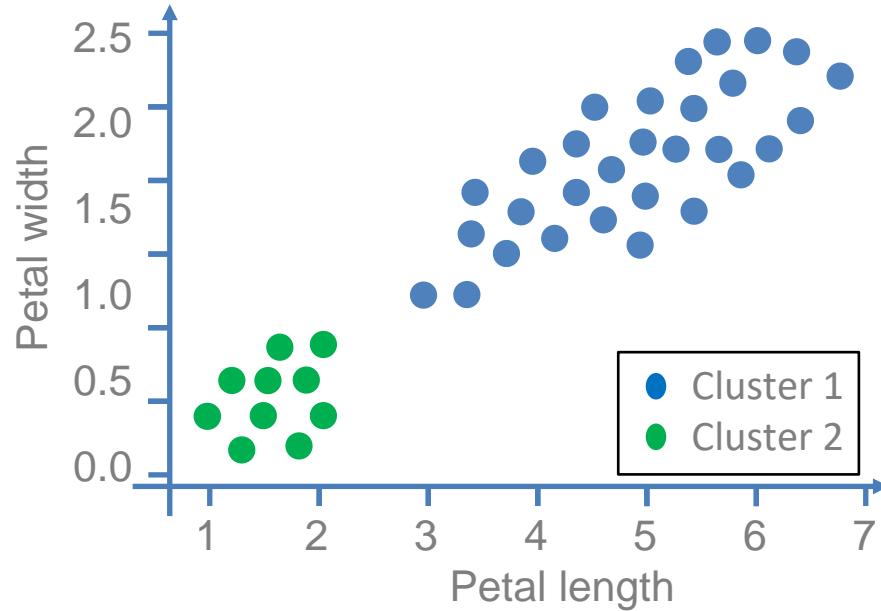


Clustering does not have labeled data and may not find all classes!

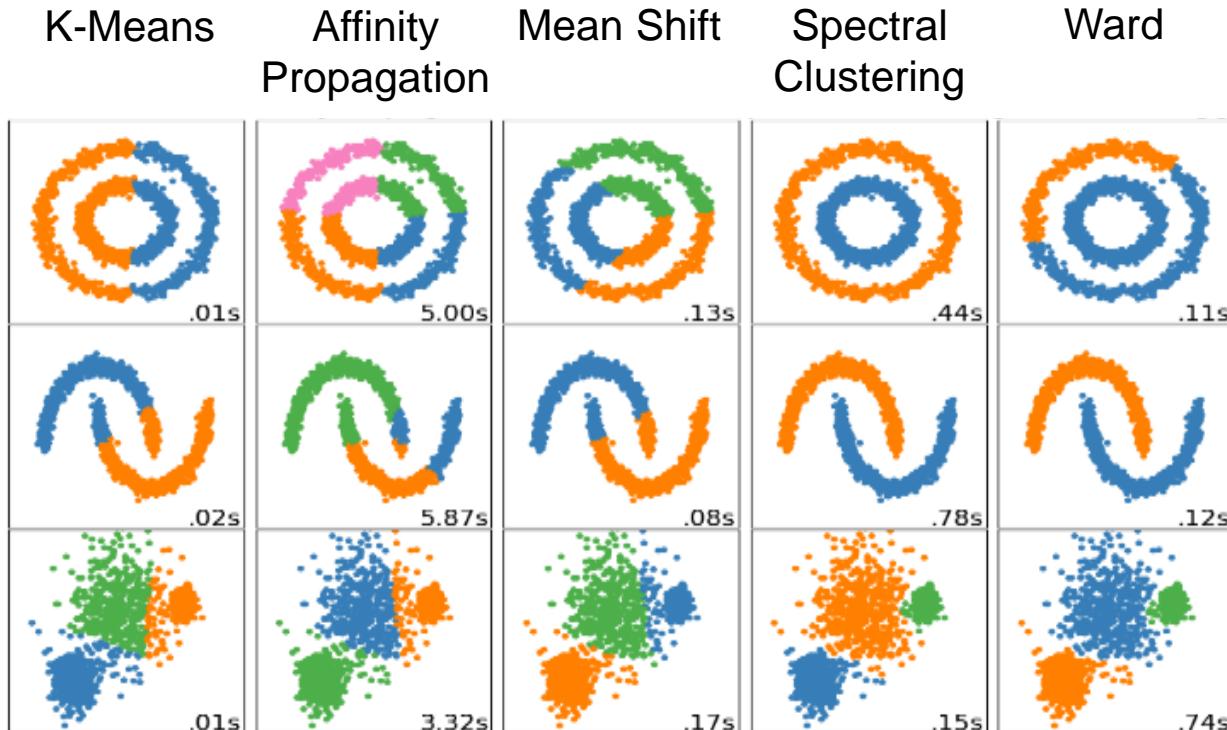
Classification



Clustering



# Clustering algorithms



**Different clustering methods produce different results**  
 e.g. some algorithms “find” more clusters than others

**Example:**  
 K-Means performs “well” on the third dataset, but not on dataset one and two

**Reason:**  
 K-Means can only identify “circular” clusters

# Curse of Dimensionality

“As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.” – Charles Isbell

The intuition in lower dimensions does not hold in higher dimensions:

- Almost all samples are close to at least one boundary
- Distances (e.g. euclidean) between all samples are similar
- Features might be wrongly correlated with outputs
- Finding decision boundaries becomes more complex

→ **Problems become much more difficult to solve!**

Source: Chen L. (2009) Curse of Dimensionality. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA.  
[https://doi.org/10.1007/978-0-387-39940-9\\_133](https://doi.org/10.1007/978-0-387-39940-9_133)

## Example: Curse of Dimensionality

The production system has  $N$  sensors attached with either the input set to “On” or “Off”

**Question:** How many samples do we need, to have **all possible sensor states** in the dataset?

## Example: Curse of Dimensionality

The production system has  $N$  sensors attached with either the input set to “On” or “Off”



**Question:** How many samples do we need, to have **all possible sensor states** in the dataset?

$$N = 1 \quad : |D| = 2^1 = 2$$

$$N = 10 \quad : |D| = 2^{10} = 1024$$

$$N = 100 \quad : |D| = 2^{100} = 1.2 \times 10^{30}$$

For  $N = 100$ , the **number of points** are even more than the number of atoms in the universe!

# Dimensionality Reduction



## The goal:

Transform the samples from a high to a lower-dimensional representation!

## Ideally:

Find a representation, which solves your problem!

## Typical Approaches:

- Feature Selection
- Feature Extraction



	S0	S1	S2	S3	S4	S5	S6	S7	S8
Sample0	0.2	0.1	11.1	2.2	Off	7	1.1	0	1.e-1
Sample1	1.2	-0.1	3.1	-0.1	On	9	2.3	-1	1.e-4
Sample2	2.7	1.1	0.1	0.1	Off	10	4.5	-1	1.e-9
Sample3	3.1	0.1	1.1	0.2	Off	1	6.6	-1	1.e-1
						⋮			

	T0	T1	T2	T3
Sample0	11.3	0.1	-1	7.8
Sample1	4.3	-0.1	1	6.8
Sample2	2.8	1.1	-1	7.1
Sample3	4.2	0.1	1	6.9
			⋮	

# Dimensionality Reduction

**The goal:**

Transform the samples from a high to a lower dimensional representation!

**Ideally:**

Find a representation, which solves your problem!

Typical Approaches:

- **Feature Selection**
- Feature Extraction



	S0	S1	S2	S3	S4	S5	S6	S7	S8
Sample0	0.2	0.1	11.1	2.2	Off	7	1.1	0	1.e-1
Sample1	1.2	-0.1	3.1	-0.1	On	9	2.3	-1	1.e-4
Sample2	2.7	1.1	0.1	0.1	Off	10	4.5	-1	1.e-9
Sample3	3.1	0.1	1.1	0.2	Off	1	6.6	-1	1.e-1

Identical

	T0	T1	T2	T3
Sample0	11.3	0.1	-1	7.8
Sample1	4.3	-0.1	1	6.8
Sample2	2.8	1.1	-1	7.1
Sample3	4.2	0.1	1	6.9

⋮

⋮

# Dimensionality Reduction

**The goal:**

Transform the samples from a high to a lower dimensional representation!

**Ideally:**

Find a representation, which solves your problem!

Typical Approaches:

- Feature Selection
- **Feature Extraction**

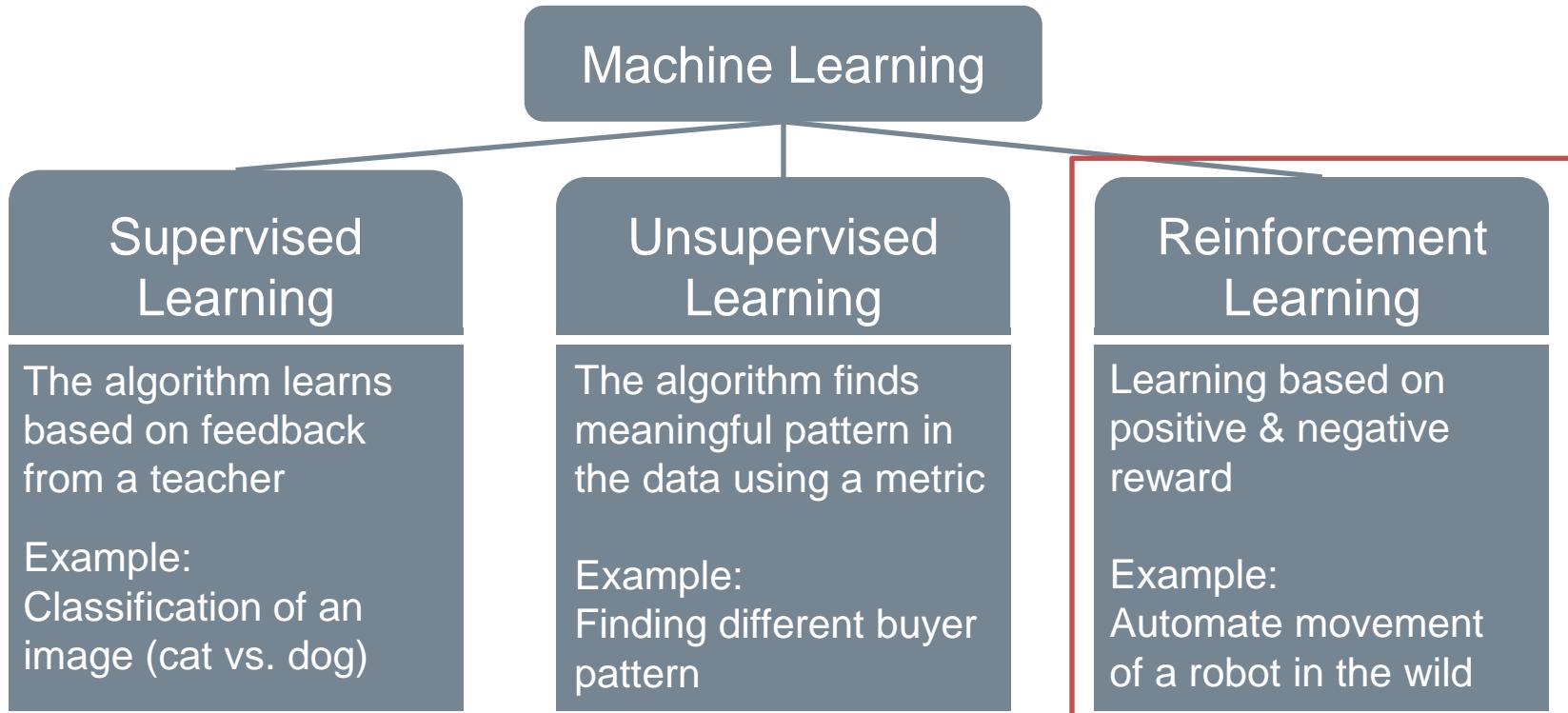


	S0	S1	S2	S3	S4	S5	S6	S7	S8
Sample0	0.2	0.1	11.1	2.2	Off	7	1.1	0	1.e-1
Sample1	1.2	-0.1	3.1	-0.1	On	9	2.3	-1	1.e-4
Sample2	2.7	1.1	0.1	0.1	Off	10	4.5	-1	1.e-9
Sample3	3.1	0.1	1.1	0.2	Off	1	6.6	-1	1.e-1
⋮									

Applied a function f

	T0	T1	T2	T3
Sample0	11.3	0.1	-1	7.8
Sample1	4.3	-0.1	1	6.8
Sample2	2.8	1.1	-1	7.1
Sample3	4.2	0.1	1	6.9
⋮				

# Machine Learning Categories



Source: Business Intelligence and its relationship with the Big Data, Data Analytics and Data Science, Armando Arroyo, 2017

# Reinforcement Learning

Reinforcement learning observes some example Input (Features) – No Labels! - and finds the **optimal action** i.e maximizes its future reward

Key Aspects:

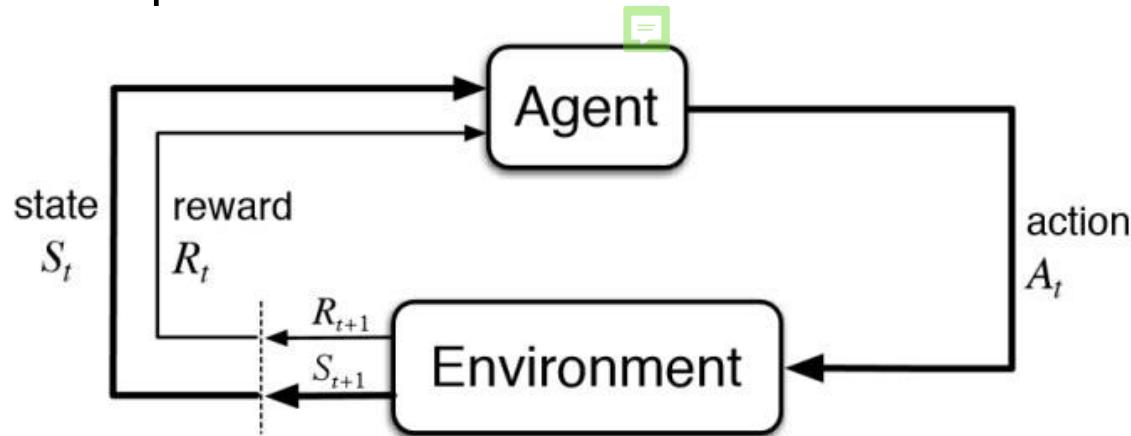
- Learning is **implicit**
- Learning using **indirect feedback**  
based on  trials and reward signals
- Actions are **affecting future measurements (i.e. inputs)**

Resolves **control** and **decision** problems  
i.e. controlling agents in games or robots

# Reinforcement Learning

**Goal:** Agents should take actions in an environment which maximize the cumulative reward.

To achieve this RL uses **reward and punishment** signals based on the previous actions to optimize the model.



Source: Reinforcement Learning: An Introduction, Richard S. Sutton, Andrew G. Barto, 2018

# Reinforcement Learning vs. (Un)supervised Learning

(Un)supervised Learning	Reinforcement Learning
<ul style="list-style-type: none"><li>The feedback is given by a <b>supervisor</b> or a <b>metric</b></li></ul>	<ul style="list-style-type: none"><li>The feedback is given by a <b>reward signal</b></li></ul>
<ul style="list-style-type: none"><li>Feedback is <b>instantaneous</b></li></ul>	<ul style="list-style-type: none"><li>Feedback can be <b>delayed</b> (credit assignment problem)</li></ul>
<ul style="list-style-type: none"><li>Learning by using <b>static data</b> (no re-recording of data necessary)</li></ul>	<ul style="list-style-type: none"><li>Training is <b>based on trials</b> i.e. interaction between environment and agent (re-recording necessary)</li></ul>
<ul style="list-style-type: none"><li>Prediction does <b>not affect future measurements</b> – The data is assumed Independent Identically Distributed (i.i.d)</li></ul>	<ul style="list-style-type: none"><li>The prediction (actions) <b>affect future measurements</b> i.e. the measurements are no necessarily i.i.d!</li></ul>

Source: Reinforcement Learning: An Introduction, Richard S. Sutton, Andrew G. Barto, 2018

# Example: Flexible handling of components

Google designed an experiment to handle objects using a robot

After training the RL algorithm: The robots **succeeded in 96% of the grasp attempts** across 700 trial grasps on previously unseen objects



## Training and data acquisition:

- 7 industrial robots
- 10 GPU's
- Numerous CPU's
- 580,000 gripping attempts

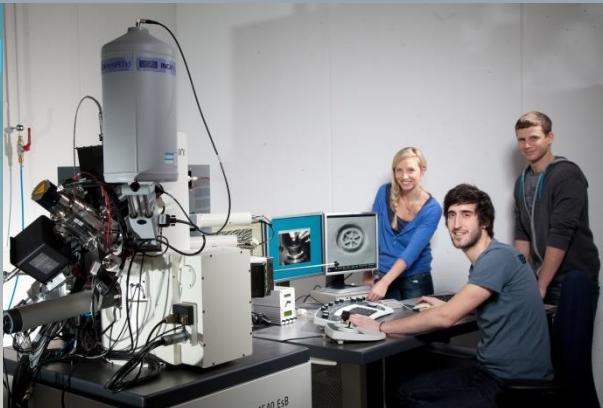
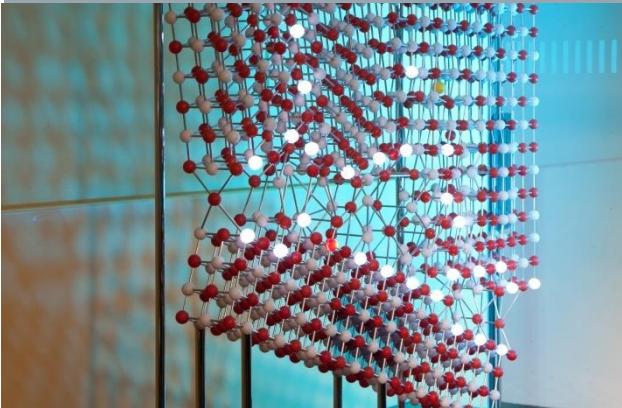


Source: <https://ai.googleblog.com/2018/06/scalable-deep-reinforcement-learning.html>



# Introduction to Machine Learning

## Machine Learning Pipeline



Bilder: TF / Malter

# The Machine Learning Pipeline

A concept that provides **guidance** in a machine learning project

- Step-by-step process
- Each step has a **goal** and a **method**

There exist multiple pipelines and concepts,  
however the idea behind all pipelines and steps are the same!

The 8-step machine learning pipeline by Artasanchez & Joshi :



Source: Artificial Intelligence with Python - Second Edition, by Alberto Artasanchez, Prateek Joshi

# 1. Problem Definition

Our aim with machine learning is to develop a solution for a problem. In order to develop a satisfying solution, we need to define the problem. This definition of the problem lays the foundation to our solution. It shows us what kind of data we need, what kind of algorithms we can use.

Examples:

- If we are trying to find a faulty equipment, we have a classification problem
- If we are trying to predict a continuous number, we have a regression problem

## Prediction of the energy consumption of a milling machine – problem definition

- Want to predict the necessary energy to conduct a process step
- The prediction should be made on the basis of the milling parameters
- Therefore we want to get transparency
- Further this should allow the optimization of the parameters 



## 2. Data Ingestion

- After defining the problem, we have to define which data we need to describe the problem
- The data should represent the problem and the information needed to successfully predict the target value
- If we have defined the data we need, we can gather them in databases or record them with trials

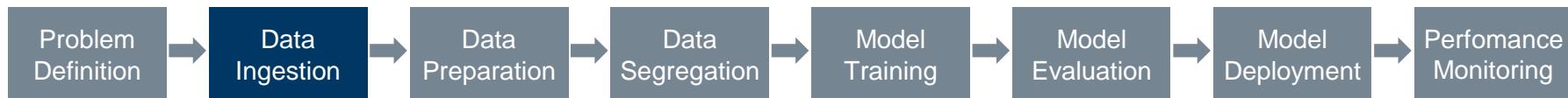
Prediction of the energy consumption of a milling machine – data ingestion

**Data:**

- Feed rate
- Speed
- Path
- Energy consumption



- Data can be collected with sensors 
- Trials can be conducted on an existing machine 



### 3. Data Preparation

We gathered the data and created our dataset. But it is highly unlikely that we can use our data as soon as we gather it. We have to prepare it for our machine learning solution. For example:

- We can get rid of instances with missing values or outliers
- We can use dimensionality reduction for higher dimensional dataset, 
- We can normalize the data to transform our data to a common scale.

**The quality of a machine learning algorithm is highly dependent on the quality of data!**

Prediction of the energy consumption of a milling machine – data preparation

In our dataset we have outliers  
and missing values



- We have enough instances with no outliers or missing values
- Therefore the easiest way to treat the instances with missing values and outliers is to delete them



# An Example Dataset and Terminology

Value of the instance  
for the attribute

Attribute

Label

Instance

Target Value

Day	Weather	Temp.	Humidity	Wind	Tennis recommended?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Mild	High	Weak	Yes
...	...	...	...	...	...

## 4. Data Segregation

Before we train a model we have to separate our data set:

- We have to separate the **target variable**
- **Training Set:** This set is used to fit our model
- **Validation Set:** Validation set is used to test our fit of the model and **tune it's hyperparameters**
- **Test Set:** Test set is used to evaluate the performance of the **final version** of our model

### Prediction of the energy consumption of a milling machine – data segregation

Attributes/ independent variable: Target variable:

- |             |                      |
|-------------|----------------------|
| • Feed rate | • Energy consumption |
| • Speed     |                      |
| • Path      |                      |

- 
- We use an training/test split of 80%/20%
  - Because we do not perform hyperparameter optimization we need no validation set



## 5. Model Training

Next step is to select an algorithm to train our data. So the question is which algorithm to use:

- Labeled data → supervised learning algorithms
  - Prediction of a discrete value → regression algorithm
  - Prediction of a class → classification algorithm
- There are numerous algorithms with their strengths and weaknesses
- To find out which algorithm is the best for our data set we have to test them



Prediction of the energy consumption of a milling machine – model training

We have labeled data  
→ **Supervised problem**

We have a discrete value  
Energy consumption (kWh)  
→ **Regression problem**

We use the following regression algorithms  
→ **Linear regression**  
→ **Random forest regression**  
→ **Support vector regression**



## 6. Model Evaluation

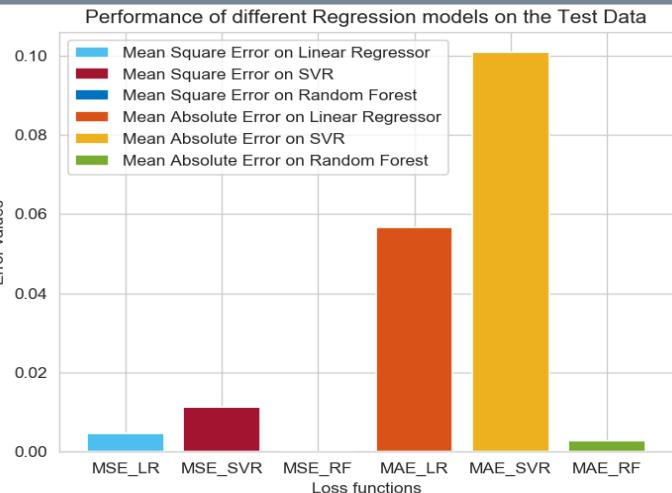
Training and evaluation of the model are iterative processes:

1. First, we train our model with the training set
2. Then evaluate its performance with validation set with evaluation metrics
3. Based on this information, we tune our algorithm's hyperparameters

This iterative process continues unless we decide that we can't improve our algorithm anymore

Then we use the test set to see the performance on unknown data

### Prediction of the energy consumption of a milling machine – model evaluation



Random Forest Regressor achieves the best results  
 → We choose this model



# Classification - Confusion Matrix and Accuracy

**Confusion Matrix** gives us a matrix as output and describes the performance of the model.

There are 4 important terms:

**True Positives**: The cases in which we predicted YES, and the actual output was also YES

**True Negatives**: The cases in which we predicted NO, and the actual output was NO

**False Positives**: The cases in which we predicted YES, and the actual output was NO

**False Negatives**: The cases in which we predicted NO, and the actual output was YES

		Predicted: NO	Predicted: YES
n=165	Actual: NO	50	10
	Actual: YES	5	100

Confusion Matrix

**Classification Accuracy** is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$\text{Accuracy} = (150/165) = 0.909$$

# Regression – Error Metrics

## Mean Absolute Error (MAE):

- Average difference between the original and predicted values
- Measure how far predictions were from the actual output
- Does not give and idea about the direction of error.

$$\text{Mean Absolute Error} = \frac{1}{N} \sum_{j=1}^N |y_j - \hat{y}_j|$$

## Mean Squared Error (MSE):

- Similar to MAE
- It takes the average of the square of the difference between original and predicted value
- Larger errors become more pronounced, so that the model can focus on larger errors.

$$\text{Mean Squared Error} = \frac{1}{N} \sum_{j=1}^N (y_j - \hat{y}_j)^2$$

## 7. Model Deployment

We have trained our model, evaluated it with evaluation metrics and tuned the hyperparameters with the help of our validation set. After finalization of our model, we evaluated it with test set. Finally we are ready to deploy our model to our problem. Here we have often to consider the following issues:

- Real time requirements
- Robust hardware (sensors and processor)

prediction of the energy consumption of a milling machine – model deployment

We are using the model to predict the energy consumption based on production parameters

Axis=2  
Feed=800  
Path=60

Predicted energy consumption = 0.046



## 8. Performance Monitoring

We must monitor the performance of our model and make sure it still produces satisfactory results. Unsatisfactory results might be caused by different reasons. For example,

- Sensors can malfunction and provide wrong data
- The data can be out of the trained range of the model

Monitoring the performance **can mean** the monitoring of the input data regarding any changes as well as the comparison of the prediction and the actual value after defined time periods

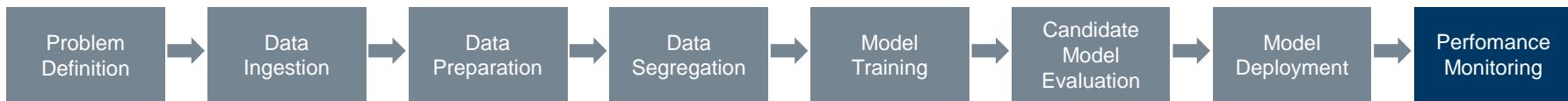
prediction of the energy consumption of a milling machine – performance monitoring

We measure the actual energy consumption of 10 work steps after each maintenance procedure

We calculate the error of the predictions of the energy consumption

**Within defined range:** model is working sufficiently

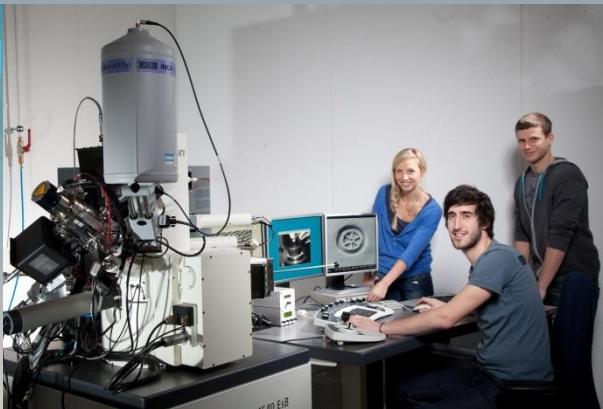
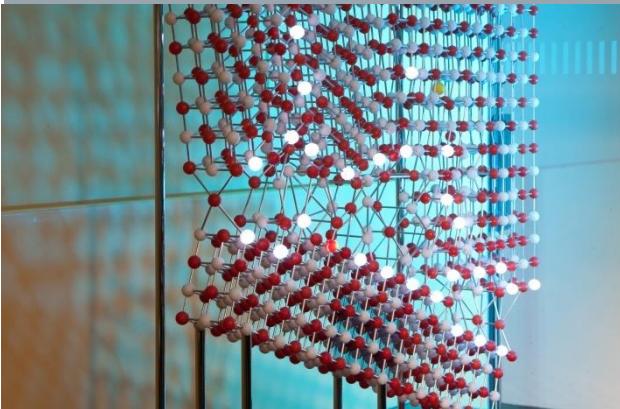
**Out defined range:** model is working insufficient  
→ stop using and root cause analysis





# Introduction to Machine Learning

## Summary



Bilder: TF / Malter

# Summary

## In this chapter we talked about:

- The history of machine learning and recent trends
- The different types of machine learning types such as supervised learning, unsupervised learning and reinforcement learning
- The steps involved in a common machine learning pipeline

## After studying this chapter, you should be able to:

- Identify the type of machine learning needed to solve a given problem
- Understand the differences of regression and classification tasks
- Setup a generic project pipeline following containing all relevant steps from problem definition to performance monitoring





# 1. Introduction to Machine Learning

◀ Summary

## Questions

What are the three popular types of machine learning?

1.
2.
3.



Also correct are:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- supervised
- unsupervised
- reinforcement



Correct!

Submit

What are the most important traits that distinct supervised and unsupervised learning from each other?



- The libraries used to solve the machine learning problem
- Existence of labeled data
- Both have different kind of learning (implicit vs. explicit)
- The algorithms to solve the machine learning problem



Correct!

Please choose the statement(s) that explain regression?

- A statistical method where we fit a line to segregate data into different groups.
- A supervised machine learning task, where we predict discrete values based on historical data
- A statistical method to reduce the dimensions of a dataset
- A programming technique to predict values



**Correct!**

Please choose the statement(s) that explain classification?

- A machine learning task where we decide which data to use in machine learning pipeline
- A machine learning task where we cluster similar data points together
- A machine learning task where we classify data into different classes based on labeled data
- A machine learning technique under supervised learning
- A specific algorithm that differentiates between spam and non-spam emails



**Correct!**

Please choose the statement(s) that explain clustering?

- A machine learning task where we cluster data into similar groups without using labeled data
- Clustering different machine learning algorithms together to improve their results
- Gathering the data for the machine learning task
- A machine learning task where we cluster similar features together, to reduce the dimensions of the dataset.
- An unsupervised machine learning technique



Correct!

Submit

Please choose the statement(s) that explain dimensionality reduction?

- Subfield of geometry
- A machine learning task where we reduce dimensions of high-dimensional data
- A mathematical technique that allows us to avoid curse of dimensionality
- Creating 2D visualization from 3D surfaces to improve processing



Correct!

Submit

Please choose the statement(s) that explain reinforcement learning?

- A decryption algorithm
- Where an agent learns to maximize its cumulative reward
- A combination of supervised and unsupervised learning to reinforce their results
-

A technique to increase the computation power of computer, so that time required to train a machine learning algorithm is reduced.



**Correct!**

Submit

Please choose the statement(s) that explain Machine Learning Pipeline?

- A machine learning software with a GUI.
- A software that conducts the steps of machine learning solution
- An automated solution for machine learning
- A concept that proves guidance during a project



**Correct!**

Submit

What is data segregation in the scope of machine learning and how many parts do we split data into?

- Segregation of data into two groups: useful or useless
- Splitting data into three groups called train, validation and test set
- Clustering similar data points together to see their similarities
- Classifying data into respective groups that we know of



**Correct!**

Submit

Please order the evaluation metrics below based on regression or classification.

[Reset Ordering](#)

Regression

Classification

Mean Squared Error

Confusion Matrix

Classification Accuracy

Mean Absolute Error



Correct!

[Submit](#)

Below, you can see the steps of a classical machine learning pipeline. Please, order these steps as you have learned in lecture start ordering beginning with the first step in the top.

Problem Definition

Data Ingestion

Data Preparation

Data Segregation

Model Training

Candidate Model Evaluation

Model Deployment

Performance Monitoring



Correct!

Submit

◀ Summary

Add Comment

Sort Ascending



Lohr, Tim [il34ifyn] - 09. May 2022

Dear [aw00ehor], thanks for your notification about question 1. This indeed makes sense. I changed it so the correct answer should be correct for both lower and upper case and also for single words, where the word "learning" is not required.

aw

[aw00ehor] - 09. May 2022

In the first question I typed in the correct answers but with small and not with capital letters in the beginning of the words. That was seen as a mistake. It would be nice if both solutions are accepted.



Lohr, Tim [il34ifyn] - 04. May 2022

Dear Patrick, thanks for your remark. This is indeed true. I changed the question accordingly!



Weber, Patrick [sa77dolu] - 03. May 2022

The wording of the question "What is the most important difference between supervised and unsupervised learning?" implies that only one answer is expected. I would like it to be clearer.



Lohr, Tim [il34ifyn] - 02. May 2022

Thank you [ly34tici], I just fixed this issue.

**ly**

[ly34tici] - 28. Apr 2022

In the last task, the bullet points are messed up



Please choose the statement(s) that explain regression?

A következő válasz „A supervised machine learning task, where we predict discrete values based on historical data” azért nem jó, mert

Regression: The output are continuous or real values

Classification: The output variable must be a discrete value (class)

Discrete data contains distinct or separate values.

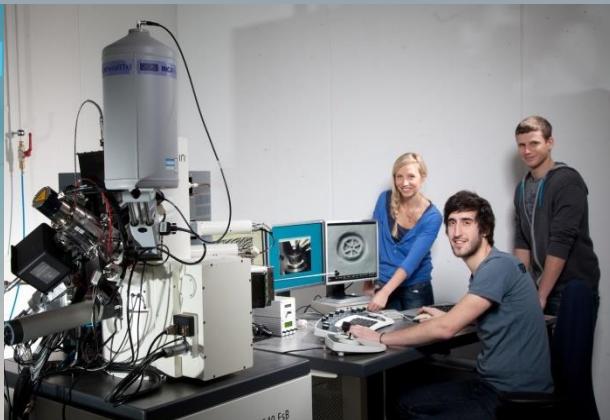
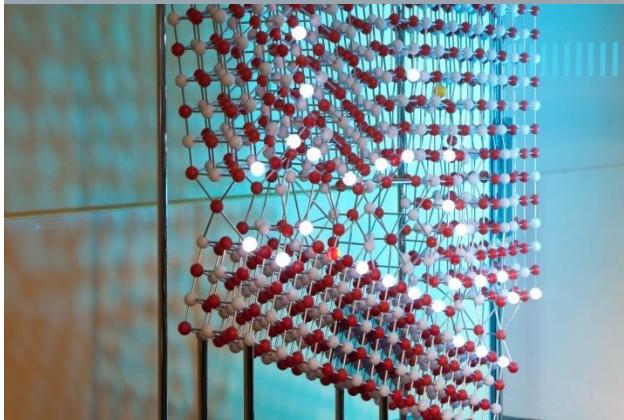
Continuous data includes any value within the preferred range.

Please choose the statement(s) that explain classification?

nem jó válasz, de érdkes: A specific algorithm that differentiates between spam and non-spam emails: classification

# Machine Learning for Engineers

## Linear Regression - Motivation



Bilder: TF / Malter

# Motivation

Linear regression is used in multiple scenarios each and every day!

## Use Cases:

- Trend Analysis in financial markets, sales and business
- Computer Vision problems i.e. registration & localization problems
- etc.



Source: <https://www.tradingview.com/script/OZZpxf0m-Linear-Regression-Trend-Channel-with-Entries-Alerts/>

# When do we use it?

The problem has to be simple:

- Dataset is small
- Linear model is enough i.e. Trend Analysis
- Linear models are the basis for complex models  
i.e. Deep Networks

→ Let's have a look at regression  
i.e. **prediction of a real-valued output**

# Example: Growth Reference Dataset (WHO)

Age	Height
4.1 years	108 cm
5.2 years	112 cm
5.6 years	108 cm
5.7 years	116 cm
6.2 years	112 cm
6.3 years	116 cm
6.6 years	120 cm
6.7 years	122 cm
⋮	

The dataset contains Age (5y – 12y) and Height information from people in the USA

# Example: Growth Reference Dataset (WHO)

Age	Height
4.1 years	108 cm
5.2 years	112 cm
5.6 years	108 cm
5.7 years	116 cm
6.2 years	112 cm
6.3 years	116 cm
6.6 years	120 cm
6.7 years	122 cm
⋮	

The dataset contains Age (5y – 12y) and Height information from people in the USA

We want to answer Questions like:

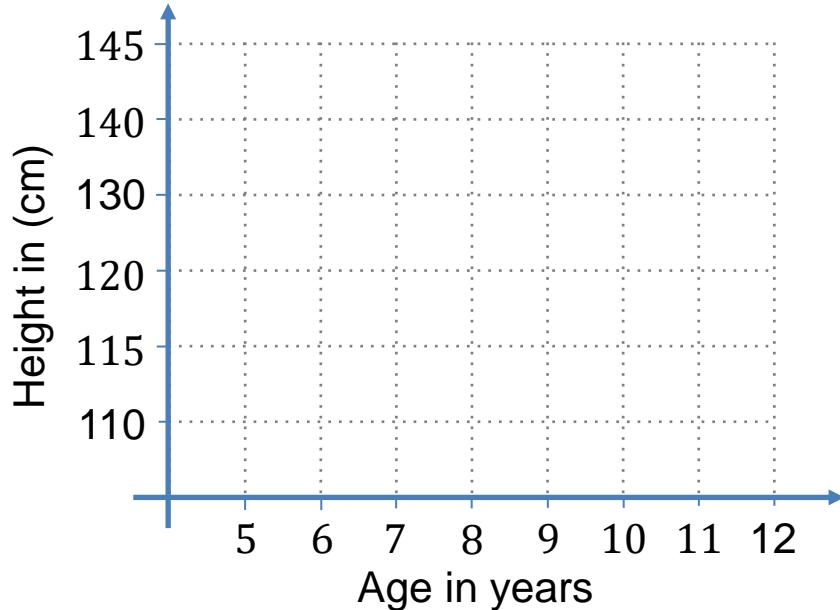
What is the height of my child,  
when it is **14 years old**?

What is the height of my child,  
when it is **30 years old**?

# Example: 1. Visualize the data

Age	Height
4.1 years	108 cm
5.2 years	112 cm
5.6 years	108 cm
5.7 years	116 cm
6.2 years	112 cm
6.3 years	116 cm
6.6 years	120 cm
6.7 years	122 cm

⋮  
⋮  
⋮

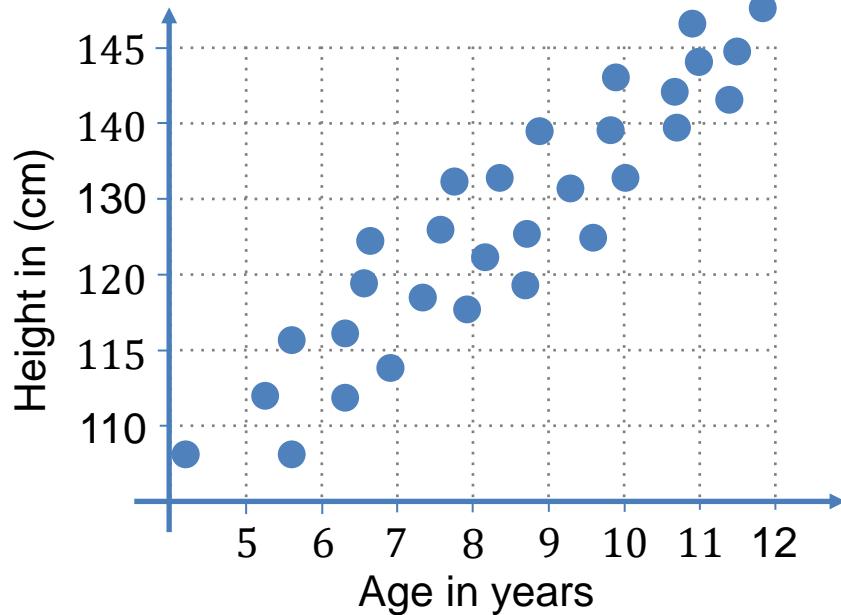


Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

# Example: 1. Visualize the data

Age	Height
4.1 years	108 cm
5.2 years	112 cm
5.6 years	108 cm
5.7 years	116 cm
6.2 years	112 cm
6.3 years	116 cm
6.6 years	120 cm
6.7 years	122 cm

⋮  
⋮  
⋮



Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

## Example: 2. Fit a model



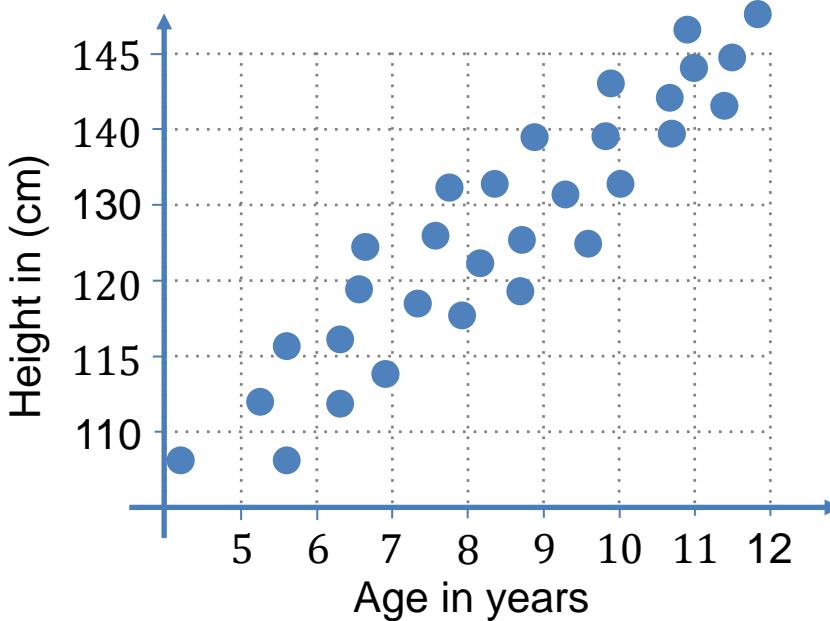
Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

## Example: 2. Fit a model

What “model” i.e. function describes our data?

- Linear model
- Polynomial model
- Gaussian model

⋮



Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

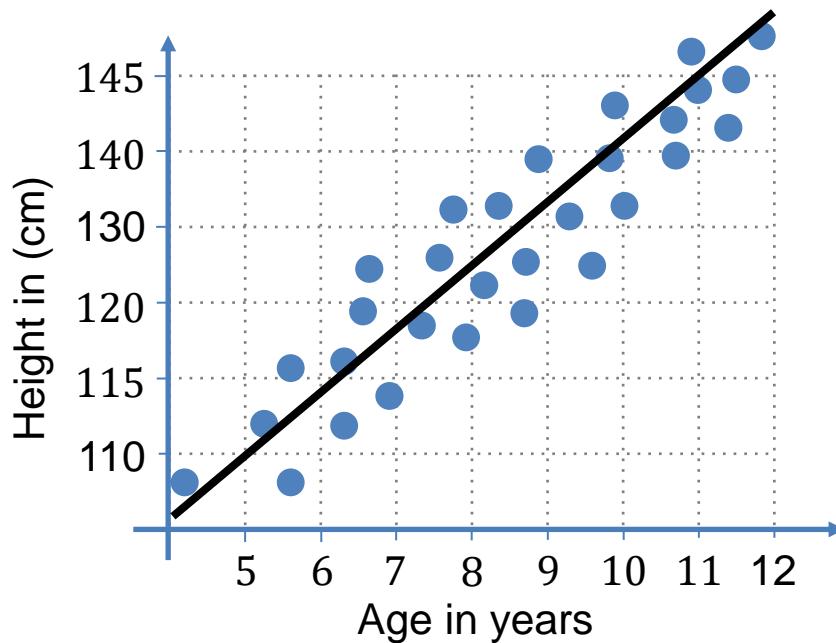
## Example: 2. Fit a model

What “model” i.e. function describes our data?

- Linear model
- Polynomial model
- Gaussian model

⋮

Answer: Linear Model!

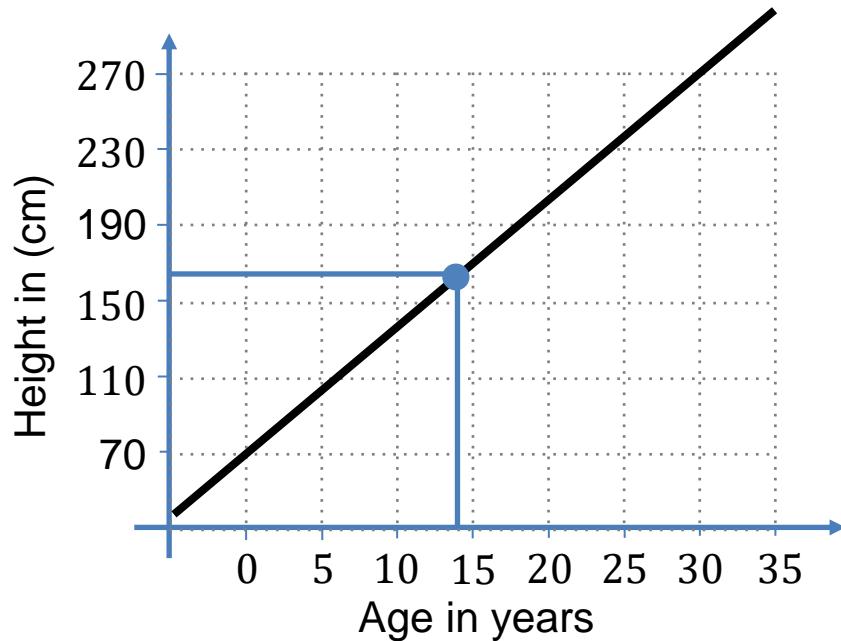


Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

## Example: 3. Answer your Questions

What is the height of my child,  
when it is **14 years old**?

→ 165 cm



Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

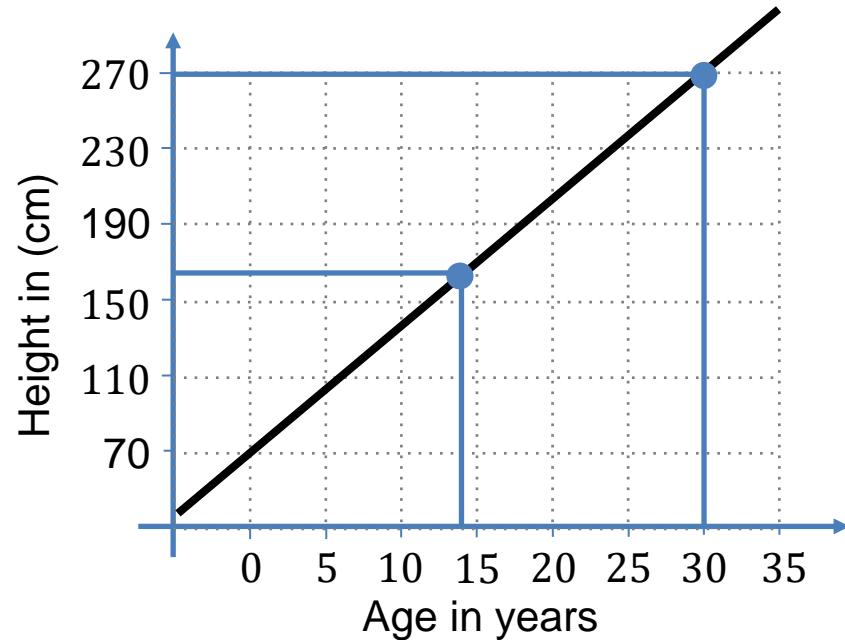
## Example: 3. Answer your Questions

What is the height of my child,  
when it is **14 years old**?

→ 165 cm

What is the height of my child,  
when it is **30 years old**?

→ 270 cm



Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

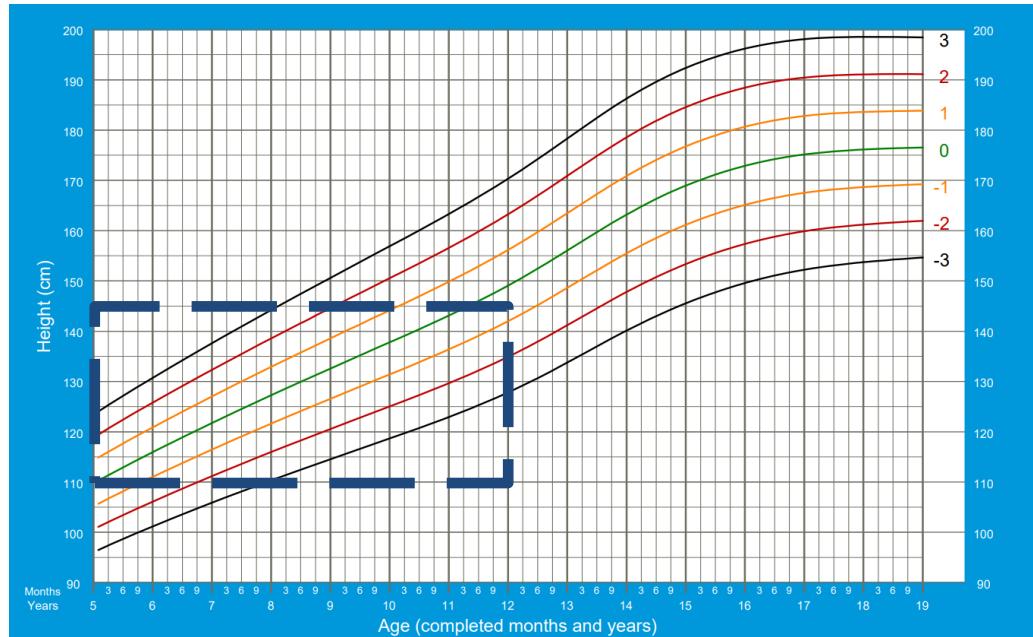
# Example: Actual Answer

What is the height of my child,  
when it is **14 years old**?  
**(We estimated 165 cm)**

→ ~165 cm 

What is the height of my child,  
when it is **30 years old**?  
**(We estimated 270 cm)**

→ ~175 cm 



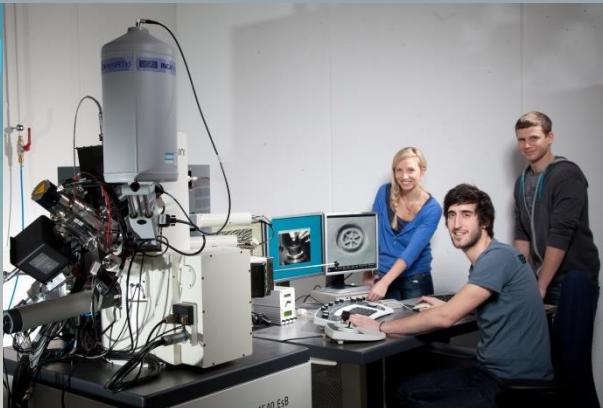
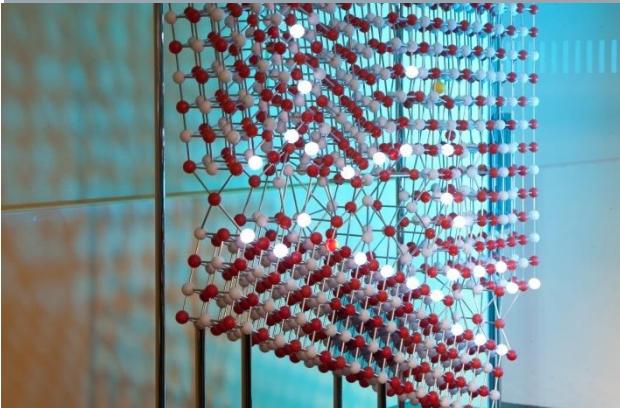
Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

## Next in this Lecture:

- What is the Mathematical Framework?
- How do we **fit** a linear model?

# Machine Learning for Engineers

## Linear Regression - Mathematics



Bilder: TF / Malter

# Overview

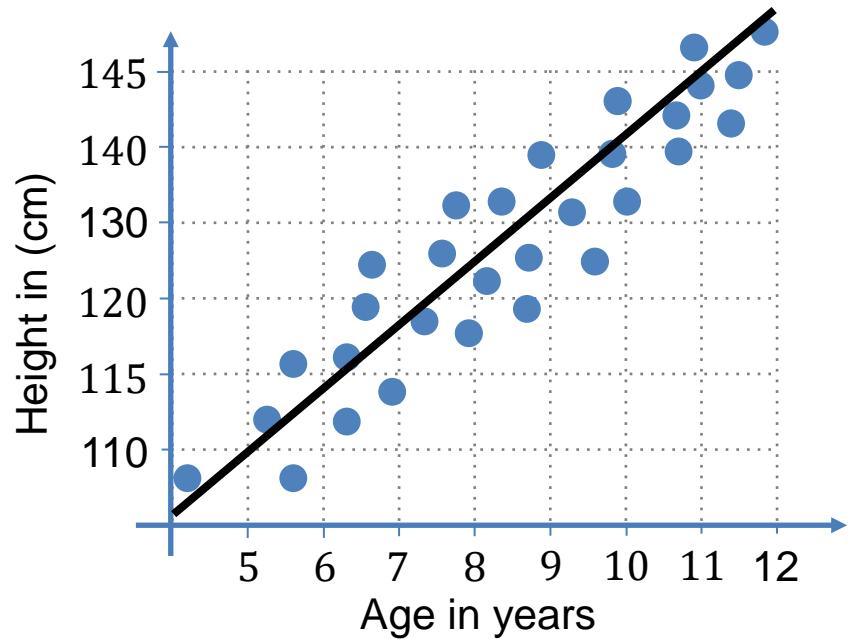
- Overall Picture
- The Linear Model
- Optimization
- Basis Function Expansion

# Overview

- **Overall Picture**
- The Linear Model
- Optimization
- Basis Function Expansion

# Overall Picture

Linear Regression is a method to fit **linear models** to our data!



Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Note: **x** (input) is called the **independent/predictor variable** and **y** (output) is called **dependent/outcome variable**

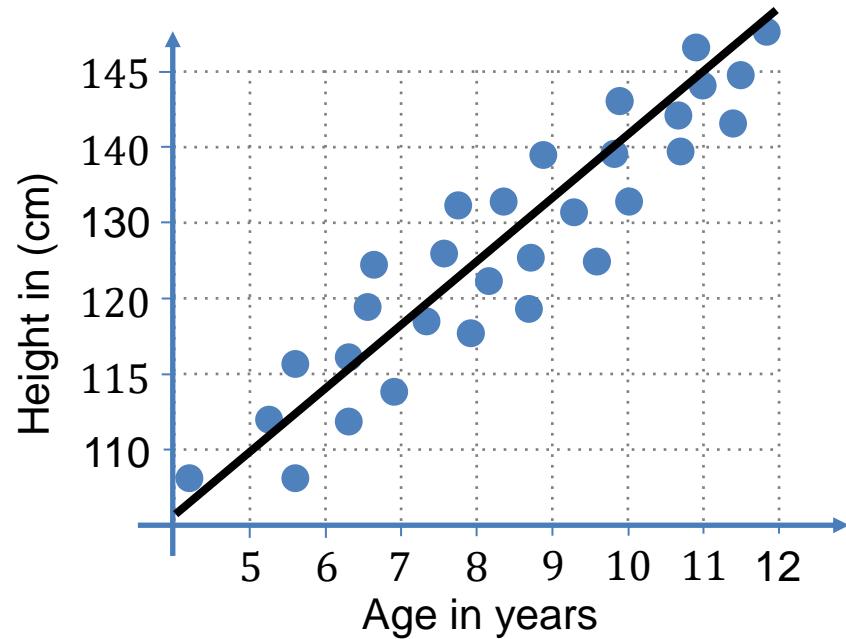
Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

# Overall Picture

Linear Regression is a method to fit **linear models** to our data!

The linear model:

$$f(\mathbf{x}) = w_0 \cdot x_0 + w_1 \cdot x_1 = \mathbf{y}$$



Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Note: **x** (input) is called the **independent/predictor variable** and **y** (output) is called **dependent/outcome variable**

Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

# Overall Picture

Linear Regression is a method to fit **linear models** to our data!

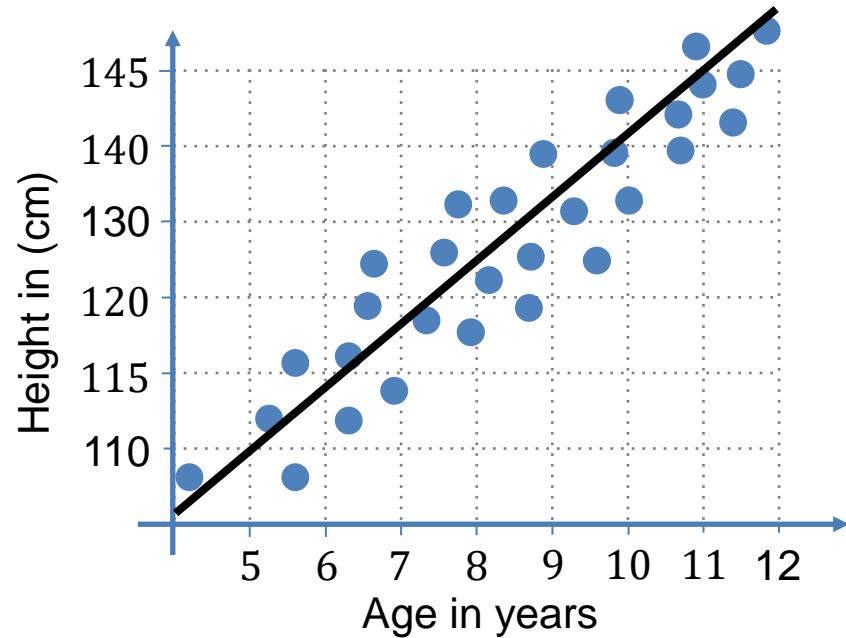
The linear model:

$$f(\mathbf{x}) = w_0 \cdot x_0 + w_1 \cdot x_1 = \mathbf{y}$$

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \text{Age in Years} \end{pmatrix}$$

$\mathbf{y}$  = Height

$\mathbf{w}$  = Weights



Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Note: **x** (input) is called the **independent/predictor variable** and **y** (output) is called **dependent/outcome variable**

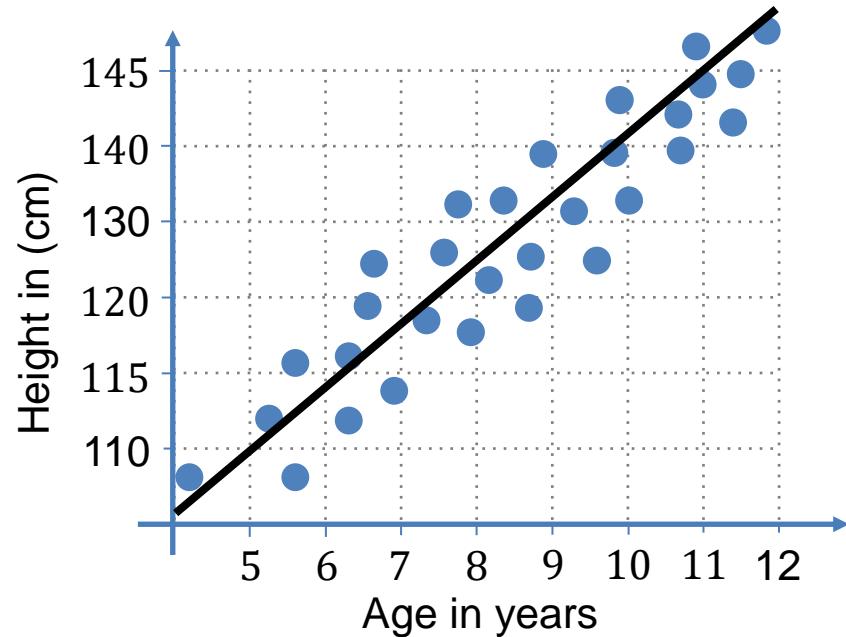
Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

# Overall Picture

Linear Regression is a method to fit **linear models** to our data!

The linear model (in our example):

$$f(\mathbf{x}) = 70 \cdot x_0 + 6.5 \cdot x_1 = \mathbf{y}$$



Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Note: **x** (input) is called the **independent/predictor variable** and **y** (output) is called **dependent/outcome variable**

Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

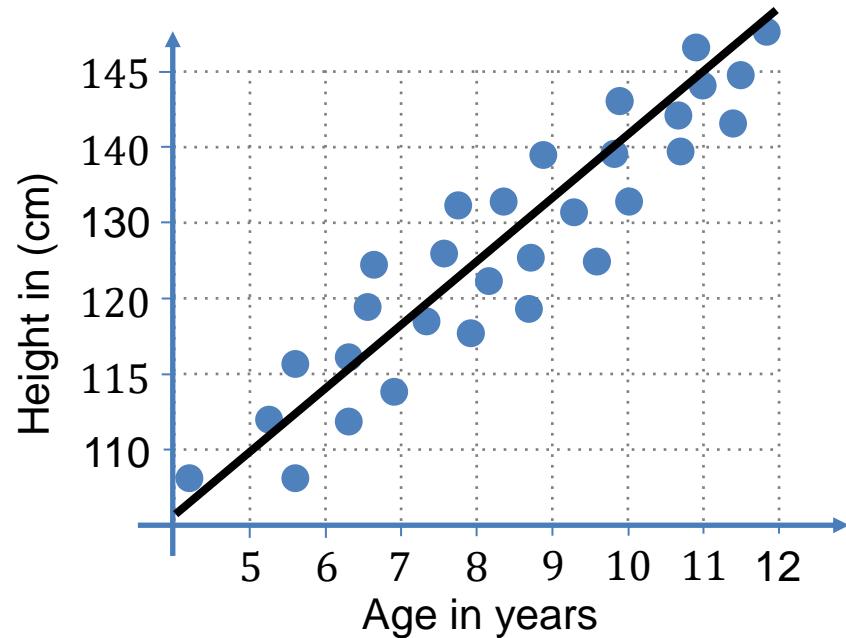
# Overall Picture

Linear Regression is a method to fit **linear models** to our data!

The linear model (in our example):

$$f(\mathbf{x}) = 70 \cdot x_0 + 6.5 \cdot x_1 = \mathbf{y}$$

→ Finding a good  $w_0$  and  $w_1$  is called **fitting**!



Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Note: **x** (input) is called the **independent/predictor variable** and **y** (output) is called **dependent/outcome variable**

Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

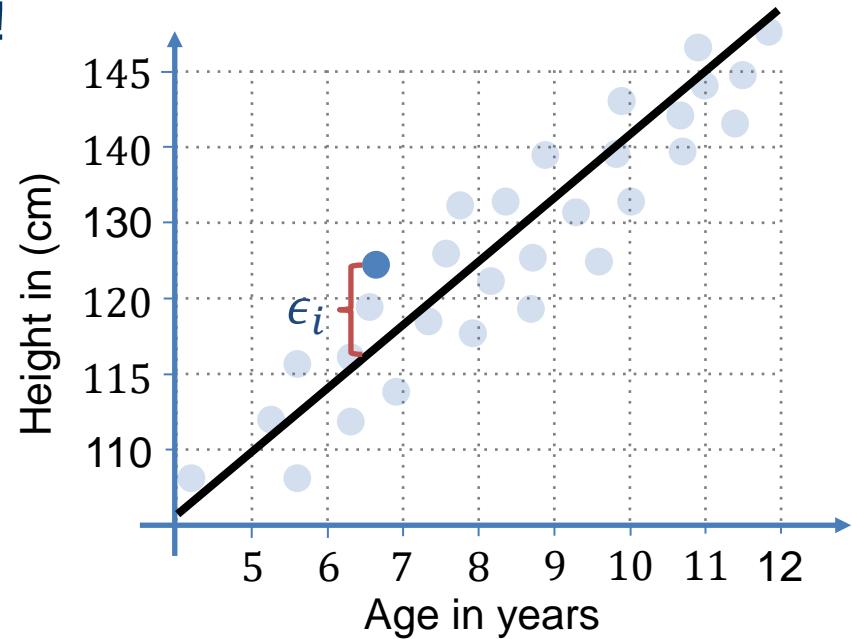
# Overview

- Overall Picture
- **The Linear Model**
- Optimization
- Basis Function Expansion

# The Linear Model

The previous description of the linear model is not accurate!

**Reason:** Real Systems produce noise!



Note: In this case we assume that the noise  $\epsilon_i$  is **Gaussian distributed**

Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

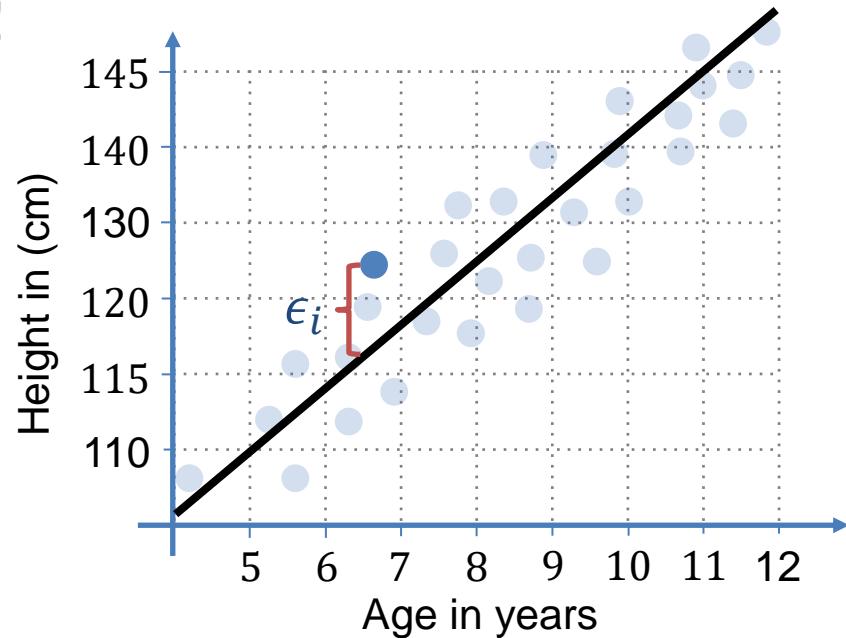
# The Linear Model

The previous description of the linear model is not accurate!

**Reason:** Real Systems produce noise!

Linear model with noise:

$$f(\mathbf{x}) = w_0 \cdot x_0 + w_1 \cdot x_1 + \epsilon_i$$



- 1) In this case we assume that the noise  $\epsilon_i$  is **Gaussian distributed**

Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

# The Linear Model

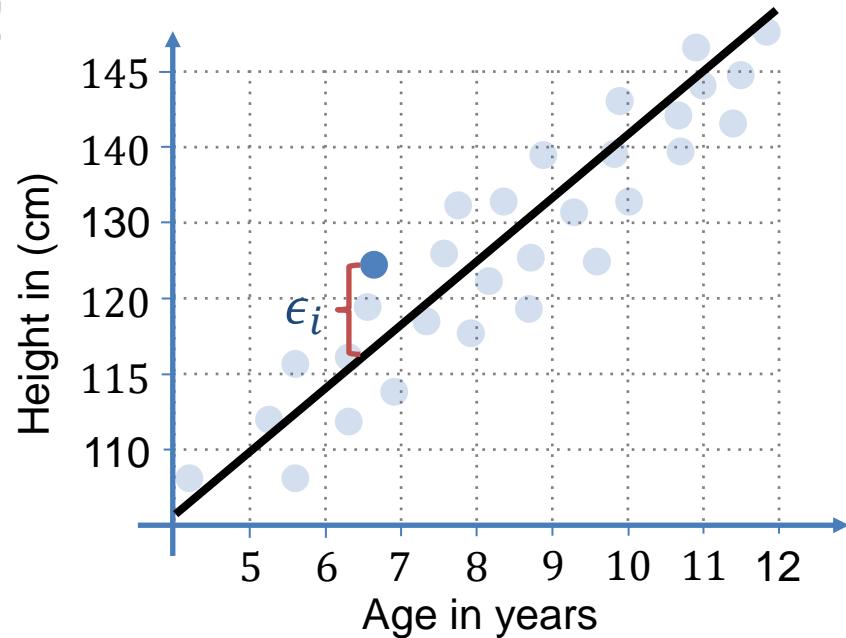
The previous description of the linear model is not accurate!

**Reason:** Real Systems produce noise!

Linear model with noise:

$$f(\mathbf{x}) = w_0 \cdot x_0 + w_1 \cdot x_1 + \epsilon_i$$

The  $\epsilon_i$ <sup>1)</sup> and the summation  $\epsilon$  of all samples is called **Residual Error**!



1) In this case we assume that the noise  $\epsilon_i$  is **Gaussian distributed**

Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

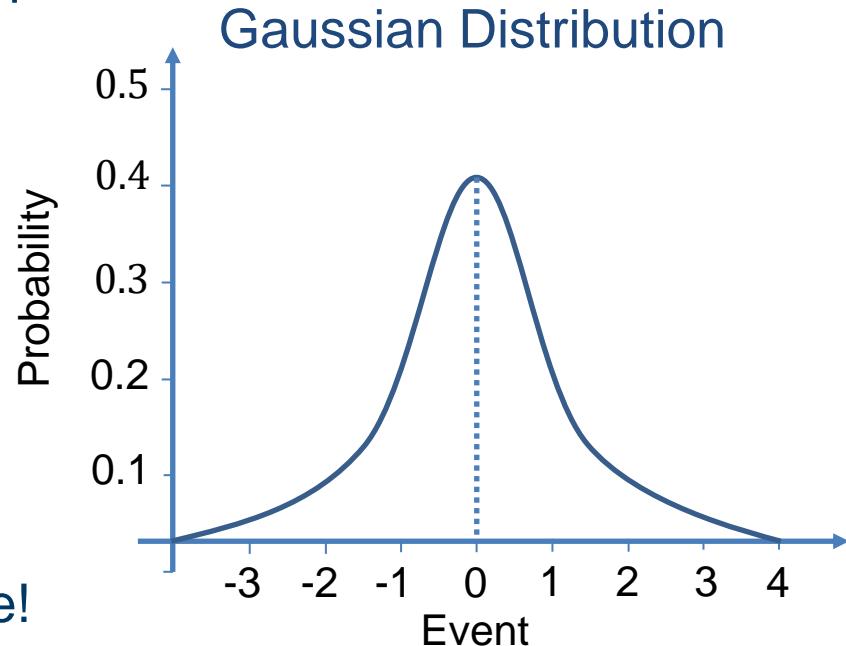
# The Residual Error $\epsilon$

The residual error  $\epsilon$  is **not** part of the input!

It is a **random variable!**

Typically: We assume  $\epsilon$  is **Gaussian Distributed** (i.e. Gaussian noise)

$$p(\epsilon) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\epsilon-\mu}{\sigma}\right)^2} \quad 1)$$



But other distributions are also possible!

1) The short form is  $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$

Note: In our use-case  $\mu = 0$  and  $\sigma$  is small. That means our samples are only "slightly" deviate

# The distribution of $y$

The noise is Gaussian:

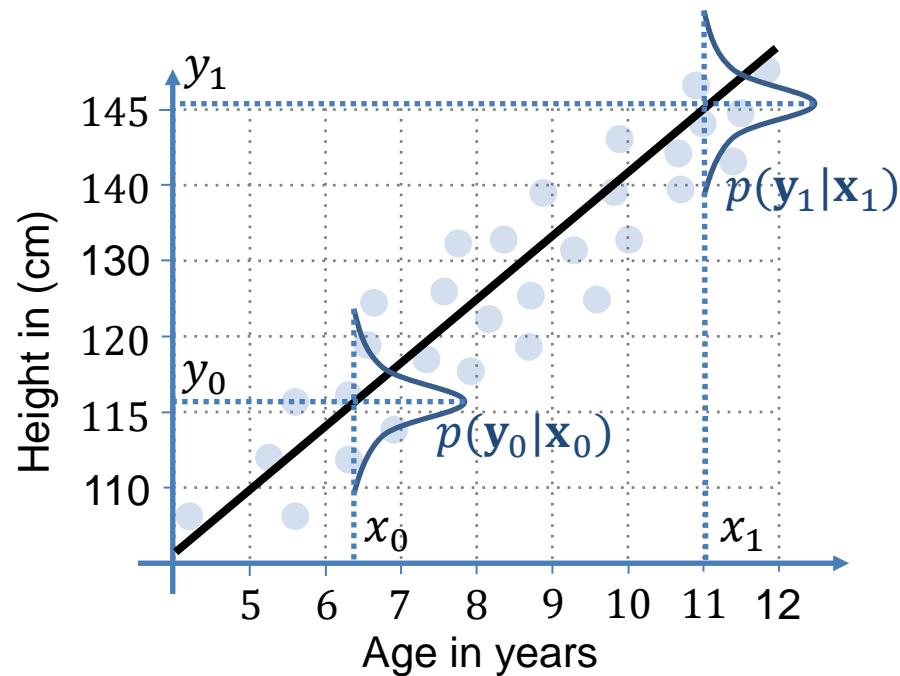
$$p(\epsilon) = \mathcal{N}(\epsilon | 0, \sigma^2)$$

“Inserting” the linear function:

$$p(y | x, w, \sigma) = \mathcal{N}(y | w^T x, \sigma^2)$$

This is the **conditional probability density** for the target variable  $y$ !

Note for later use:  $p(y | x, \theta) = p(y | x, w, \sigma)$



# General Linear Model

A more general formulation:

$$f(\mathbf{x}) = \sum_{j=1}^D w_j x_j + \epsilon = \mathbf{y}$$

Vector Notation:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \epsilon = \mathbf{y}$$

Note: **Bold** variables are vectors and/or matrices, non-bold variables are scalars!

Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

# What parameters do we have to estimate?

The model is linear:

$$f(\mathbf{x}) = \sum_{j=1}^D w_j x_j + \epsilon = \mathbf{y}$$



Note: In this case, we assume that the noise is Gaussian distributed! i.e.  $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$

Note: We assume  $\mu = 0$  : That means we only have to estimate  $\sigma$ ! Homework: Why can we assume  $\mu = 0$ ?

# What parameters do we have to estimate?

The model is linear:

$$f(\mathbf{x}) = \sum_{j=1}^D w_j x_j + \epsilon = \mathbf{y}$$

The **model parameters** are:

  $\theta = \{w_0, \dots, w_n, \sigma\} = \{\mathbf{w}, \sigma\}$  

Note: In this case, we assume that the noise is Gaussian distributed! i.e.  $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$

Note: We assume  $\mu = 0$  : That means we only have to estimate  $\sigma$ ! **Homework:** Why can we assume  $\mu = 0$ ?

# Overview

- Overall Picture
- The Linear Model
- **Optimization**
- Basis Function Expansion

# What are optimal model parameters?

We define the **optimal parameters** as:

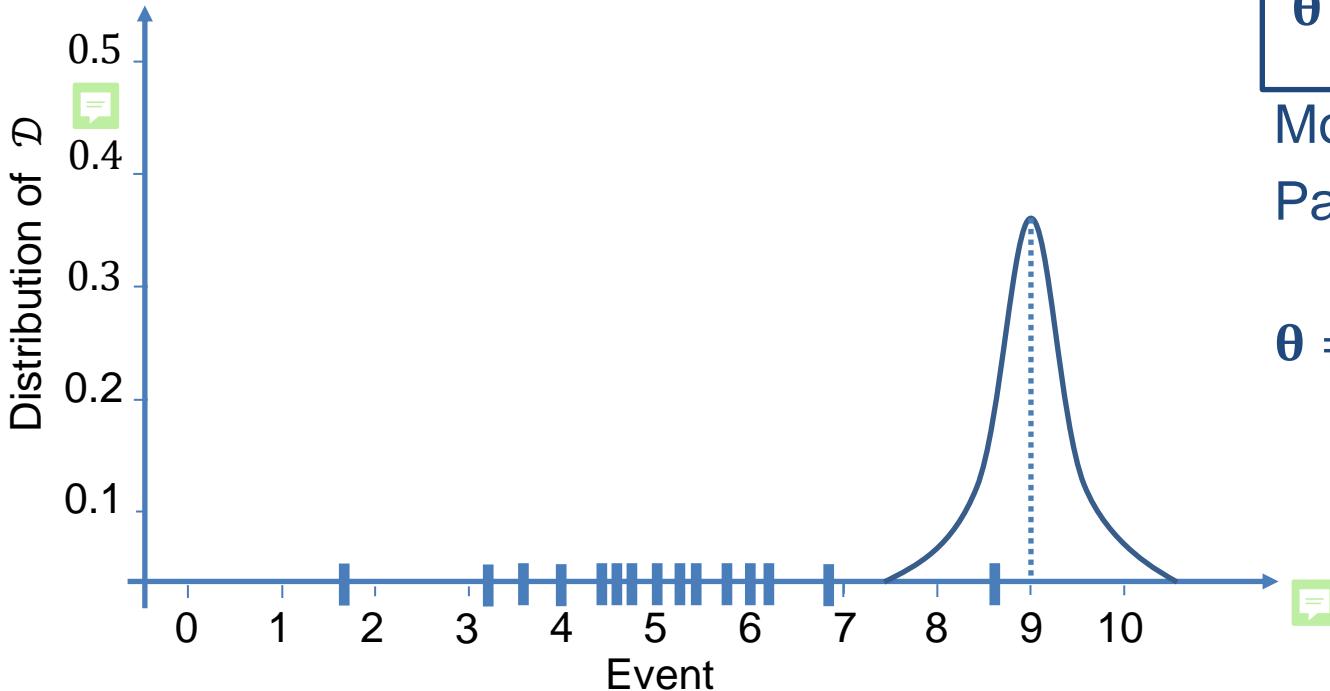
$$\theta^* = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}|\theta)$$

That means:

- We want to find the optimal parameters  $\theta^*$
- We search over all possible  $\theta$
- We select the  $\theta$  which **most likely generated** our training dataset  $\mathcal{D}$

Reminder:  $\theta$  are the parameters,  $\mathcal{D}$  is the training dataset!

# Intuitive Example



$$\theta^* = \operatorname{argmax}_{\theta} p(\mathcal{D}|\theta)$$

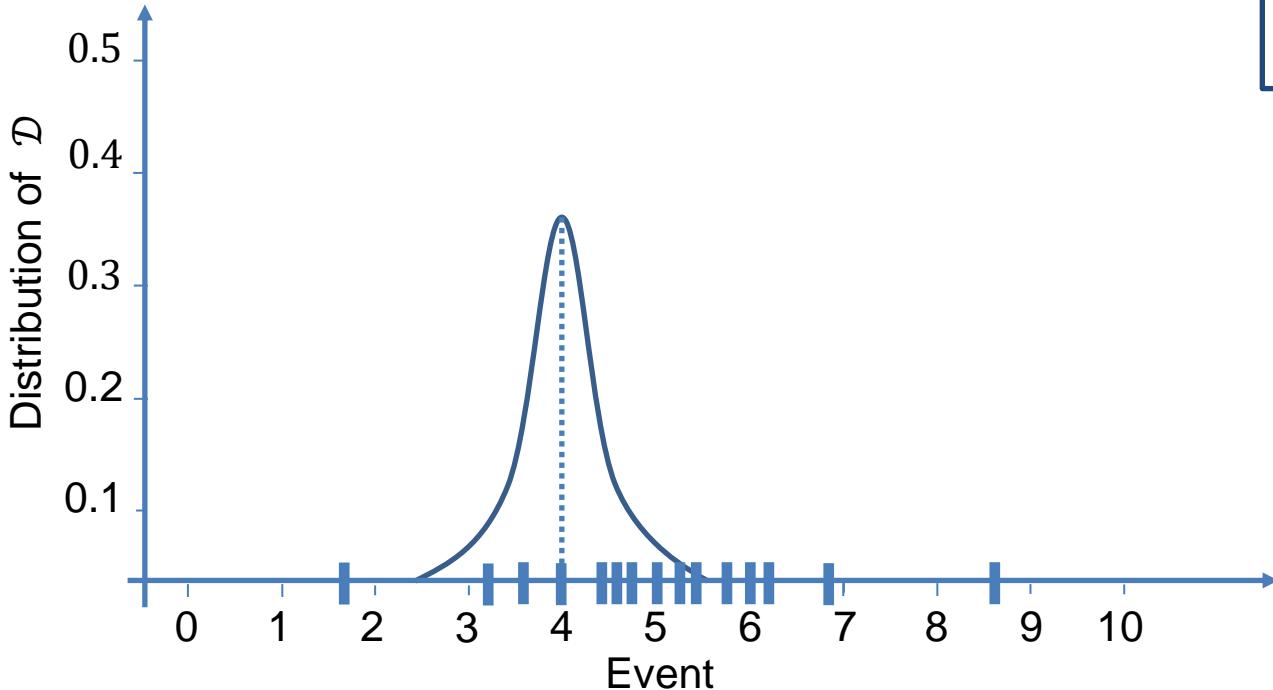
Model:  $\mathcal{D} \sim \mathcal{N}(\mu, \sigma^2)$

Para.:  $\theta = \{\mu, \sigma\}$

$\theta = \{9, 0.9\}$  : Bad

Reminder:  $\theta$  are the parameters,  $\mathcal{D}$  is the training dataset!

# Intuitive Example



$$\theta^* = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}|\theta)$$

Model:  $\mathcal{D} \sim \mathcal{N}(\mu, \sigma^2)$

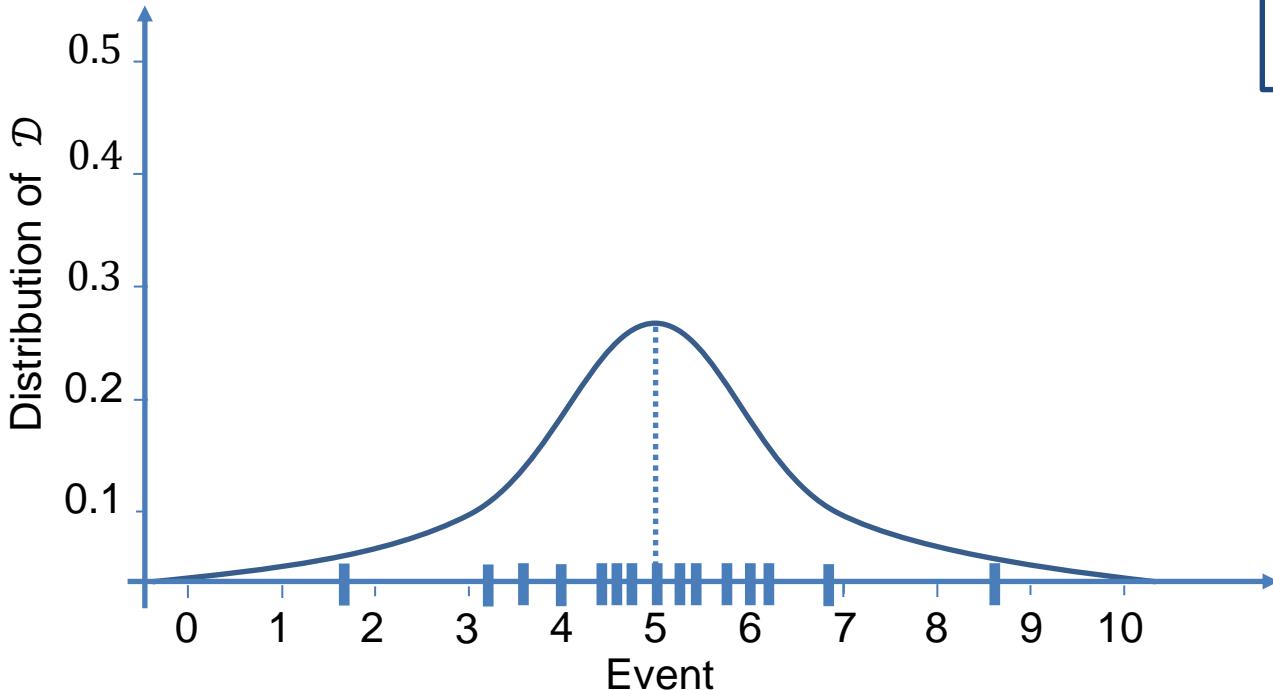
Para.:  $\theta = \{\mu, \sigma\}$

$\theta = \{9, 0.9\}$  : **Bad**

$\theta = \{4, 0.9\}$  : **Better**

Reminder:  $\theta$  are the parameters,  $\mathcal{D}$  is the training dataset!

# Intuitive Example



$$\theta^* = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}|\theta)$$

Model:  $\mathcal{D} \sim \mathcal{N}(\mu, \sigma^2)$

Para.:  $\theta = \{\mu, \sigma\}$

$\theta = \{9, 0.9\}$  : **Bad**

$\theta = \{4, 0.9\}$  : **Better**

$\theta = \{5, 1.5\}$  : **Best**

Reminder:  $\theta$  are the parameters,  $\mathcal{D}$  is the training dataset!

# Maximum Likelihood Estimation

This process is called **Maximum Likelihood Estimation (MLE)**

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} p(\mathcal{D}|\boldsymbol{\theta})$$

**Important:**  $l(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})$  is called **Likelihood!**



Note: It can also written as  $p(\mathcal{D}|\boldsymbol{\theta}) = l(\boldsymbol{\theta})$

# The Likelihood Function

We assume each sample is **independent, identically distributed (i.i.d)!\***

$$\begin{aligned}\mathcal{L}(\theta) &= p(\mathcal{D}|\theta) \\ &= p(s_0, s_1, \dots, s_n | \theta) \\ &\stackrel{*}{=} p(s_0 | \theta) \cdot p(s_1 | \theta) \cdot \dots \cdot p(s_n | \theta) \\ &= \prod_{i=1}^N p(s_i | \theta)\end{aligned}$$

→ This is the basis to judge, how good our parameters are!

Note: In our case a sample  $s_i$  is  $\mathbf{x}_i$  and  $\mathbf{y}_i$ ; Just assume both are "tied" together i.e. like a vector

# The Log-Likelihood

The  product of small numbers in a computer is unstable!

We transform the likelihood into the quasi-equivalent **Log-Likelihood**:

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{s}_i | \boldsymbol{\theta}) \stackrel{*}{\Leftrightarrow} \sum_{i=1}^N \log[p(\mathbf{s}_i | \boldsymbol{\theta})] = \ell(\boldsymbol{\theta})$$

\* The logarithm „just“ scales the problem. Likelihood and Log-Likelihood **both** have to be maximized!

# The conditional distribution $p(s_i|\theta)$

We know the conditional is the Gaussian distribution of  $y$  (Slide 35):

$$p(s_i|\theta) = p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \mathcal{N}(y_i | \mathbf{w}^T \mathbf{x}_i, \sigma^2)$$

We insert this into the log-likelihood:

$$\sum_{i=1}^N \log[p(s_i|\theta)]$$

# The conditional distribution $p(s_i|\theta)$

We know the conditional is the Gaussian distribution of  $y$  (Slide 35):

$$p(s_i|\theta) = p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \mathcal{N}(y_i | \mathbf{w}^T \mathbf{x}_i, \sigma^2)$$

We insert this into the log-likelihood:

$$\sum_{i=1}^N \log[p(s_i|\theta)] = \sum_{i=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T \mathbf{x}_i)^2} \right]$$



# The conditional distribution $p(\mathbf{s}_i|\boldsymbol{\theta})$

We know the conditional is the Gaussian distribution of  $\mathbf{y}$  (Slide 35):

$$p(\mathbf{s}_i|\boldsymbol{\theta}) = p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \mathcal{N}(\mathbf{y}_i | \mathbf{w}^T \mathbf{x}_i, \sigma^2)$$

We insert this into the log-likelihood:

$$\begin{aligned} \sum_{i=1}^N \log[p(\mathbf{s}_i|\boldsymbol{\theta})] &= \sum_{i=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2} \right] \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2) \end{aligned}$$

# The conditional distribution $p(\mathbf{s}_i|\boldsymbol{\theta})$

We know the conditional is the Gaussian distribution of  $\mathbf{y}$  (Slide 35):

$$p(\mathbf{s}_i|\boldsymbol{\theta}) = p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \sigma) = \mathcal{N}(\mathbf{y}_i | \mathbf{w}^T \mathbf{x}_i, \sigma^2)$$

We insert this into the log-likelihood:

$$\begin{aligned} \sum_{i=1}^N \log[p(\mathbf{s}_i|\boldsymbol{\theta})] &= \sum_{i=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2} \right] \\ &= \boxed{-\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2)} \end{aligned}$$

# Interpreting the Loss

We can “simplify” the current log-likelihood:

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2)$$

# Interpreting the Loss

We can “simplify” the current log-likelihood:

$$\ell(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2)$$



# Interpreting the Loss

We can “simplify” the current log-likelihood:

$$\ell(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2)$$

Residual Sum of Squares 

Constant  Constant 

## Interpreting the Loss

We can “simplify” the current log-likelihood:

$$\ell(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (\mathbf{y}_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{N}{2} \log(2\pi\sigma^2)$$

Constant

Constant

Residual Sum of Squares

→ We only have to **minimize** the **Residual Sum of Squares!**

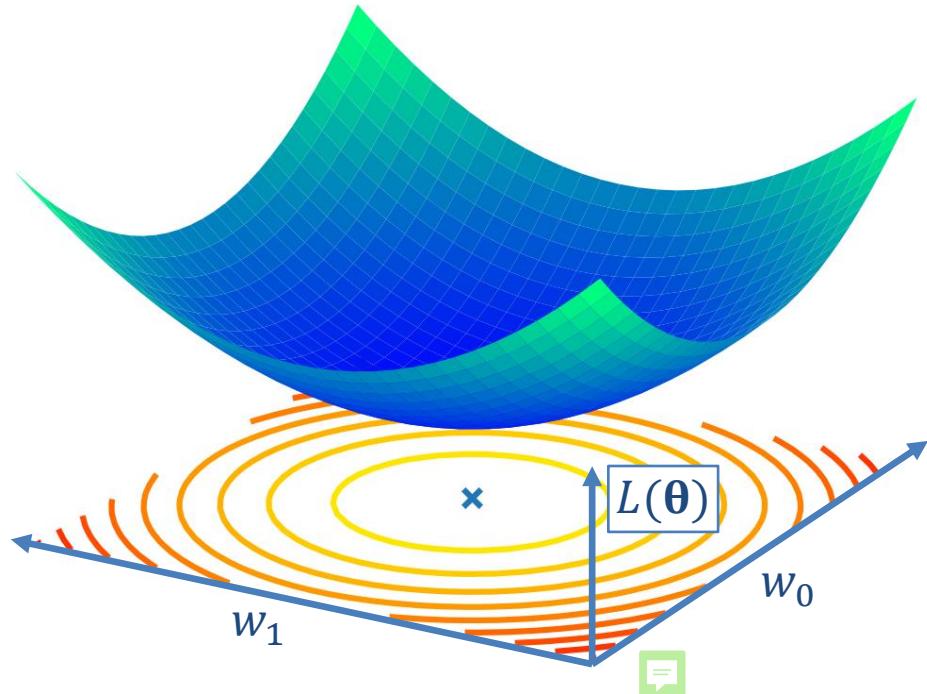
# The Residual Sum of Squares

We minimize the loss term:

$$L(\theta) = \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

For linear regression: The loss is shaped like a bowl with a unique minimum!

The cross indicates the optimal parameters i.e. where the loss is minimal



# Optimization

Multiple ways to find the minimum:

- Analytical Solution
- Gradient Descend
- Newtons Method

... and of course many, many more methods!<sup>1)</sup>

1) Examples: Particle Swarm Optimization, Genetic Algorithms, etc.

# Optimization

Multiple ways to find the minimum:

- **Analytical Solution**
- Gradient Descend
- Newtons Method

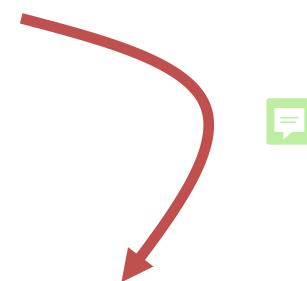
... and of course many, many more methods!<sup>1)</sup>

1) Examples: Particle Swarm Optimization, Genetic Algorithms, etc.

# Analytical Solution

For the analytical solution, we want a “simpler” form:<sup>1)</sup>

$$L(\boldsymbol{\theta}) = \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$



$$NLL(\boldsymbol{\theta}) = \frac{1}{2} (\mathbf{y} - \mathbf{x}\mathbf{w})^T (\mathbf{y} - \mathbf{x}\mathbf{w})$$

1) This form uses the “**Negative Log Likelihood**”, which can also be derived from the likelihood

# Analytical Solution

For the analytical solution, we want a “simpler” form:<sup>1)</sup>

$$\text{NLL}(\boldsymbol{\theta}) = \frac{1}{2} (\mathbf{y} - \mathbf{x}\mathbf{w})^T (\mathbf{y} - \mathbf{x}\mathbf{w})$$

We find the minimum conventionally by using **the derivative**:

$$\text{NLL}'(\boldsymbol{\theta}) = \mathbf{x}^T \mathbf{x}\mathbf{w} - \mathbf{x}^T \mathbf{y}$$



Equating the gradient to zero:



$$\mathbf{x}^T \mathbf{x}\mathbf{w} = \mathbf{x}^T \mathbf{y}$$

1) This form uses the “**Negative Log Likelihood**”, which can also be derived from the likelihood

# Analytical Solution

The solution i.e. the minimum is:

$$\mathbf{w} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y}$$

**In practice too time consuming to compute!**

**Reason:**

The more training data,  
the longer the calculation!

$$\mathbf{x}^T \mathbf{y} = \sum_{i=1}^N \mathbf{x}_i \mathbf{y}_i$$

$$\mathbf{x}^T \mathbf{x} = \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$$

$$= \sum_{i=1}^N \begin{pmatrix} x_{i,1}^2 & \cdots & x_{i,1} \cdot x_{i,D} \\ \vdots & \ddots & \vdots \\ x_{i,D} \cdot x_{i,1} & \cdots & x_{i,D}^2 \end{pmatrix}$$

Note:  $N$  : Amount of Samples in the Training Dataset ;  $D$  : Amount of Dimensions (length of the input vector)

# What is the quality of the fit?

The goodness of fit should be evaluated to verify the learned model!

A typical measure for quality is :  $R^2$

It is a measure for the **percentage of variability of the dependent variable (y)** in the data explained by the model!

# The $R^2$ – Measure

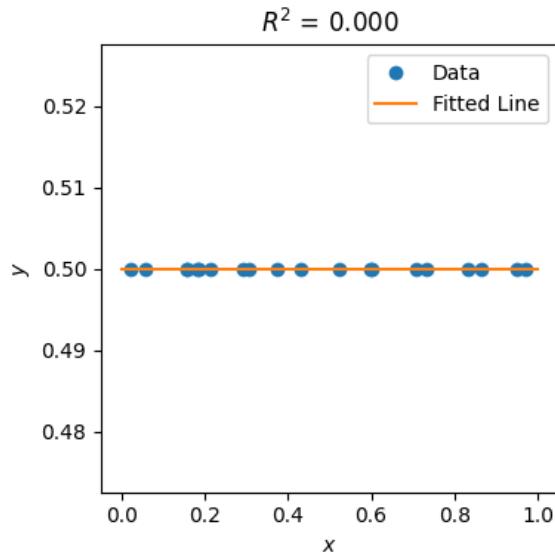
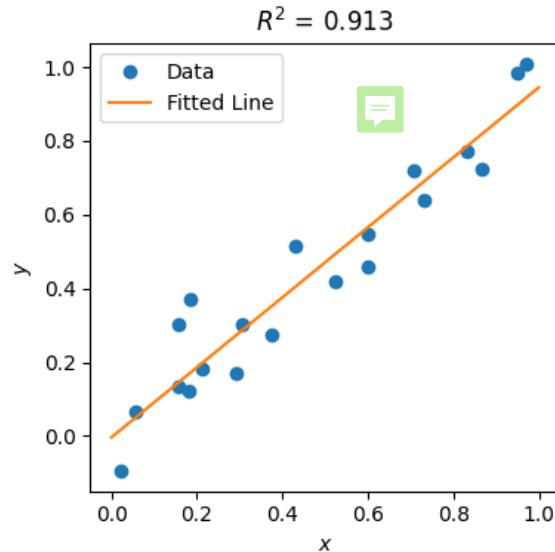
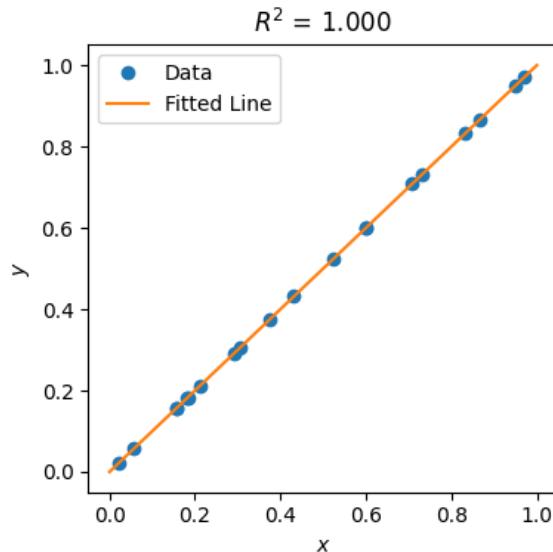
The  $R^2$  – Measure is calculated as:

Let  $\hat{y}$  be the mean of the labels, then 

- **Total sum of squares:**  $SS_{tot} = \sum_i (y_i - \hat{y})^2$  
- **Residual sum of squares:**  $SS_{res} = \sum_i (y_i - f(x_i))^2$  
- **Coefficient of determination:** 
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$
 

→ This is different from the Mean Squared Error (MSE)! 

# Linear Regression: Evaluation



Model can completely (100%)  
explain variations in  $y$

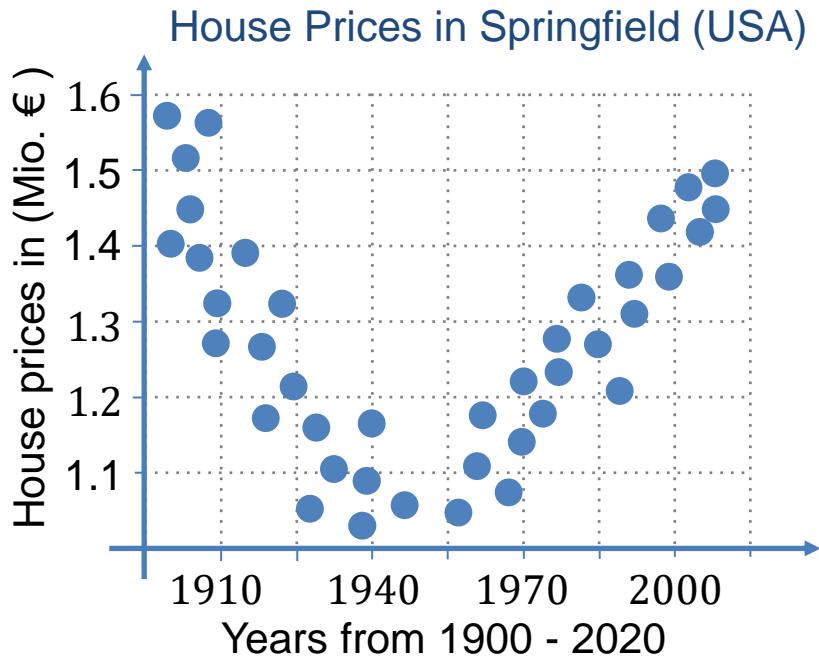
Model can partially (91.3%)  
explain variations in  $y$ , but still  
considered **good**

Model cannot explain any  
(0%) variation in  $y$ , because  
it's **its own average**

# Overview

- Overall Picture
- The Linear Model
- Optimization
- **Basis Function Expansion**

# Can we fit the following dataset?



Source: <https://www.who.int/toolkits/growth-reference-data-for-5to19-years/indicators/height-for-age>

# Polynomial Regression

The answer is – of course – yes!

**Polynomial regression** fits a **nonlinear relationship** between input of  $x$  and the corresponding conditional mean of  $y$

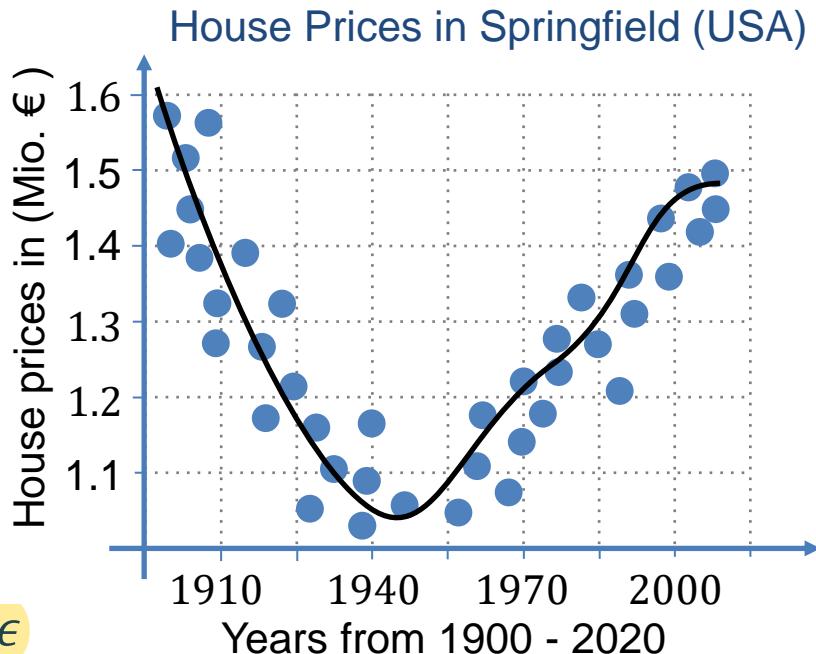


Example of a polynomial model:

$$y = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m + \epsilon$$



Note:  $m$  is called the degree of the polynomial



# How can we use polynomials in linear regression?

Trick: Basis function expansion!

We can just apply a function **before** we predict our “linear line”:

$$f(\mathbf{x}) = \sum_{j=1}^D w_j \Phi_j(\mathbf{x}) + \epsilon = \mathbf{y}$$

That means: We transform our Input-Space by using the function  $\Phi$ !

→ The estimation  from “normal” linear regression is the same!

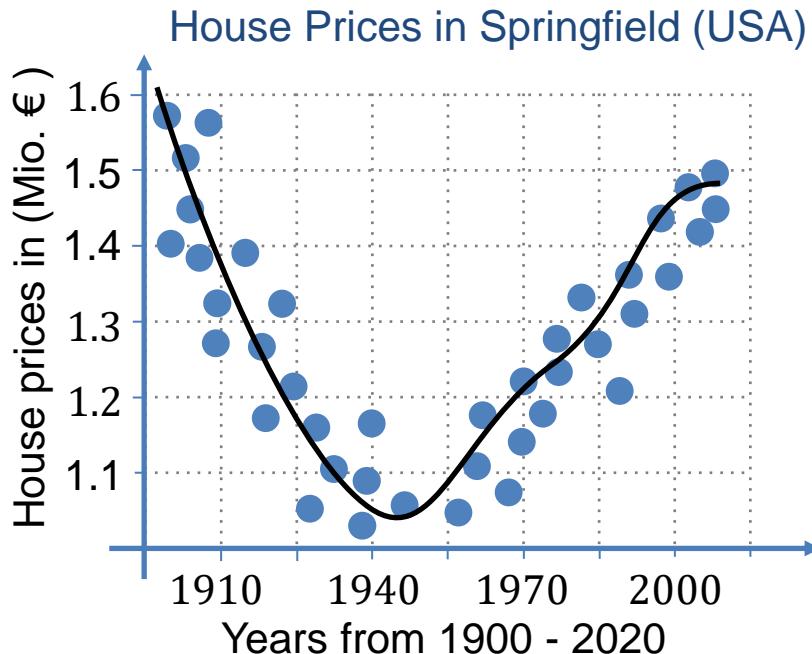
# Example: Polynomial Regression

The Basis Function for our example:

$$\Phi(x) = \begin{pmatrix} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 \end{pmatrix}^T$$

The model parameters<sup>1)</sup> are:

$$\boldsymbol{\theta} = \{w_0, w_1, w_2, w_3, w_4, w_5, w_6, \sigma\}$$

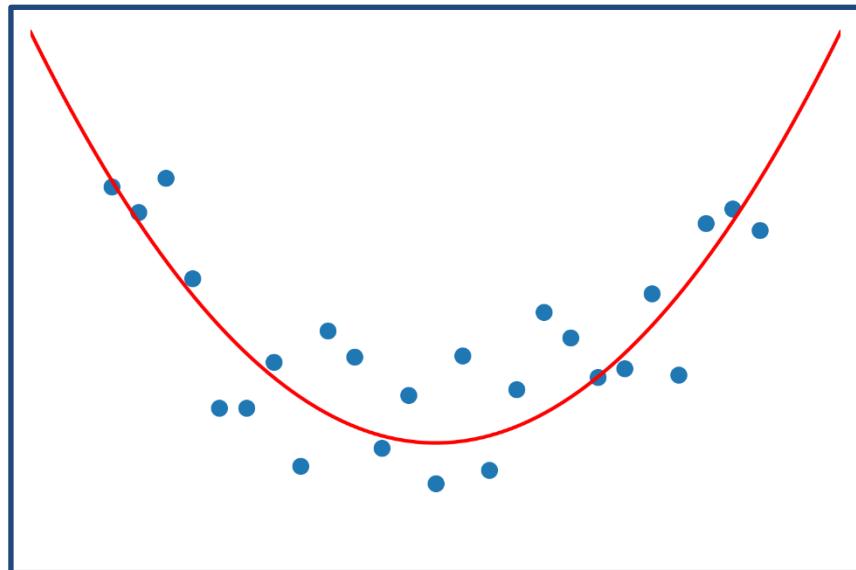


1) We assume that the noise is Gaussian distributed!

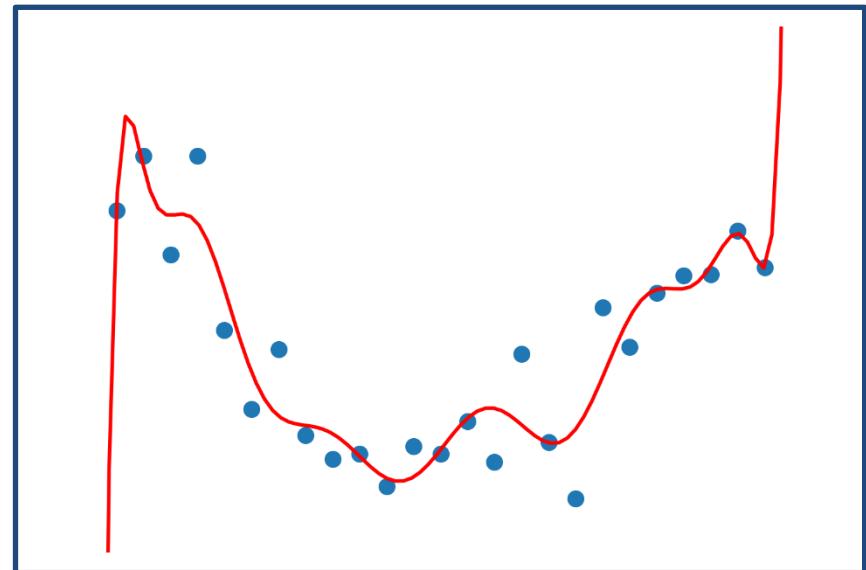
Note: The input here is NOT a vector! It is a scalar! However, the result is a 7-dim vector!

# Complexity of Polynomial Regression

The **greater degrees** of the polynomial, the **more complex** relationships we can express!



Degree  $m = 2$



Degree  $m = 13$

# Basis Functions

We can use **any arbitrary function**, with **one condition**:

The resulting vector has to have a constant “length”  
**i.e. linear in respect to the parameters!**

Possible basis functions:

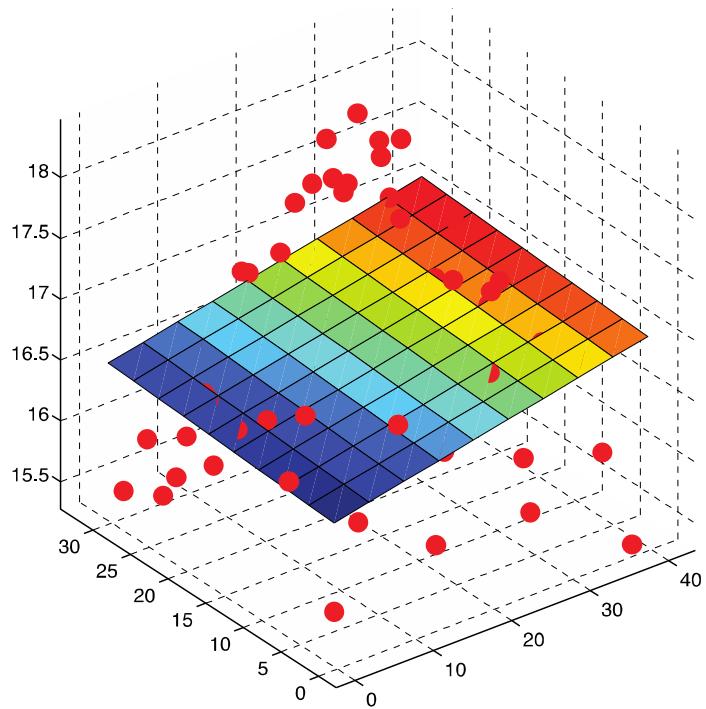


- $\Phi(\mathbf{x}) = (1 \ x_0 \ x_1 \dots x_D)^T$
- $\Phi(\mathbf{x}) = (1 \ x \ x^2 \dots x^m)^T$
- $\Phi(\mathbf{x}) = (1 \ x_0 \ x_1 \ x_0x_1 \ x_0^2 \ x_1^2 \ x_2^2)^T$
- etc.

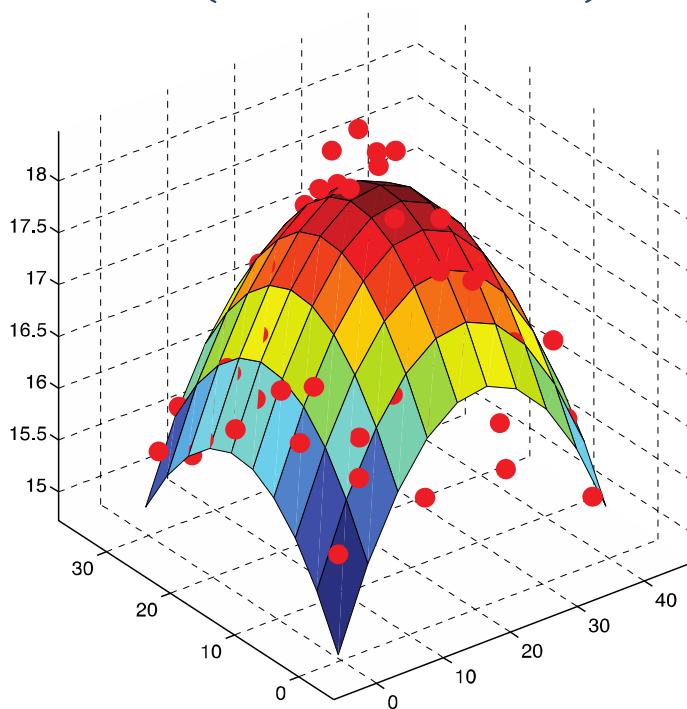
Note: That is the reason, why polynomial regression can be considered linear

# Examples for Basis Functions

$$\Phi(\mathbf{x}) = (1 \ x_0 \ x_1)^T$$



$$\Phi(\mathbf{x}) = (1 \ x_0 \ x_1 \ x_2 \ x_1^2 \ x_2^2)^T$$

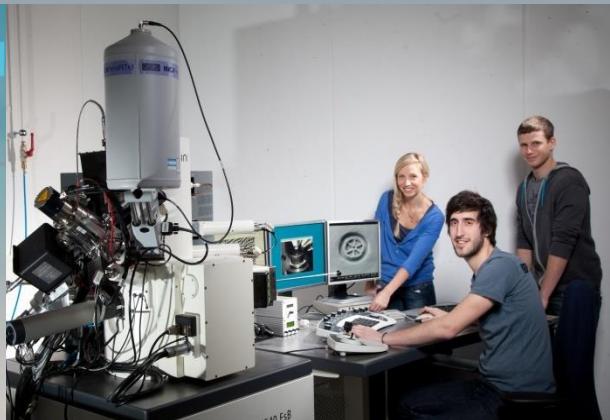
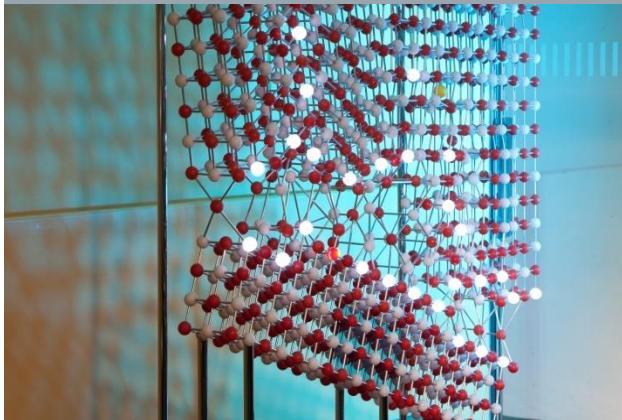


Thank you for listening!



# Machine Learning for Engineers

## Logistic Regression - Motivation



Bilder: TF / Malter

# Motivation

Logistic regression is the application of linear regression to classification!

## Use Cases:

- Credit Scoring
- Medicine
- Text Processing
- etc.

→ Especially useful for “**explainability**” 

Source: <https://activewizards.com/blog/5-real-world-examples-of-logistic-regression-application>

# When do we use it?

The problem has to be simple:

- Dataset is small
- Linear model is enough
- Basis for complex models

→ Let's have a look at classification

i.e. **prediction of a categorial-valued output**

# Example: Iris Flower Dataset

Label	Petal Width	Petal Length
Setosa	5.0 mm	9.2 mm
Versi.	9.2 mm	26.1 mm
Setosa	7.7 mm	18.9 mm
Versi.	9.1 mm	32.1 mm
Setosa	7.9 mm	15.5 mm
Setosa	5.7 mm	12 mm
Setosa	2.5 mm	13.5 mm
Versi.	15.0 mm	39 mm
⋮		

Contains 50 samples of the flowers **Iris setosa**, **Iris virginica** and **Iris versicolor**

We want to answer Questions like:

My flower has a **Petal Width of 7mm** and a **Petal Length of 15mm**.

Is my flower a Iris Setosa or Iris Versicolor?

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

# Example: Transform the Label

Label	Petal Width	Petal Length
Setosa	5.0 mm	9.2 mm
Versi.	9.2 mm	26.1 mm
Setosa	7.7 mm	18.9 mm
Versi.	9.1 mm	32.1 mm
Setosa	7.9 mm	15.5 mm
Setosa	5.7 mm	12 mm
Setosa	2.5 mm	13.5 mm
Versi.	15.0 mm	39 mm
⋮		

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

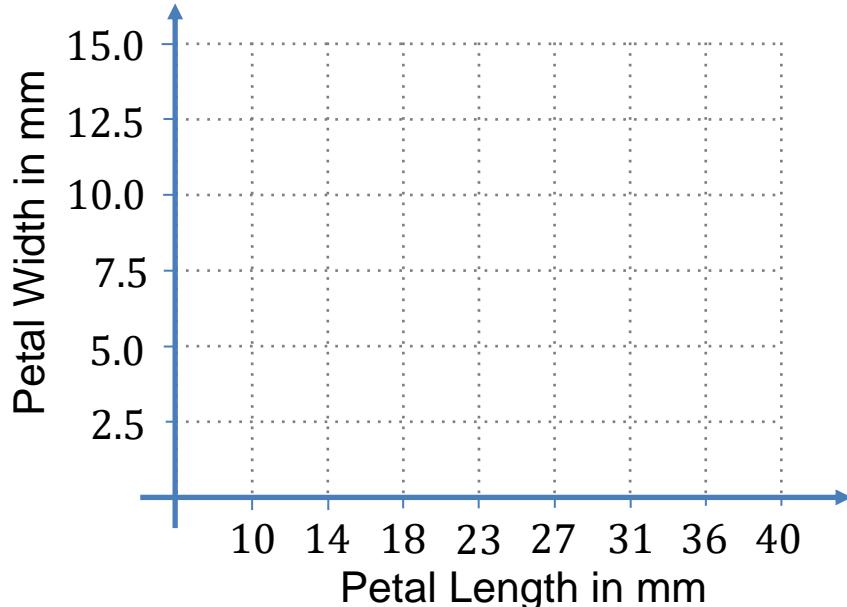
# Example: Transform the Label

Label	Petal Width	Petal Length
0	5.0 mm	9.2 mm
1	9.2 mm	26.1 mm
0	7.7 mm	18.9 mm
1	9.1 mm	32.1 mm
0	7.9 mm	15.5 mm
0	5.7 mm	12 mm
0	2.5 mm	13.5 mm
1	15.0 mm	39 mm
⋮		

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

# Example: 1. Visualize the data

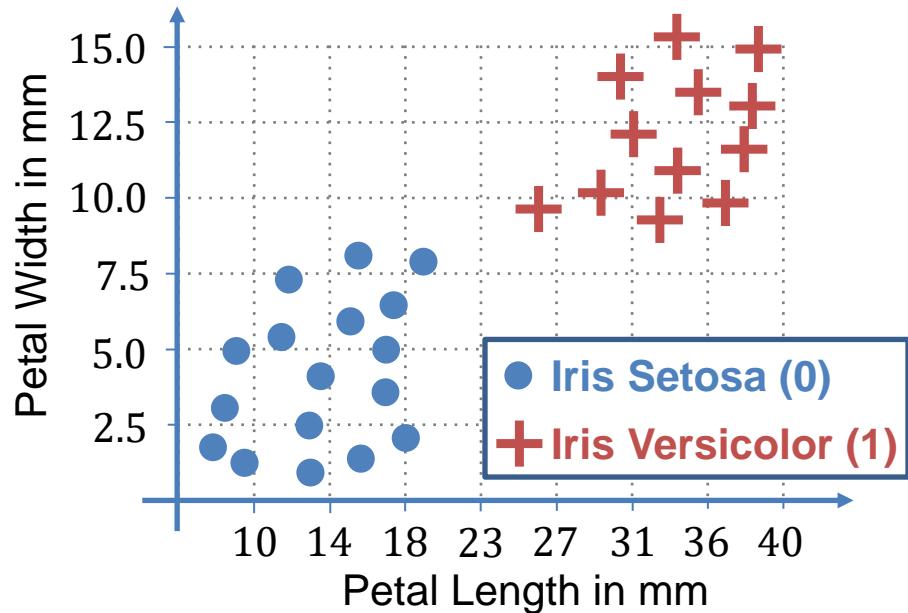
Label	Petal Width	Petal Length
0	5.0 mm	9.2 mm
1	9.2 mm	26.1 mm
0	7.7 mm	18.9 mm
1	9.1 mm	32.1 mm
0	7.9 mm	15.5 mm
0	5.7 mm	12 mm
0	2.5 mm	13.5 mm
1	15.0 mm	39 mm
⋮		



Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

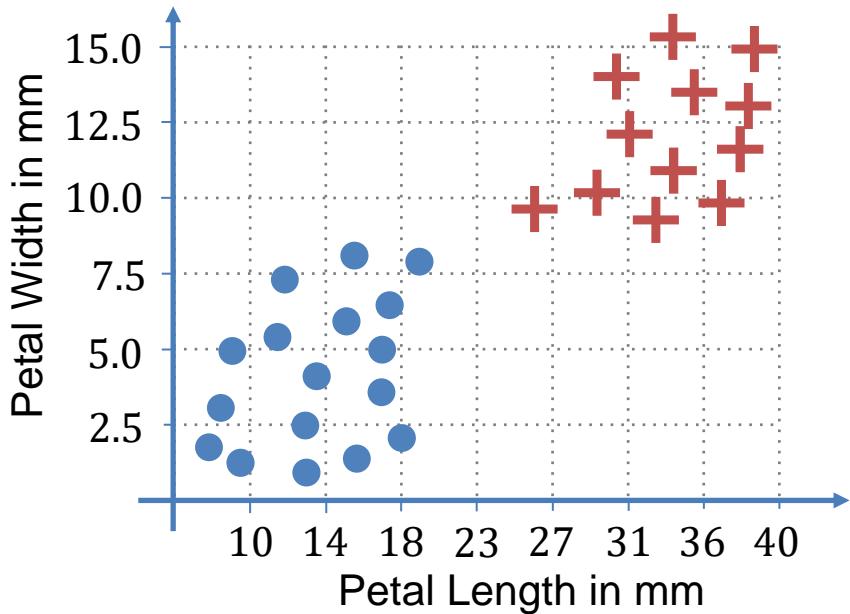
# Example: 1. Visualize the data

Label	Petal Width	Petal Length
0	5.0 mm	9.2 mm
1	9.2 mm	26.1 mm
0	7.7 mm	18.9 mm
1	9.1 mm	32.1 mm
0	7.9 mm	15.5 mm
0	5.7 mm	12 mm
0	2.5 mm	13.5 mm
1	15.0 mm	39 mm
⋮		



Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Example: 2. Find a decision boundary



Note: Setosa ● Versicolor +

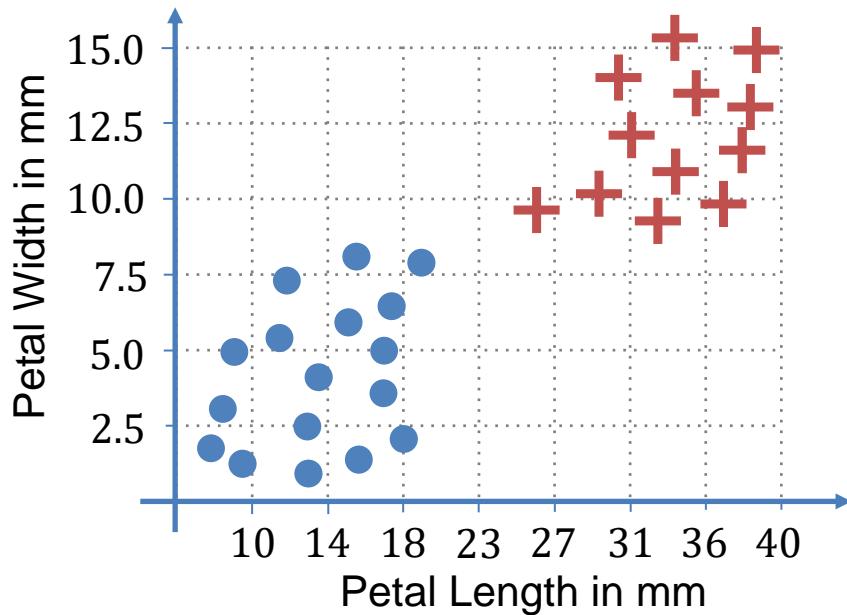
Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Example: 2. Find a decision boundary

What decision boundary i.e. function separates the data?

- Linear boundary
- Polynomial boundary
- Gaussian boundary

⋮



Note: Setosa ● Versicolor +

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

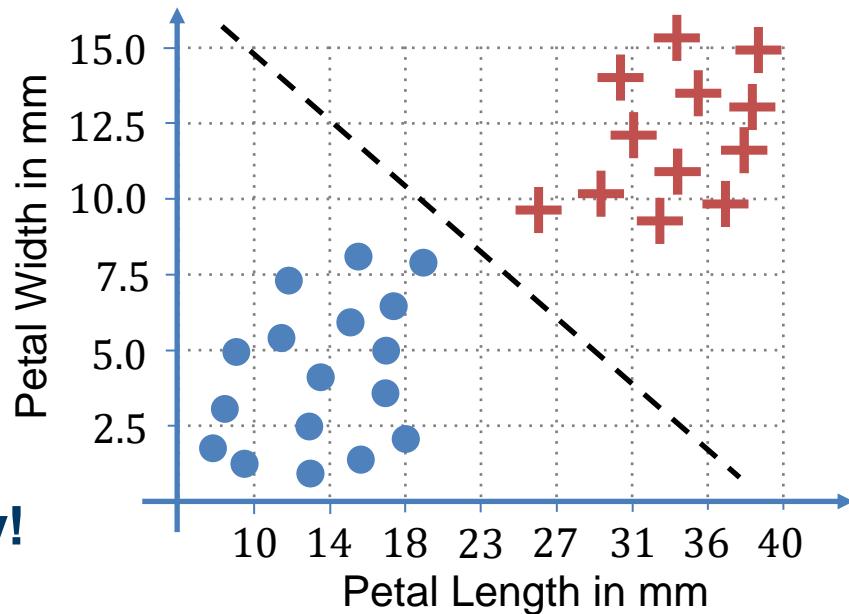
## Example: 2. Find a decision boundary

What decision boundary i.e. function separates the data?

- Linear boundary
- Polynomial boundary
- Gaussian boundary

⋮

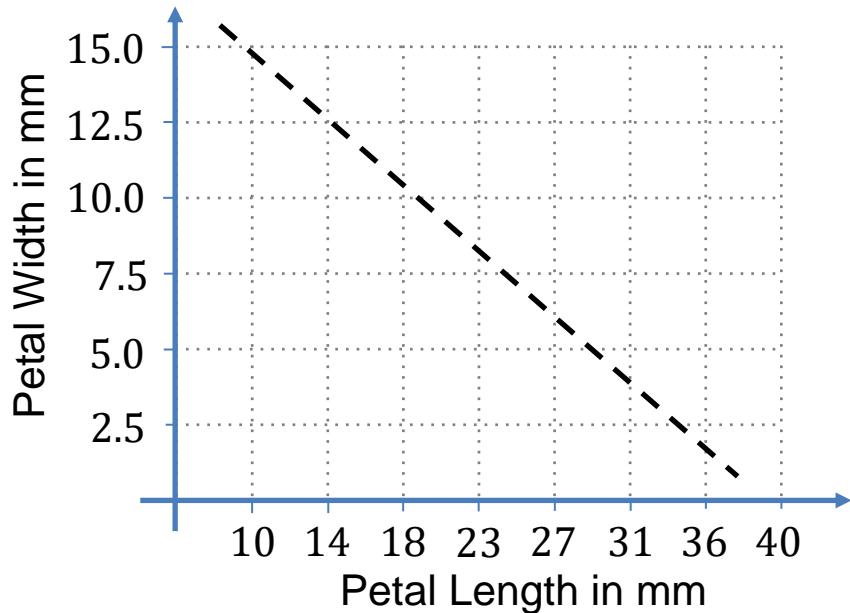
Answer: **Linear Decision Boundary!**



Note: Setosa ● Versicolor +

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Example: 3. Answer your Question



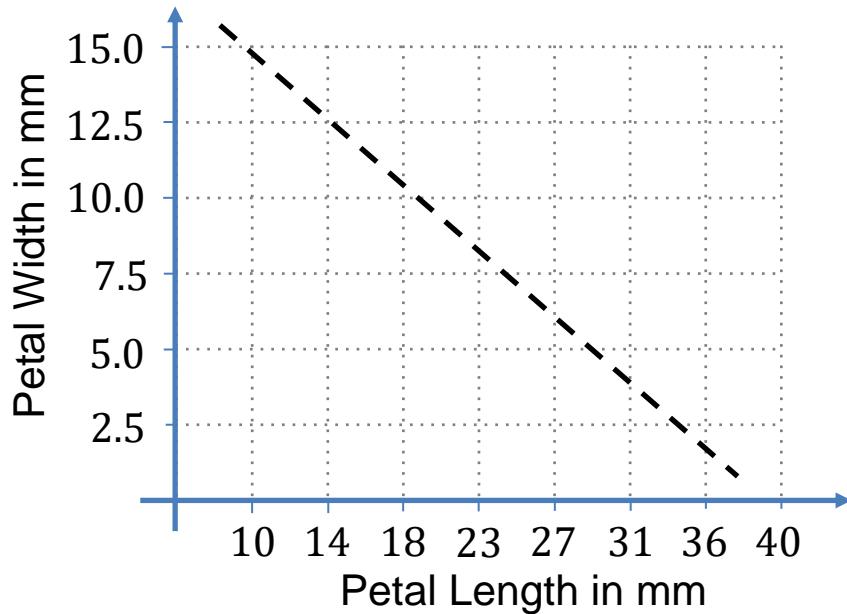
Note: Setosa ● Versicolor +

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Example: 3. Answer your Question

My flower has a **Petal Width** of 7mm and a **Petal Length** of 15mm.

Is my Flower a Iris Setosa or Iris Versicolor?



Note: Setosa ● Versicolor +

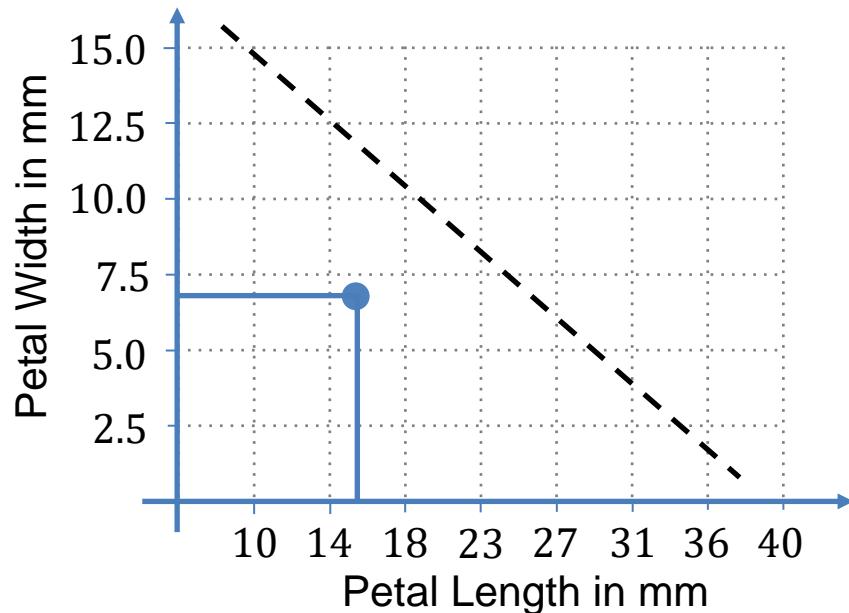
Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Example: 3. Answer your Question

My flower has a **Petal Width** of 7mm and a **Petal Length** of 15mm.

Is my Flower a Iris Setosa or Iris Versicolor?

→ Iris Setosa



Note: Setosa ● Versicolor + | Reason: The point is on the “left side” of the decision boundary

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

## Next in this Lecture:

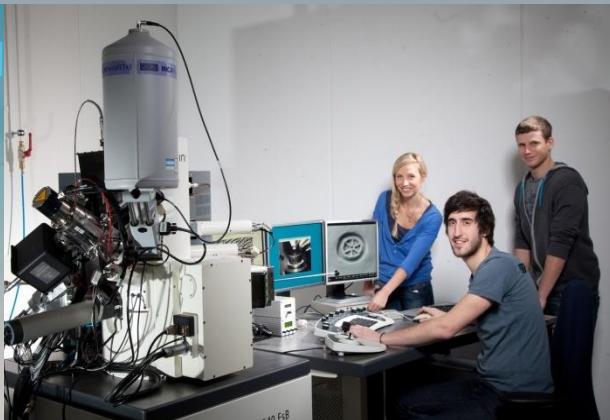
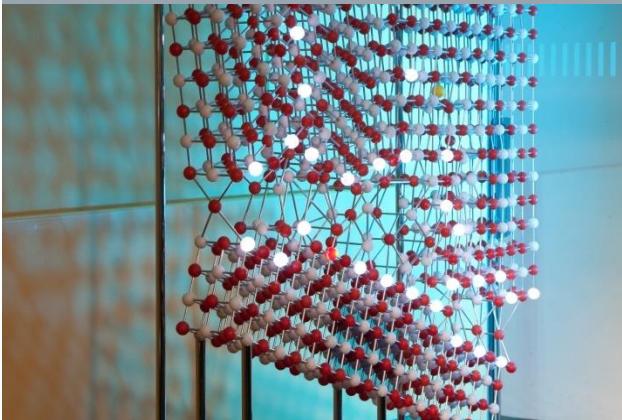
- What is the Mathematical Framework?
- How do we classify using a linear model?

Thank you for listening!



# Machine Learning for Engineers

## Logistic Regression



Bilder: TF / Malter

# Overview

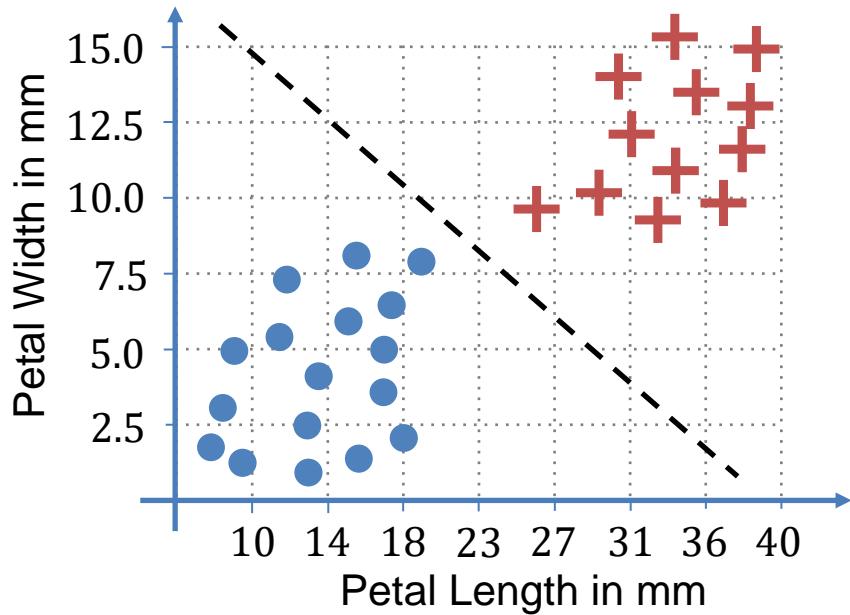
- The Logistic Model
- Optimization

# Overview

- **The Logistic Model**
- Optimization

# Logistic Regression

How do we describe the linear model as **decision boundary**?



Note: Setosa ● Versicolor +

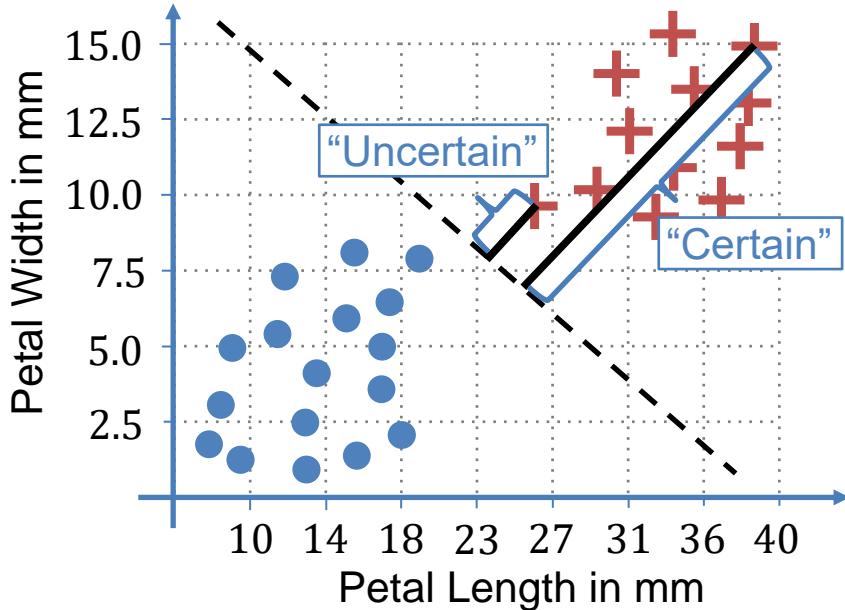
Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

# Logistic Regression

How do we describe the linear model as **decision boundary**?

**Thumb-Rule:**

The **larger the distance** of the input  $x$  to the decision boundary,  
**the more certain** it belongs to either Setosa (left of the line) or Versicolor (right of the line)!



Note: Setosa ● Versicolor +

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

# From distance to “probability”

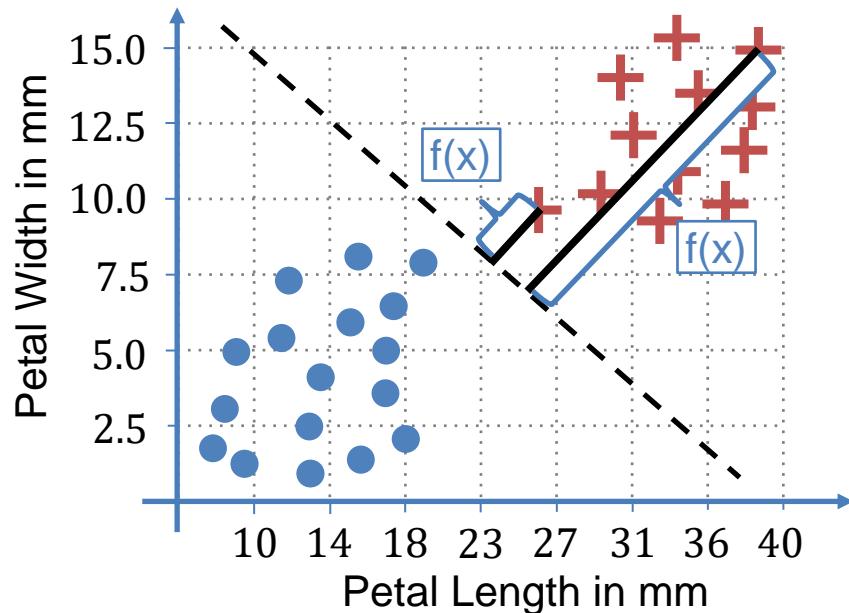
The linear model:

$$f(\mathbf{x}) = \sum_{j=1}^D w_j x_j$$



Calculates a signed<sup>1)</sup> distance,  
between the input and the linear  
model

→ How do we get a “probability”?



Note: Setosa ● Versicolor + | 1) negative (when left of the line), positive (when right of the line)

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

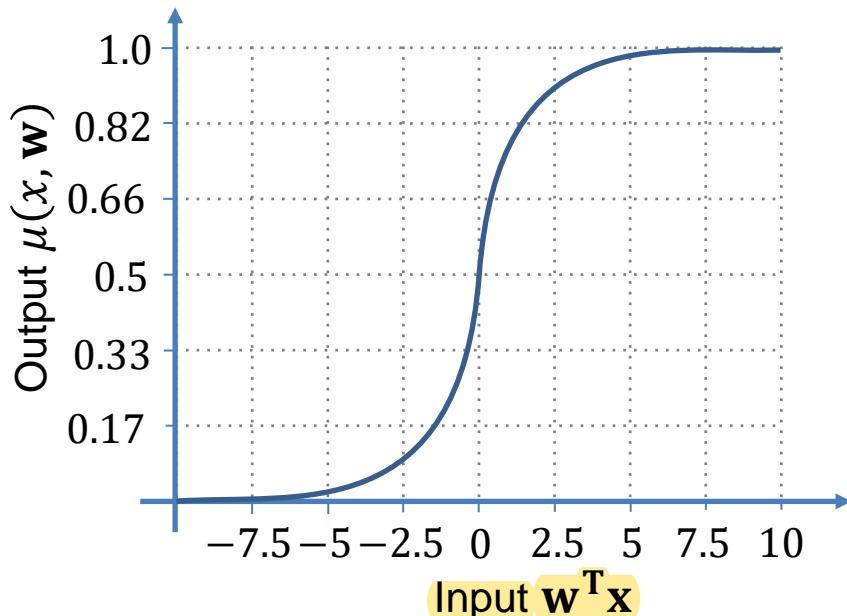
# The sigmoid function

The **sigmoid (logit, logistic)** function maps to the range [0,1]! 

That means the model is now:

$$\mu(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

→ „Only“ the probability for the event **versicolor**... 



Fun fact: The sigmoid function is sometimes lovingly called “squashing function”

Note: Here we already inserted the function  $f$ ! Homework: What is the general form of the sigmoid function? 

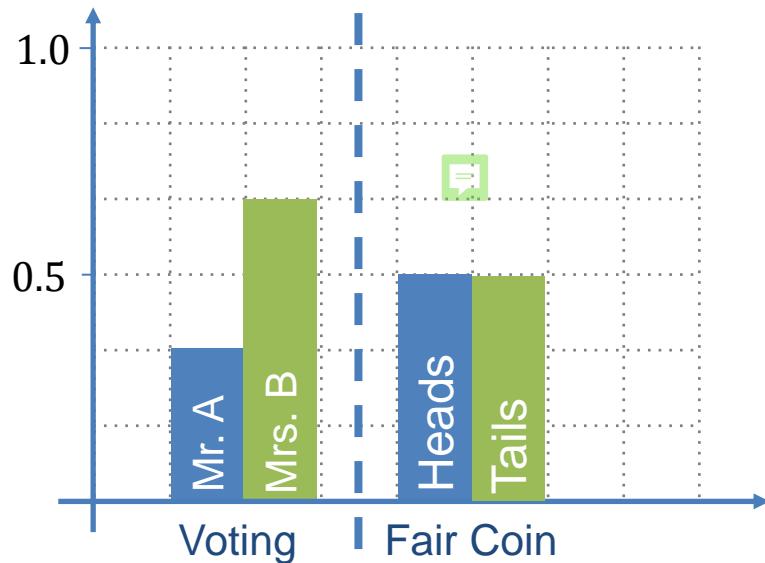
# Bernoulli distribution

The Bernoulli distribution can model both events (yes-or-no event):

$$\begin{aligned} p(y | \mathbf{x}, \mathbf{w}) &= \text{Ber}(y | \mu(\mathbf{x}, \mathbf{w})) \\ &= \mu(\mathbf{x}, \mathbf{w})^y (1 - \mu(\mathbf{x}, \mathbf{w}))^{1-y} \end{aligned}$$

## Problem:

How do we get the label, based on the calculated probability?



Note: We use the above distribution for the MLE estimation! Basically we replace " $p(s_i|\theta)$ " in the log-likelihood with this!

# The Decision Rule

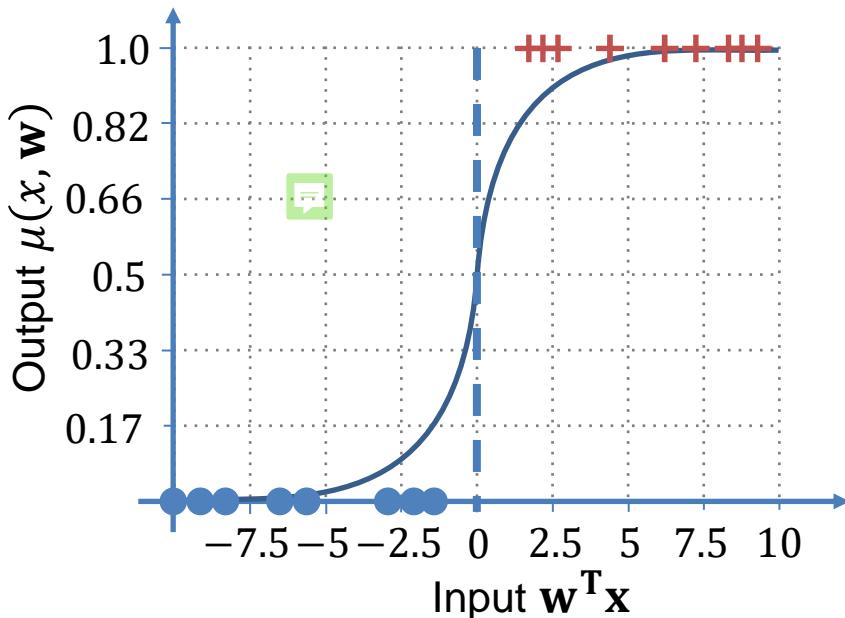
Based on  $\mu(x, w)$  we can decide, which class is more “likely”!

The Decision Rule is:

$$y = \begin{cases} 1, & \text{if } \mu(x, w) > 0.5 \\ 0, & \text{if } \mu(x, w) \leq 0.5 \end{cases}$$

**Question:**

How do we find the optimal parameters?



Note: Setosa ● Versicolor + | 1) negative (when left of the line), positive (when right of the line)

Source: Fisher, R.. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." Annals of Human Genetics 7 (1936): 179-188.

# Overview

- The Logistic Model
- **Optimization**

# Constructing the Loss

## Reminder: The Log-Likelihood

$$\ell(\theta) = \sum_{i=1}^N \log[p(y_i | \mathbf{x}_i, \theta)]$$

# Constructing the Loss

**Reminder:** The Log-Likelihood

$$\ell(\theta) = \sum_{i=1}^N \log[p(y_i | \mathbf{x}_i, \theta)]$$

$$= \sum_{i=1}^N \log \left( \mu(\mathbf{x}_i, \mathbf{w})^{y_i} (1 - \mu(\mathbf{x}_i, \mathbf{w}))^{1-y_i} \right)$$

Bernoulli Distribution

$$p(y | \mathbf{x}, \mathbf{w}) = \\ = \mu(\mathbf{x}, \mathbf{w})^y (1 - \mu(\mathbf{x}, \mathbf{w}))^{1-y}$$

\* We just insert the Probability from Slide 99 

# Constructing the Loss

**Reminder:** The Log-Likelihood

$$\ell(\theta) = \sum_{i=1}^N \log[p(y_i | \mathbf{x}_i, \theta)]$$

$$= \sum_{i=1}^N \log \left( \mu(\mathbf{x}_i, \mathbf{w})^{y_i} (1 - \mu(\mathbf{x}_i, \mathbf{w}))^{1-y_i} \right)$$

$$= \sum_{i=1}^N y_i \log(\mu(\mathbf{x}_i, \mathbf{w})) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i, \mathbf{w}))$$

Bernoulli Distribution

$$p(y | \mathbf{x}, \mathbf{w}) = \\ = \mu(\mathbf{x}, \mathbf{w})^y (1 - \mu(\mathbf{x}, \mathbf{w}))^{1-y}$$

\* We just insert the Probability from Slide 99

# The Cross Entropy Loss

Instead of Maximizing the Log-Likelihood, we **minimize the Negative Log-Likelihood!**

This Loss is called the **Cross Entropy**: 

$$L(\boldsymbol{\theta}) = - \sum_{i=1}^N y_i \log(\mu(\mathbf{x}_i, \mathbf{w})) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i, \mathbf{w}))$$

# The Cross Entropy Loss

Instead of Maximizing the Log-Likelihood, we **minimize the Negative Log-Likelihood!**

This Loss is called the **Cross Entropy**:

$$L(\boldsymbol{\theta}) = - \sum_{i=1}^N y_i \log(\mu(\mathbf{x}_i, \mathbf{w})) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i, \mathbf{w}))$$

- Unique minimum
- No analytical solution possible!

→ Optimization with Gradient descent.



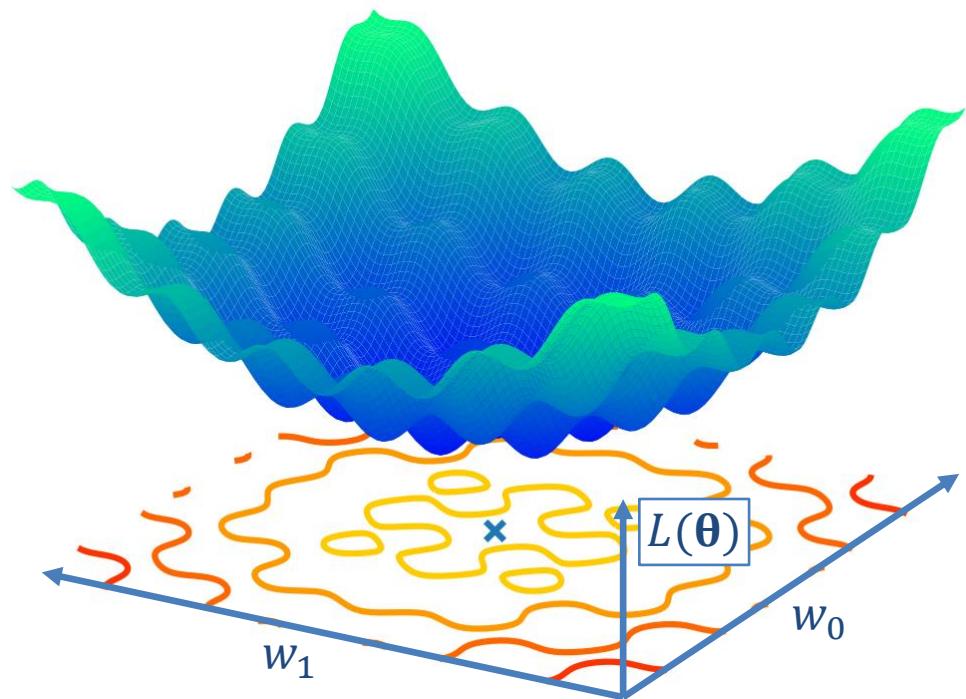
# Optimization Method: Gradient Descend

## Observation:

The loss is like a mountainous landscape!

## Idea:

We find the minimum, by “walking down” the slope of the mountain



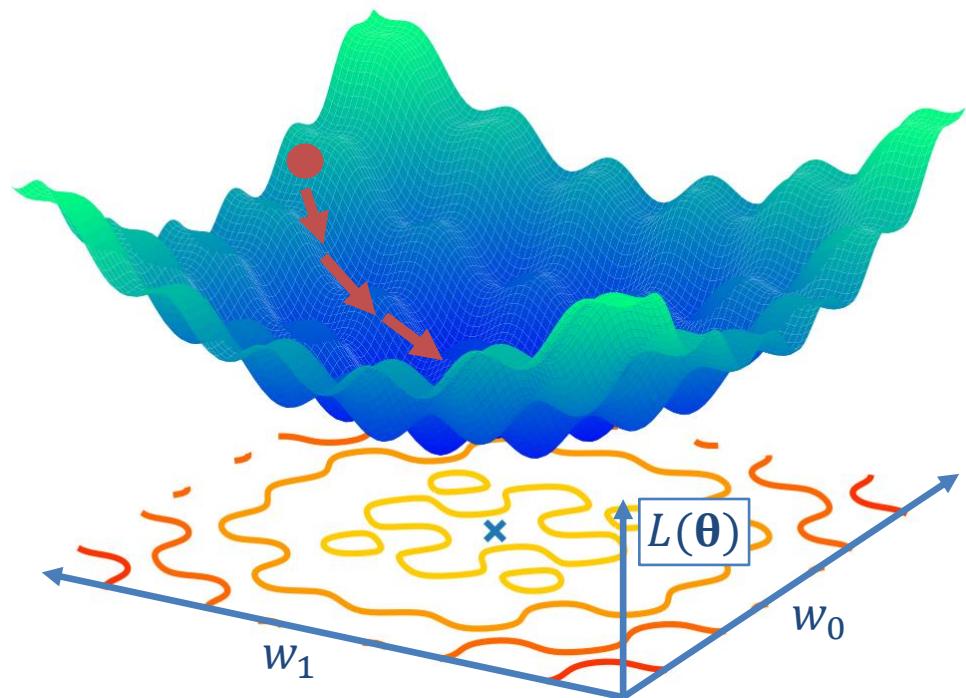
# Optimization Method: Gradient Descend

## Approach:

1. Start with random weights
2. Calculate: The direction of **steepest descend** 
3. Step in that direction
4. Repeat from step 2

## Result:

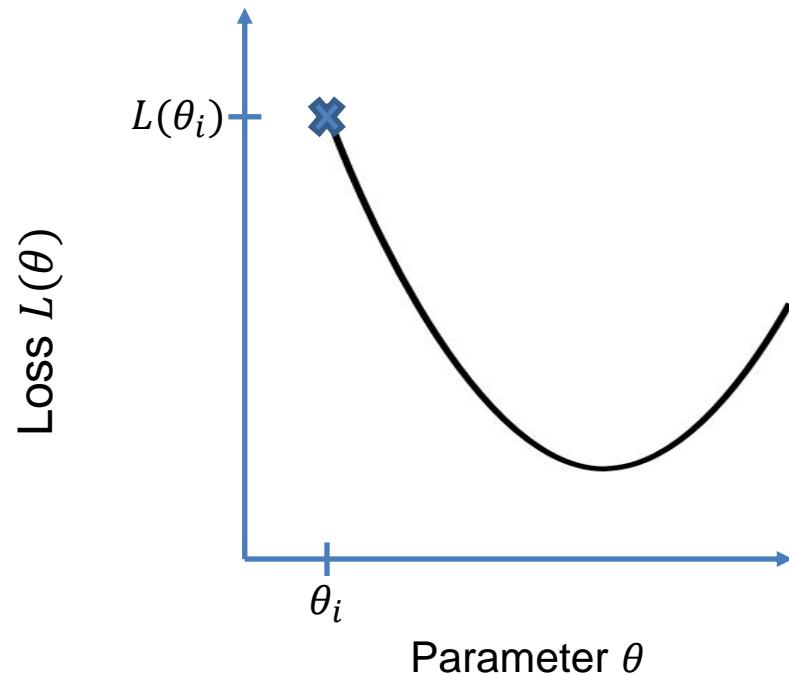
Some local minimum is reached!



# Example: Gradient descend

## Algorithm:

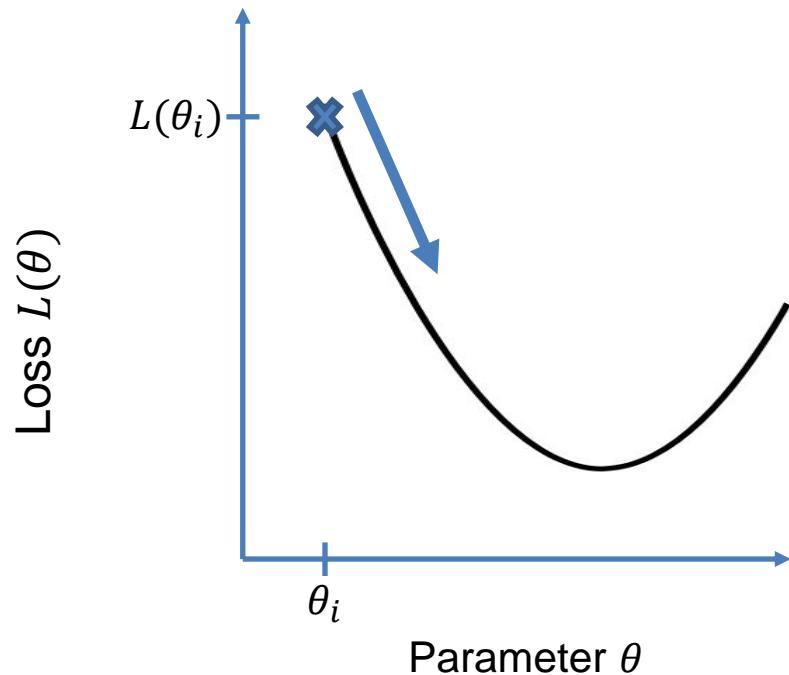
- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$



# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient   $\nabla L(\theta_i)$



Note: The gradient in matrix-vector form is:  $\mathbf{X}^T(\mathbf{\mu} - \mathbf{y})$

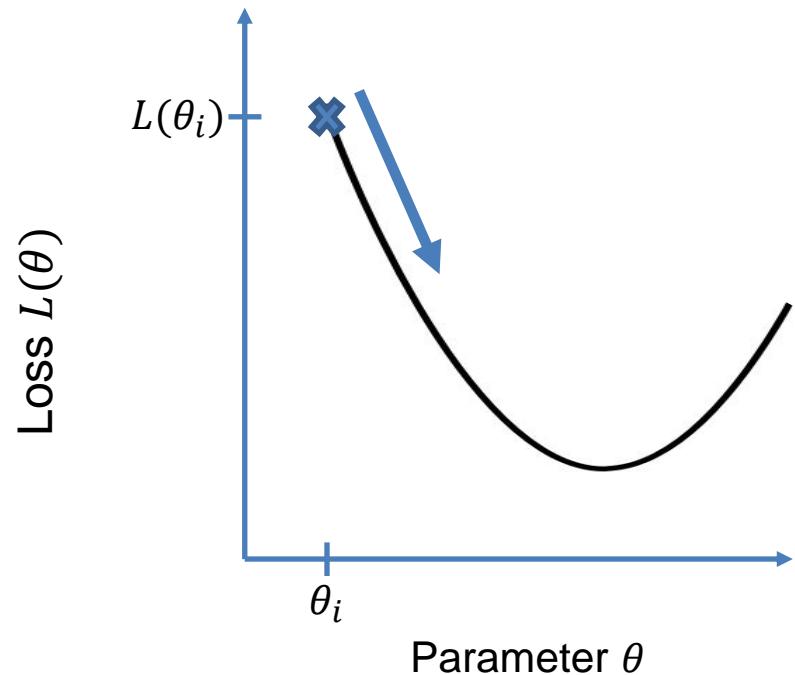
# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$

The Gradient for Logistic Regression:

$$\nabla L(\theta_i) = \sum_i (\mu(\mathbf{x}_i, \mathbf{w}) - y_i) \mathbf{x}_i$$

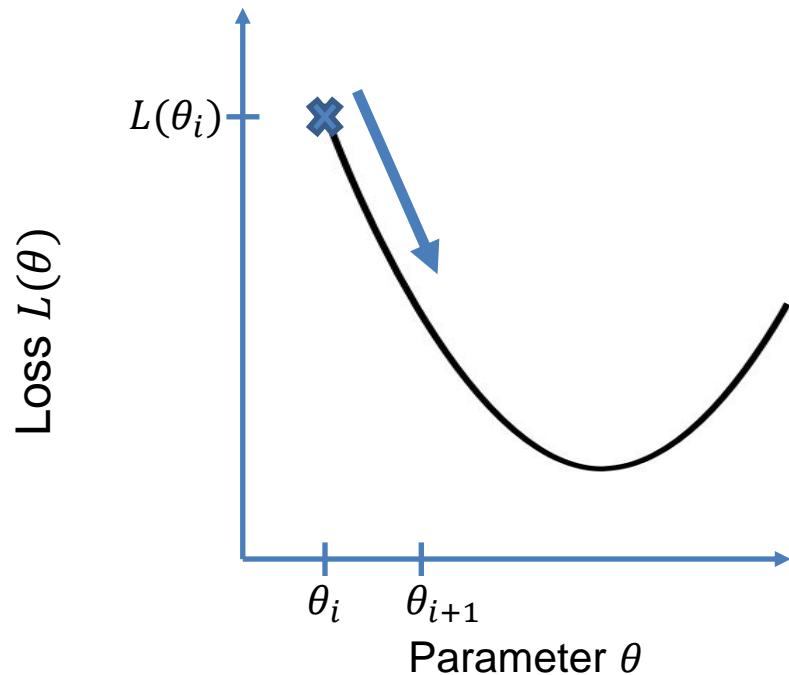


Note: The gradient in matrix-vector form is:  $\mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$

# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$



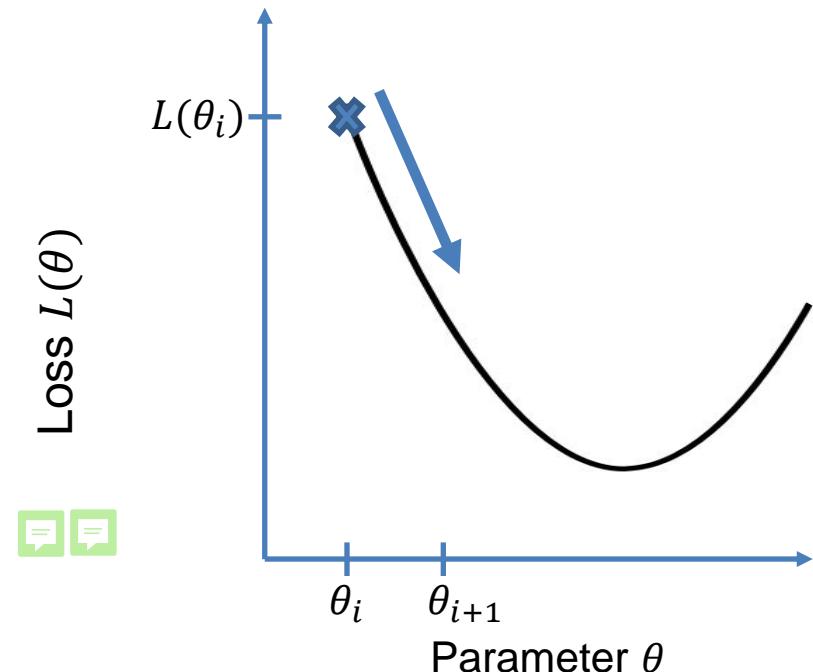
# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$

$\eta$  is called the **learning rate**

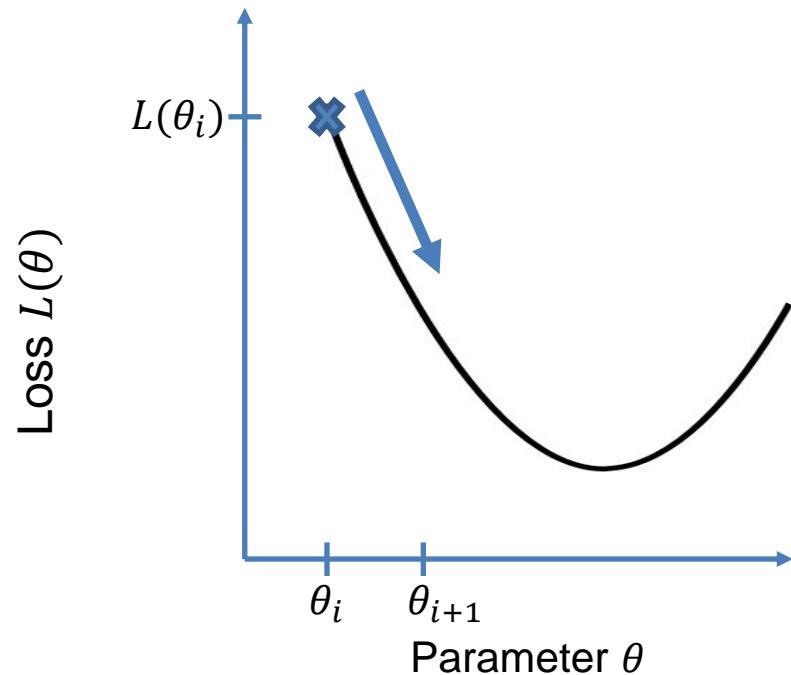
- If  $\eta$  is too large  
→ Overshooting the minimum
- If  $\eta$  is too small  
→ Minimum is not reached



# Example: Gradient descend

## Algorithm:

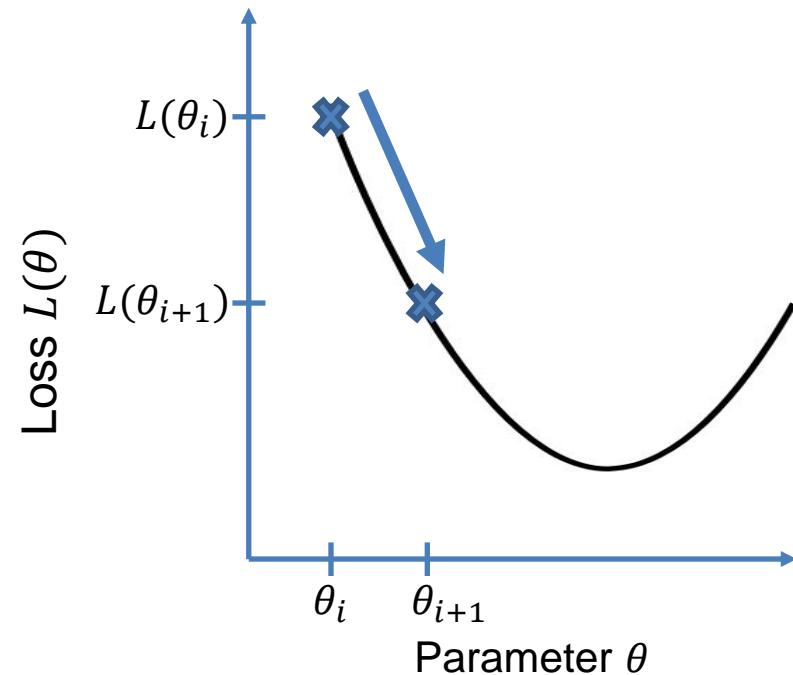
- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$



# Example: Gradient descend

## Algorithm:

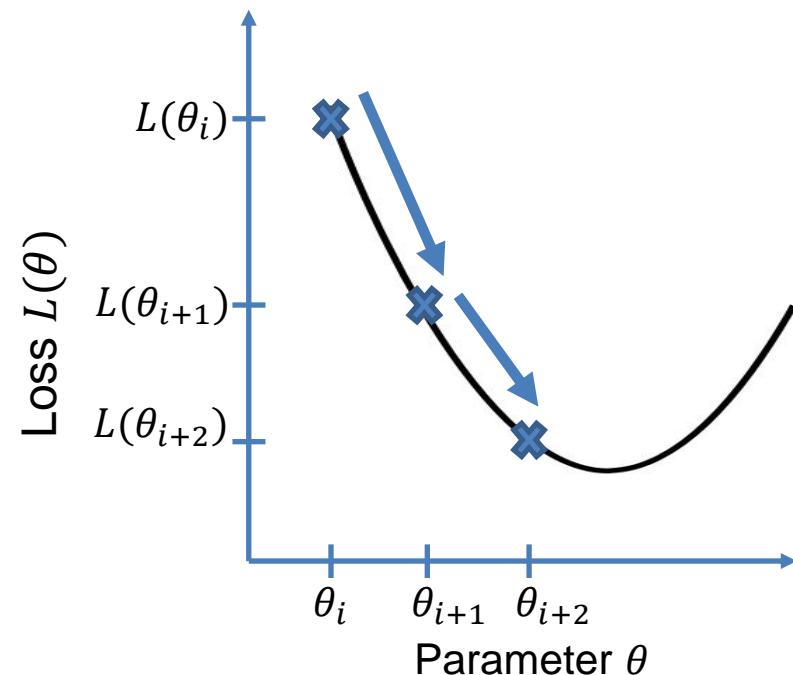
- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$
- 5  $i = i + 1$



# Example: Gradient descend

## Algorithm:

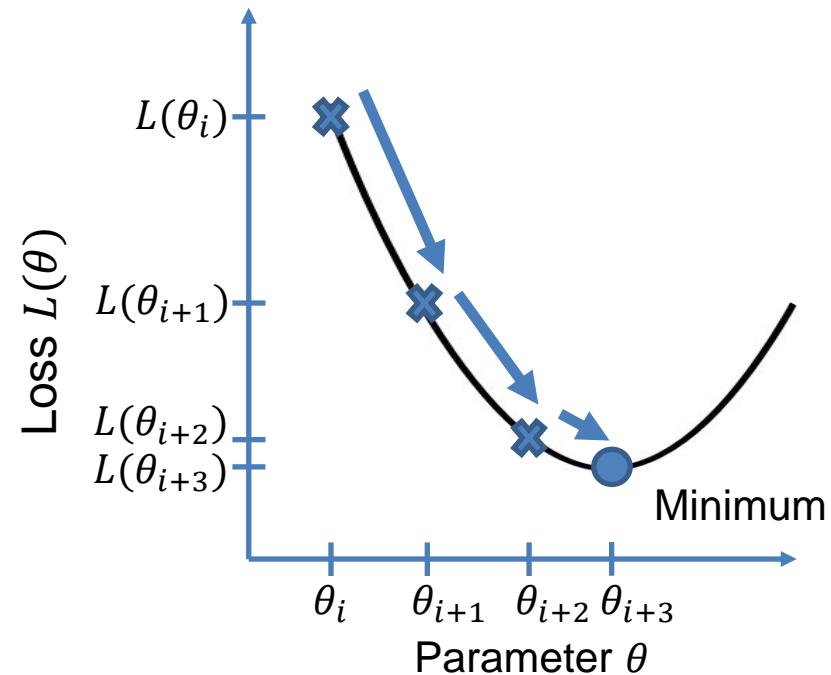
- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$
- 5  $i = i + 1$



# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$
- 5  $i = i + 1$

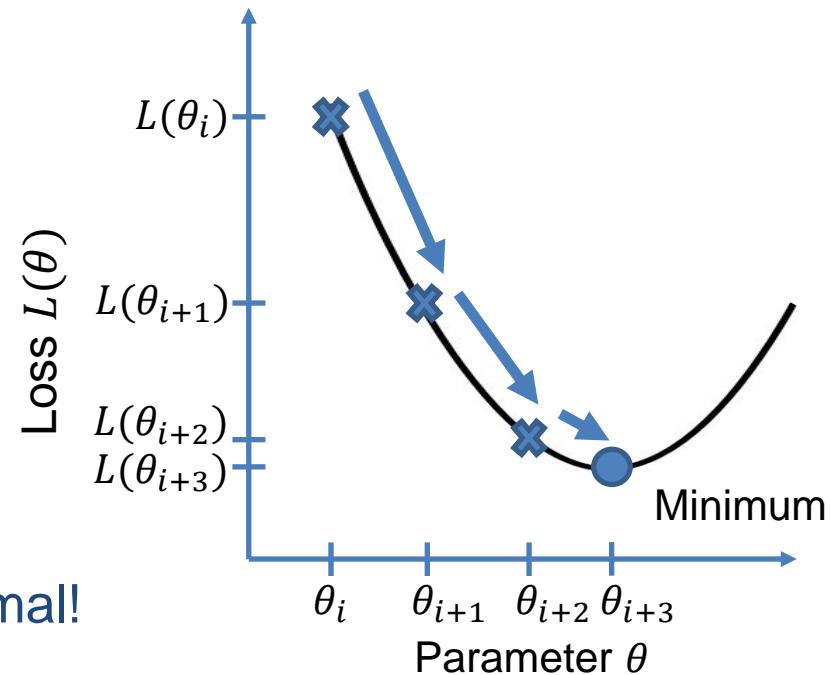


# Example: Gradient descend

## Algorithm:

- 1 Repeat for each iteration  $i$ :
- 2 Calculate loss  $L(\theta_i)$
- 3 Calculate gradient  $\nabla L(\theta_i)$
- 4  $\theta_{i+1} = \theta_i - \eta \cdot \nabla L(\theta_i)$
- 5  $i = i + 1$

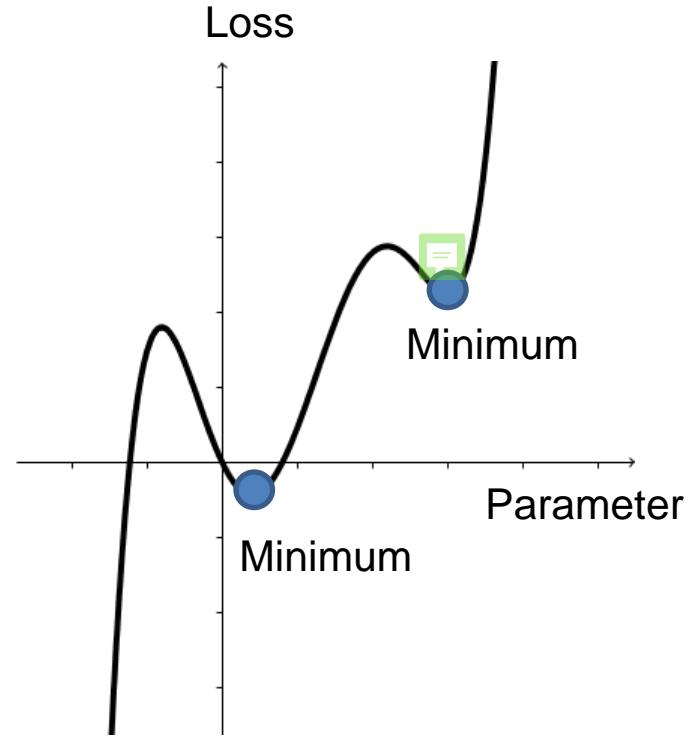
Repeat the process until the loss is minimal!



# General weaknesses of Gradient Descend

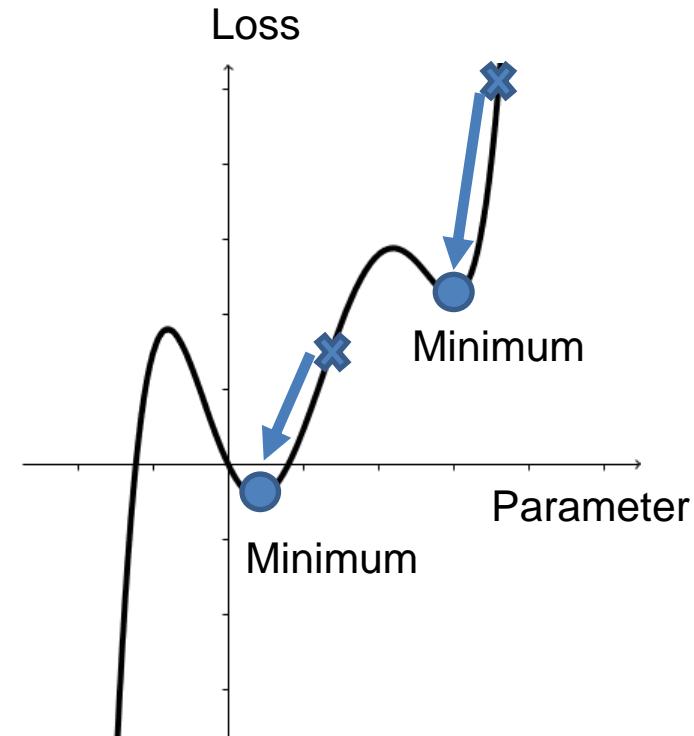
# General weaknesses of Gradient Descend

- Multiple local minima are common



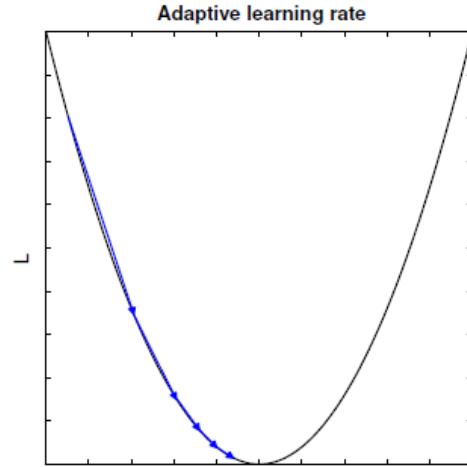
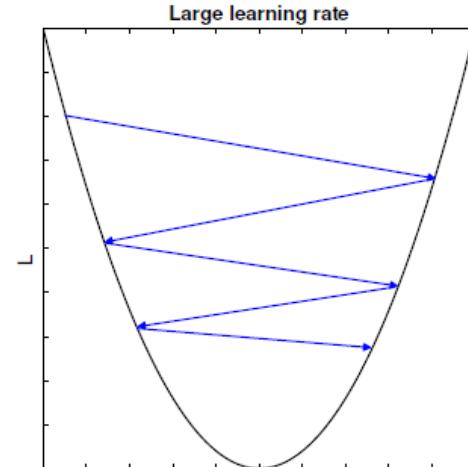
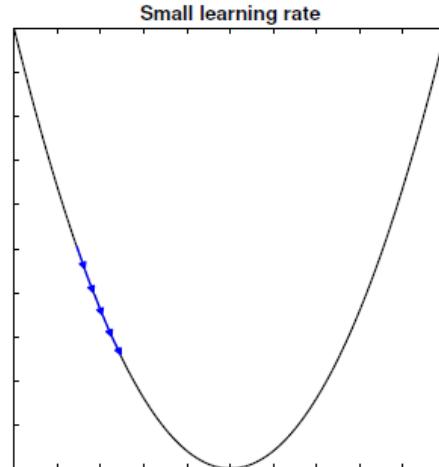
# General weaknesses of Gradient Descend

- Multiple local minima are common
- Into which the network converges to depends heavily on random initialization



# General weaknesses of Gradient Descend

- Multiple local minima are common
- Into which the network converges to depends heavily on random initialization
- Success depends on learning rate  $\eta$

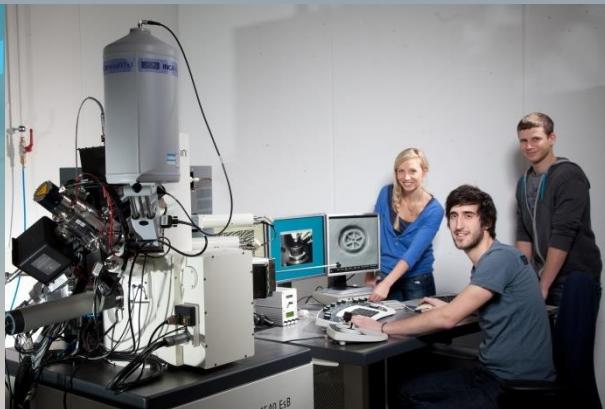
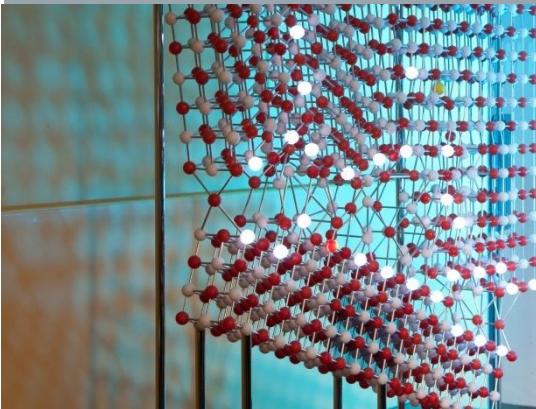


Thank you for listening!



# Machine Learning for Engineers

## Overfitting and Underfitting

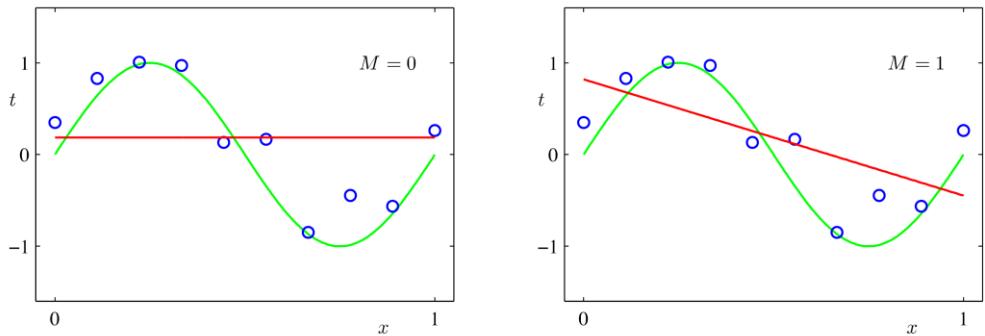


Bilder: TF / Malter

# Overfitting and Underfitting

## Optimal function vs. Estimated function

For  $M = 0$  and  $M = 1$  the function fails to model the data, as the chosen model is too simple (**Underfitting**)

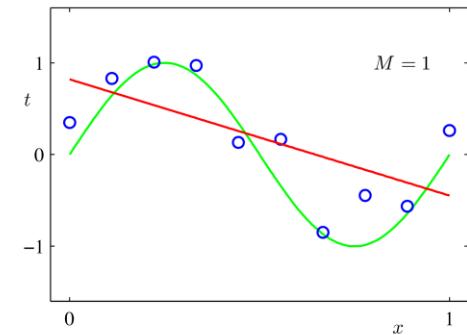
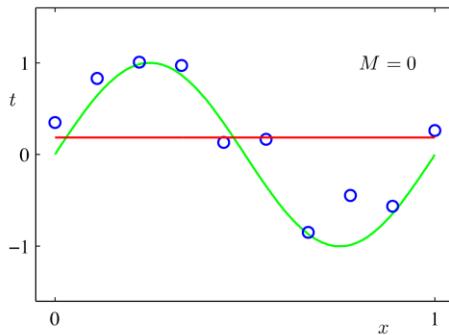


Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

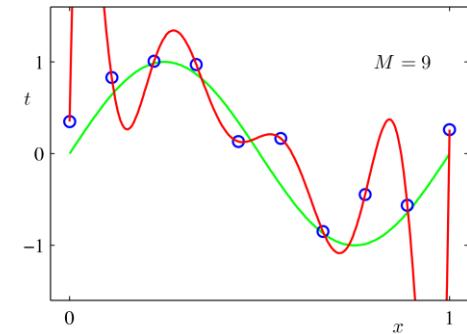
# Overfitting and Underfitting

## Optimal function vs. Estimated function

For  $M = 0$  and  $M = 1$  the function fails to model the data, as the chosen model is too simple (**Underfitting**)



For  $M = 9$  the function exactly models the given training data, as the chosen model is too complex (**Overfitting**)

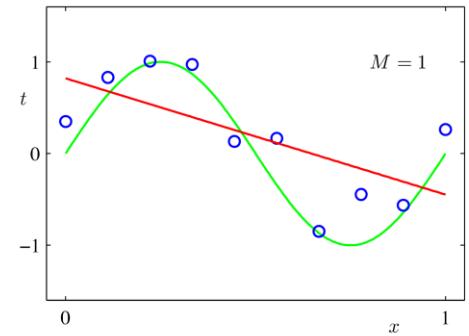
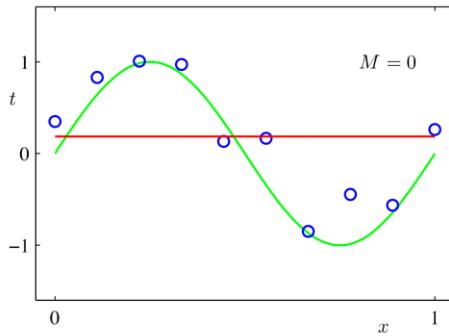


Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

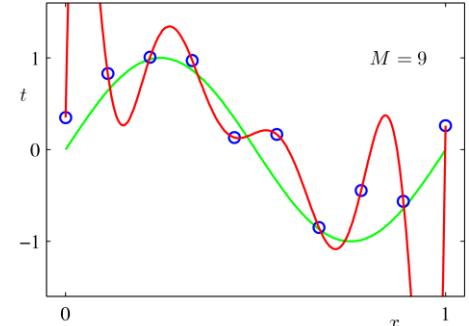
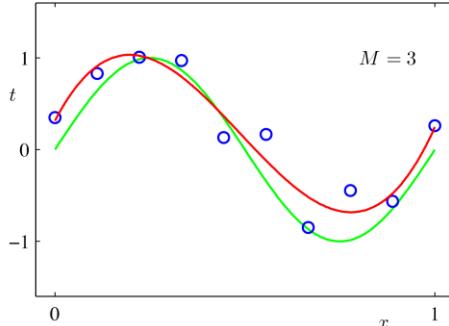
# Overfitting and Underfitting

## Optimal function vs. Estimated function

For  $M = 0$  and  $M = 1$  the function fails to model the data, as the chosen model is too simple (**Underfitting**)



For  $M = 9$  the function exactly models the given training data, as the chosen model is too complex (**Overfitting**)

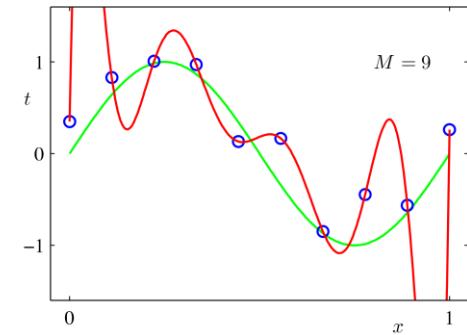
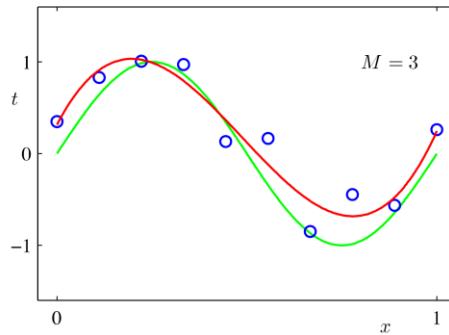
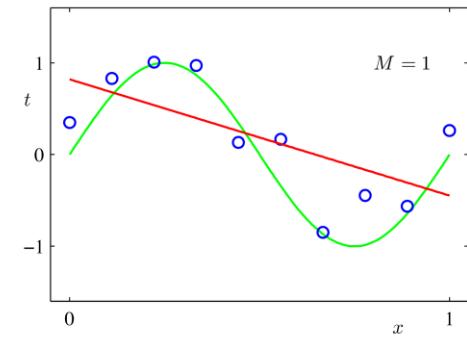
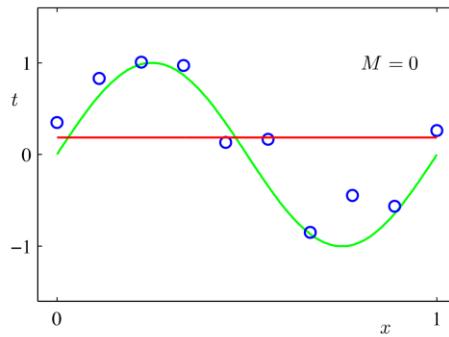


For  $M = 3$  the function closely matches the expected function

Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

# Overfitting and Underfitting

UNDERFITTING	OVERFITTING
Error on the training data <u>very high</u>	Error on the training data is <u>very low</u>
Testing error is <u>high</u>	Testing error is <u>high</u>
Examples are $M = 0$ and $M = 1$	Example is $M = 9$



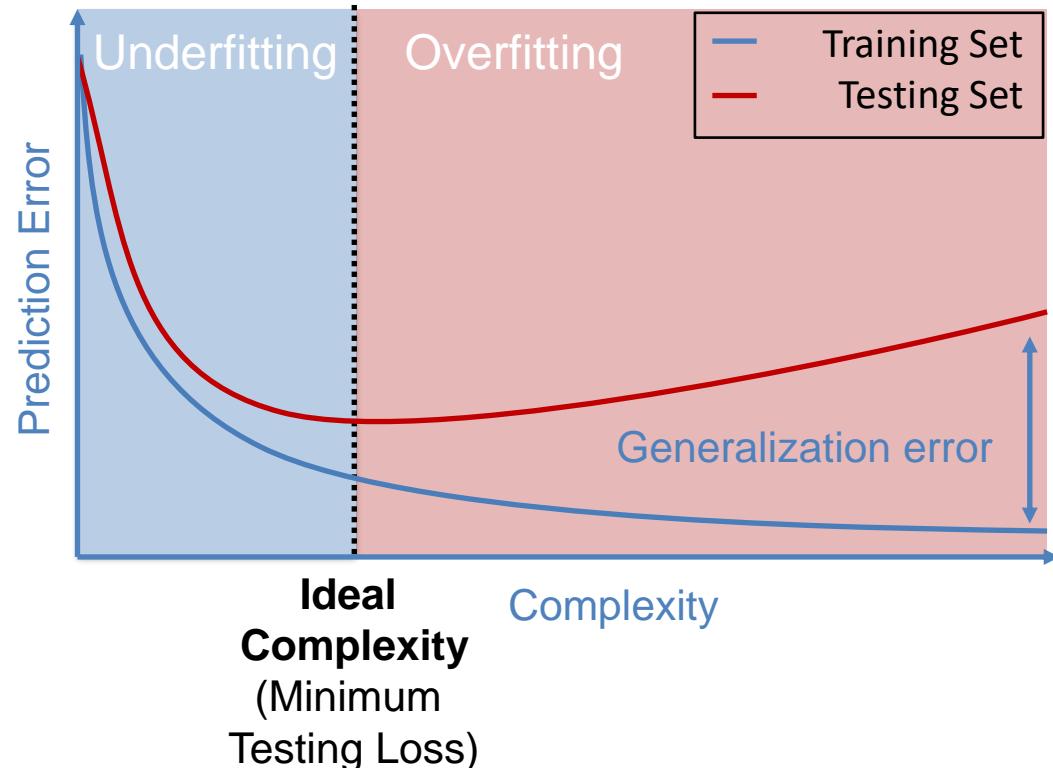
Source: Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, p.19

# Complexity vs. Generalization Error

Plotting  over all complexities, typically reveals a “sweet spot” (i.e. an ideal complexity)

The prediction error is affected by two kind of errors:

- Bias error
- Variance error

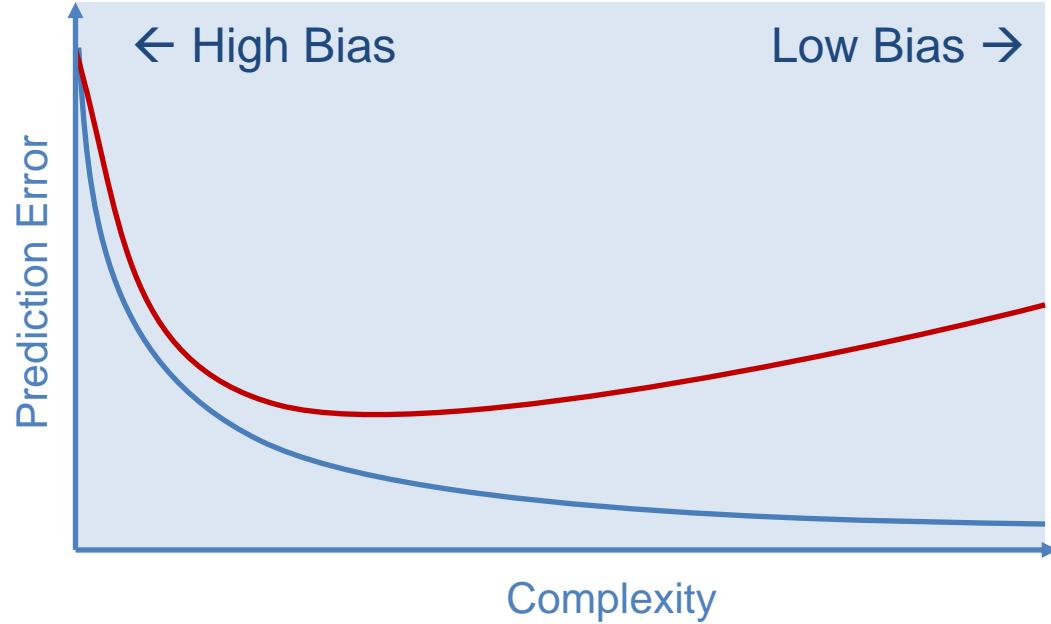


Note: Complexity does not mean parameters! It means the mathematical complexity! i.e. the ability of the model to capture a relationship!

# Complexity vs. Generalization Error

## Bias:

Error induced by simplifying assumptions of the model



# Complexity vs. Generalization Error

## Bias:

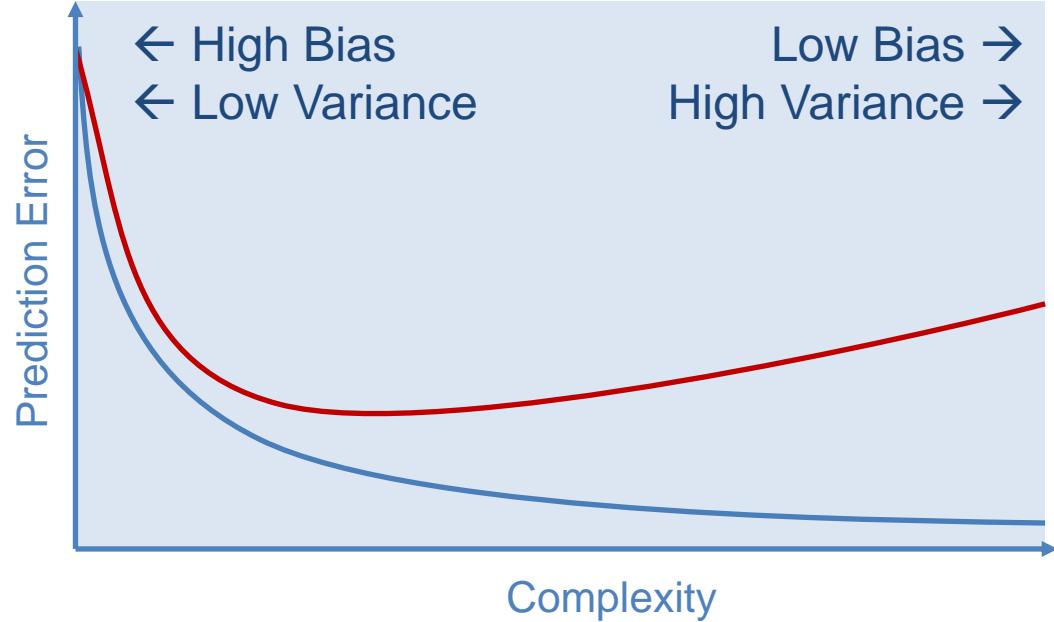


Error induced by simplifying assumptions of the model

## Variance:



Error induced by differing training data i.e. how sensitive is the model to “noise”



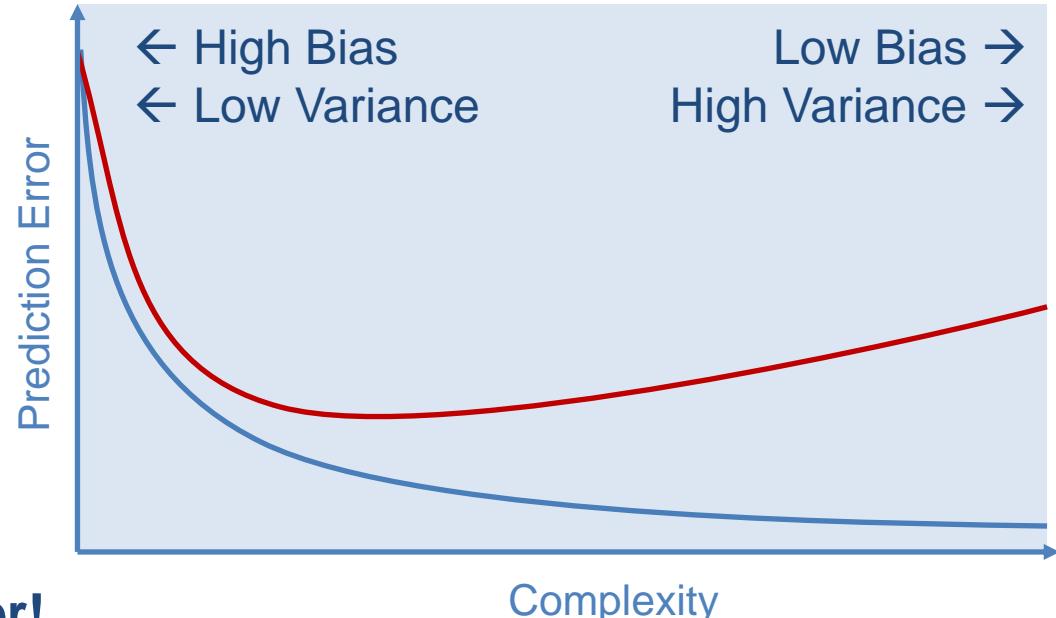
# Complexity vs. Generalization Error

## Bias:

Error induced by simplifying assumptions of the model

## Variance:

Error induced by differing training data i.e. how sensitive is the model to “noise”



**Both errors oppose each other!**

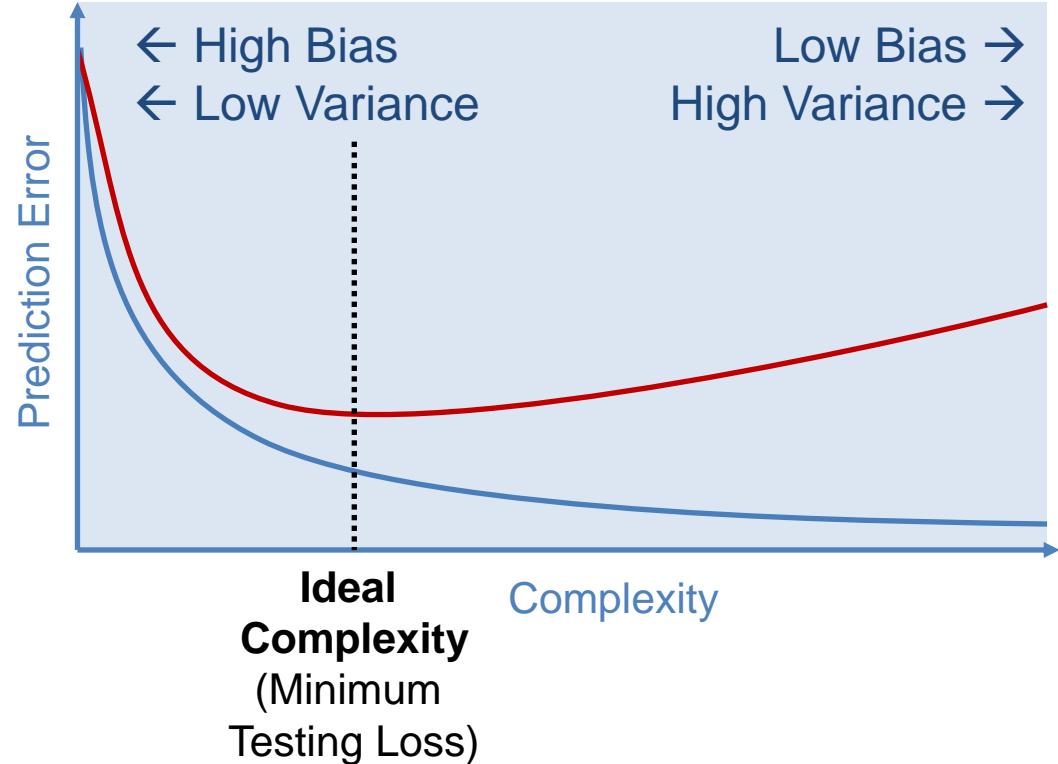
When reducing bias, you increase variance!  
(and vice versa)

# Complexity vs. Generalization Error

The **sweet spot** is located at the minimum of the test curve

It has **both** a low bias and a low variance!

This spot is – usually – found using **hyperparameter search**



# Hyperparameter search

**Approach:** Calculate the prediction error **on the validation set**<sup>1)</sup> and select the hyperparameters with the smallest loss

Example for hyperparameters:

- Amount of degrees in a polynomial
- Learning rate  $\eta$
- Different Basis Functions (See BFE)
- In Neural Networks: The amount of neurons

→ There exists multiple methods to solve this!

1) We use this instead of the test set. This is important! You only touch the test set for the **final** evaluation!

# Hyperparameter search methods

There exist multiple ways to find good hyperparameters...

- ... manual search (Trial-and-Error)
- ... random search
- ... grid search
- ... Bayesian methods
- ... etc.

→ In practice you typically use **Trial-and-Error & Grid-Search**

Note: Basically every known optimization technique can be used. Examples: Particle Swarm Optimization, Genetic Algorithms, Ant Colony Optimization...

# Hyperparameter search on the data splits

Typically you split the data into a **training and testing split**

**Hyperparameter search is forbidden on the testing split!**

# Hyperparameter search on the data splits

Typically you split the data into a **training and testing split**

**Hyperparameter search is forbidden on the testing split!**

## Solution:

Splitting the training data further into a **train split and validation split**

# Hyperparameter search on the data splits

Typically you split the data into a **training and testing split**

**Hyperparameter search is forbidden on the testing split!**

## Solution:

Splitting the training data further into a **train split and validation split**

If the dataset is large “enough” → 

# Hyperparameter search on the data splits

Typically you split the data into a **training and testing split**

**Hyperparameter search is forbidden on the testing split!**

## Solution:

Splitting the training data further into a **train split and validation split**

If the dataset is large “enough” → 

If the dataset is small ... → 

# Cross Validation

Problem of the **Static Split**:

The data<sup>1)</sup> is split into 80% train set and 20% validation set

If you are “unlucky”: The validation split is not representative!  
i.e. your prediction error is a **bad estimate** for the generalization performance!

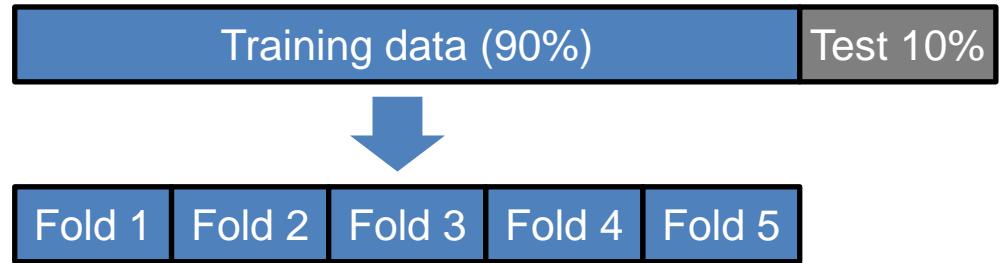
→ Solution: **Cross Validation!**

1) We assume the test data is already split and put away. A typical split of all the data is 80% train – 10% validation - 10% testing

# K-Fold Cross Validation

## Approach:

Split the training data into  $k$ -folds  
(in our example  $k = 5$ )



# K-Fold Cross Validation

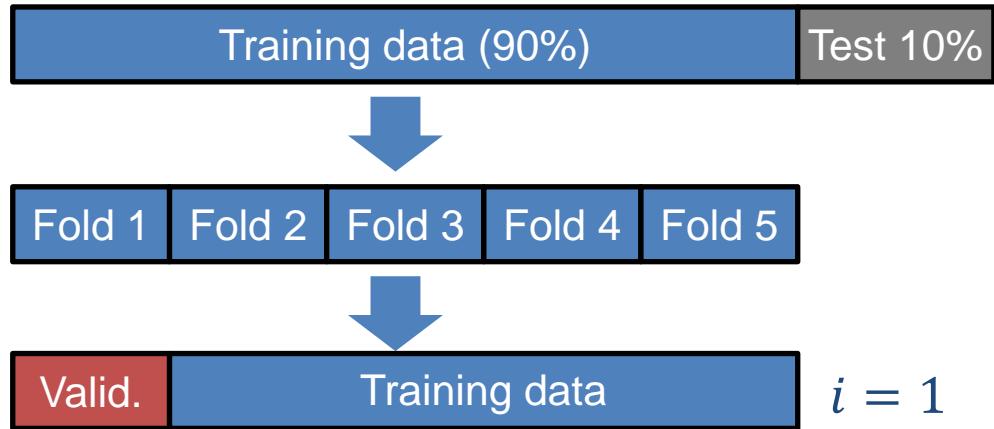
## Approach:

Split the training data into  $k$ -folds  
(in our example  $k = 5$ )

For each split  $i$ :

1. Train on all splits,  
except the  $i$ -th Fold
2. Evaluate on the  $i$ -th Fold

**Finally:** Average over all results



# K-Fold Cross Validation

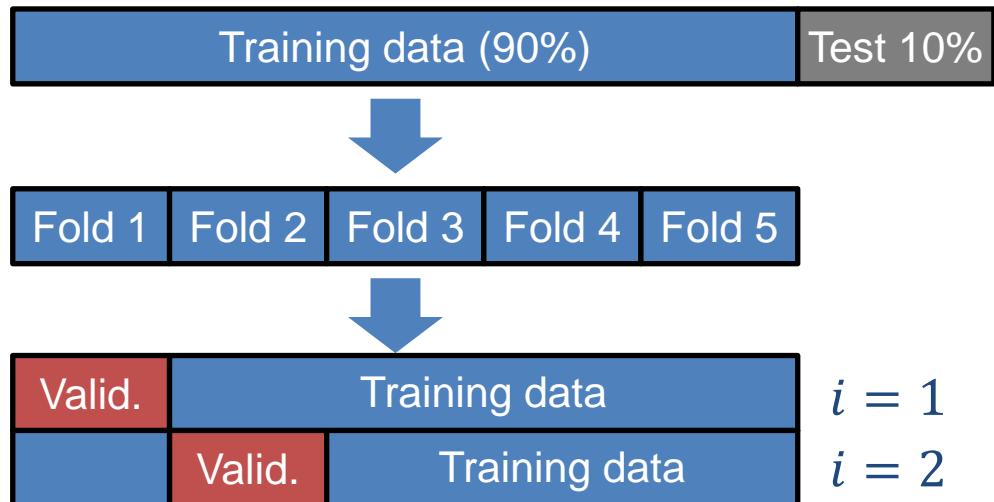
## Approach:

Split the training data into  $k$ -folds  
(in our example  $k = 5$ )

For each split  $i$ :

1. Train on all splits,  
except the  $i$ -th Fold
2. Evaluate on the  $i$ -th Fold

**Finally:** Average over all results



# K-Fold Cross Validation

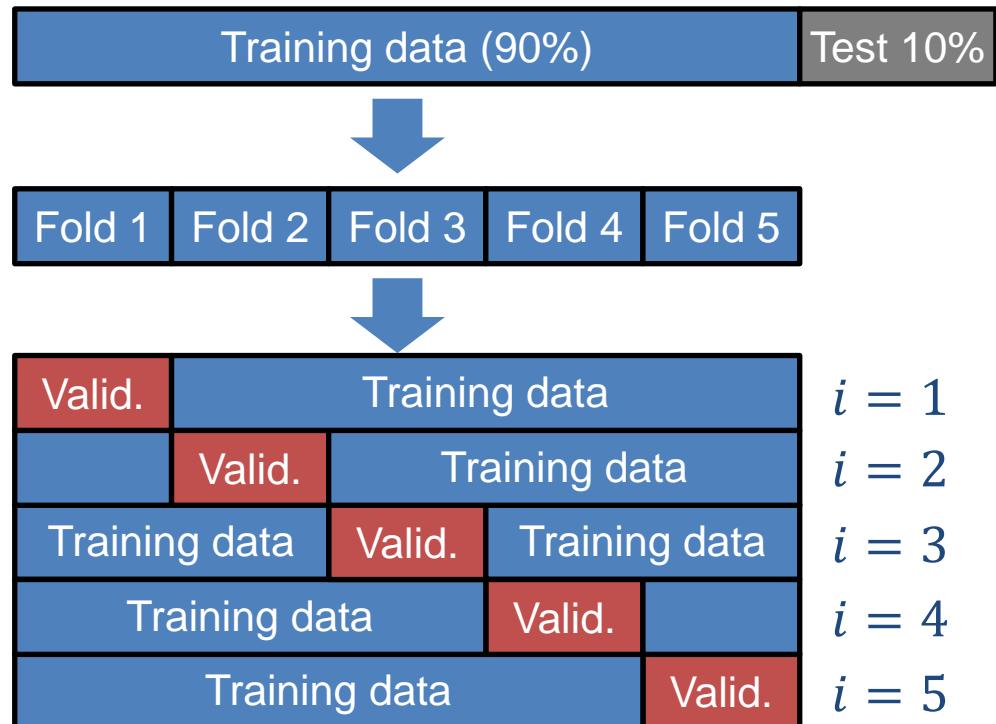
## Approach:

Split the training data into  $k$ -folds  
(in our example  $k = 5$ )

For each split  $i$ :

1. Train on all splits,  
except the  $i$ -th Fold
2. Evaluate on the  $i$ -th Fold

**Finally:** Average over all results



# Cross Validation

K-Fold Cross validation is the typical form of cross validation

When  $k=1$ , the process is called Leave-one out cross validation (or LOOCV)

There exist other variants as well, like:

- Grouped Cross Validation
- Nested Cross Validation
- Stratified Cross Validation

→ Each has its unique use-case!

Thank you for listening!



## 2. Linear and Logistic Regression

### ◀ Overfitting and Underfitting.

#### Questions

What does a regression task do?

- Map from an input to real-valued output
- Both are correct
- Find a function that fits the given data



**Correct!**

**Submit**

When is a model in Machine Learning considered linear?

- The regression function is linear w.r.t features (or basis functions ( $x^n$ ))
- The regression function is linear w.r.t weights ( $w_n$ ) of basis functions
- The result of model is linear i.e. a line



**Correct!**

**Submit**

What determines the set of model parameters in a linear regression?

- Weights of a model and noise variance
- Weights of a model
- Characteristics of noise



**Correct!**

Submit

When we learn a linear regression model, we learn the models.... ?

- Basis functions and set of parameters
- Basis functions
- Set of parameters



**Correct!**

Submit

How does the surface of the root sum of squared (RSS) look (for a linear regression problem)?

- It has one minimum point
- It has many minimum and maximum points
- It has many minimum points



**Correct!**

Submit

What does a classification task do?

- Learning a function that separate different classes
- Both are correct
- Mapping from input to the categorical outputs or labels



Correct!

Submit

What is the role of sigmoid function in logistic regression?

- Improving model accuracy
- Improving model performance
- Discretize the model output to two classes



Correct!

Submit

Please name the basic assumption for any Maximum Likelihood Estimation method!

1.

Also correct are:

- iid
- i.i.d.
- i.i.d
- independent, identically distributed
- independent identically distributed
- independent identically distributed (i.i.d)
- independent identically distributed (iid)
- independent identically distributed (i.i.d.)
- independent, identically distributed (i.i.d.)
- independent, identically distributed (i.i.d.)



Correct!

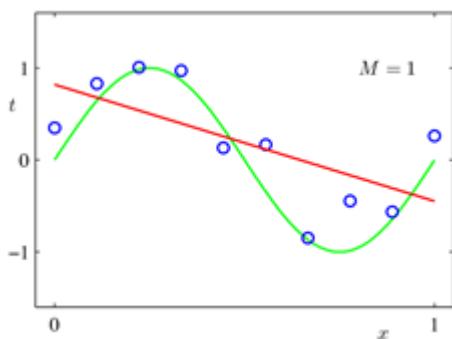
Please fill out the following text correctly!

Over- and Underfitting are common problems during training of machine learning models. Overfitting for example produces an extremely  ✓ [low] training error, but has an extremely  ✓ [high] testing error. The latter indicates, that the model cannot  ✓ [generalize] well and therefore should not be used in the real world. Although, there exist multiple ways to combat overfitting like regularization or just more - varying - training data, it is still an especially hard problem in deep learning. Underfitting on the other hand has a very  ✓ [high] training error and a  ✓ [high] testing error. In that sense, undefined models are not useable in the real world as well. Although, most of the time using more complex models solves this problem.

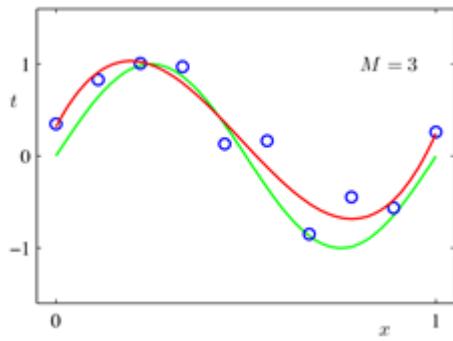


Correct!

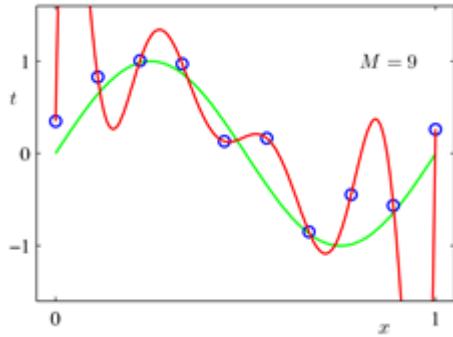
### Example 1



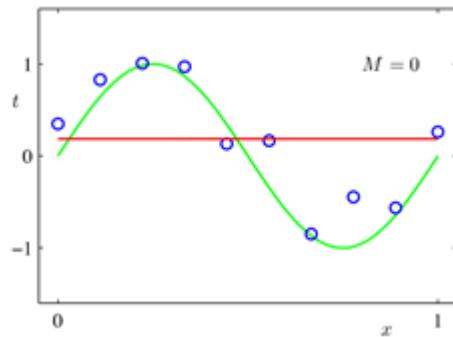
### Example 2



### Example 3



### Example 4



Please assign each example at least one possibility:

- Undefitting
- Overfitting
- Ideal Fitting

[Reset Ordering](#)

Overfitting

Underfitting

Ideal Fit



Example 1

Example 2

Example 3

Example 4



Correct!

Submit

## ◀ Overfitting and Underfitting

Add Comment

Sort Ascending

we

[we85bixe] - 31. Jul 2022

Hello together,

I got the same problem as Jonathan and I really tried a lot of words.  
Could you please help me?

Thank you.



Löhr, Tim [il34ifyn] - 26. May 2022

Thanks [xy39tuqi] for answering the question :). Jonathan did you get it right now? Otherwise let me know and I will give you a more detailed explanation.

Kind regards,  
Tim

**xy**

[xy39tuqi] - 23. May 2022

Hi Jonathan,

the answer was mentioned in the lecture, that if something is too specific, it cannot be generalized anymore.

Regards

**JQ**

Quaß, Jonathan [uw02agon] - 20. May 2022

Hi,

I don't get what missing word has to be put into this sentence:

"The latter indicates, that the model cannot **X** well and therefore should not be used in the real world."

I tried: perform, estimate, predict, do ... but none seemed to be right, so I guess, I'm missing what it wants to tell me.

Could you please clarify for me?

Thank you,

best regards

Jonathan



Löhr, Tim [il34ifyn] - Last edited on 04. May 2022

Hello [ta56tigo],

thank you for your interesting question. The answer to that is that an independently and identically distributed assumption (i.i.d.) is normally the basic assumption yes, but it is not necessarily the case. We make this assumption because if we do so, we can take the log and just add up all the log likelihoods. It makes it very feasible for computation. If we don't make the i.i.d. assumption, we cannot take the log and need to use e.g. Monte-Carlo sampling for approximation. So i.i.d. just makes the MLE and MAP computation much easier, because we are then allowed to use the closed-form solution for computation.

I hope this clarifies your questions, otherwise don't hesitate to ask more questions. Questions are always warmly welcomed!

Kind regards.

**ta**

[ta56tigo] - 04. May 2022

hello, is it possible to say the answer "Please name the basic assumption for any Maximum Likelihood Estimation method! "

is it independency and identically distributed?

## 2.Linear and Logistic Regression kérdések

SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables. SVM is based on geometrical properties of the data while logistic regression is based on statistical approaches

When is a model in Machine Learning considered linear?

67.slide, a weighsteknek kell linearnak maradnia, az x lehet bármilyen, pl 71.dia

What determines the set of model parameters in a linear regression?

39.slide, epsilon, epsilon noise variance-nak kell híjni, és nem standard deviaton of nose-nak

When we learn a linear regression model, we learn the models.... ?

Ha basis function expansion van, akkor már nem linear model (69. slide), basis functiont szerintem nem így kell kiszámítani (bár hyperparameter search-nél hyperparaméterek számít ez is, ez volt a slideon), ha basis function expansion van, akkor meg már nem linear regressionnek hívják

one condition:

The resulting vector has to have a constant “length” i.e. linear in respect to the parameters! ->  
That is the reason, why polynomial regression can be considered linear: de polynomial regression-nek hívja

How does the surface of the root sum of squared (RSS) look (for a linear regression problem)?

szerintem ez residual sum of squares akart lenni, az meg 56. slide

What does a classification task do?

"Learning a function that separate different classes": 86. slide

What is the role of sigmoid function in logistic regression?

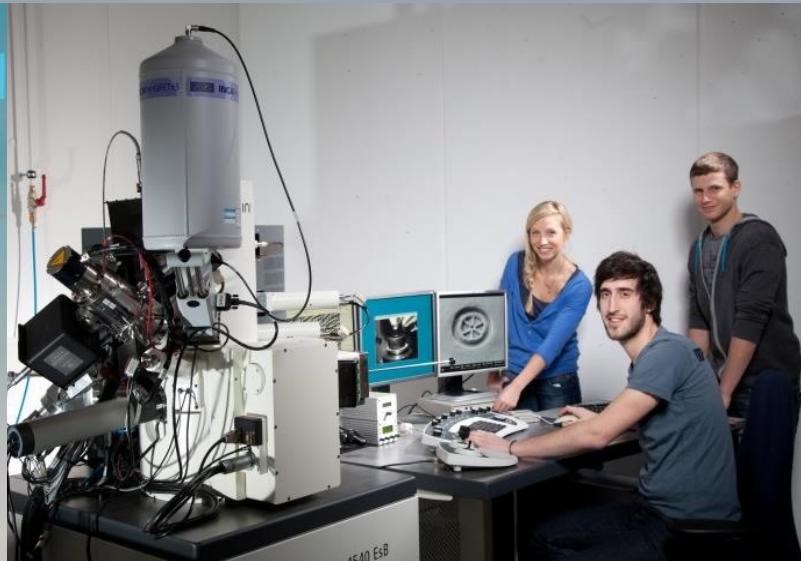
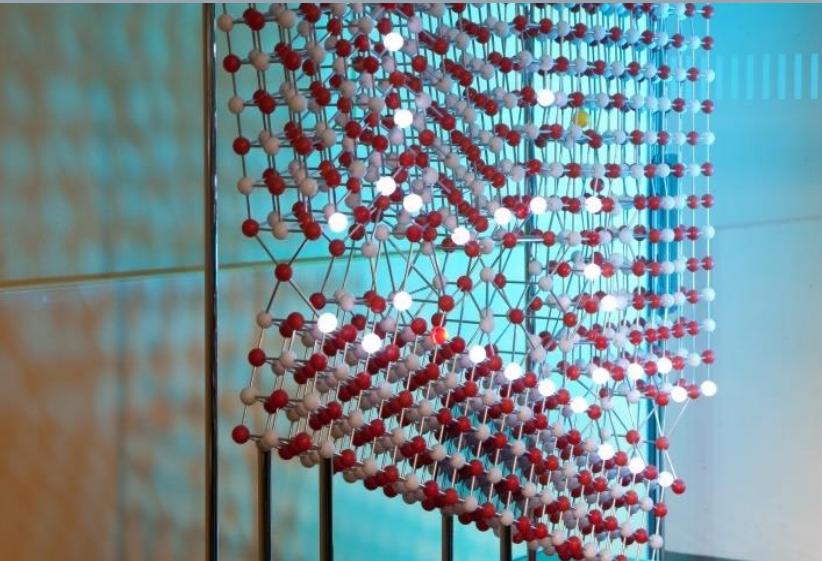
98. slide, a sigmoid function az a  $u(x,w)$

Please name the basic assumption for any Maximum Likelihood Estimation method!

45 -46.slide

# Machine Learning for Engineers

## Support Vector Machines – Problem Statement



Bilder: TF / Malter

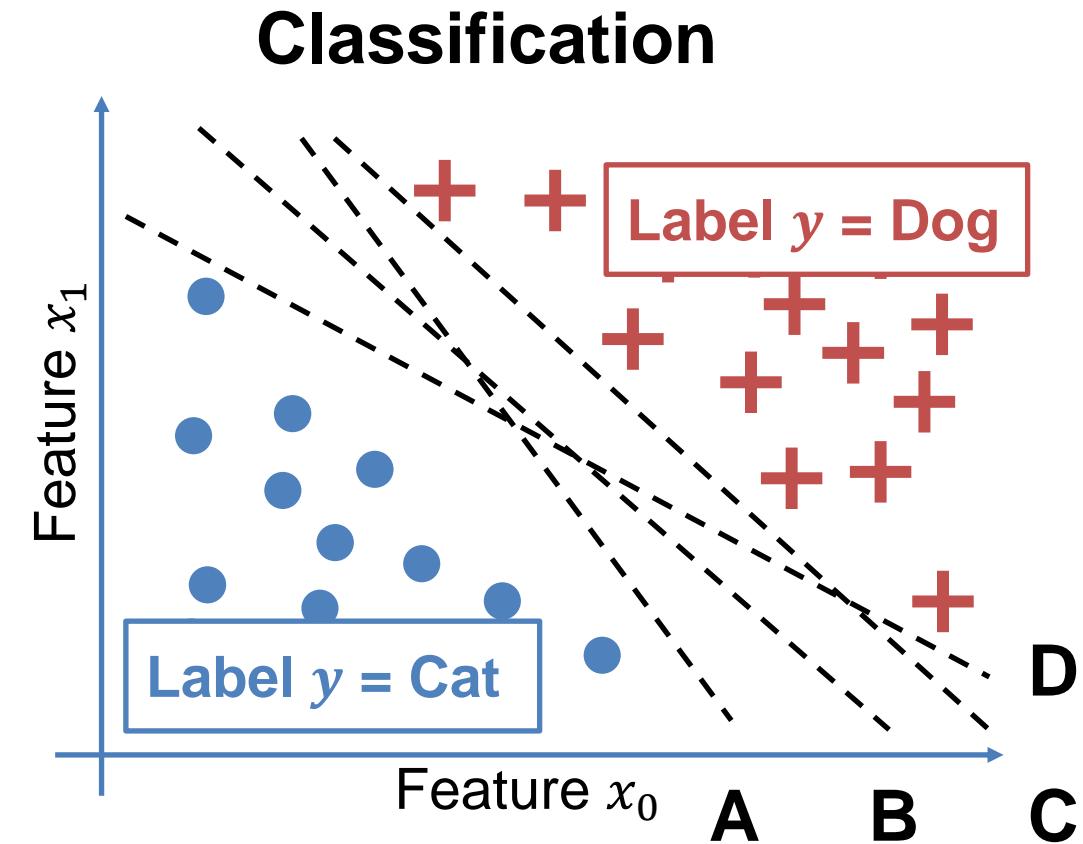
# Recap: Logistic Regression

**Goal:** Find a linear decision boundary separating the two classes

**Problem:** Multiple linear decision boundaries solve the problem with equal accuracy

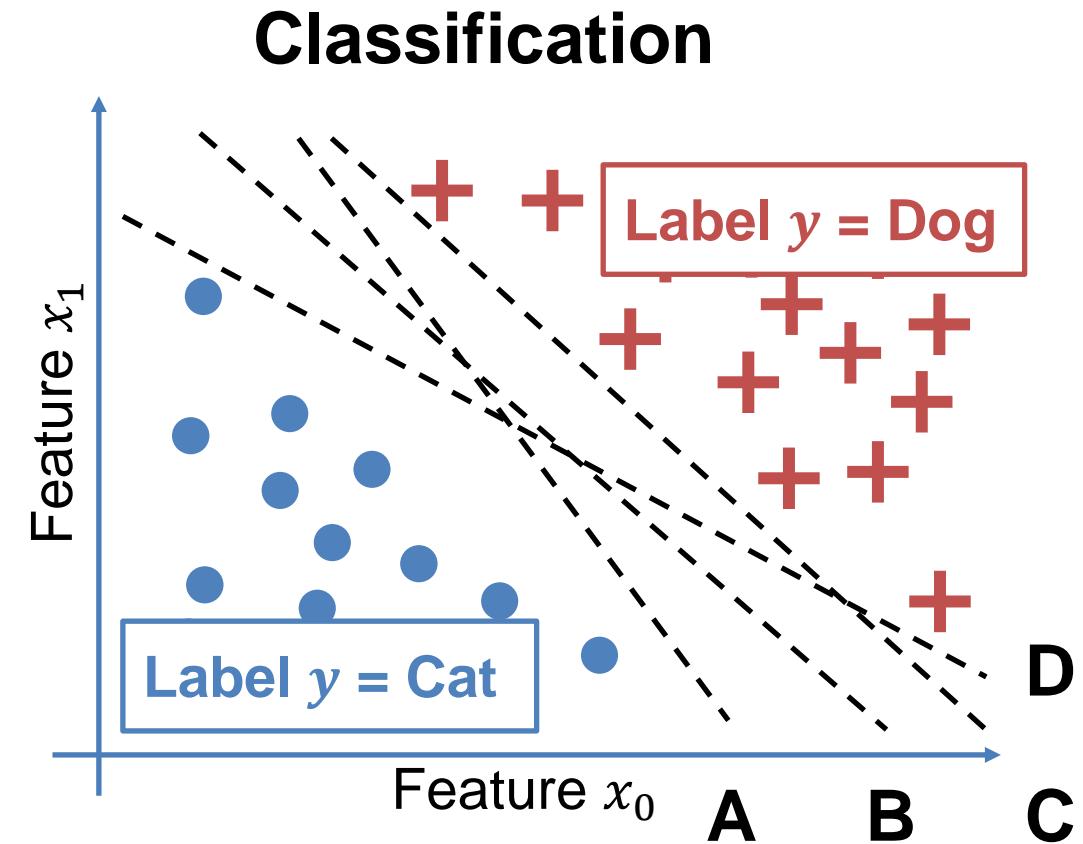
**Question:** Which one is the best?

A    B    C    D



# Intuition behind Support Vector Machines

- A is closer to Dog at the top and to Cat at the bottom
- B is far away from both Dog and Cat throughout the boundary
- C always is closer to Dog than Cat
- D almost touches Dog at the bottom and Cat at the top

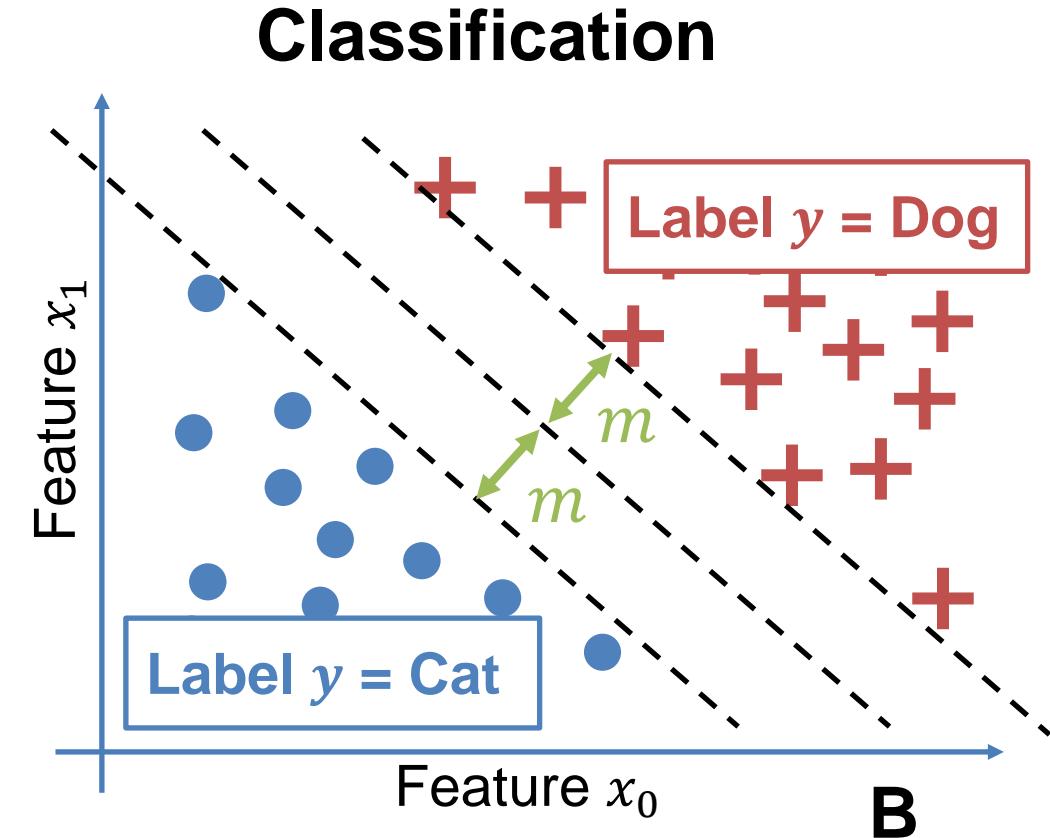


# Intuition behind Support Vector Machines

**B** is far away from both Dog and Cat throughout the boundary

To put this in more mathematical terms, we say it has the largest margin  $m$

**Updated Goal:** Find linear decision boundary with the largest margin.



# Mathematical Model for Decision Boundary

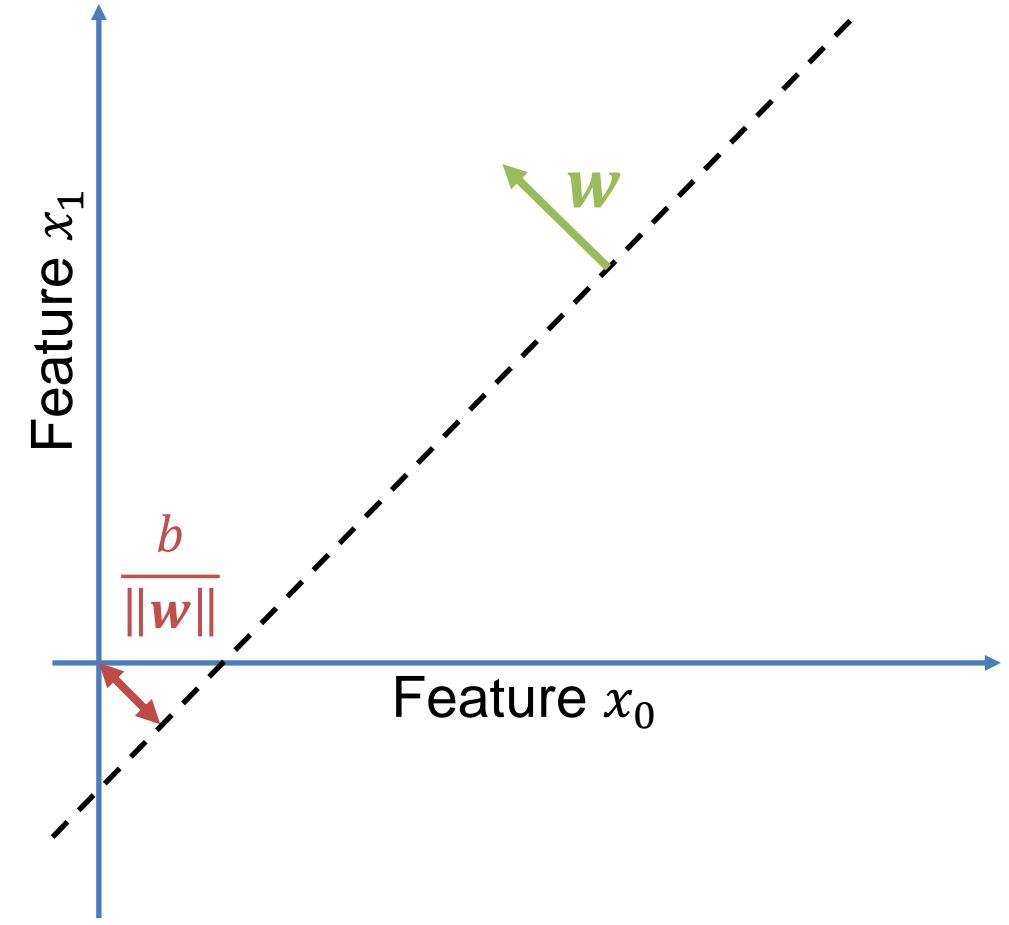
Mathematically, any hyperplane is given by the equation

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

$\mathbf{w}$  is the normal vector

$\mathbf{x}$  is any arbitrary feature vector

$\frac{b}{\|\mathbf{w}\|}$  is the distance to the origin



# Mathematical Model for Margins

Decision boundary is given by

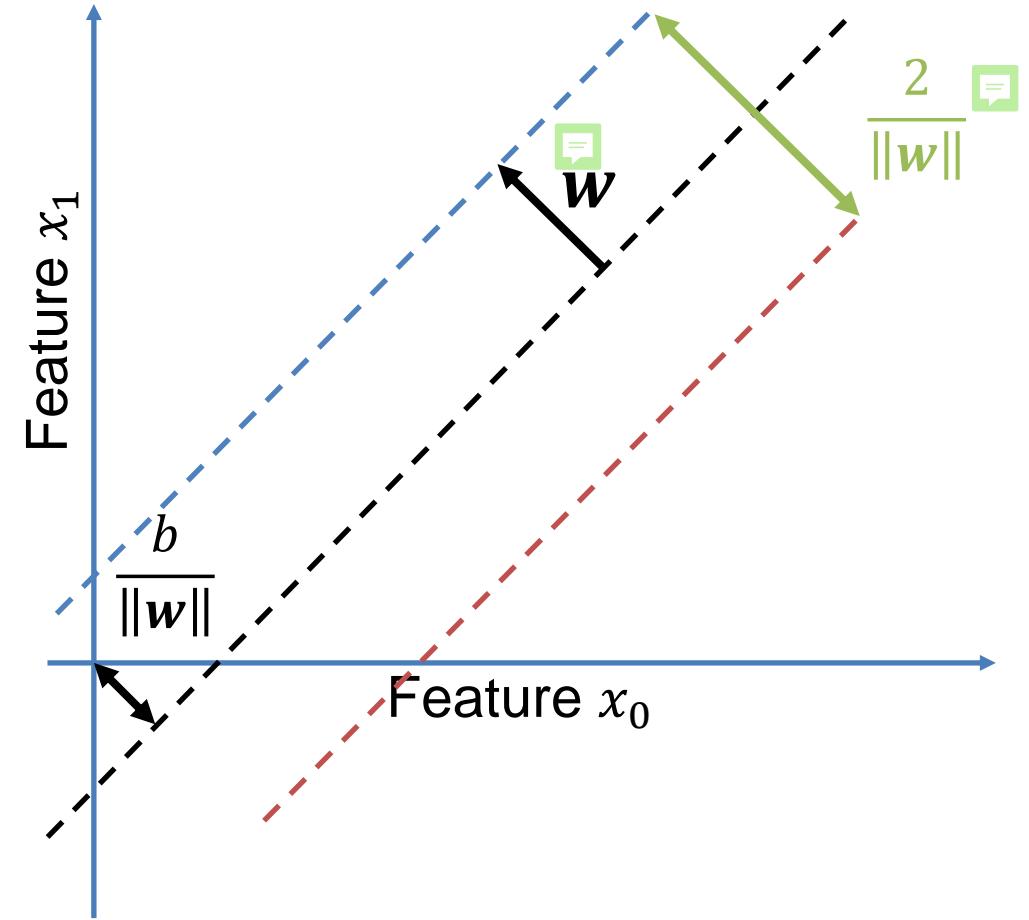
$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

Furthermore, by definition, the margin boundaries are given by

$$\mathbf{w} \cdot \mathbf{x} - b = +1$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1$$

$\frac{2}{\|\mathbf{w}\|}$  is thus the distance between the two margins → maximize



# Mathematical Model for Classes



Two-class problem  $y_1, \dots, y_n = \pm 1$

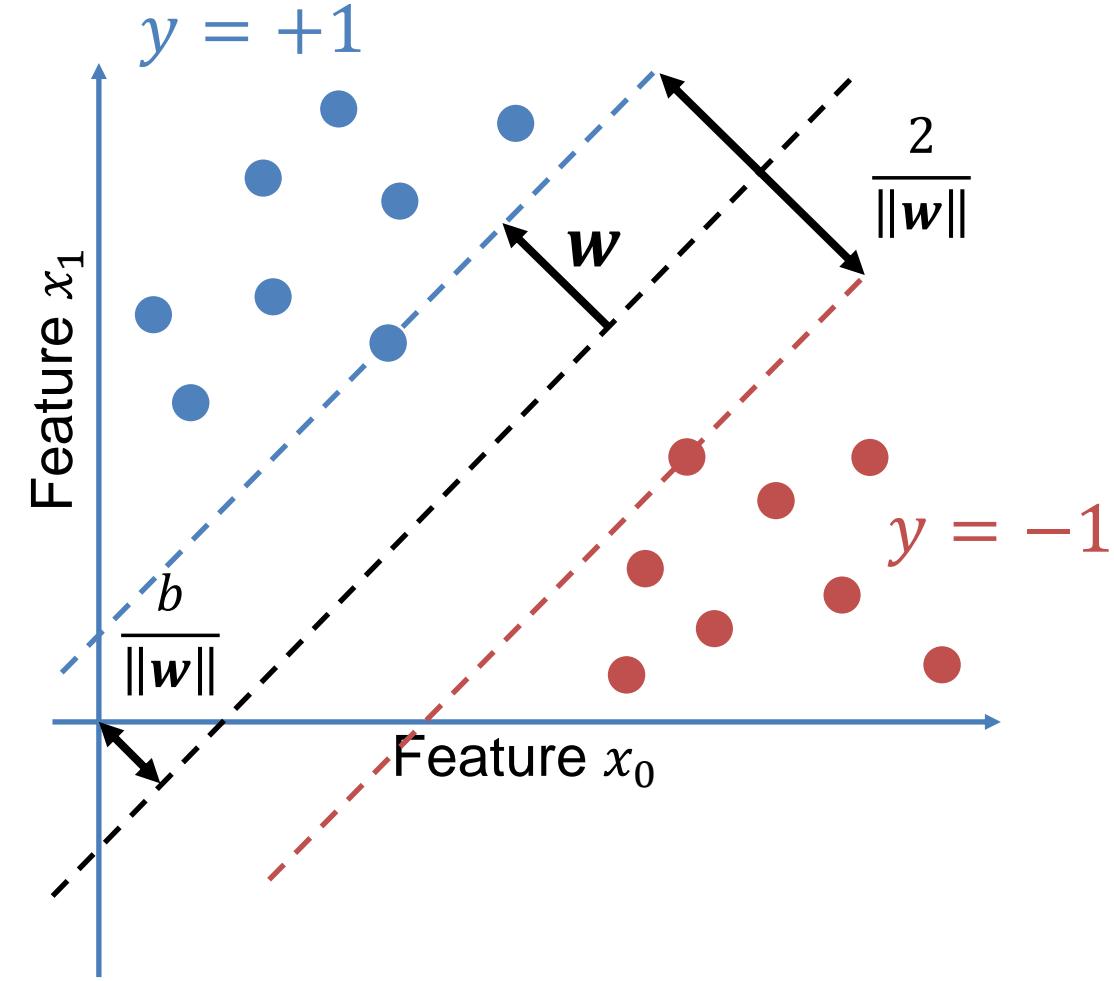
All training data  $x_1, \dots, x_n$  needs to be correctly classified outside margin

$$w \cdot x_i - b \geq 1 \quad \text{if } y_i = +1$$

$$w \cdot x_i - b \leq -1 \quad \text{if } y_i = -1$$

Due to the label choice, this can be simplified as

$$y_i(w \cdot x_i - b) \geq 1$$



# Optimization of Support Vector Machines

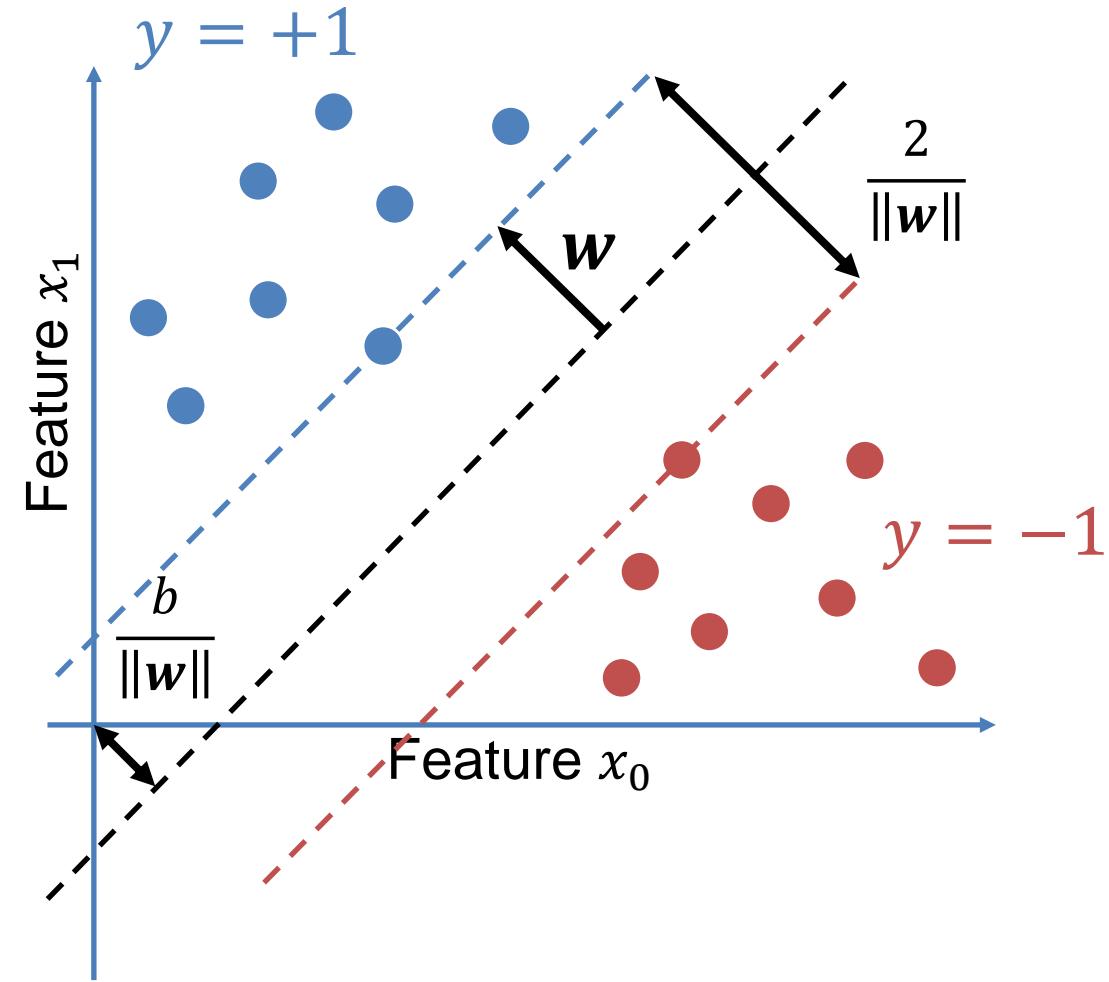
This leads us to a constrained optimization problem.

1. Maximizing the margin  $\frac{2}{\|w\|}$ , which is equal to minimizing  $\frac{1}{2} \|w\|^2$

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

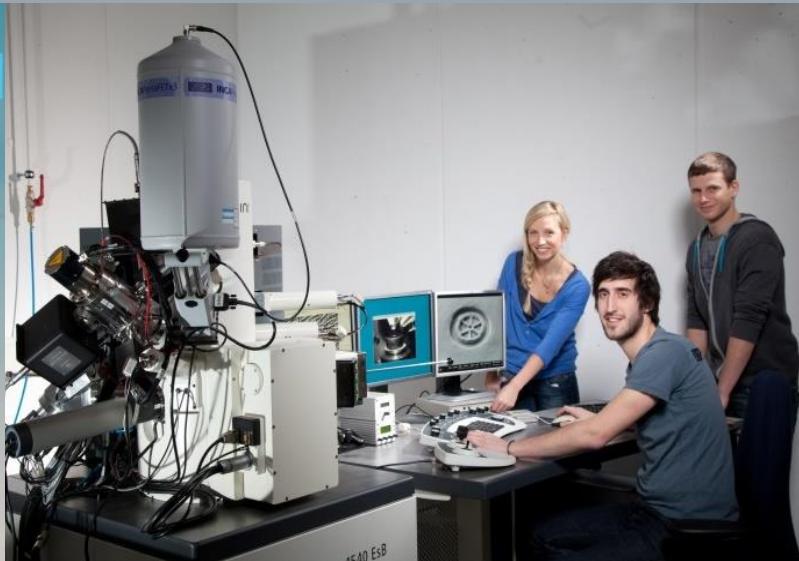
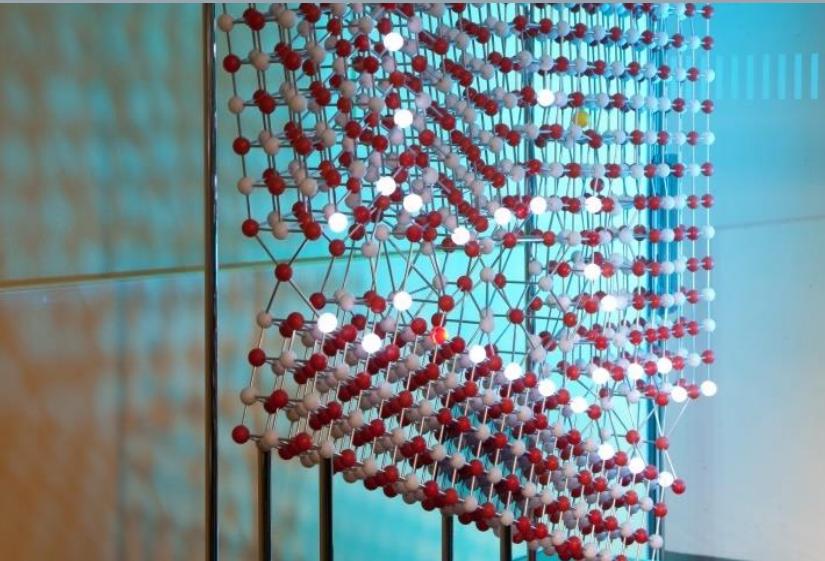
2. Subject to no misclassification on the training data

$$y_i(w \cdot x_i - b) \geq 1$$



# Machine Learning for Engineers

## Support Vector Machines – Optimization



Bilder: TF / Malter

# Optimization of Support Vector Machines

In the previous section, we derived the constrained optimization problem for Support Vector Machines (SVMs):

$$\begin{aligned} & \min_{w,b} \frac{1}{2} \|w\|^2 \\ \text{s. t. } & y_i(w \cdot x_i - b) \geq 1 \end{aligned}$$

This is a *quadratic programming* problem, minimizing a quadratic function subject to some inequality constraints.

→ We can thus introduce Lagrange multipliers

# Introduction of Lagrange Multipliers

For each of the  $n$  inequality constraints, introduce  $\alpha_i$  Lagrange multiplier<sup>1)</sup>:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i - b) - 1]$$

The Lagrange multipliers need to be maximized, resulting in the following derived optimization problem:

$$\min_{\mathbf{w}, b} \max_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$$

1) While not relevant for the exam, a more detailed derivation is available in C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006, Appendix E.

# Primal versus Dual Formulation

This optimization problem is called the **primal** formulation, with solution  $p^*$ :

$$p^* = \min_{\mathbf{w}, b} \max_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$$

Alternatively, there's also the **dual** formulation, with solution  $d^*$ :

$$d^* = \max_{\boldsymbol{\alpha}} \min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$$

Since **Slater's condition** holds for this **convex optimization problem**, we can guarantee that  $p^* = d^*$  and solve the dual problem instead.

→ We can thus first solve the minimization problem for  $\mathbf{w}$  and  $b$  

# Solution using Partial Derivatives

We now want to minimize the following function for  $w$  and  $b$ :

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i(w \cdot x_i - b) - 1]$$

At the minimum, we know that the derivatives  $\frac{\partial \mathcal{L}}{\partial w}$  and  $\frac{\partial \mathcal{L}}{\partial b}$  need to be zero:

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^n \alpha_i y_i x_i \quad \text{💡} \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^n \alpha_i y_i \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

# Solution using Partial Derivatives

We then eliminate  $w$  and  $b$  by inserting both equations into the function:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

This directly leads us to the remaining maximization problem for  $\alpha$ :

$$\begin{aligned} & \max_{\alpha} \mathcal{L}(\mathbf{w}, b, \alpha) \\ \text{s. t. } & \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

This quadratic programming problem can be solved using sequential minimal optimization, but which is not a topic of this lecture.

# The Karush–Kuhn–Tucker Conditions

The given problem fulfills the Karush-Kuhn-Tucker conditions<sup>1)</sup>:

$$\alpha_i \geq 0$$

$$y_i(w \cdot x - b) - 1 \geq 0$$

$$\alpha_i[y_i(w \cdot x - b) - 1] = 0$$

Based on the last condition, we can derive that either  $\alpha_i = 0$  or  $y_i(w \cdot x - b) = 1$  for each of the training point.

💡 When  $\alpha_i$  is non-zero, the training point is on the margin and thus a so-called “support vector”.

1) While not relevant for the exam, a more detailed derivation is available in C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006, Appendix E.

# Optimization Summary

The dual formulation leads to a quadratic programming problem on  $\alpha$ :<sup>1)</sup>

$$\max_{\alpha} \mathcal{L}(\mathbf{w}, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

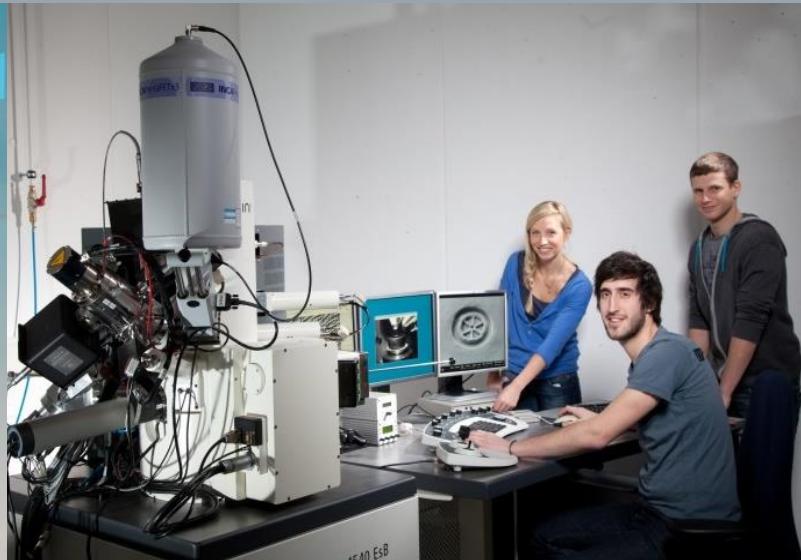
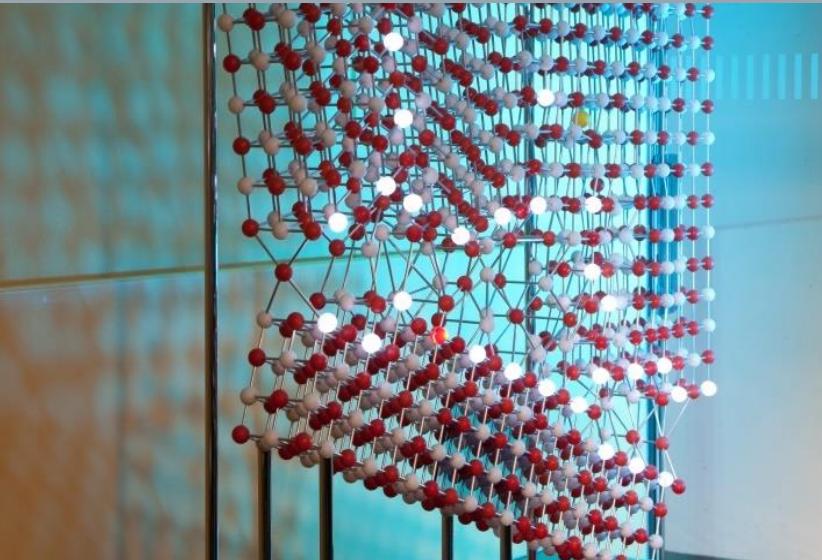
The solution  $\alpha^*$  can then be used to compute  $\mathbf{w}$  and make predictions on previously unseen data points  $\mathbf{x}$ :<sup>2)</sup>

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn} \left( \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} \right)$$

- 1) This is simplified for the purposes of this summary. For the additional constraints that are also required refer to slide 14
- 2) Remember the equation for  $\mathbf{w}$  resulting from the derivative in slide 13

# Machine Learning for Engineers

## Support Vector Machines – Non-Linearity and the Kernel Trick



Bilder: TF / Malter

# Recap: Basis Functions

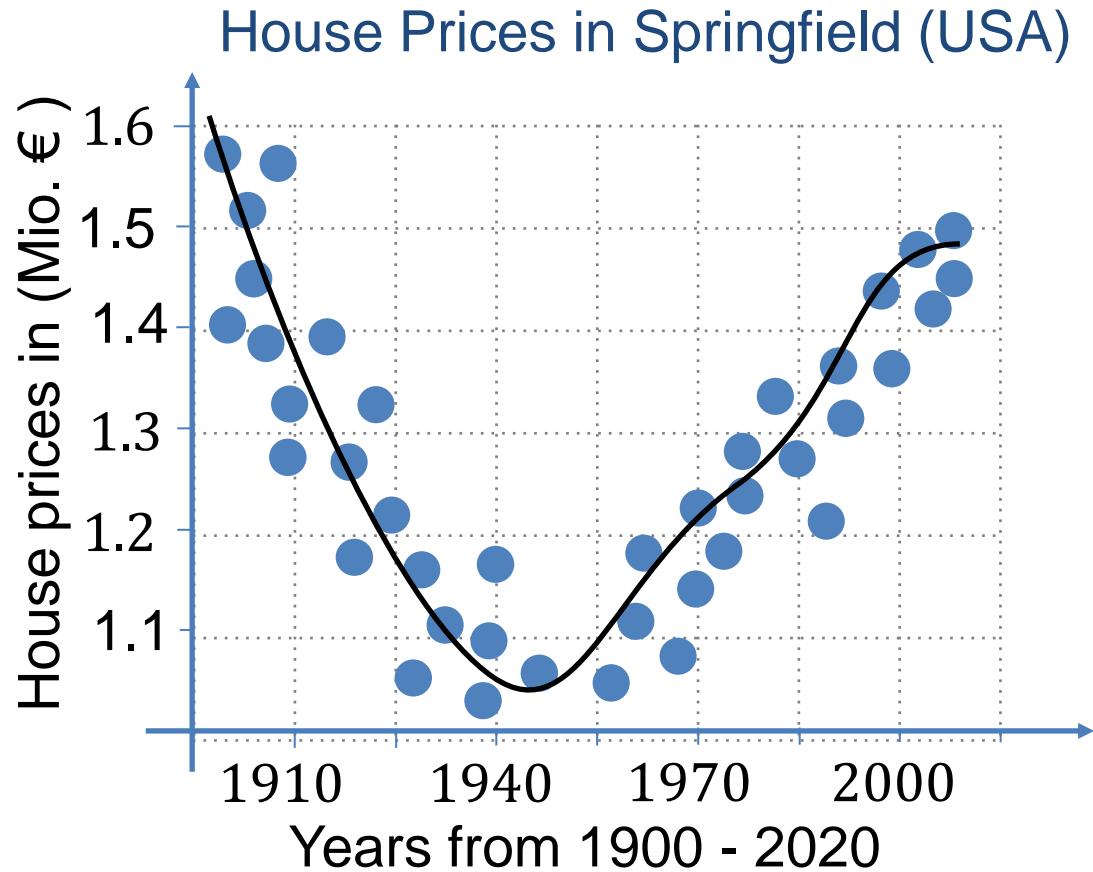
For linear regression we transform the input space using a polynomial basis function  $\Phi$ :



$$\Phi(x) = (1 \ x \ x^2 \ x^3 \ x^4 \ x^5 \ x^6)^T$$

This can lead to

- Computational issues depending on the number of points
- Memory issues depending on the dimensionality of the output 



# Introduction of Basis Functions

We've previously derived the optimization and prediction functions for SVM. Next, we can also introduce a basis function  $\Phi(x)$  to allow for non-linearity.

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$$

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn} \left( \sum_{i=1}^n \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \right)$$

💡 Note how the basis function is applied at each of the dot products.

# Solving Computational Issues with Sparsity

There are  $n^2$  summands for the optimization and  $n$  for the prediction, that is, both functions naively scale with the number of points  $n$ .

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$$

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn} \left( \sum_{i=1}^n \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \right)$$

# Solving Computational Issues with Sparsity

However, from the Karush-Kuhn-Tucker conditions we know that  $\alpha_i$  is non-zero only for the few support vectors.

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$$

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn} \left( \sum_{i=1}^n \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \right)$$

Most summands are thus zero. This property is called **sparsity** and greatly simplifies the computations.



# Solving Memory Issues with Kernel Trick

Since the basis function  $\Phi: \mathbb{R}^n \xrightarrow{\text{ }} \mathbb{R}^m$  is applied explicitly on each of the points, the memory scales with the output dimensionality m.

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$$

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn} \left( \sum_{i=1}^n \alpha_i^* y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \right)$$

# Solving Memory Issues with Kernel Trick

However, instead of explicitly computing  $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ , we can instead replace it with a kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$ .

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn}\left(\sum_{i=1}^n \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x})\right)$$

The kernel function usually doesn't require explicit computation of the basis function. This method of replacement is called the **kernel trick**.

# The Linear Kernel

The kernel function is given by:

$$K(x_i, x_j) = \frac{1}{2\sigma^2} x_i \cdot x_j$$

- $\sigma$  is a length-scale parameter
- Feature space mapping is basically just the identity function
- Only useful for linearly separable classification

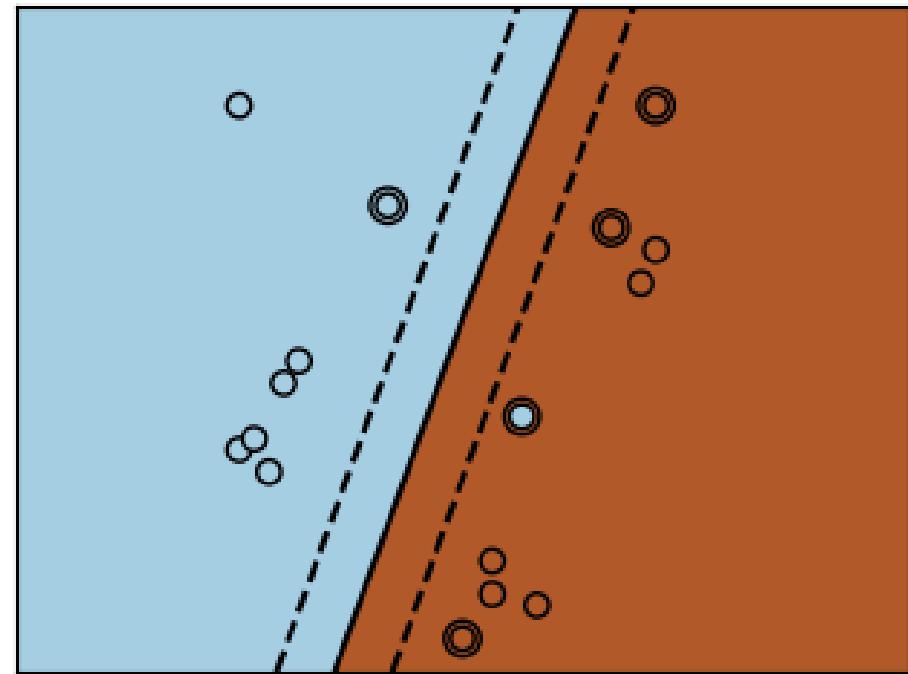


Image from [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_svm\\_kernels.html](https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html)

# The Radial Basis Function Kernel

The kernel function is given by:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

- $\sigma$  is a length-scale parameter
- Useful for non-linear classification with clusters
- Probably the most popular kernel

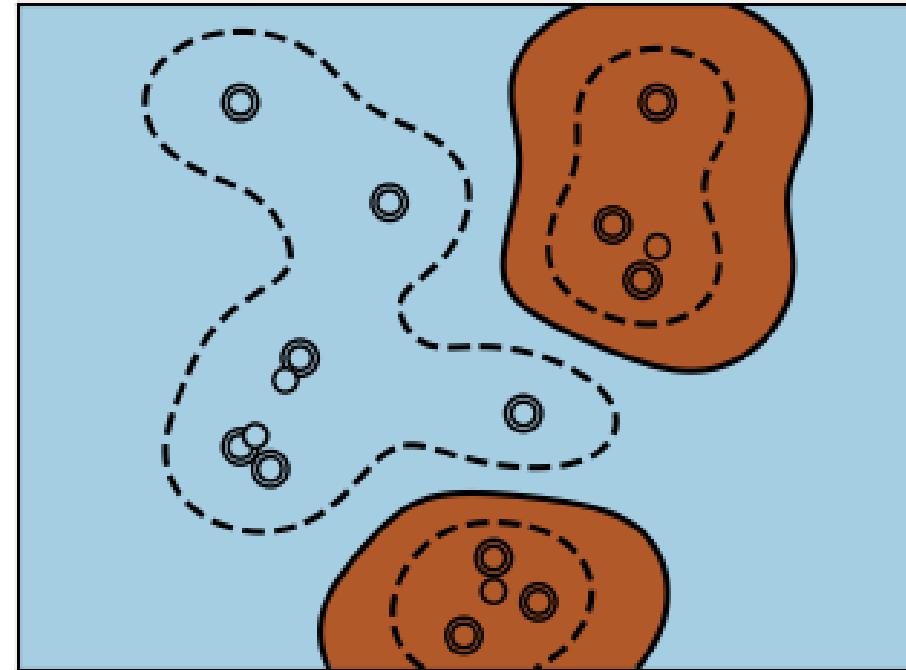


Image from [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_svm\\_kernels.html](https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html)

# The Polynomial Kernel

The kernel function is given by:

$$K(x_i, x_j) = \left( \frac{1}{2\sigma^2} x_i \cdot x_j + r \right)^d$$

- $\sigma$  is a length-scale parameter
- $r$  is a free parameter for the trade-off between lower- and higher-order 
- $d$  is the degree of the polynomial, a typical choice is  $d = 2$

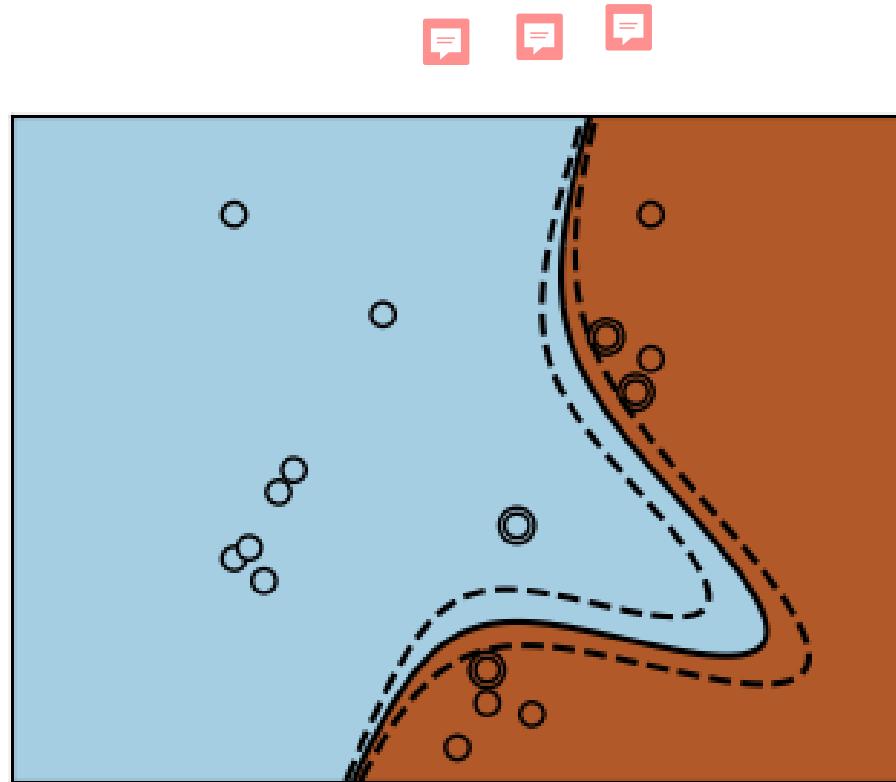
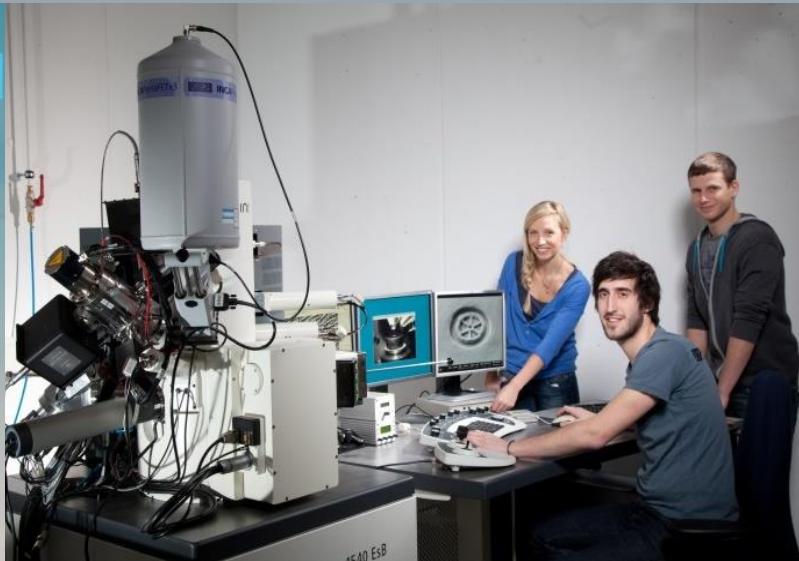
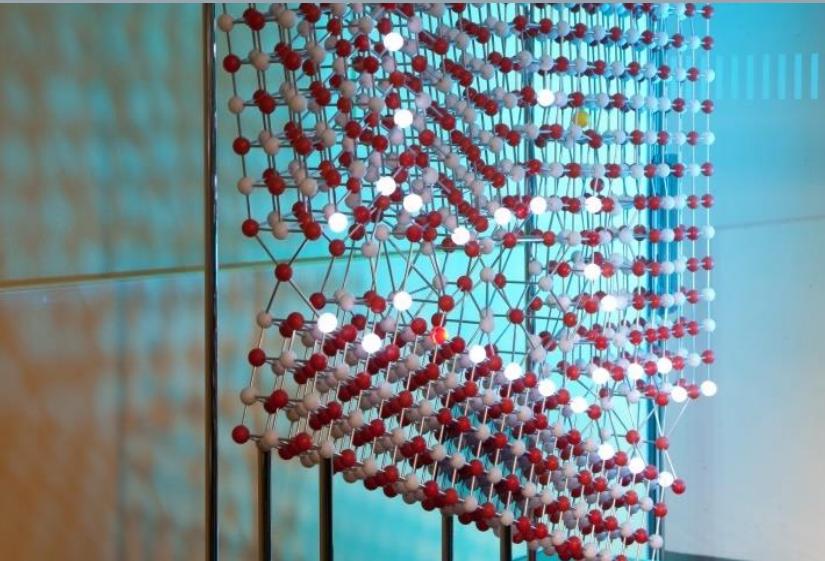


Image from [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_svm\\_kernels.html](https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html)

# Machine Learning for Engineers

## Support Vector Machines – Hard Margin and Soft Margin



Bilder: TF / Malter

# Hard Margin and Soft Margin

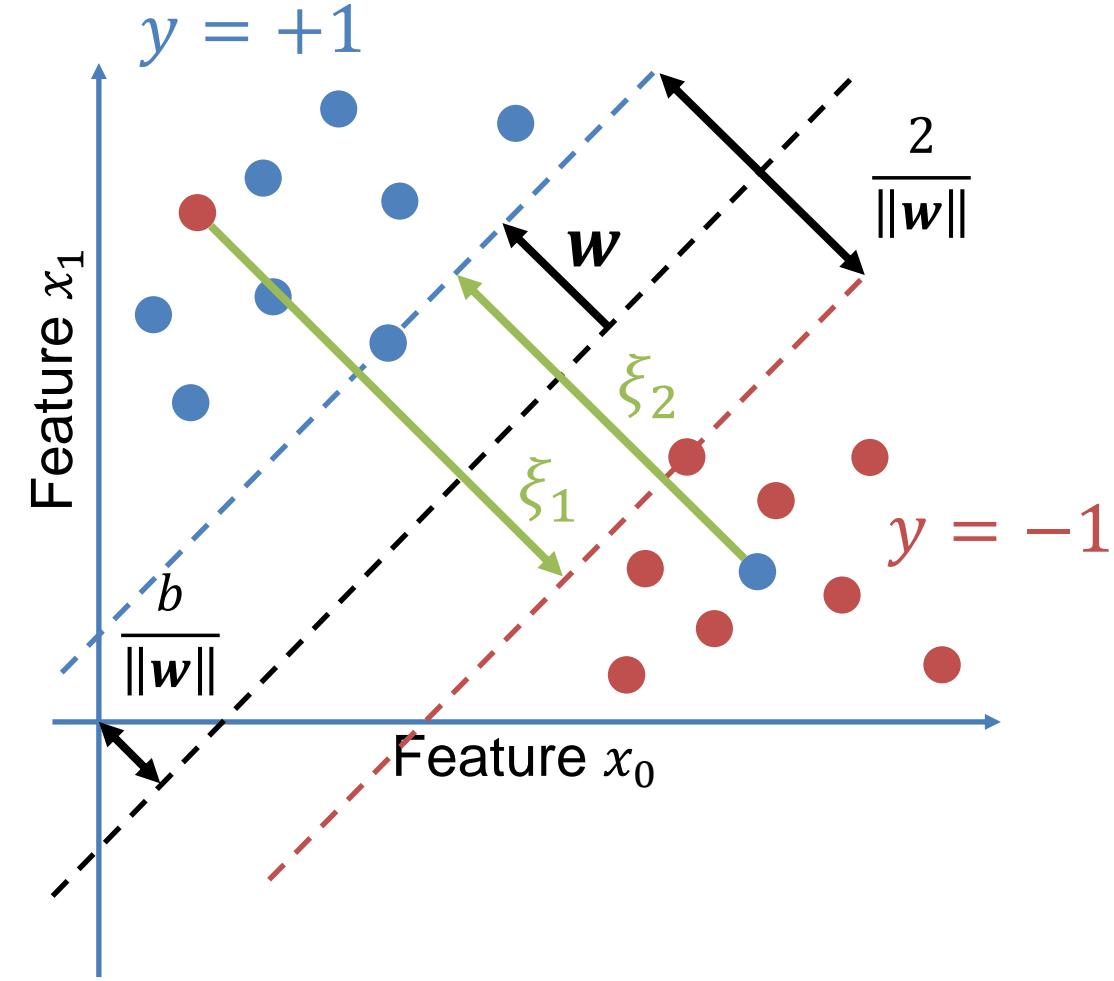
Up until now we have always implicitly assumed classes are linearly separable

→ So-called hard margin

What happens when the classes overlap and there is no solution?

→ So-called soft margin

Introduce slack variables  $\xi_1, \xi_2$  that move points onto the margin



# Soft Margin – Slack Variables

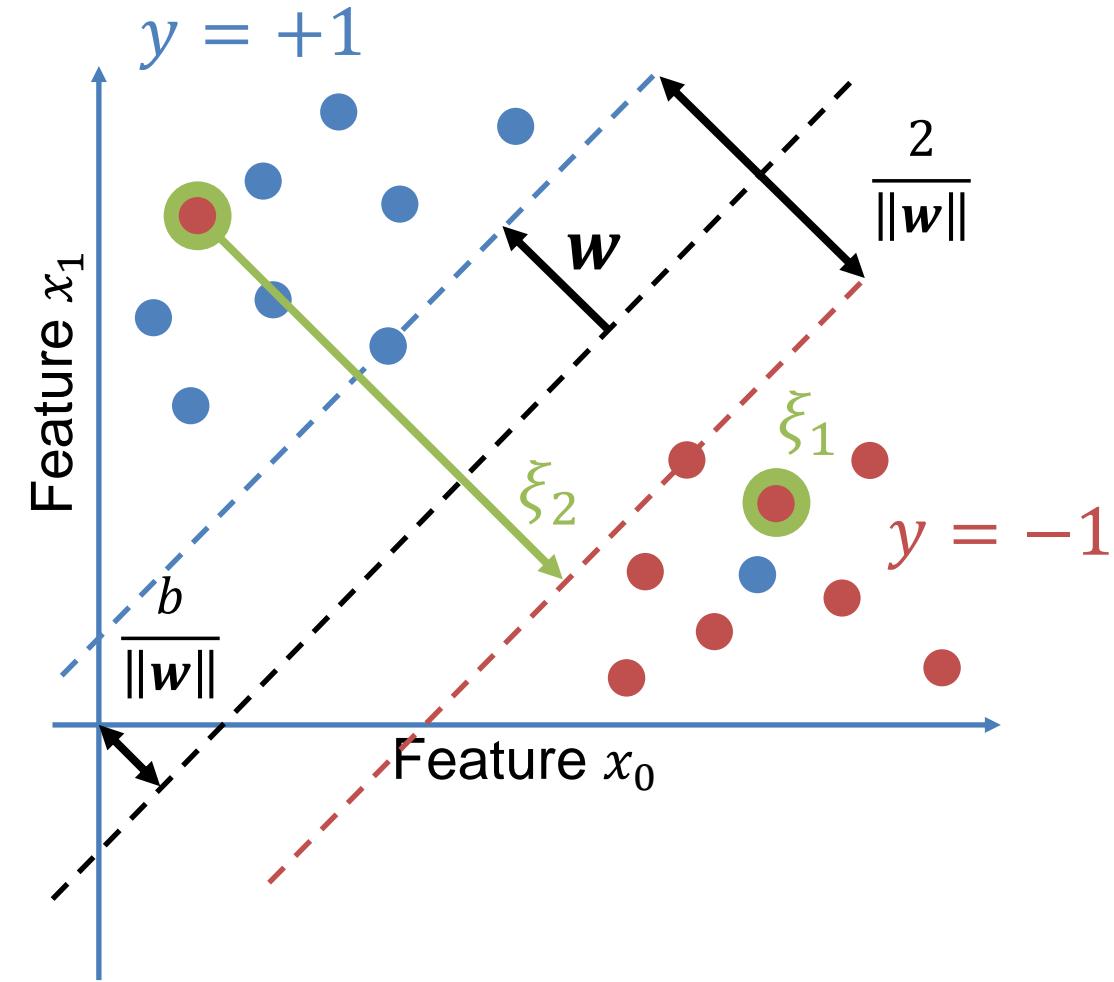
Slack variable  $\xi_i$  is the distance required to correctly classify point  $x_i$

If correctly classified and outside margin

- We know  $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1$  
- No correction, thus  $\xi_i = 0$

If incorrectly classified or inside margin

- We know  $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) < 1$  
- Move by  $\xi_i = 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i - b)$



# Updated Optimization Problem

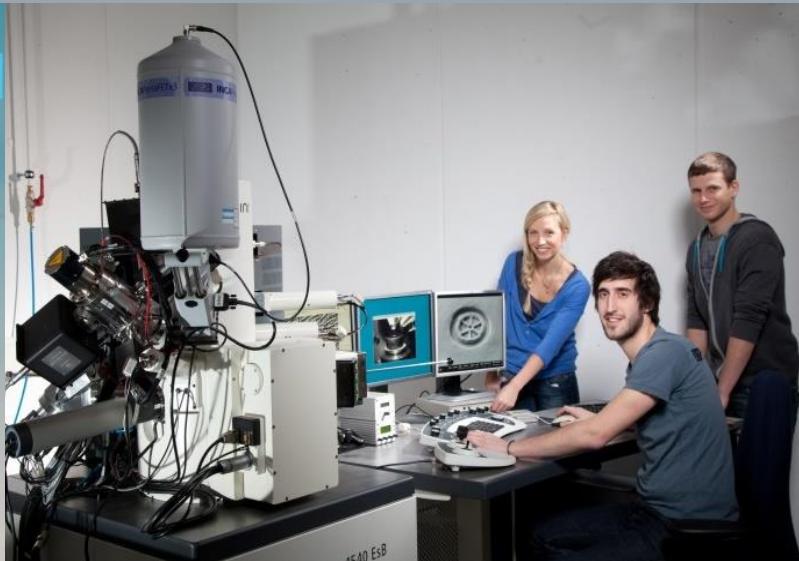
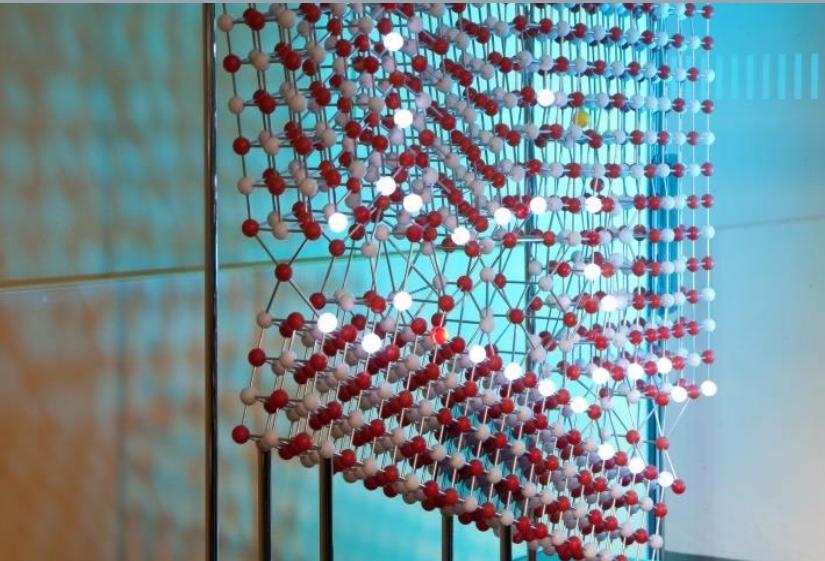
We can update the original optimization problem with the **slack variables**  $\xi$ , which leads us to the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} & \frac{1}{2} \|\mathbf{w}\|^2 + \lambda \frac{1}{n} \sum_{i=1}^n \xi_i \\ \text{s.t. } & y_i (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i \end{aligned}$$

The optimization procedure and kernel trick can similarly be derived for this, but we'll spare you the details 😊.

# Machine Learning for Engineers

## Support Vector Machines – Regression



Bilder: TF / Malter

# Intuition behind Support Vector Regression

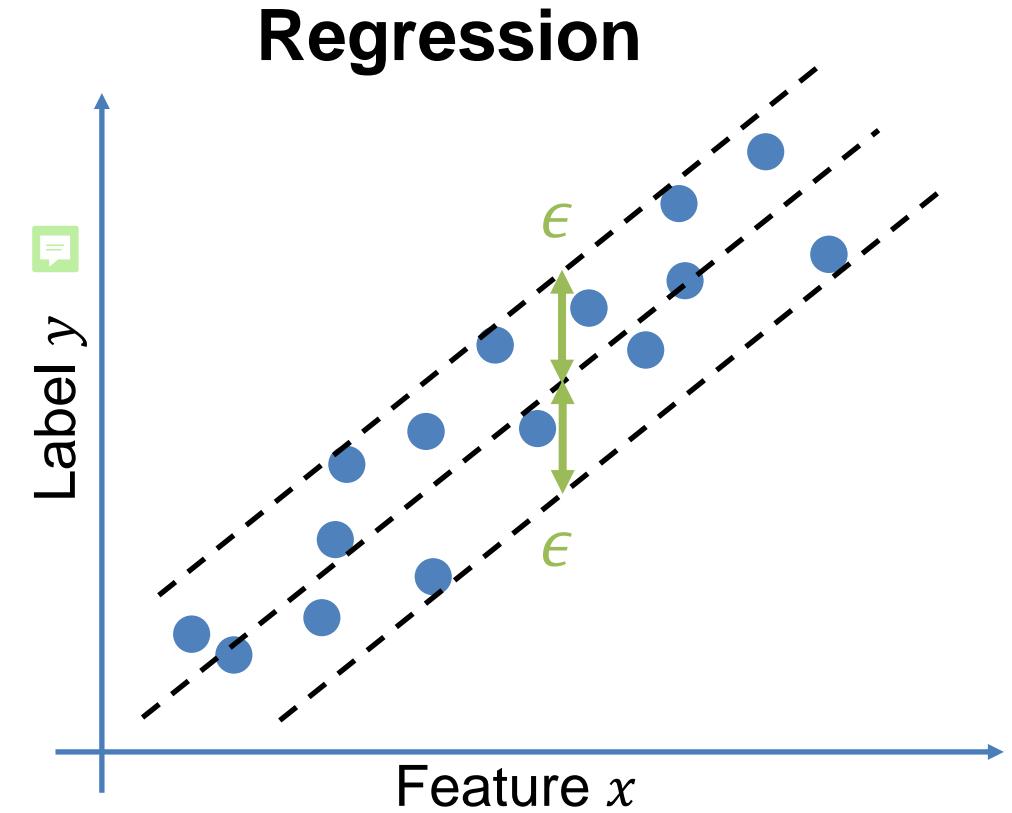
The Support Vector Machine is a method for **classification**.

Let us now turn to **regression**.

With training data  $(x_1, y_1), \dots, (x_n, y_n)$ , we want to find a function of form:

$$y = \mathbf{w} \cdot \mathbf{x} + b$$

For Support Vector Regression, we want to keep everything in a  $\epsilon$ -tube



# Keeping Everything inside The Tube

Thus, for each point  $(x_i, y_i)$ , the following conditions need to hold:

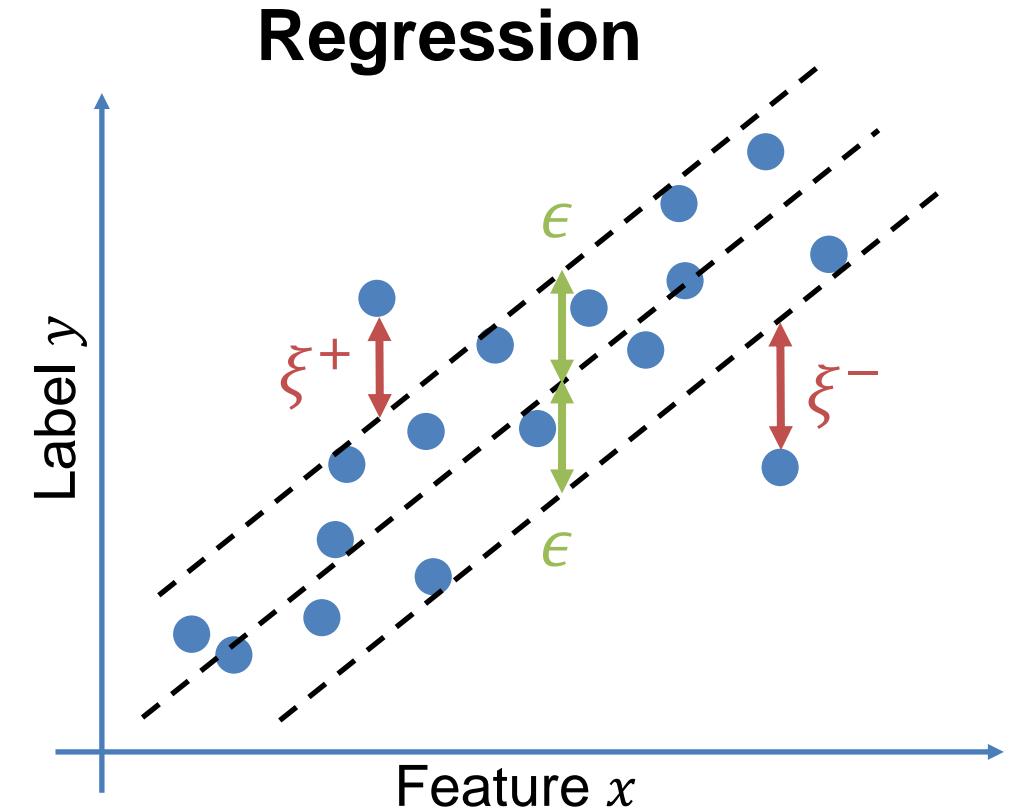
$$y_i \leq (\mathbf{w} \cdot \mathbf{x}_i + b) + \epsilon$$

$$y_i \geq (\mathbf{w} \cdot \mathbf{x}_i + b) - \epsilon$$

For outliers, we introduce slack variables  $\xi^+$  and  $\xi^-$ :

$$y_i \leq (\mathbf{w} \cdot \mathbf{x}_i + b) + \epsilon + \xi_i^+ \quad \square$$

$$y_i \geq (\mathbf{w} \cdot \mathbf{x}_i + b) - \epsilon - \xi_i^- \quad \square$$



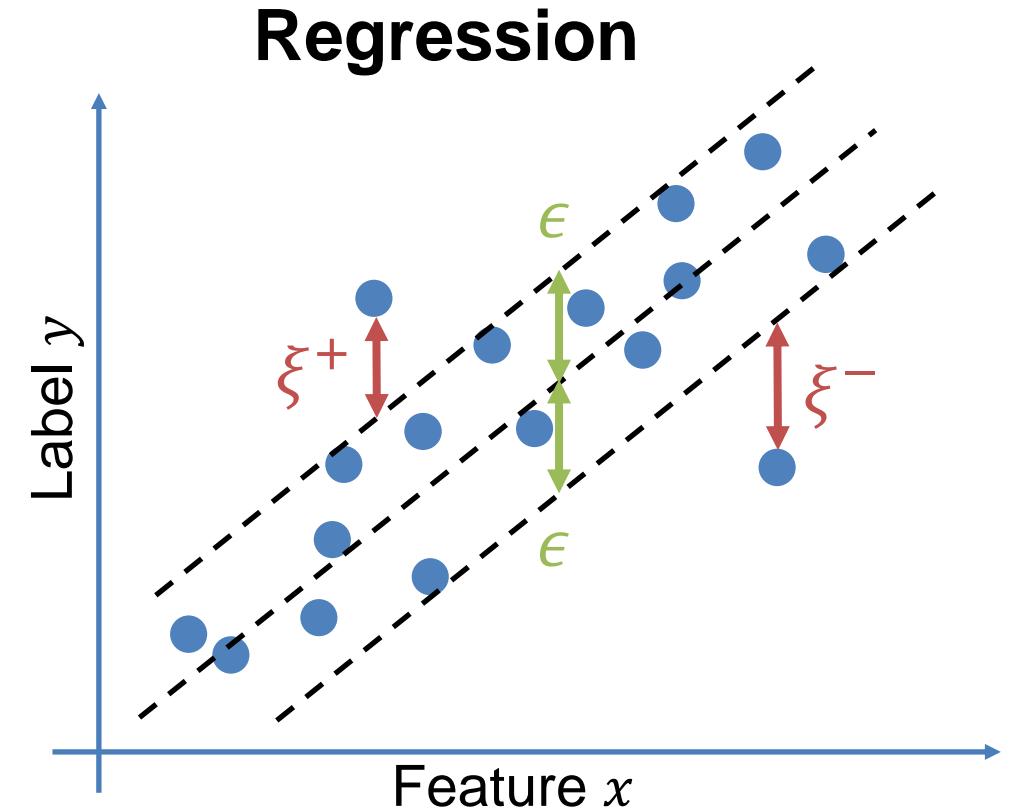
# Optimization of Support Vector Regression

This leads us to another constrained optimization problem.

- Minimizing  $\frac{1}{2} \|\mathbf{w}\|^2$  and the slack variables  $\xi_i^+$  and  $\xi_i^-$

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n (\xi_i^+ + \xi_i^-)$$

$C$  is a free parameter specifying the trade-off for outliers



# Optimization of Support Vector Regression

This leads us to another constrained optimization problem.

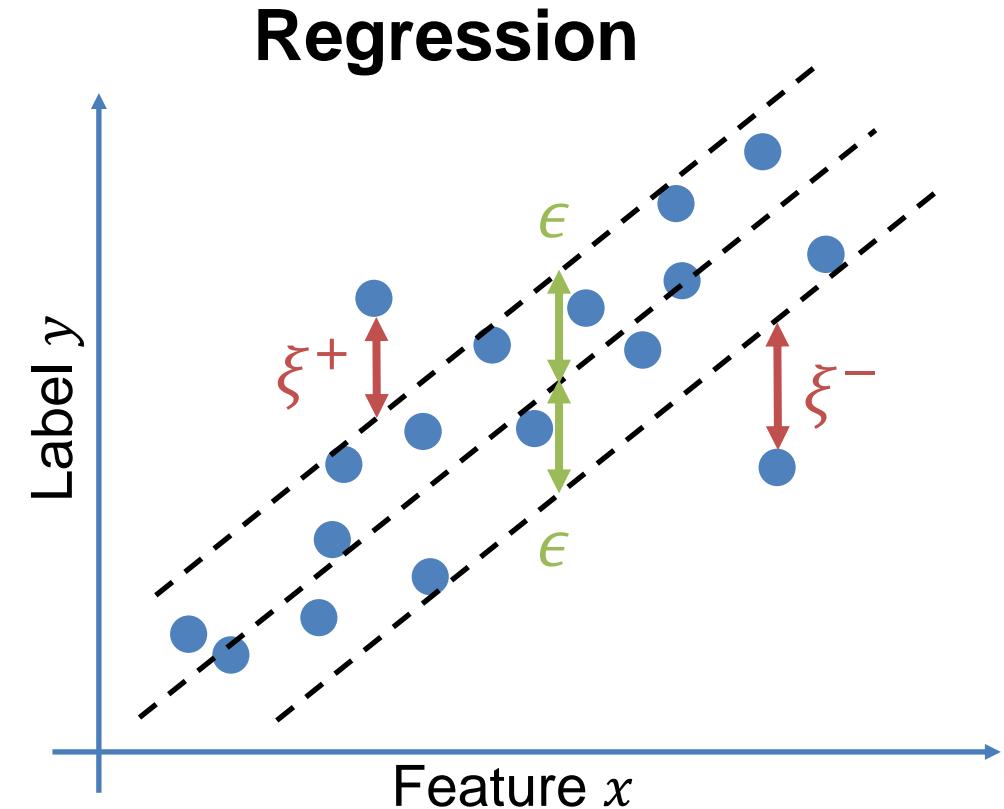
2. Subject to all  $y_i$  being inside the tube specified by  $\epsilon$

$$y_i \leq (\mathbf{w} \cdot \mathbf{x}_i + b) + \epsilon + \xi_i^+$$

$$y_i \geq (\mathbf{w} \cdot \mathbf{x}_i + b) - \epsilon - \xi_i^-$$

$$\xi_i^+ \geq 0$$

$$\xi_i^- \geq 0$$



# Pros and Cons of Support Vector Regression

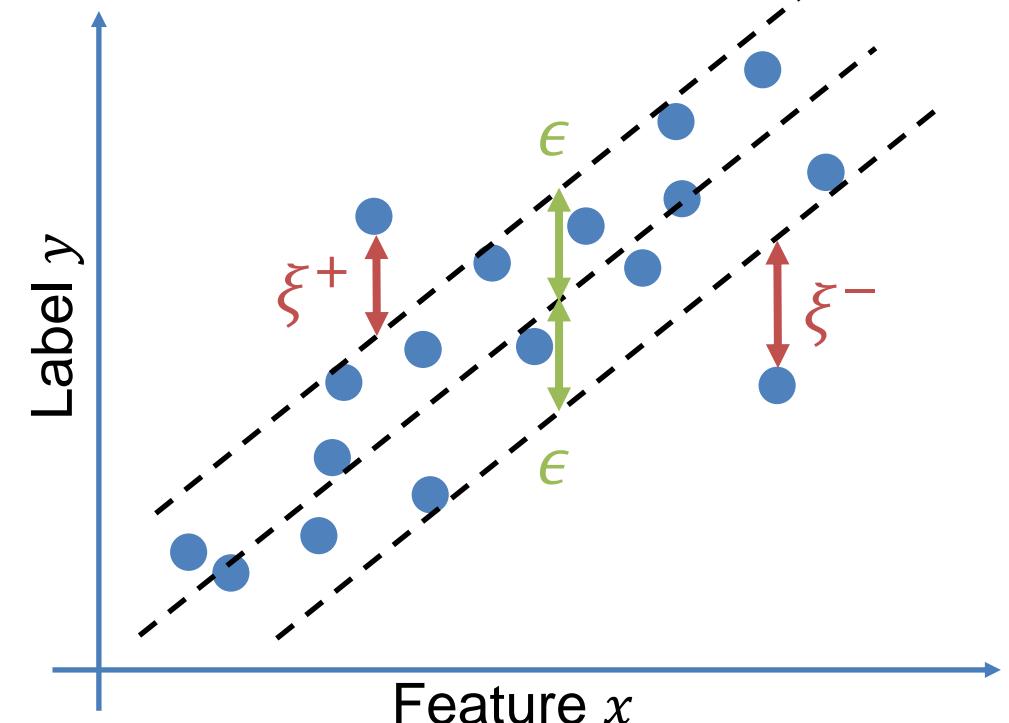
## Advantages

- Less susceptible  to outliers when compared to linear regression

## Disadvantages

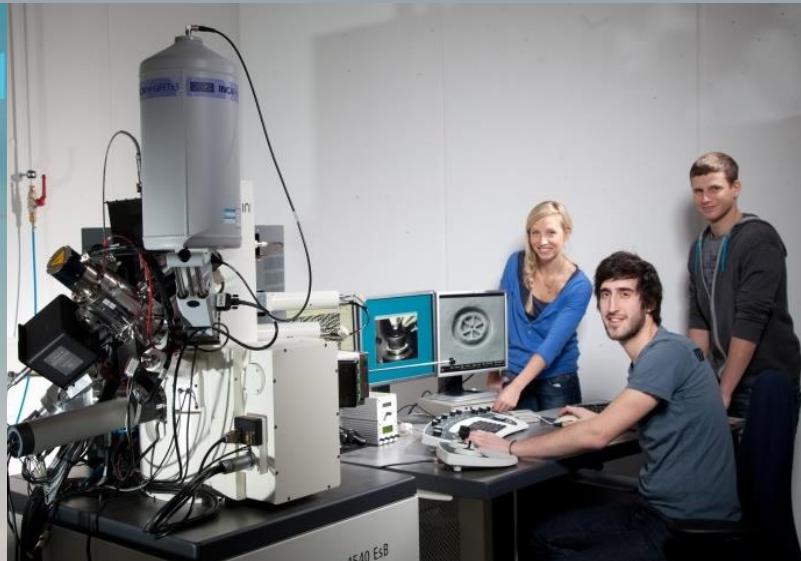
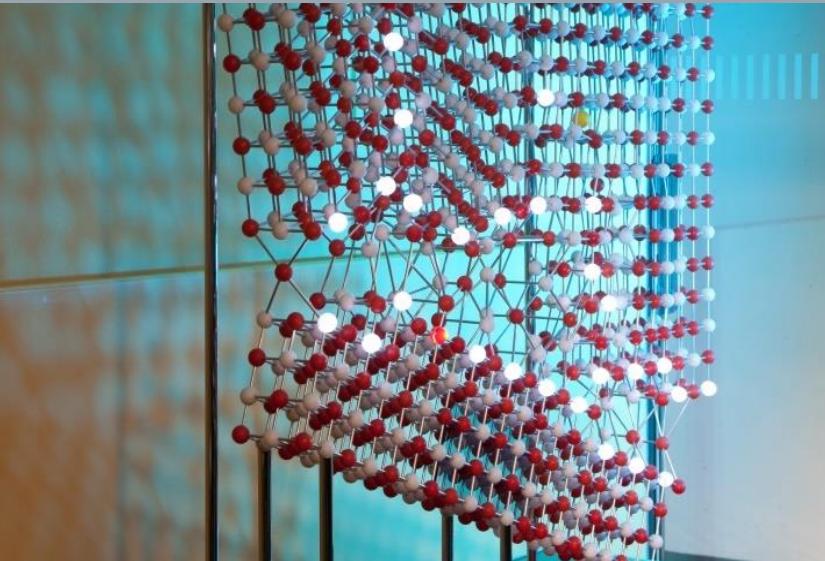
- Requires careful tuning of the free parameters  $\epsilon$  and  $C$
- Doesn't scale to large data sets 

## Regression



# Machine Learning for Engineers

## Support Vector Machines – Summary



Bilder: TF / Malter

# Summary of Support Vector Machines

## Support Vector Machine Advantages

- Elegant to optimize, since they have **one** local and global optimum
- Efficiently scales to high-dimensional features due to the **sparsity** and the **kernel trick**

## Support Vector Machine Disadvantages

- Difficult to choose good combination of **kernel function** and other free parameters, such as **regularization coefficient  $C$**
- Does not scale to large amount of training data

# References

1. D. Sontag. (2016). Introduction To Machine Learning: Support Vector Machines [Online]. Available: <https://people.csail.mit.edu/dsontag/courses/ml16/>
2. A. Zisserman. (2015). Lecture 2: The SVM classifier [Online]. Available: <https://www.robots.ox.ac.uk/~az/lectures/ml/>
3. K. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
4. B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2018.
5. A. Kowalczyk, *Support Vector Machines Succinctly*. Morrisville, NC, USA: Syncfusion, 2017.

## 3. Support Vector Machine

### ◀ Summary

### Questions

When is the decision boundary of SVM optimal?

- The margin is maximal
- There is no margin
- The margin is minimal



**Correct!**

**Submit**

Please fill out the following gaps in the text!

In Support Vector Machines the optimization term is

$$\min_{w,b} \frac{1}{2} \|w\|^2 \text{ under the constraints s.t. } y_i(w \cdot x_i - b) \geq 1.$$

This is known as the  ✓ [quadratic programming] problem, minimizing a

✓ [quadratic function] subject to

some  ✓ [inequality constraint | inequality constraints]. For these problems the mathematical framework of  ✓ [Lagrangian multipliers | Lagrange multipliers | lagrange multipliers | lagrangian multipliers] is used.</span>



**Correct!**

**Submit**

What does a convex function mean?

- A convex function has only one minimum and it is the global minimum
- A convex function has only a local minimum
- A convex function has several local minimum and a global minimum



**Correct!**

Submit

What is the purpose of using Lagrange Multipliers in SVM optimization?

- To deal with constraints in SVM optimisation
- Lagrange Multipliers are part of the convex optimization
- Lagrange Multipliers are penalty of SVM weights  $w$



**Correct!**

Submit

In the SVM context, why we solve a dual optimization problem, why not the primal form?

- Dual form increases optimisation performance
- Primal form is not a valid SVM formulation
- Dual form makes it possible (easier) to take gradient of parameters to minimise them



**Correct!**

Name three kernels, which are used with Support Vector Machines

1.
2.
3.

Also correct are:

- Linear
- Radial Basis Function
- Polynomial
- RBF



**Correct!**

What does a soft margin mean in the SVM context?

- We do not need to learn the margin parameters
- It penalises misclassifications using slack variables
- Data is separable using a linear separator



**Correct!**

Please fill out the following gaps in the text!

The advantage of support vector machines is, that they are  
[less] susceptible to  
[outliers] when compared to linear regression. However, there are two disadvantages! For one, SVMs require careful [tuning] of the free [parameters]  $\epsilon$  and  $C$ . On the other hand, it doesn't [scale] to large [datasets | data sets].

 **Correct!**

**Submit**

 **Summary**

**Add Comment**

**Sort Ascending**



Löhr, Tim [il34ifyn] - 26. May 2022

Thanks for noticing, I changed it.



Berisha, Dardan [in63ykyl] - 26. May 2022

"However" is misspelled in the last questions (as "Howeber")

When is the decision boundary of SVM optimal?

Problem statement 4.slide

Please fill out the following gaps in the text!

In Support Vector Machines the optimization term is  $\min_{w,b} \frac{1}{2} \|w\|^2$  under the constraints s.t.  $y_i(w \cdot x_i - b) \geq 1$ .

Optimization 2.slide

What does a convex function mean?

Summary 2.slide: Elegant to optimize, since they have one local and global optimum  
és Optimization 4.slide

What is the purpose of using Lagrange Multipliers in SVM optimization?

nem helyes válasz: „Lagrange Multipliers are part of the convex optimization”: a convex optimization-nek is része a Lagrange m-ek, viszont nem ez a céljuk (és ez volt a kérdés)  
helyes válasz: To deal with constraints in SVM optimisation, mert „For each of the  $n$  inequality constraints, introduce  $\alpha_i$  Lagrange multiplier”

In the SVM context, why we solve a dual optimization problem, why not the primal form?

válasz: 5. slideon, hogy a paraméterek gradiensét tudjuk venni, azaz az 1. deriváltat 0-val egyenlővé tenni

Optimization 4. slide sárgával kiemelt rész, és 5. slide kapcsolatban van a Gradient-tel: A gradiens a függvények deriválásának általánosítása többváltozós függvényekre. Ennek vektormező az eredménye, ami azt mutatja meg, hogy hogyan változik a függvény, és megadja a skalármező legnagyobb megváltozását.

Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent is simply used in machine learning to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

Name three kernels, which are used with Support Vector Machines

Kernel Trick utolsó 3 slide

What does a soft margin mean in the SVM context?

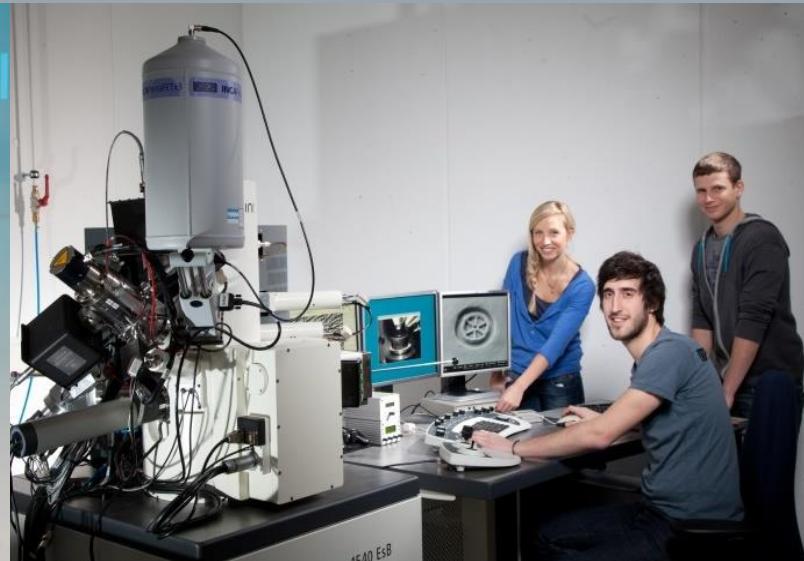
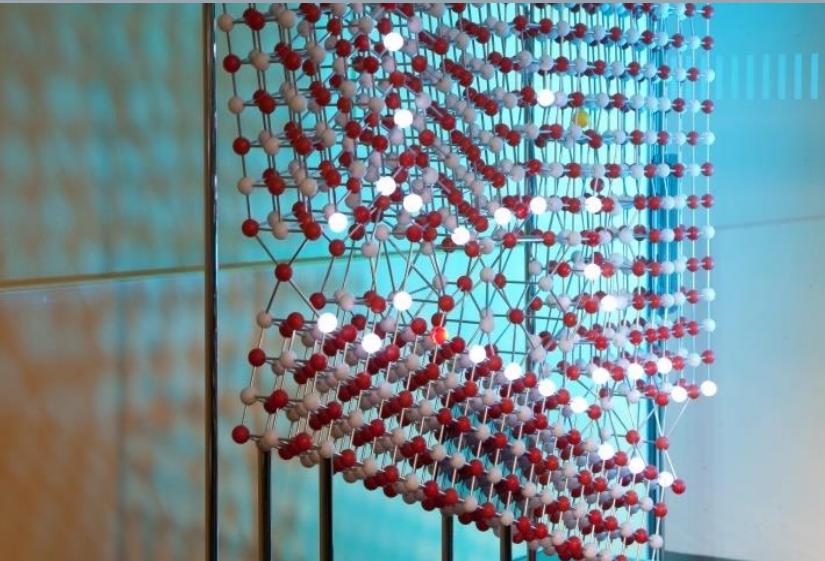
Hard and Soft Margin 2. és 3. slide

Please fill out the following gaps in the text!

Regression 6.slide, a szövegben annyi hibát találok, hogy Support Vector Regression-ről van szó, de „The advantage of support vector machines is”

# Machine Learning for Engineers

## Principal Component Analysis – Intuition



Bilder: TF / Malter

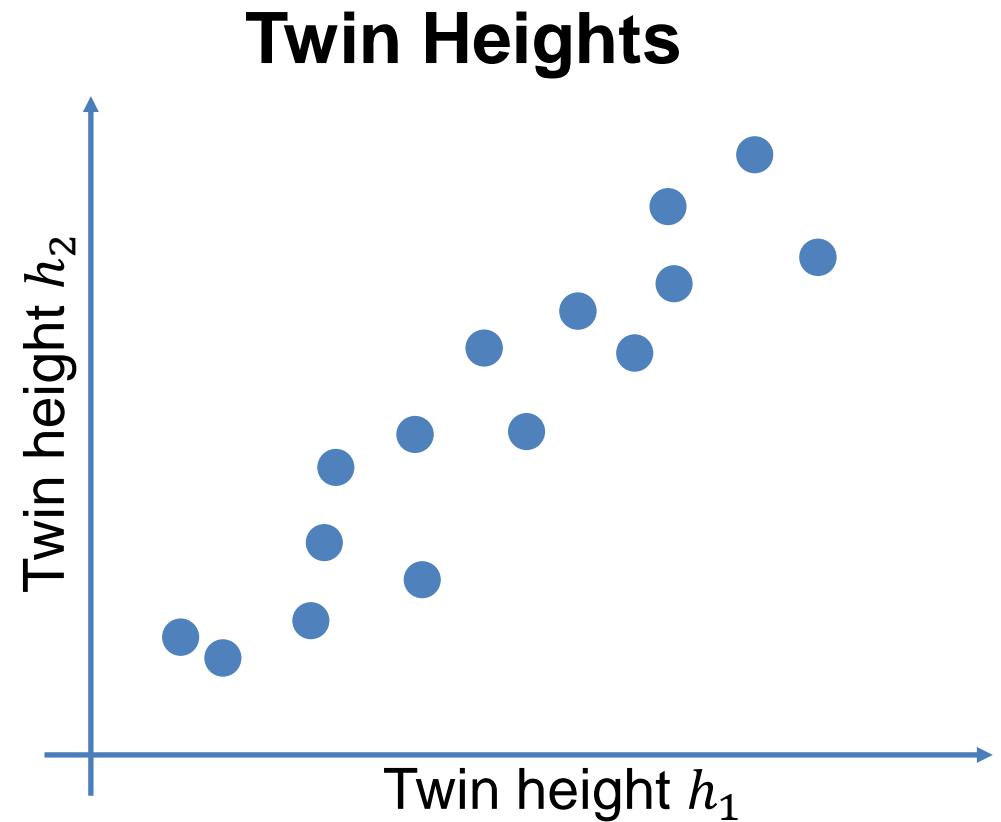
# Intuition behind Principal Component Analysis

Consider the following data with height of two twins  $h_1$  and  $h_2$ .



Now assume that

- a) We can only plot one-dimensional figures due to limitations<sup>1</sup>
- b) We have complex models that can only handle a single value<sup>1</sup>



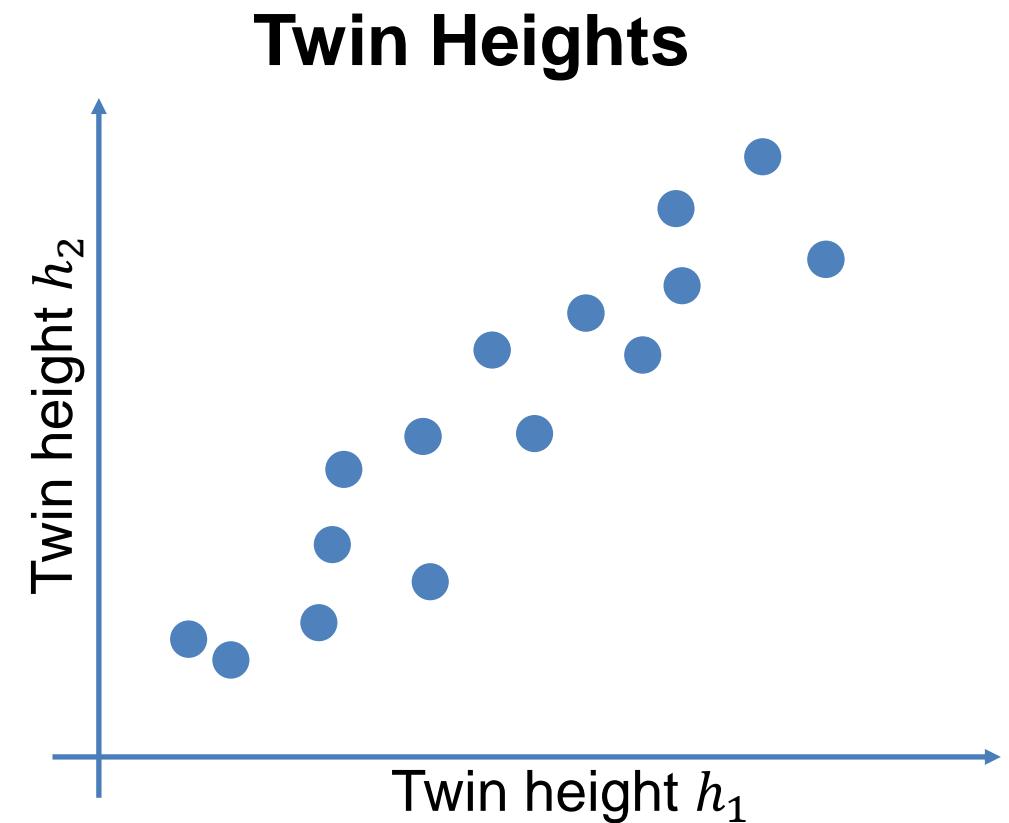
<sup>1)</sup> These are obviously constructed limitations for didactical concerns. However, in real life you will also face limitations in visualization (maximum of three dimensions) or model feasibility.

# Intuition behind Principal Component Analysis

How do we find a transformation from  
**two values to one value?**

In this specific case we could

- a) Keep the average height  $\frac{1}{2}(h_1 + h_2)$  as one value
- b) Discard the difference in height  $h_1 - h_2$



# Intuition behind Principal Component Analysis

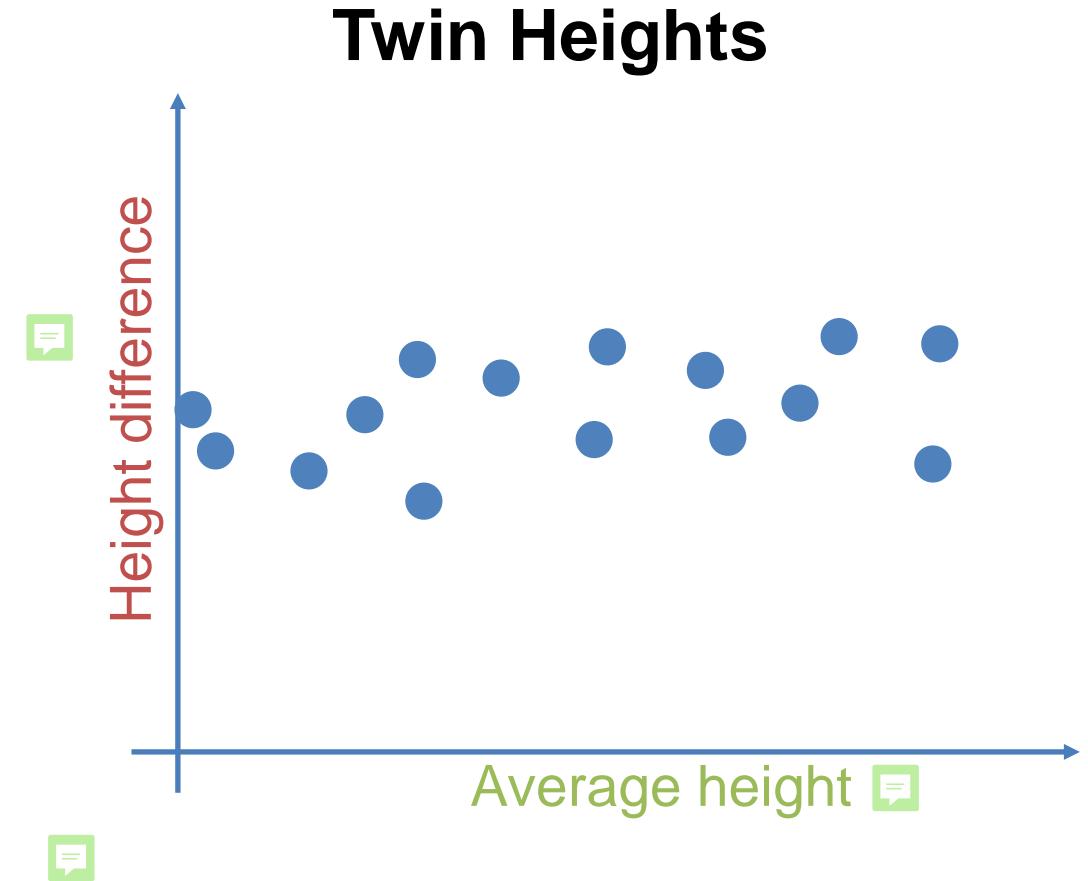
In this specific case we could

a) Keep the average height  $\frac{1}{2}(h_1 + h_2)$  as one value

b) Discard the difference in height  $h_1 - h_2$

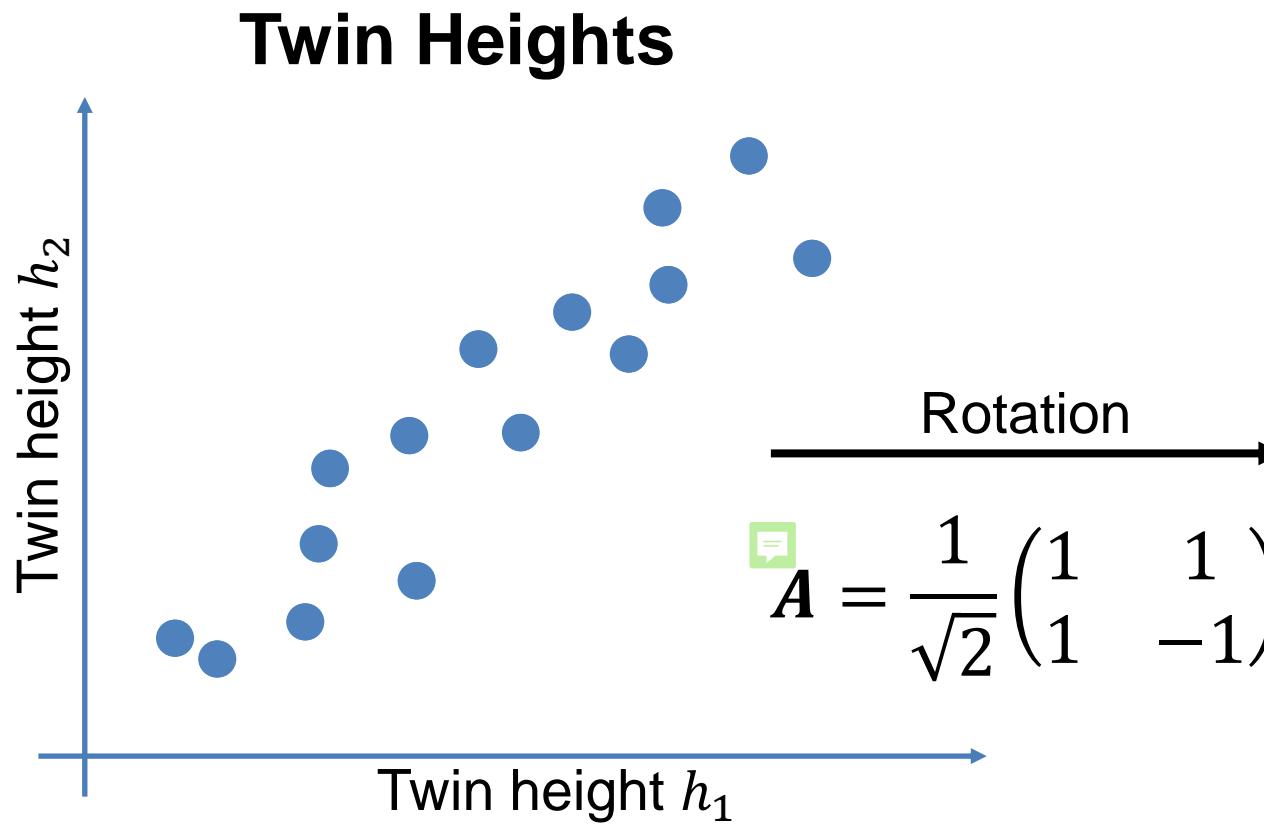
💡 This corresponds to a change in basis or coordinate frame in the plot.

→ Rotation by matrix  $A$



# Principal Component Analysis as Rotation

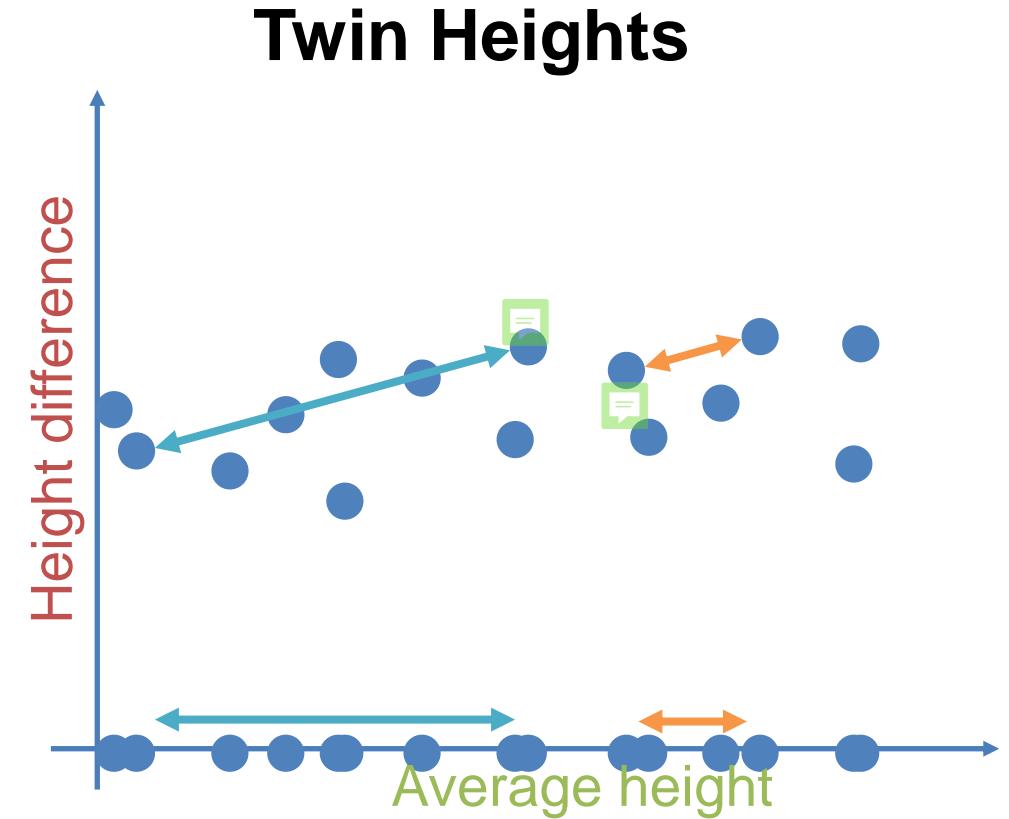
The Principal Component Analysis (PCA) is essentially a change in basis using a rotation matrix  $A$ .



# Principal Component Analysis: Preserving Distances

 **Idea:** Preserve the distances between each pair in the lower dimension

- Points that are **far apart** in the original space remain far apart
- Points that are **close** in the original space remain close

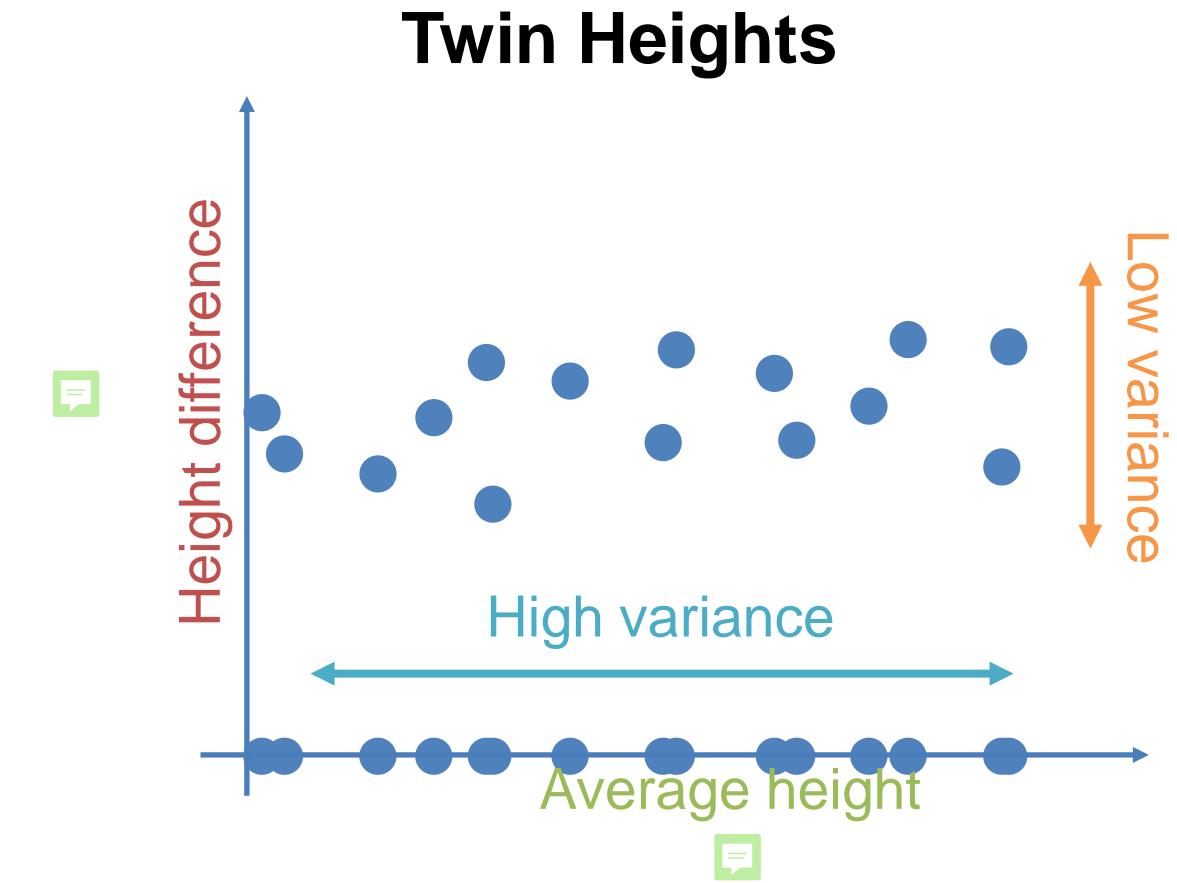


# Principal Component Analysis: Preserving Variance

The previous idea is closely coupled with the “variance”.

- Find those axes and dimensions with high variance
- Discard the axes and dimensions with increasingly low variance

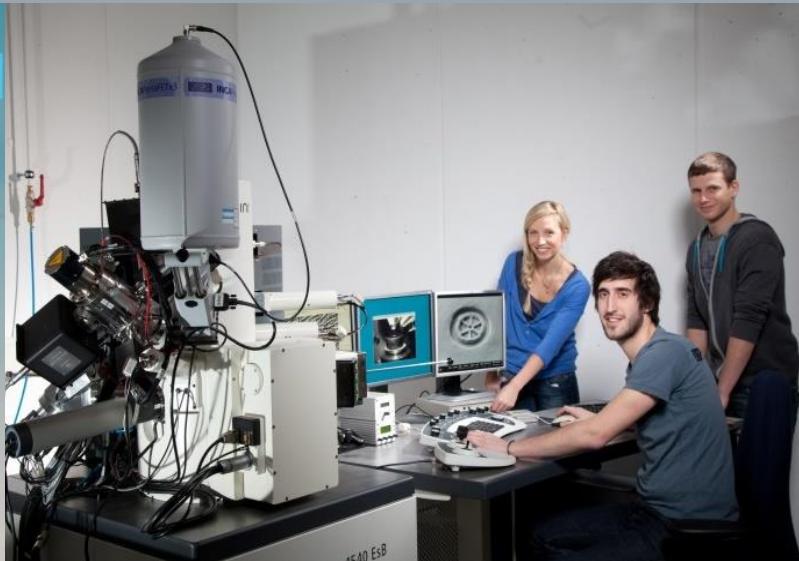
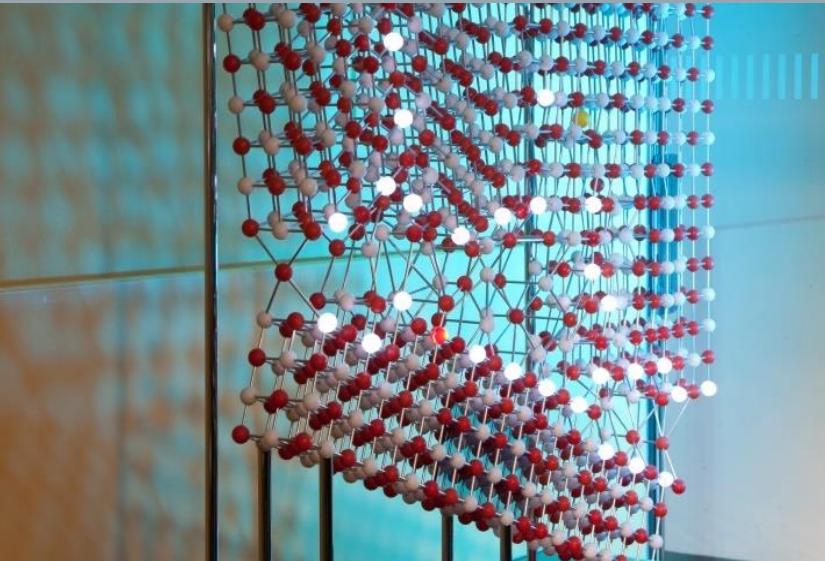
**Goal:** Algorithmically find directions with high/low variance in data<sup>1</sup>



1) In this trivial case we could hand-engineer such a transformation. However, this is not generally the case.

# Machine Learning for Engineers

## Principal Component Analysis – Mathematics



Bilder: TF / Malter

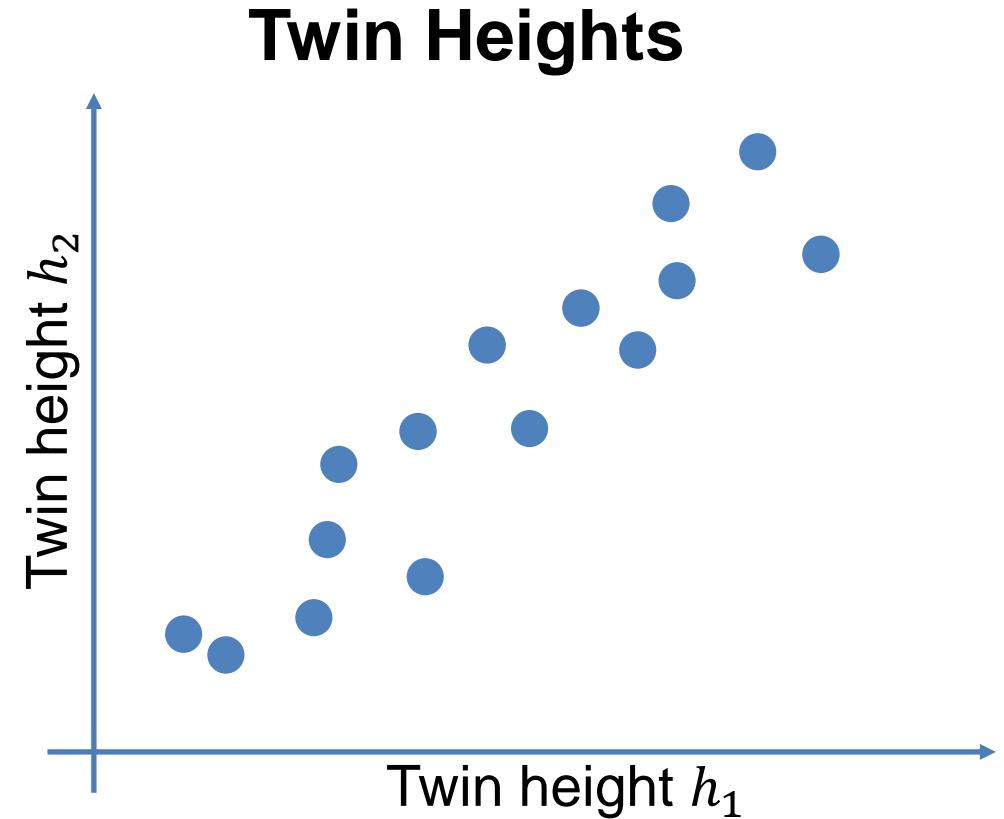
# Description of Input Data

Let  $X$  be a  $n \times p$  matrix of data points, where

- $n$  is number of points (here 15)
- $p$  is number of features (here 2)

$$X = \begin{pmatrix} 1.75 & 1.77 \\ 1.67 & 1.68 \\ \vdots & \vdots \\ 2.01 & 1.98 \end{pmatrix}$$



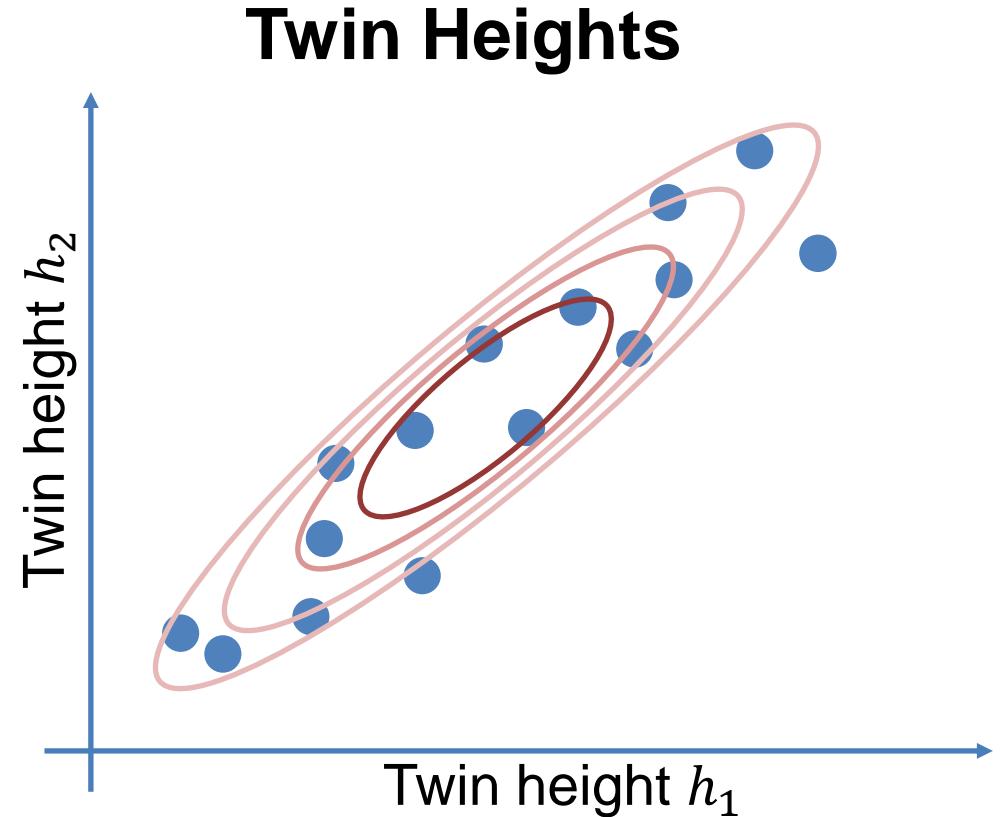


# Description of Input Data

Let  $C = X^T X$  be the  $p \times p$  covariance matrix of data points<sup>1</sup>, where

- $X$  are data points
- $p$  is number of features (here 2)

The covariance matrix generalizes the variance  $\sigma^2$  of a Gaussian distribution  $\mathcal{N}(\mu, \sigma^2)$  to higher dimensions.



1) Technically the covariance matrix of the transposed data points  $X^T$  after centering to zero mean.

# Description of Desired Output Data

We're interested in new axes that rotate the system, here  $w_1$  and  $w_2$

These are columns of a matrix

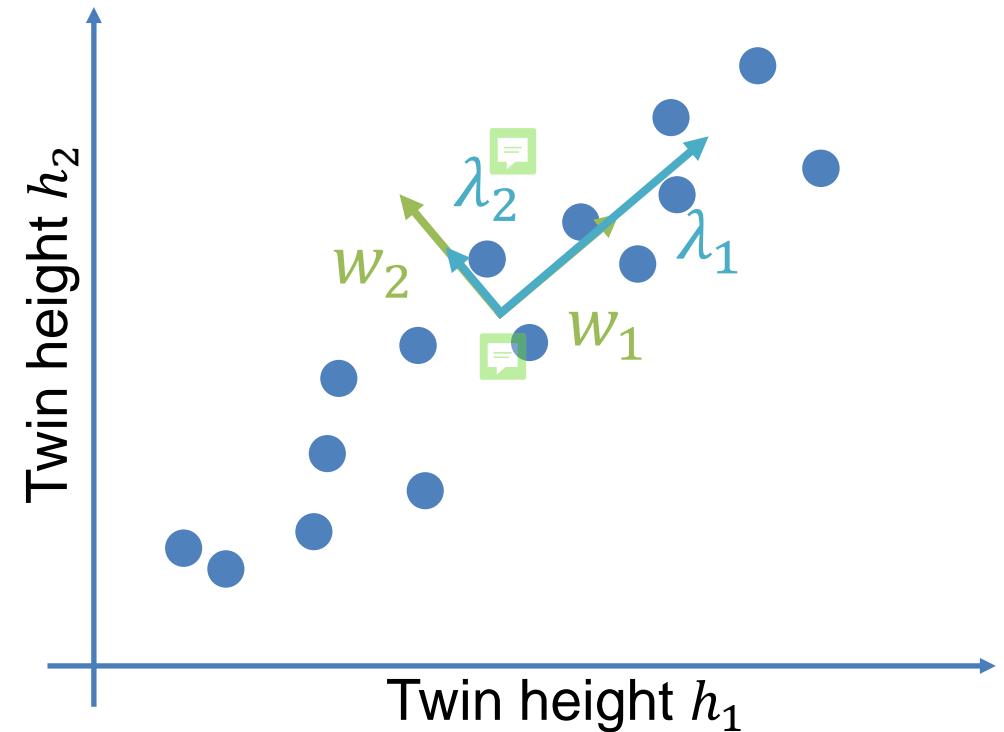
$$W = \begin{pmatrix} 0.7 & 0.7 \\ 0.7 & -0.7 \end{pmatrix}$$

$w_1$   $w_2$

We're also interested in the “variance” of each axis,  $\lambda_1$  and  $\lambda_2$

$$\lambda_1 = 2, \lambda_2 = 1$$

## Twin Heights

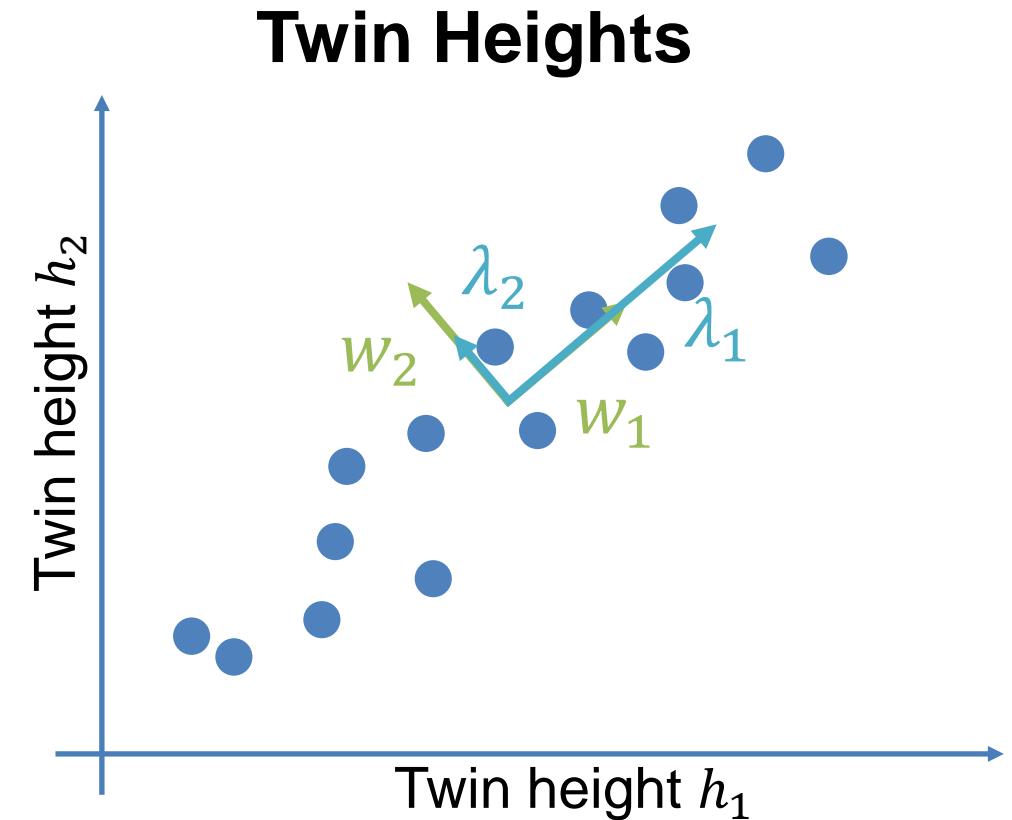


# Relationship to Eigenvectors and -values

We're in luck as this corresponds to the eigenvectors and –values of  $C$ :

$$C = WLW^T$$

- $W$  is  $p \times p$  matrix, where each column is an eigenvector
- $L$  is a  $p \times p$  matrix, with all eigenvalues on diagonal in decreasing order

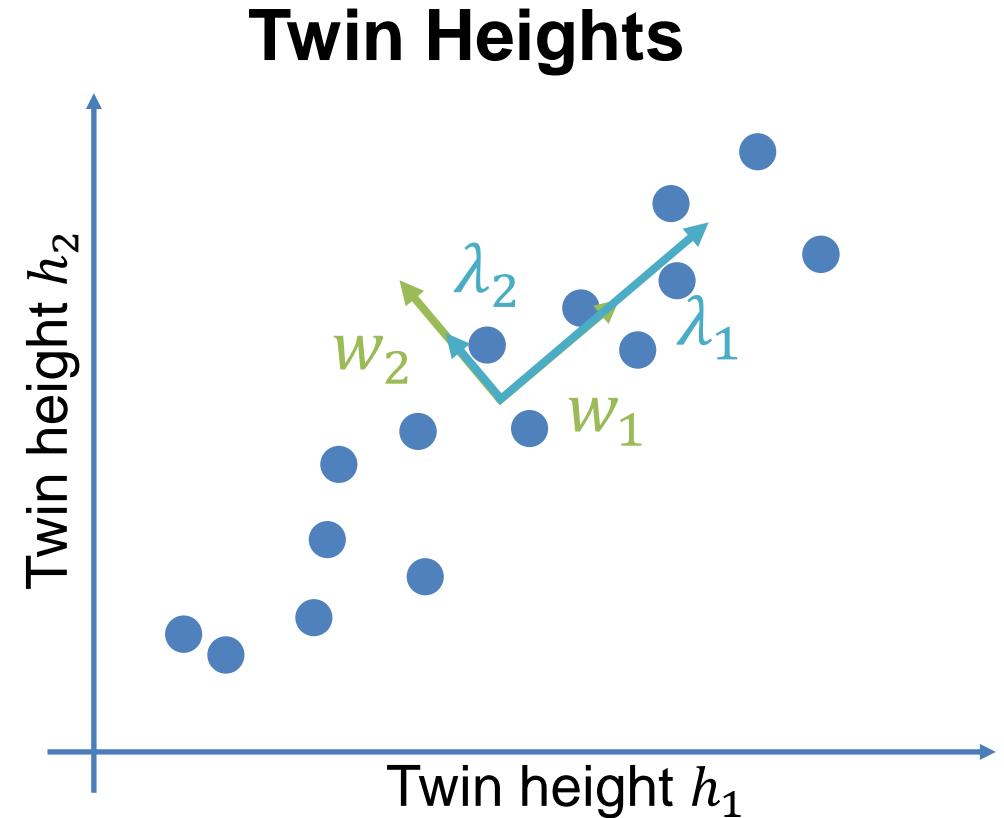


# Relationship to Singular Value Decomposition

Alternatively, one can also perform a  Singular Value Decomposition (SVD) on the data points  $X$ :

$$X = USW^T$$

- $U$  is  $n \times n$  unitary matrix
- $S$  is  $n \times p$  matrix of singular values  $\sigma_1, \dots, \sigma_p$  on diagonal
- $W$  is  $p \times p$  matrix of singular vectors  $w_1, \dots, w_2$

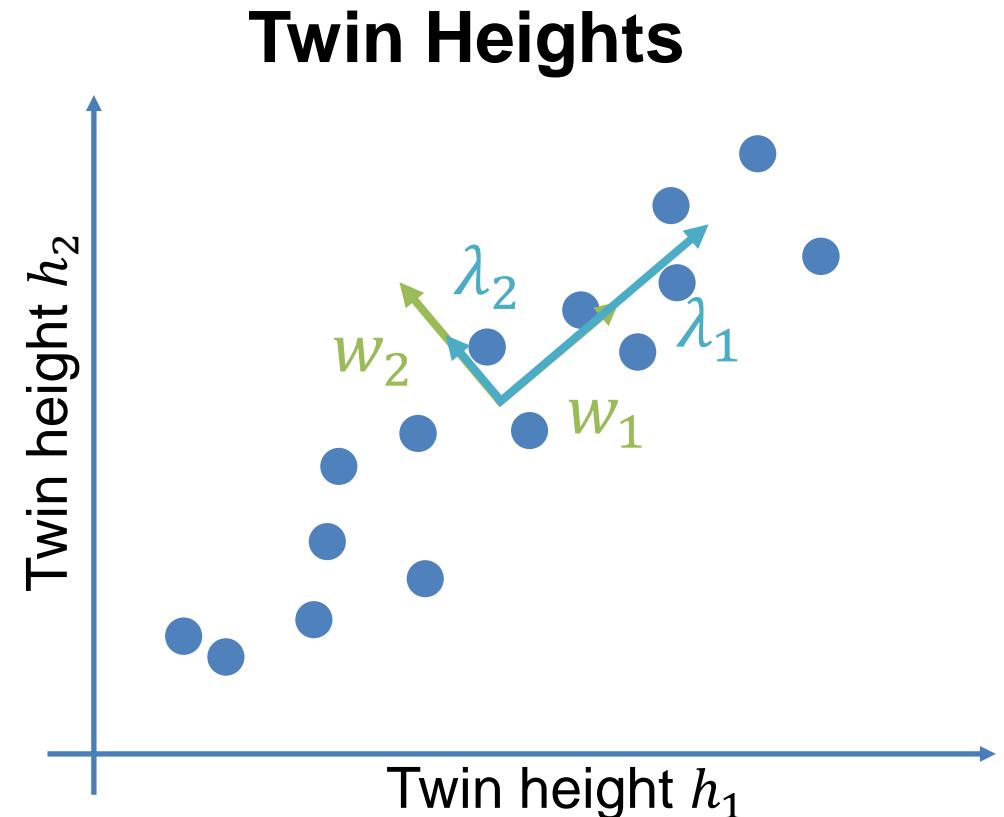


# Relationship to Singular Value Decomposition

Let us insert  $X = USW^T$  into the definition of the covariance matrix  $C$ :

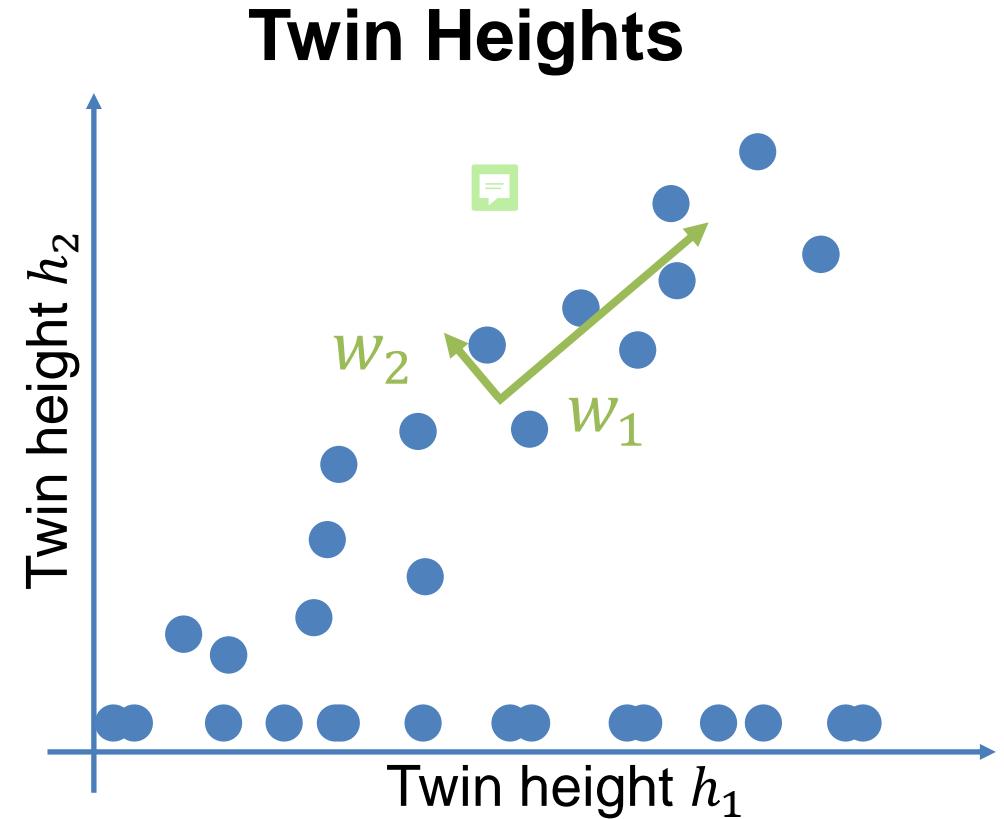
$$\begin{aligned} C &= X^T X \\ &= (USW^T)^T (USW^T) \\ &= WS^T U^T USW^T \\ &= WS^2 W^T \end{aligned}$$

- Eigenvalues correspond to squared singular values  $\lambda_i = \sigma_i^2$
- Eigenvectors directly correspond to singular vectors



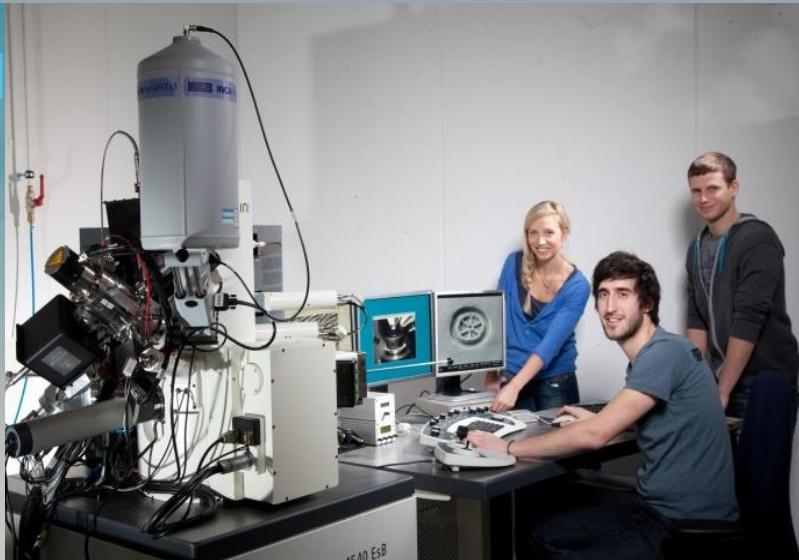
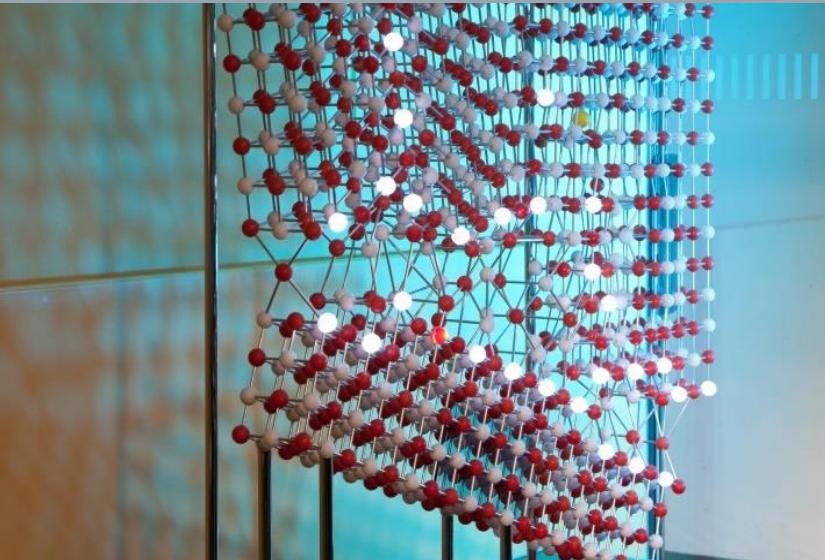
# Summary of Algorithm

1. Find principal directions  $w_1, \dots, w_p$  and eigenvalues  $\lambda_1, \dots, \lambda_p$
2. Project data points into new coordinate frame using
$$T = XW$$
3. Keep the  $q$  most important dimensions as determined by  $\lambda_1, \dots, \lambda_q$  (which are sorted by variance)



# Machine Learning for Engineers

## Principal Component Analysis – Applications



Bilder: TF / Malter

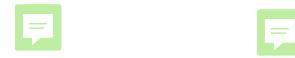
# Application: Image Compression

We can save data more efficiently  
by applying PCA!



The MaD logo ( $266 \times 266 \times 3$  px)  
can be compressed from 212 kB to

- 32 components : 8 kB
- 16 components : 4 kB
- 8 components : 2 kB



Can be used for other data as well!



Original



32 Components (97.27%)



16 Components (93.62%)



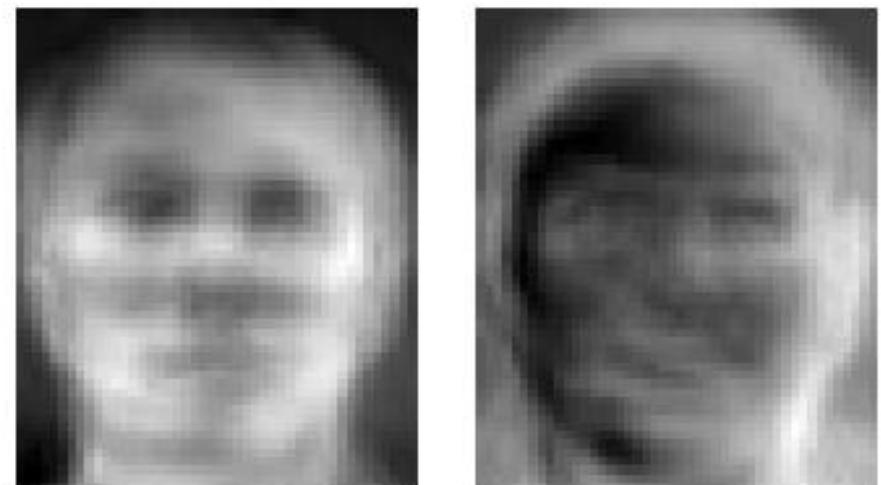
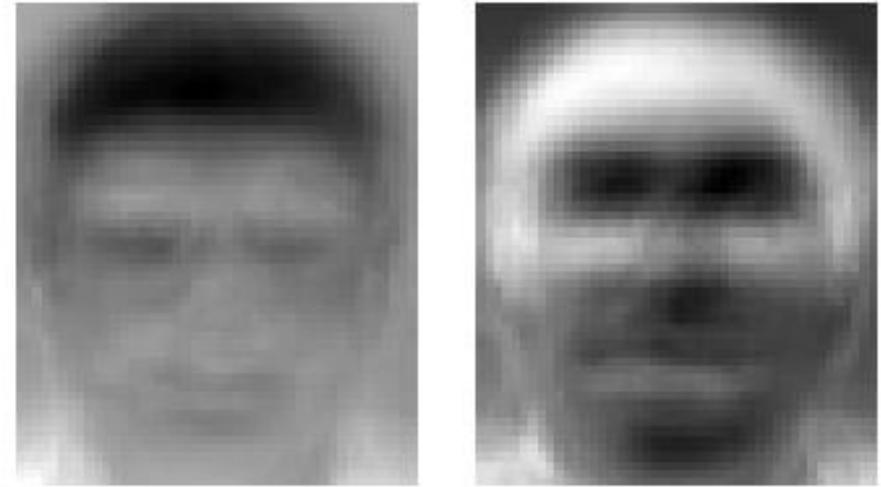
8 Components (85.50%)

# Application: Facial Recognition

Eigenfaces (1991) are a **fast** and **efficient** way to recognize faces in a gallery of facial images.

## General Idea:

1. Generate a set of eigenfaces (see right)  
based on a gallery
2. Compute the weight for each eigenface  
based on the query face image
3. Use the weights to classify (and identify)  
the person



Turk, M. and A. Pentland. "Face recognition using eigenfaces." *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991): 586-591.

# Application: Anomaly Detection

Detecting anomalous traffic in IP Networks is crucial for administration!

## General Idea:

Transform the time series of IP packages (requests, etc.) using a PCA

→ The first  $k$  (typically 10) components represent  
“normal” behavior

→ The components “after”  $k$ , represent  
anomalies and/or noise



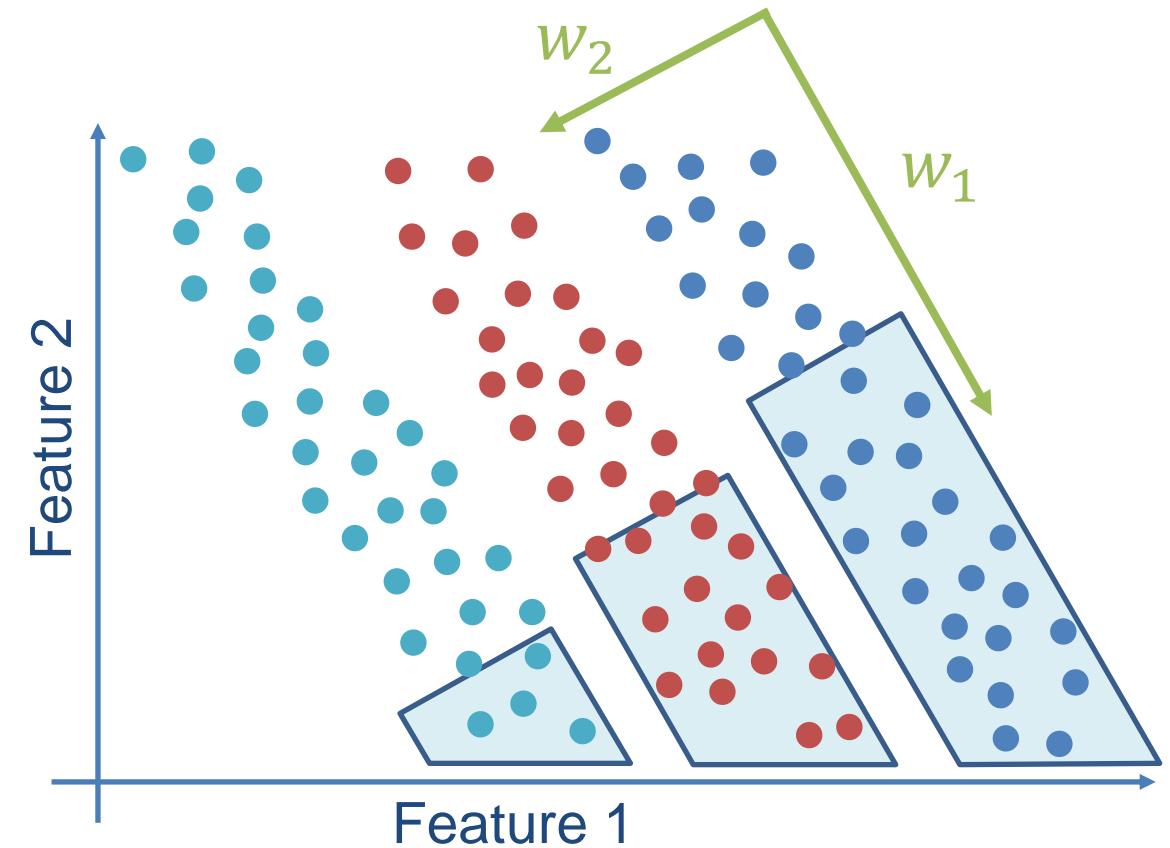
Packages mapped primarily to the latter component are classified as  
anomalous!

# Limitation: The Adidas Problem

Let the data points be distributed like the Adidas logo, with three implicit classes (teal, red, blue stripe).

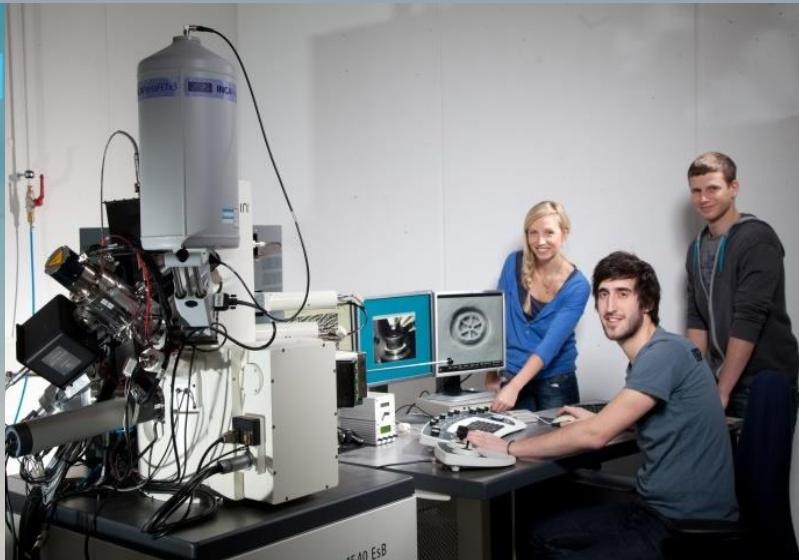
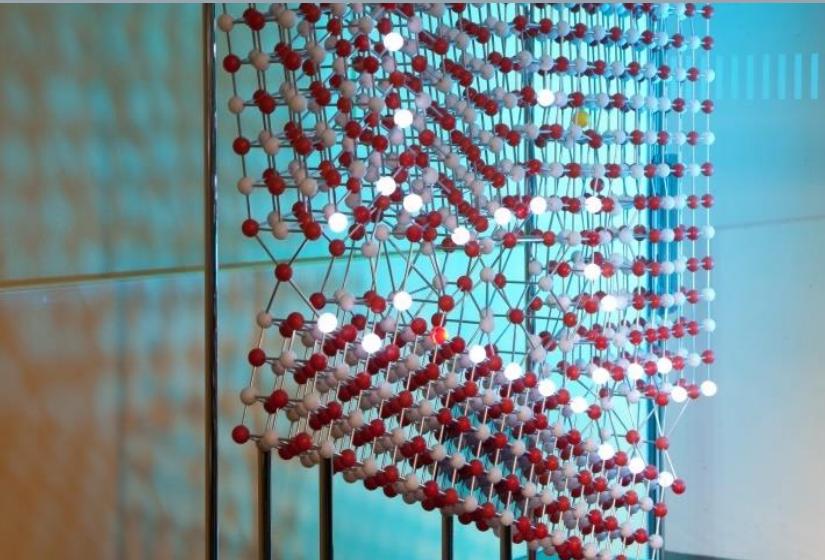
The principal directions  $w_1$  and  $w_2$  found by PCA are given. 

The first principal direction  $w_1$  does **not** preserve the information to classify the points! 



# Machine Learning for Engineers

## Principal Component Analysis – Summary



Bilder: TF / Malter

# Summary of Principal Component Analysis

- Rotate coordinate system such that all axes are sorted from most variance to least variance
- Required axes  $W$  determined using either
  - Eigenvectors and –values of covariance matrix  $C = WLW^T$
  - Singular Value Decomposition (SVD) of data points  $X = USW^T$
- Subsequently drop axes with least variance
- Variance-based feature selection has limitations (see. Adidas problem)

# References

1. P. Liang. (2009). Practical Machine Learning: Dimensionality Reduction [Online]. Available: <https://people.eecs.berkeley.edu/~jordan/courses/294-fall09/lectures/dimensionality/>
2. K. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.



## 4. Principal Component Analysis

### ◀ Summary

### Questions

Fill out the gaps in following text!

The Principal Component Analysis (PCA) is essentially a

- ✓ [change] in
- ✓ [basis] using a
- ✓ [rotation matrix] ?.



**Correct!**

**Submit**

Select the correct expressions regarding PCA Distances!

- Points that are far apart in the original space should be far apart in the target space
- Points that are far apart in the original space should be close in the target space
- Points that are close in the original space should be close in the target space
- Points that are close in the original space should be far apart in the target space



**Correct!**

**Submit**

Fill out the gaps in the following text!

The core idea of PCA is to find [directions | vectors | axes] and dimensions with high [variance]. Additionally, [directions | vectors | Axes] and dimensions with increasingly low [variance] are discarded. The overall goal is therefore to algorithmically find [vectors | axes | directions] with [high | low] / [high | low] / [variance] in data.



Correct!

Submit

The covariance is a central element needed for PCA. What does it describe?

- The transposed matrix of the high dimensional data
- The variance of the high dimensional data
- The transposed of the variance of the low dimensional data
- The variance of a high dimensional gaussian distribution



Correct!

Submit

The covariance C can be written as  $C = WLW^T$ . This is used to find the eigenvectors and eigenvalues. What components of the matrices correspond to what?

Reset Ordering

The Eigenvalues

The Eigenvectors

The diagonal of L

The Columns of W



Correct!

Submit

There exists an efficient way to find the W and D matrix to compute the eigenvalues and eigenvectors. What is full name of SVD ?

1.

Also correct is:

- Singular Value Decomposition



Correct!

Submit

Name at least three applications for PCA!

1.

2.

3.

Also correct are:

- Data Compression
- Image Compression

- Compression
- Facial Recognition
- Anomaly detection



Correct!

Submit

Fill out the gaps in the following text!

There exists a severe limitation using PCA! When the data points are distributed like the [Adidas logo], with three implicit classes (teal, red, blue stripe). The principal directions ?1 and ?2 found by PCA are given. The first principal direction ?1 does not preserve the information to classify the points, but the second component could be used for classification. This problem is called the

✓ [Adidas Problem].



Correct!

Submit

What value does indicate the data variance with respect to a particular Eigenvector?

- Magnitude of a Eigenvector
- Eigenvalue corresponds to that Eigenvector
- Both are correct



Correct!

Submit

What are the advantages of using SVD decomposition in computing the PCA method?

- SVD decomposition is a general method and applicable for any matrices
- Both are correct
- SVD decomposition is more efficient than the Eigen decomposition method



**Correct!**

**Submit**

Why variance or spread of data with respect to a particular direction is an essential factor in dimensionality reduction?

- Projecting data to a lower dimension corresponds to projecting data to directions with the lowest variance
- Preserving data only in the direction of the eigenvectors with the highest variance (largest eigenvalue) means preserving more information by reducing the dimensionality
- The highest variance corresponds to the highest reconstruction error



**Correct!**

**Submit**

**Summary**

[Add Comment](#)[Sort Ascending](#)yg

[yg69oroh] - 30. Jun 2022

Thanks, Tim!



Löhr, Tim [il34ifyn] - 27. Jun 2022

Dear Sebastian,

thanks for all of your questions. In fact, I did not create this question, so I can only assume what the learning goal of these questions is. I try to answer them subsequently:

- 1) This question shows that the magnitude of each eigenvector corresponds to a single eigenvalue per eigenvector. Your attention shall be raised that you decompose a matrix into eigenvalues and vectors, where the vector with the largest magnitude corresponds to the biggest eigenvalue.
- 2) I see what you mean, but it was not meant in this complicated way. This question was meant to clarify that SVD is a general method that finds a wide application field and that is why it is an advantage to use SVD instead of eigenvalue decomposition.
- 3) Alright, yes. We are not projecting we are preserving the eigenvectors with the largest eigenvalues. So we cut away all other vectors except for example 2 if we want to reduce the dimension to 2D. I will rephrase that answer. Thanks for noticing it!

Thanks for making all of these points Sebastian :)

Kind regards,  
Tim

yg

[yg69oroh] - 23. Jun 2022

Aller guten Dinge sind drei :)

For the question "Why variance or spread of data with respect to a particular direction is an essential factor in dimensionality reduction?", the correct answer here is "Projecting data to the direction with the highest variance means preserving more information by reducing the dimensionality".

But in fact what we do here is exactly NOT projecting into the direction of the highest variance. We are identifying the directions with highest variance, and we PRESERVE the data in this direction. A projection would instead delete all information in this direction.

I think what you mean is, that the data is projected onto the space, which is spanned by the directions of the highest variance..

yg

[y69oroh] - 23. Jun 2022

For the question "What are the advantages of using SVD decomposition in computing the PCA method?", the correct answer is "Both are correct". But I do not understand why.

SVD decomposition is a general method and applicable for any matrices. But if I understand correctly, in our application here we only have square (covariance) matrices. So it is no advantage, that SVD can be applied also to non-square matrices.

BR Sebastian

yg

[y69oroh] - 23. Jun 2022

For the question "What value does indicate the data variance with respect to a particular Eigenvector?", I would think that only the answer "Eigenvalue corresponds to that Eigenvector" is correct. Within a Eigenspace there is no single special vector, of which the magnitude is relevant.

Instead the correct solution here is "Both are correct".



Löhr, Tim [il34ifyn] - 26. May 2022

Hey Dardan, yes indeed, that is a really frustrating problem with StudOn. I have to type in every possible solution. For all upper and lower case possibilities as well as for example "dataset" and "data set" and again with lower and upper case. If you just don't get it right, write me an email and I will give you the answers.

Kind regards,

Tim



Berisha, Dardan [in63ykyl] - 26. May 2022

Furthermore there should be accepted more answers to the adidas question, e.g. that "adidas" could also be written in small letters. But I think this is more of a studon-problem. Maybe you could inform the studon-administrators that such solutions should also be accepted so you don't have to write every possible solution down.



Berisha, Dardan [in63ykyl] - 26. May 2022

In the 3rd questions is "vector" misspelled 2x in the solution. There it is spelled "vecors".

Fill out the gaps in following text!

The Principal Component Analysis (PCA) is essentially

Intuition 5. Slide

Select the correct expressions regarding PCA Distances!

Intuition 6. slide

Fill out the gaps in the following text!

The core idea of PCA is to find

Intuition 7. slide

The covariance is a central element needed for PCA. What does it describe?

Mathematics 3. slide

The covariance  $C$  can be written as  $C = W L W^T$ . This is used to find the eigenvectors and eigenvalues. What components of the matrices correspond to what?

Mathematics 5. slide

There exists an efficient way to find the  $W$  and  $D$  matrix to compute the eigenvalues and eigenvectors. What is full name of SVD ?

SVD-vel a singular values és singular vectors-t kapjuk meg, ezekből lehet computolni az eigenvalues és eigenvectorokat

Mathematics 5 – 6 – 7. slide, hogy SVD hogyan correspond az eigenvalues and eigenvectoroknak, de leírom ide is tanulásképp:

SVD:  $\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{W}^T$

$\mathbf{S}$  is  $n \times p$  matrix of singular values  $\sigma_1, \dots, \sigma_p$  on diagonal.

Covariance matrix:  $\mathbf{C} = \mathbf{W} \mathbf{L} \mathbf{W}^T$

$\mathbf{L}$  is a  $p \times p$  matrix with all eigenvalues on diagonal in decreasing order

**Relationship:** Eigenvalues correspond to squared singular values  $\lambda_i = \sigma_i^2$

---

SVD:  $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{W}^T$

$\mathbf{W}$  is  $p \times p$  matrix of singular vectors  $\mathbf{w}_1, \dots, \mathbf{w}_2$

Covariance matrix:  $\mathbf{C} = \mathbf{W}\mathbf{L}\mathbf{W}^T$

$\mathbf{W}$  is  $p \times p$  matrix, where each column is an eigenvector

**Relationship:** Eigenvectors directly correspond to singular vectors

Name at least three applications for PCA!

Application egész pdf

Fill out the gaps in the following text!

There exists a severe limitations using PCA!

Application 5. slide

What value does indicate the data variance with respect to a particular Eigenvector?

mathematics

Eigenvalue corresponds to that Eigenvector válasz:

Mathematics 4. slide:

We're also interested in the "variance" of each axis,  $\lambda_1$  and  $\lambda_2$  (ezek az eigenvaluek)

és Mathematics 5. slide eigenvector-value angol és magyar jelentése kommentekben

Magnitude of a Eigenvector:

The magnitude of a vector formula is used to calculate the length for a given vector (say  $v$ ) and is denoted as  $|v|$ . So basically, this quantity is the length between the initial point and endpoint of the vector.

The indicator showing the magnitude (tehát a vector hossza) change (change, tehát variance-cal van kapcsolatban) is called Eigenvalue. For example, if the eigenvalue is 1.2, it means that the magnitude of the vector gets larger than the original magnitude by 20% and if the eigenvalue is 0.8, it means the vector got smaller than the original vector by 20 %.

**„Tim:** This question shows that the magnitude of each eigenvector corresponds to a single eigenvalue per eigenvector. Your attention shall be raised that you decompose a matrix into eigenvalues and vectors, where the vector with the largest magnitude corresponds to the biggest eigenvalue."

What are the advantages of using SVD decomposition in computing the PCA method?

Mathematics 6-7. slide

SVD decomposition is more efficient than the Eigen decomposition method:

gondolom mert bajon a singular values, azaz szórás (variancia ezáltal),

meg mert meg tudjuk csinálni ezt: „Let us insert  $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{W}^T$  into the definition of the covariance matrix  $\mathbf{C}$ ”, amiből következnek ezek:

„• Eigenvalues correspond to squared singular values  $\lambda_i = \sigma_i^2$ , • Eigenvectors directly correspond to singular vectors”

SVD decomposition is a general method and applicable for any matrices: most az X mátrixon (2. slide) alkalmazza, és:

„This question was meant to clarify that SVD is a general method that finds a wide application field and that is why it is an advantage to use SVD instead of **eigenvalue decomposition**.”

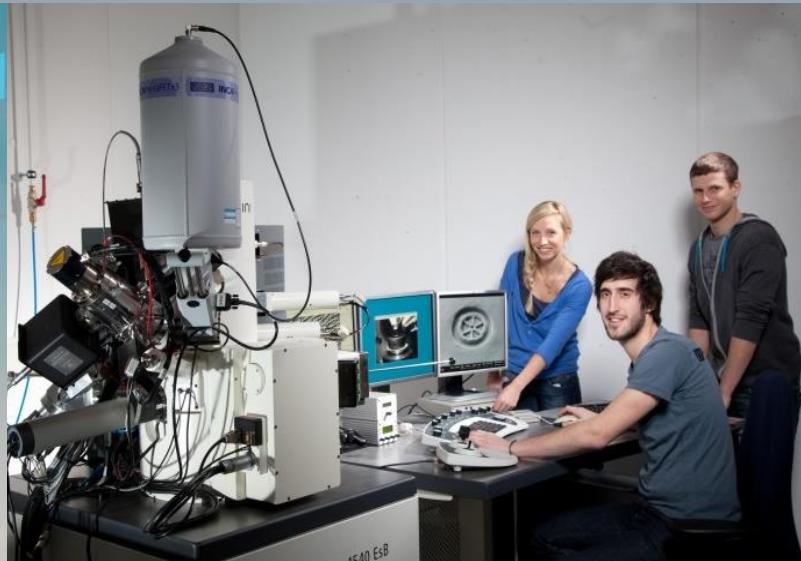
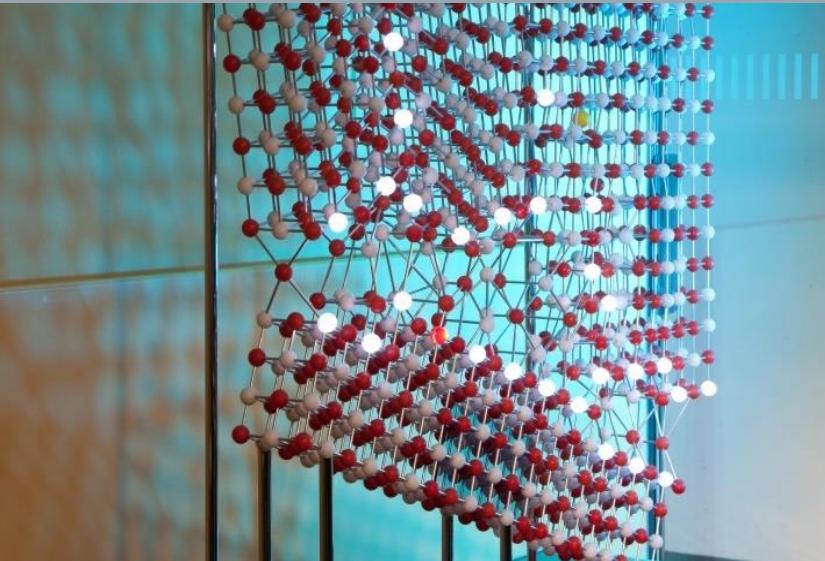
Why variance or spread of data with respect to a particular direction is an essential factor in dimensionality reduction?

egyértelmű, Intuition-ban van szerintem

Preserving data only in the direction of the eigenvectors with the highest variance (largest eigenvalue) means preserving more information by reducing the dimensionality

# Machine Learning for Engineers

## Deep Learning – Perceptron



Bilder: TF / Malter

# The Human Brain

The human brain is our reference for an intelligent agent, that

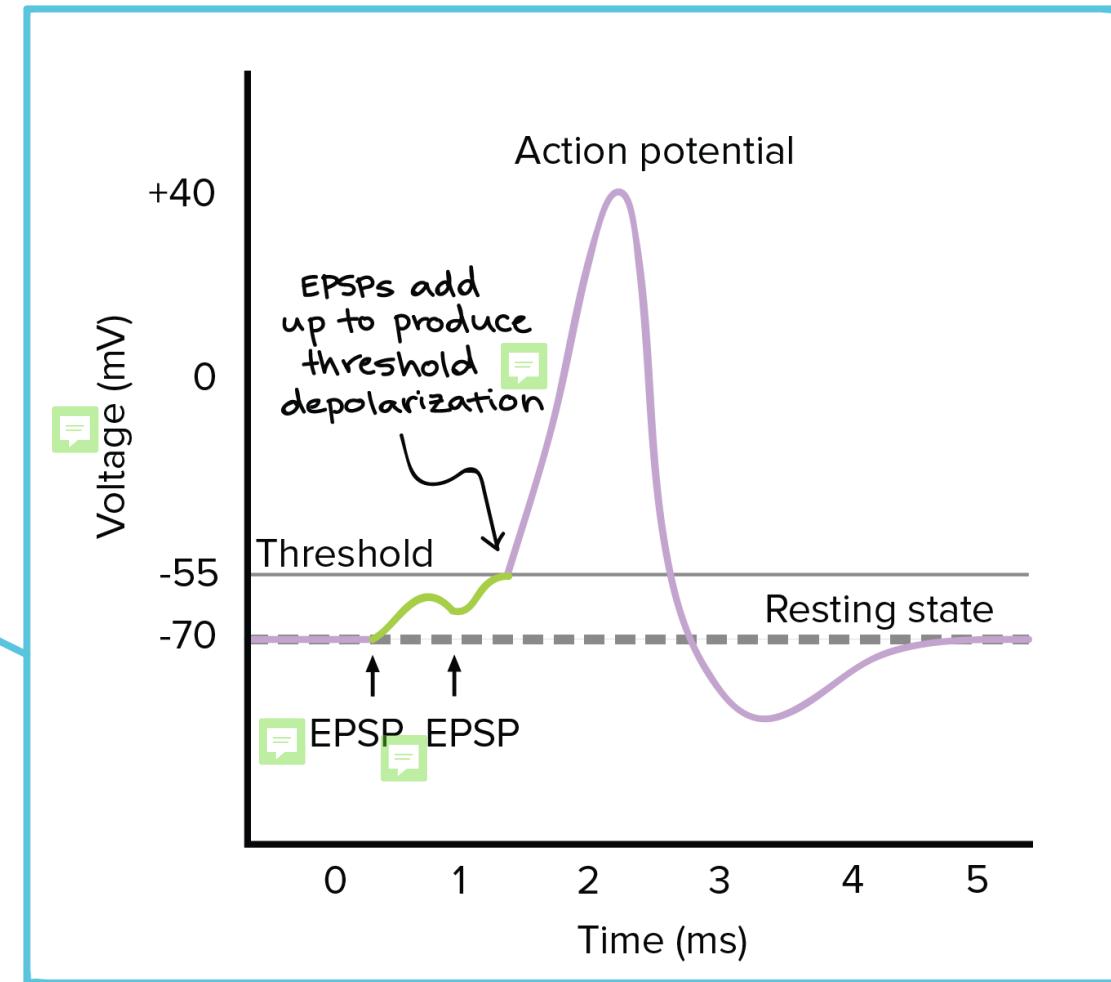
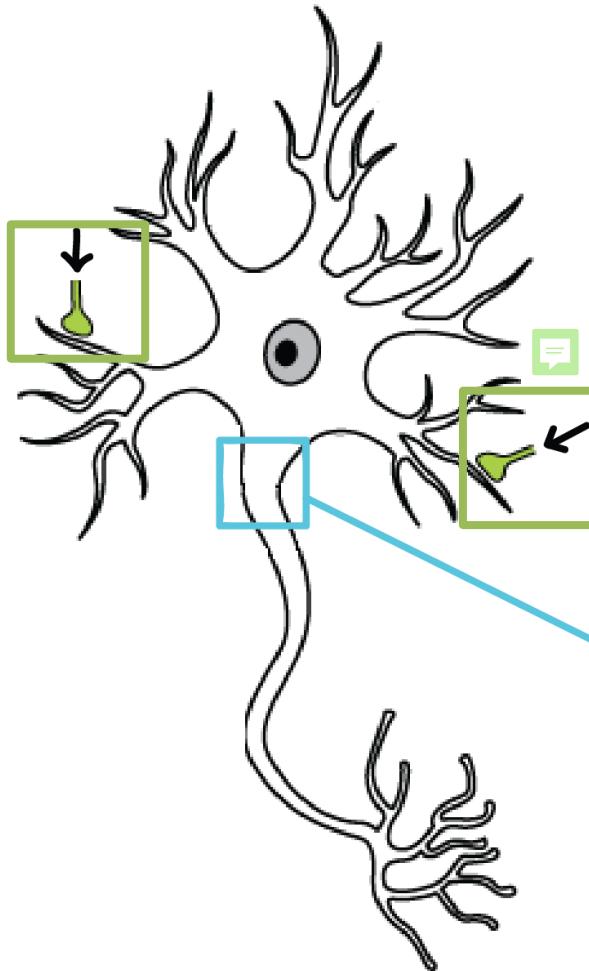
- a) ... contains different areas specialized for some tasks (e.g., the visual cortex)
- b) ... consists of neurons as the fundamental unit of “computation”



→ Let us have a closer look at the inner workings of a neuron 

Image from <https://ucresearch.tumblr.com/post/138868208574>

# The Human Neuron – Signaling Mechanism



1. Excitatory stimuli reach the neuron  
→ Input
2. Threshold is reached
3. Neuron fires and triggers action potential  
→ Output

Image from <https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/the-synapse>

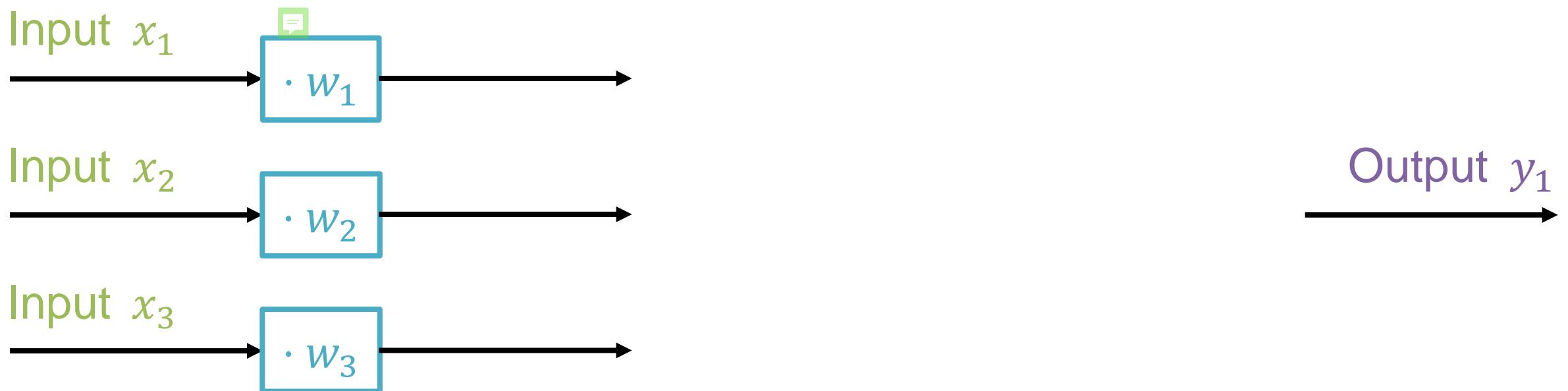
# The Perceptron – Computational Model of a Neuron

1. Let's start by adding some basic components (**input** and **output**), we're subsequently going to build the computational model step by step



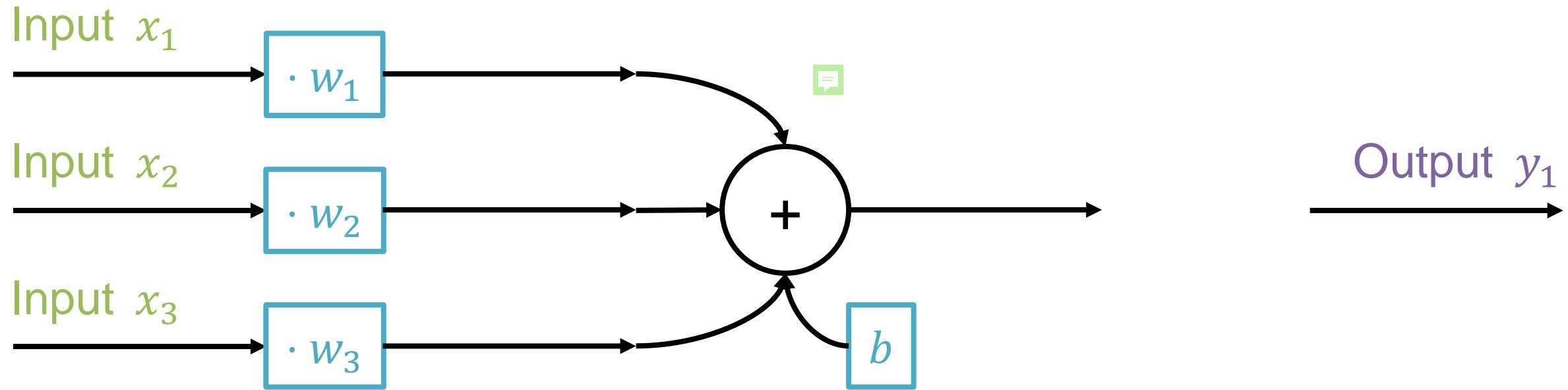
# The Perceptron – Computational Model of a Neuron

2. Next, let's add some **weights** to select and deselect **input** channels as not all are relevant to our model neuron



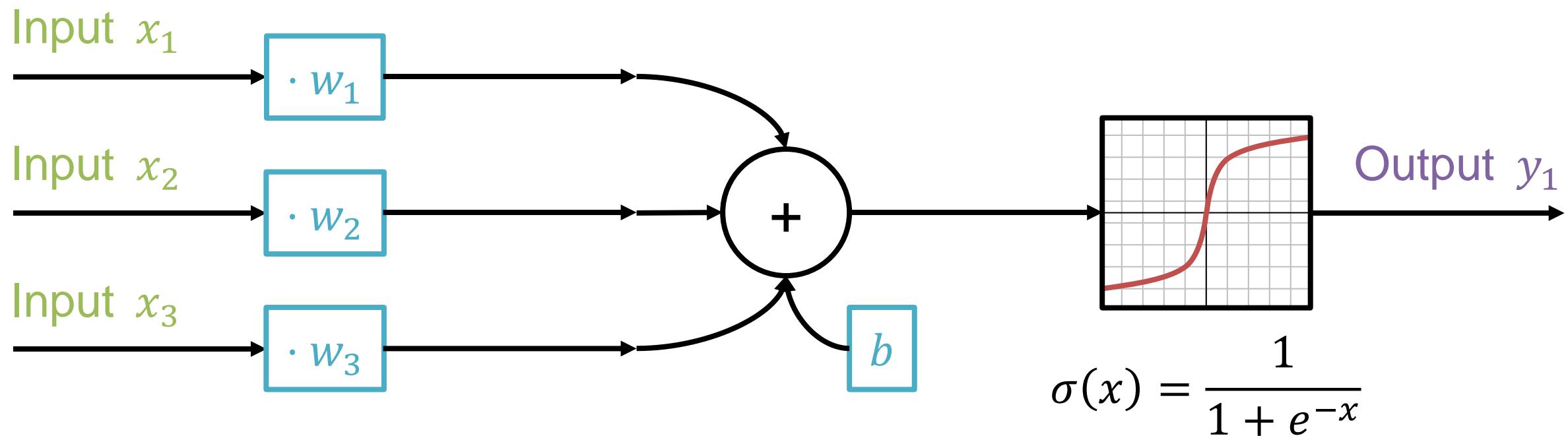
# The Perceptron – Computational Model of a Neuron

3. Then, let's add all the excitatory stimuli to the resting potential to determine the current potential of our model neuron



# The Perceptron – Computational Model of a Neuron

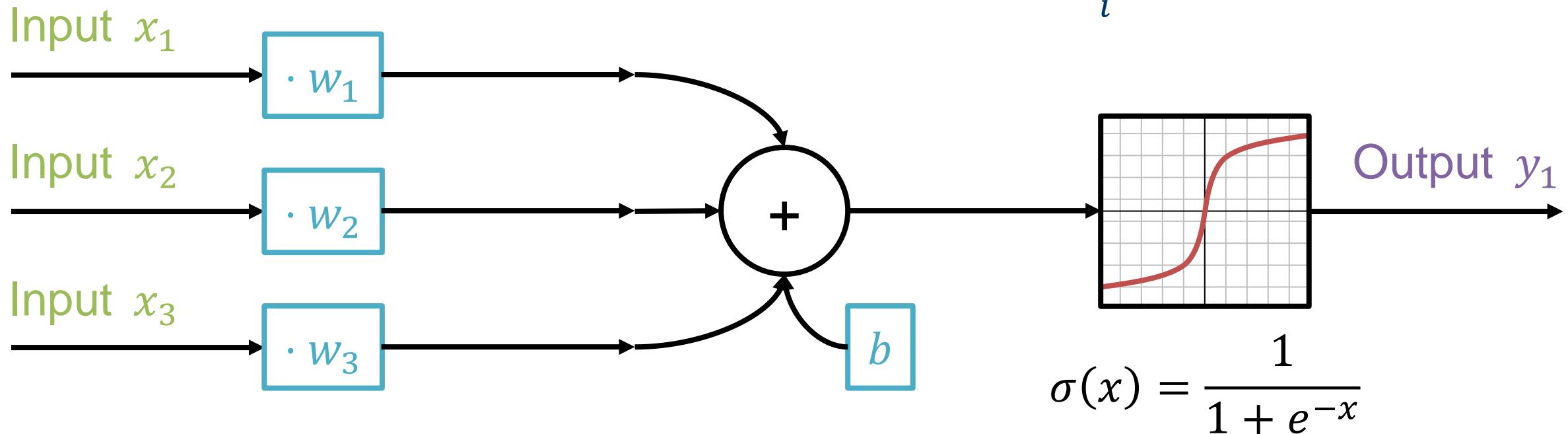
4. Finally, let's apply a **threshold** function  $\sigma$  (usually the sigmoid) to determine whether to send an action potential in the **output**



# The Perceptron – Computational Model of a Neuron

5. We can now write a perceptron as a mathematical function mapping the inputs  $x_1, x_2, x_3$  to the output  $y_1$  using channel weights  $w_1, w_2, w_3, b$ :

$$y_1 = \sigma(w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b) = \sigma\left(\sum_i w_i \cdot x_i + b\right)$$

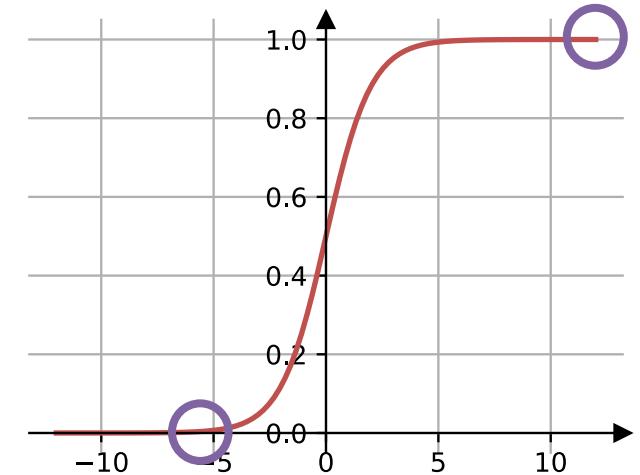


# The Perceptron – Signaling Mechanism

Given a perceptron with parameters

$w_1 = 4, w_2 = 7, w_3 = 11, b = -10$  and equation

$$y_1 = \sigma(w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b)$$



**Input  $x_1 = 1, x_2 = 0, x_3 = 0$  ("100")   Input  $x_1 = 1, x_2 = 1, x_3 = 1$  ("111")**

$$\begin{aligned} y_1 &= \sigma(4 \cdot 1 + 7 \cdot 0 + 11 \cdot 0 - 10) \\ &= \sigma(4 + 0 + 0 - 10) = \sigma(-6) \\ &= 0.00 \end{aligned}$$

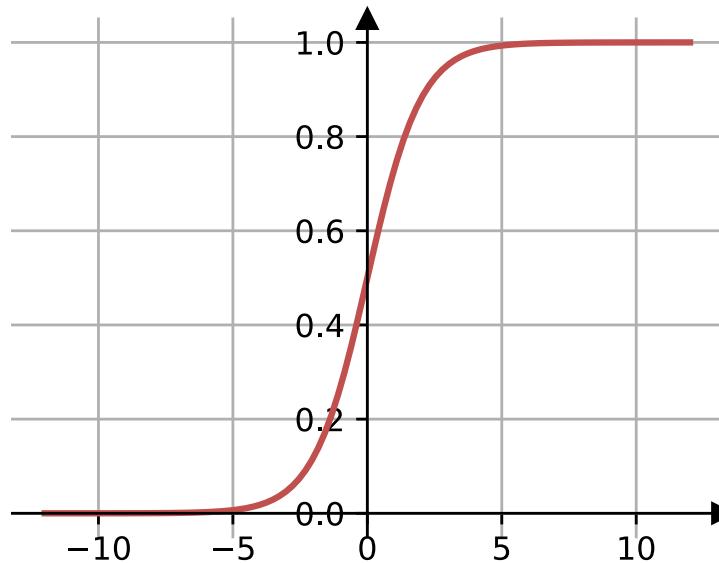
→ Output  $y_1$  not activated

$$\begin{aligned} y_1 &= \sigma(4 \cdot 1 + 7 \cdot 1 + 11 \cdot 1 - 10) \\ &= \sigma(4 + 7 + 11 - 10) = \sigma(12) \\ &= 1.00 \end{aligned}$$

→ Output  $y_1$  is activated

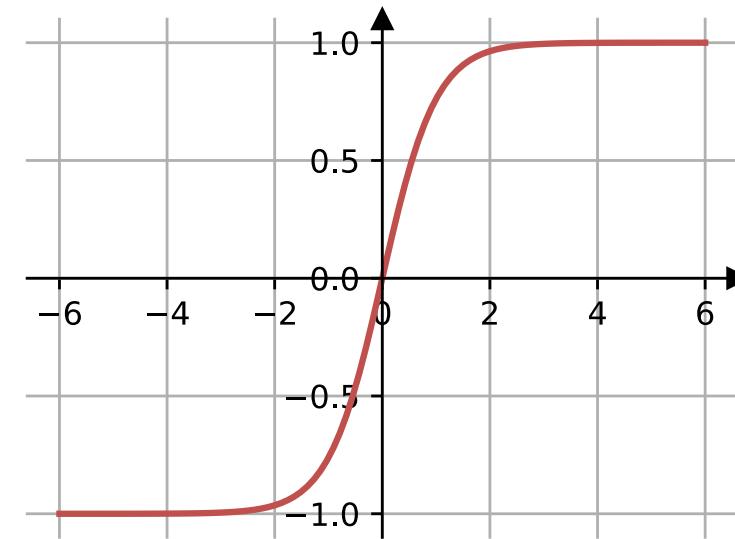
# Generalizing the Threshold: Activation Function

Sigmoid



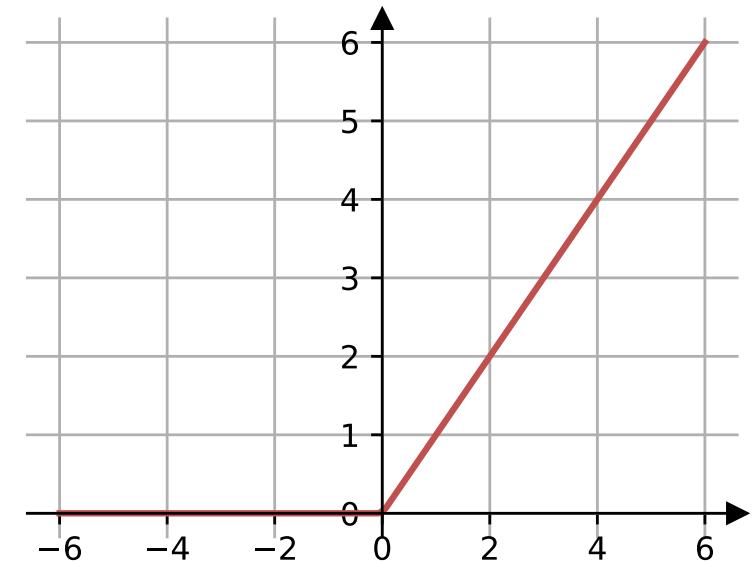
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Hyperbolic Tangent



$$\sigma(x) = \tanh(x)$$

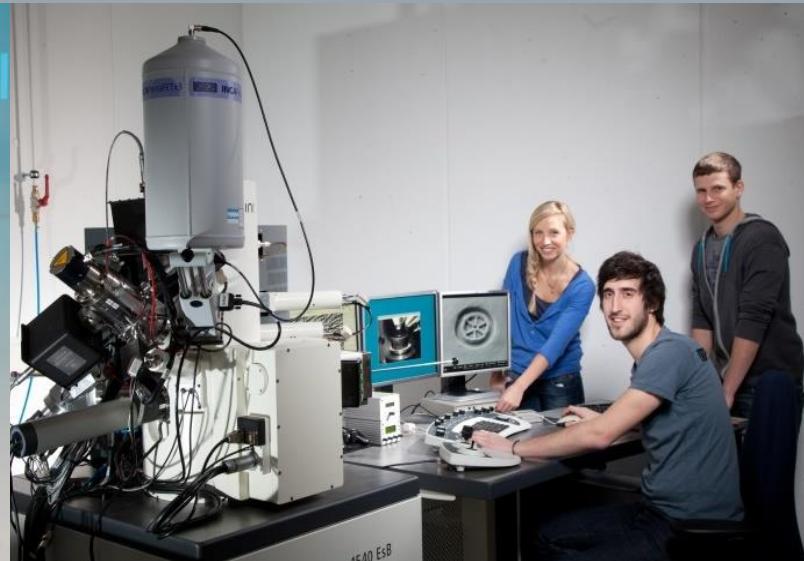
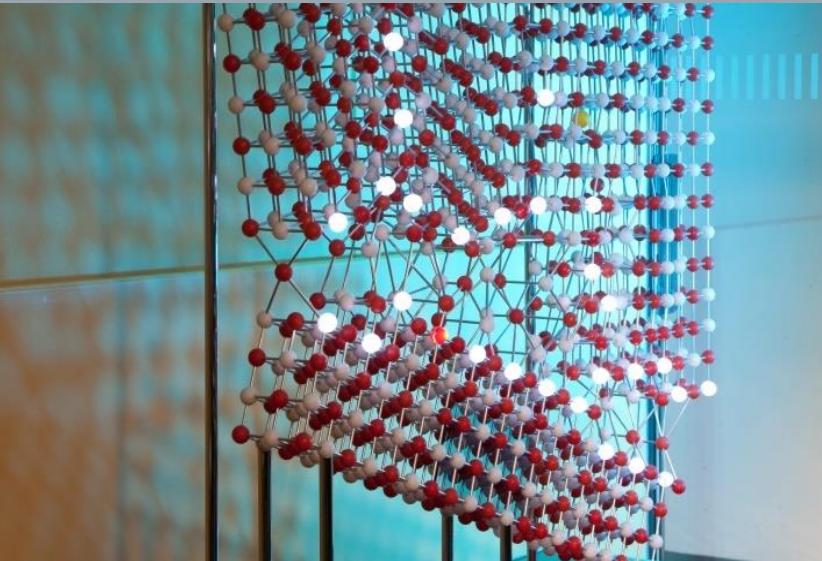
Rectified Linear Unit



$$\sigma(x) = \max(x, 0)$$

# Machine Learning for Engineers

## Deep Learning – Multilayer Perceptron



Bilder: TF / Malter

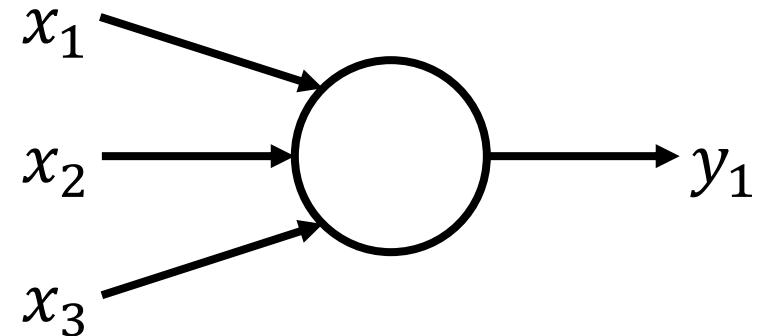
# The Perceptron – A Recap

In the last section we learned about the perceptron, a computational model representing a neuron.

**Inputs**  $x_1, \dots, x_n$

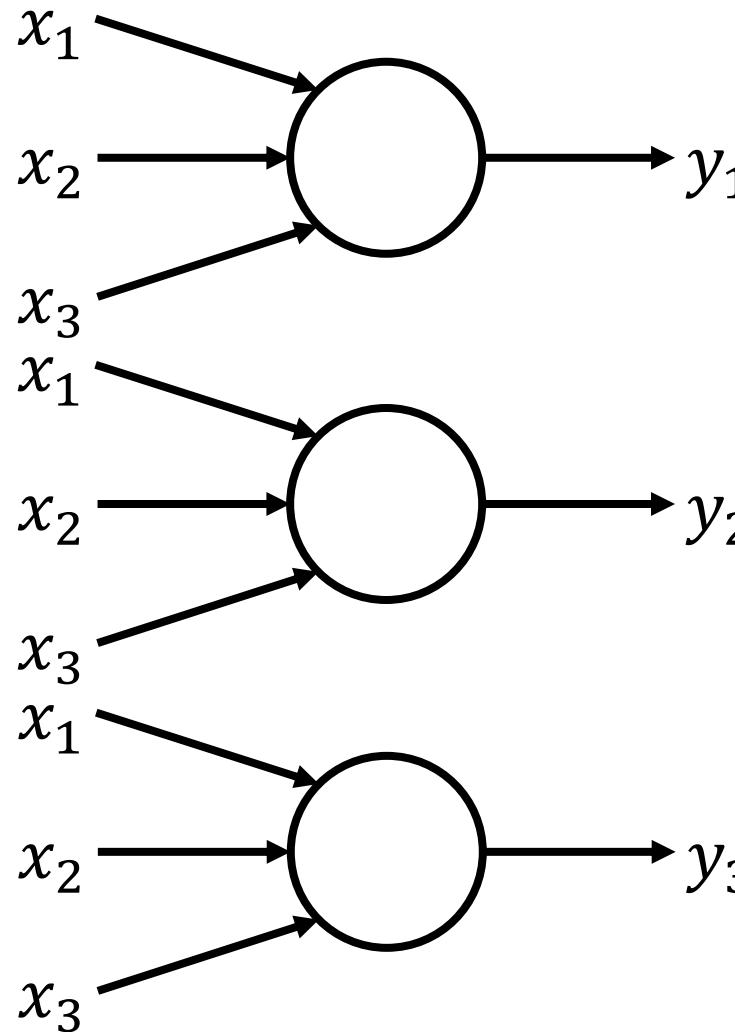
**Output**  $y_1$

**Computation**  $y_1 = \sigma(\sum_{i=1}^n w_i \cdot x_i + b)$



→ We can combine more than one perceptron for complex models 

# Combining Multiple Perceptron – Individual Notation



Let's now combine three perceptron with different outputs  $y_1$ ,  $y_2$  and  $y_3$ .

The computations for each these are:

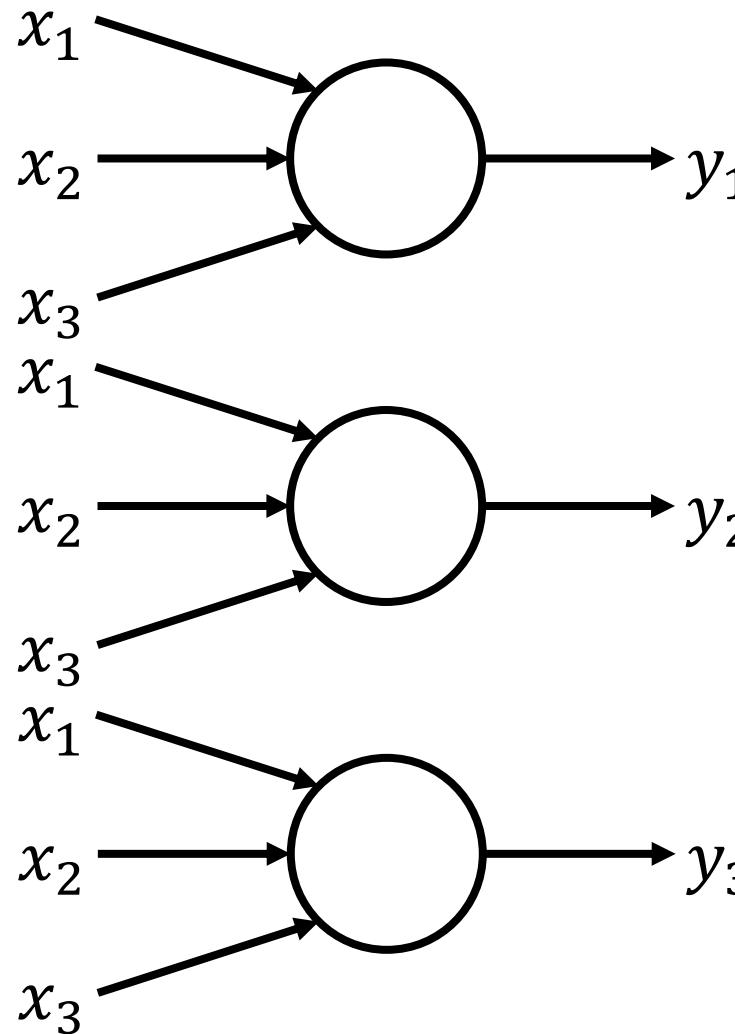
$$y_1 = \sigma(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1)$$

$$y_2 = \sigma(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2)$$

$$y_3 = \sigma(w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3)$$

Note how they all have their own parameters  
(weights  $w$  and bias  $b$ )

# Combining Multiple Perceptron – Matrix Notation



The outputs  $y$ , weights  $w$ , inputs  $x$  and bias  $b$  of each perceptron are matrices and vectors.

We can thus rewrite the three computations<sup>1)</sup> as:

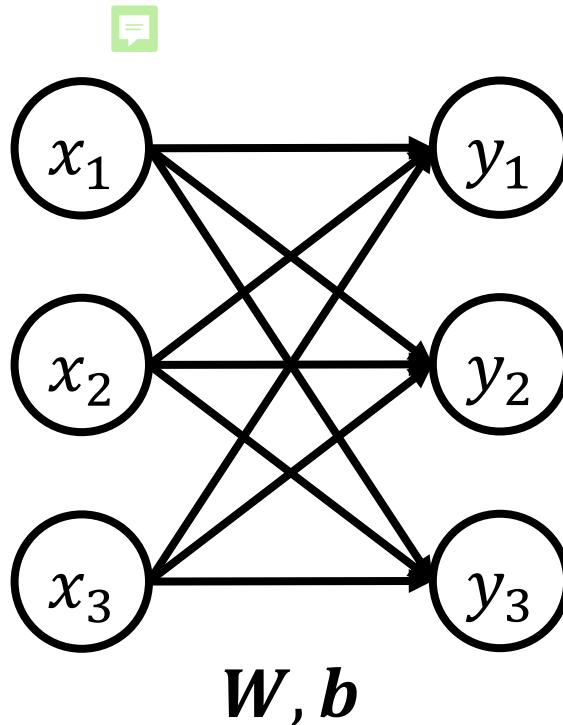
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Or in a more simplified form:

$$y = \sigma(W \cdot x + b)$$

1) This is a perfect opportunity to brush up your linear algebra skills. As an optional homework assignment, you can verify that the given matrix multiplication holds.

# Combining Multiple Perceptron – Single Layer



Instead of depicting each computation as a node or circle, we can also

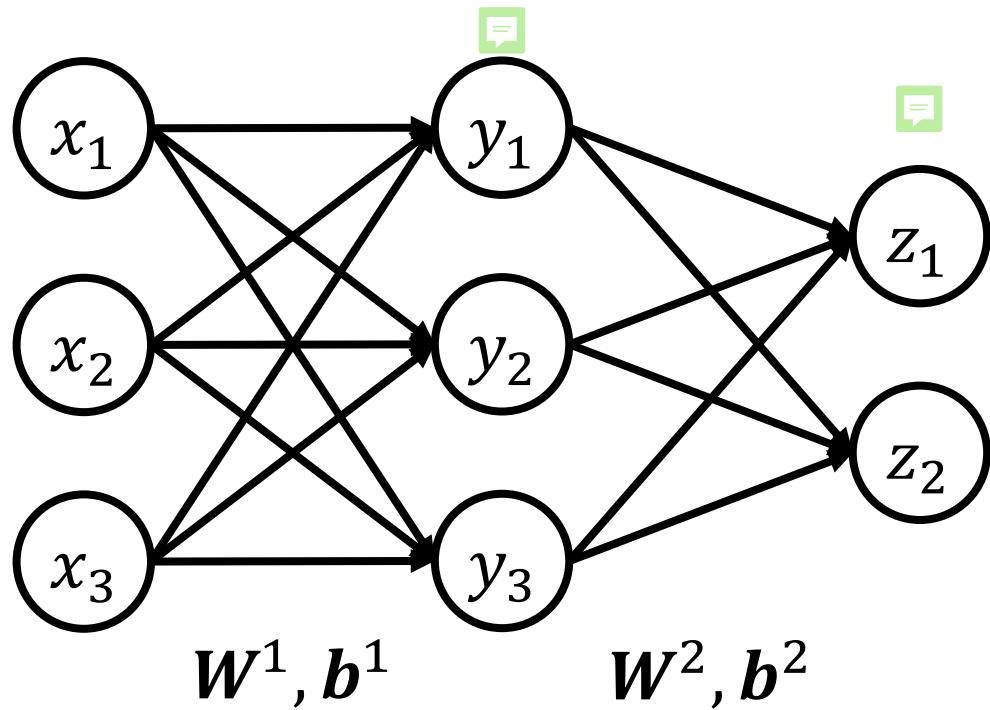
- Depict each value (input  $x$  and output  $y$ ) as a node or circle
- Depict each weighted connection as an arrow

This is the typical graphical representation<sup>1)</sup>, with the underlying computation remaining

$$y = \sigma(W \cdot x + b)$$

1) Note how the bias is implicitly assumed and not visualized.

# Combining Multiple Perceptron – Multiple Layers



There's no reason to limit ourselves to a single layer.

We can chain multiple layers, with each output being the input of the next:

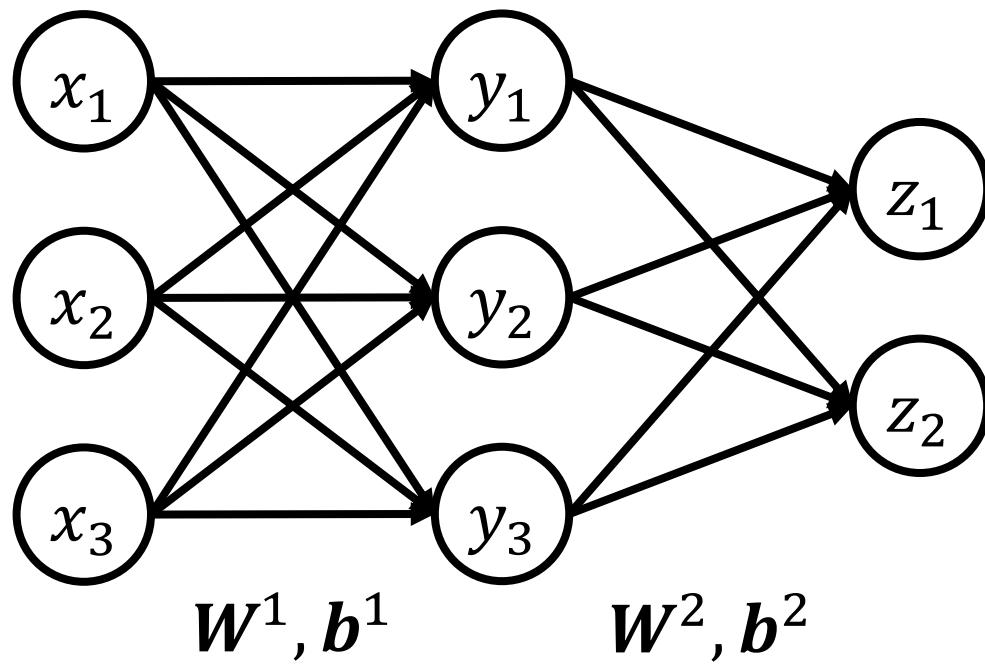
$$\mathbf{y} = \sigma(\mathbf{W}^1 \cdot \mathbf{x} + \mathbf{b}^1)$$

$$\mathbf{z} = \sigma(\mathbf{W}^2 \cdot \mathbf{y} + \mathbf{b}^2)$$

Combining these leads us to:

$$\mathbf{z} = \sigma(\mathbf{W}^2 \cdot \sigma(\mathbf{W}^1 \cdot \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2)$$

# Multilayer Perceptron – A Summary

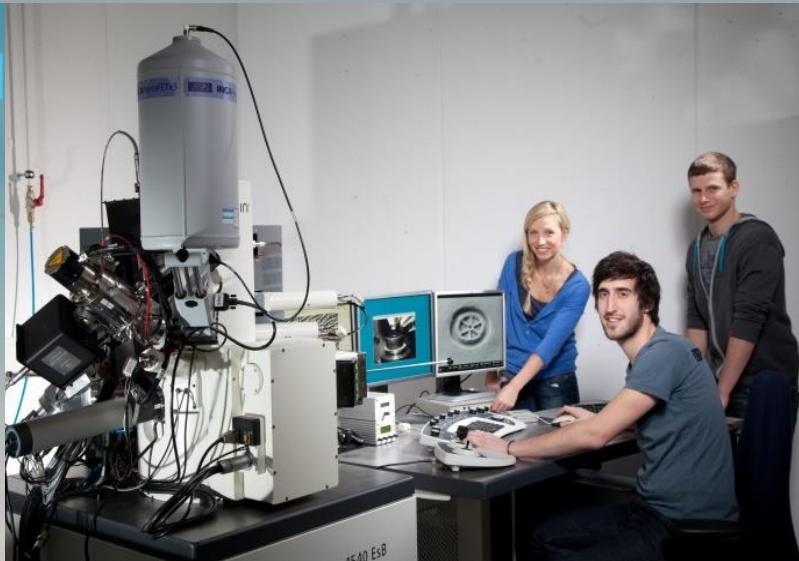
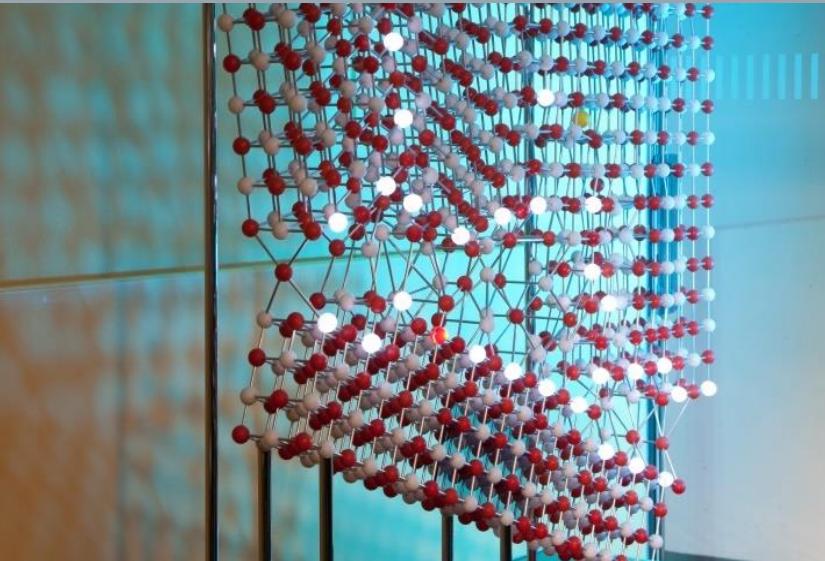


- A multilayer perceptron is a neural network with multiple perceptron
- It is oftentimes organized into layers
- Each layer has its own set of parameters (weights  $W^i$  and bias  $b^i$ )
- The underlying computation is a matrix multiplication described by

$$y^{i+1} = \sigma(W^i \cdot y^i + b^i)$$

# Machine Learning for Engineers

## Deep Learning – Loss Function



Bilder: TF / Malter

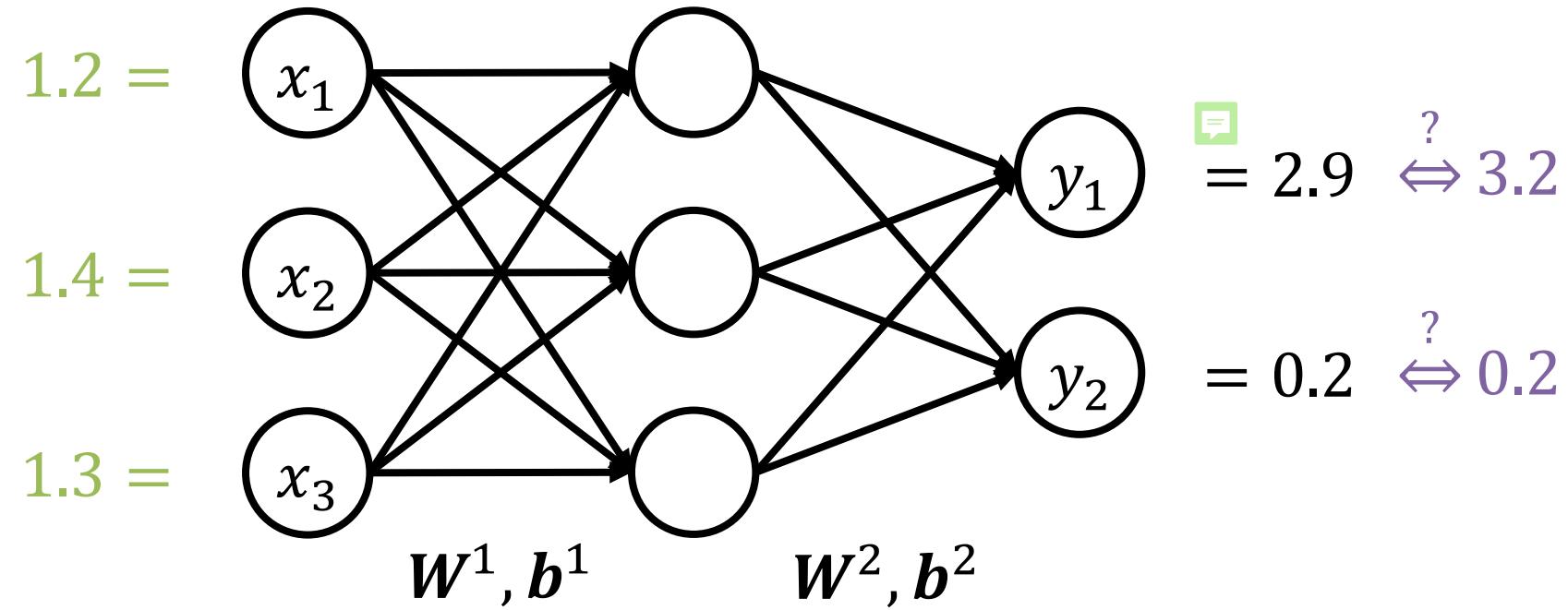
# How do our models learn?

- Up until now, the parameters  $W^i$  and  $b^i$  of the multilayer perceptron were assumed as given
- We now aim to learn the parameters  $W^i$  and  $b^i$  based on some example input and output data
- Let us assume we have a dataset of  $x$  and corresponding  $y$

$$(x_1, y_1) = \begin{pmatrix} 1.2 \\ 1.4 \\ 3.2 \\ 1.3 \end{pmatrix} \text{ and } (x_2, y_2) = \begin{pmatrix} 0.2 \\ 0.4 \\ 0.2 \\ 0.3 \end{pmatrix} \text{ and ...}$$

# How do our models learn?

- We now want to ensure that the parameters  $W^i$  and  $b^i$  can correctly predict the  $y_i$  based on the  $x_i$   
- To do this, we apply  $x_i$  on our model and compare the result with  $y_i$



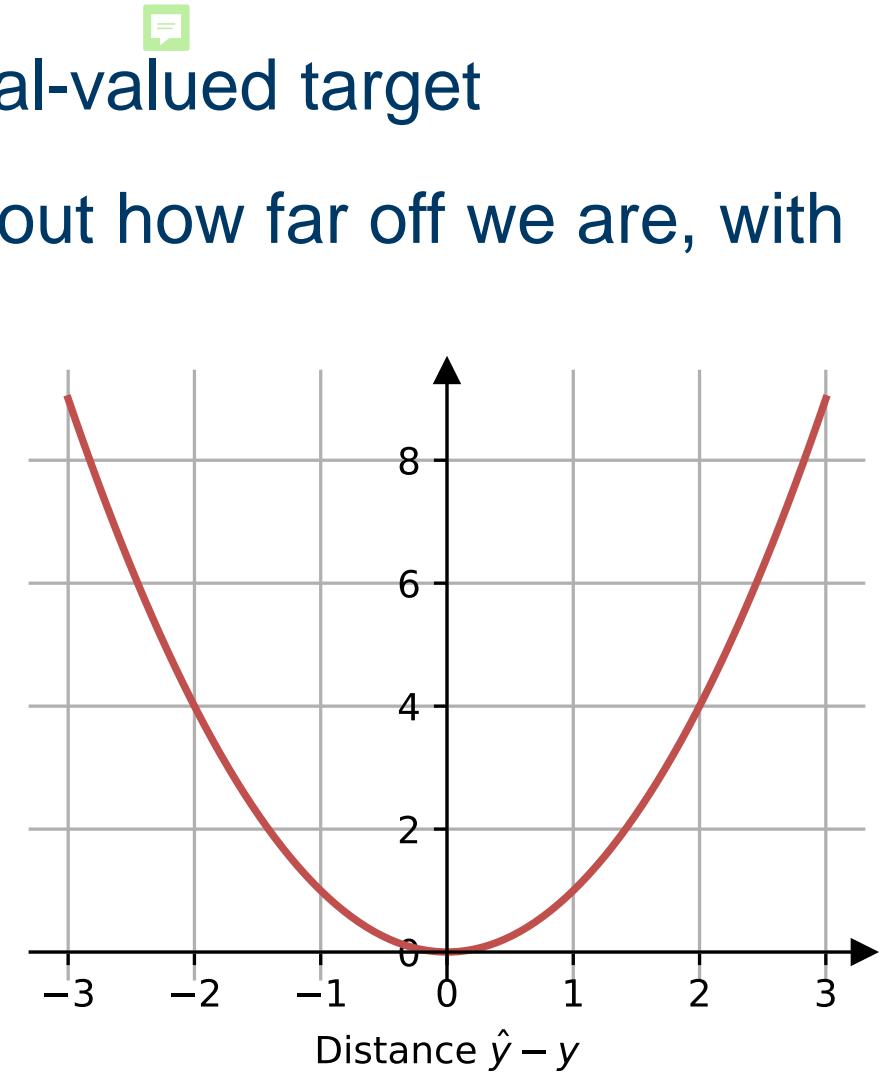
# The Loss Function

- We need a comparison metric between the predicted outputs  $\hat{y}_i$  and the expected outputs  $y_i$
- This is called the loss function, which usually depends on the type of problem the multilayer perceptron solves
  - For **regression**, a common metric is the mean squared error
  - For **classification**, a common metric is the cross entropy

# Mean Squared Error

- For regression, each output  $y_1, \dots, y_n$  is a real-valued target
- The difference  $\hat{y}_i - y_i$  tells us something about how far off we are, with the mean square error computed as

$$L(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$



# Cross Entropy – Class Probability and Softmax

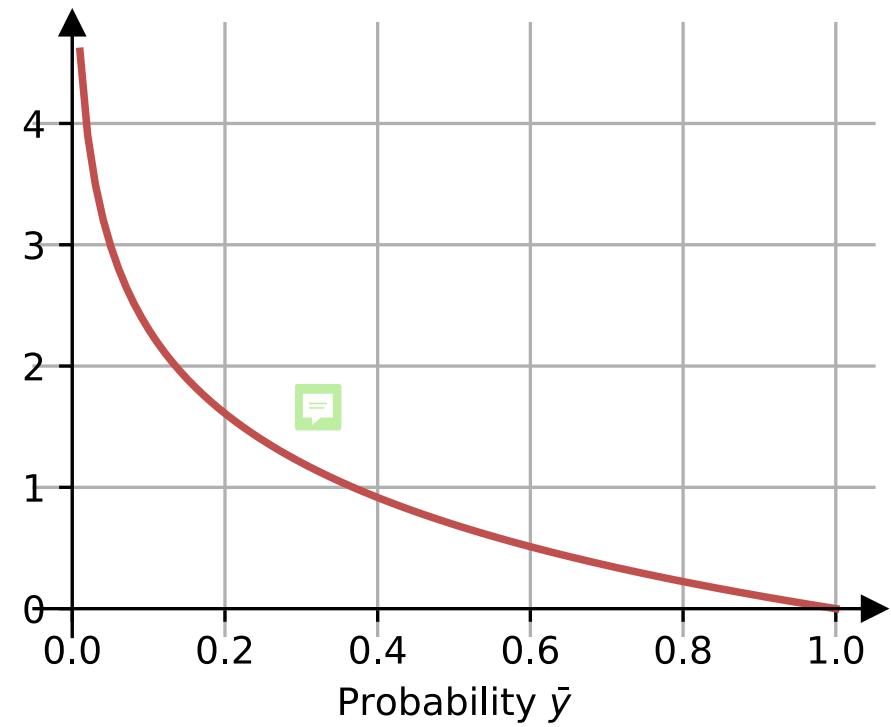
- For classification, each output  $y_1, \dots, y_n$  represents a class energy
- If the target class index is  $i$ , then only  $y_i = 1$  and all other  $y_j = 0, i \neq j$  (this is also called one-hot encoding)
  - Example for class 1:  $y = (1 \ 0 \ 0 \ 0 \ 0)^T$
  - Example for class 3:  $y = (0 \ 0 \ 1 \ 0 \ 0)^T$
- To ensure that we have probabilities, we apply a softmax transformation

$$\bar{y}_i = \frac{\exp(\hat{y}_i)}{\sum_{j=1}^n \exp(\hat{y}_j)}$$

# Cross Entropy – Negative Log Likelihood

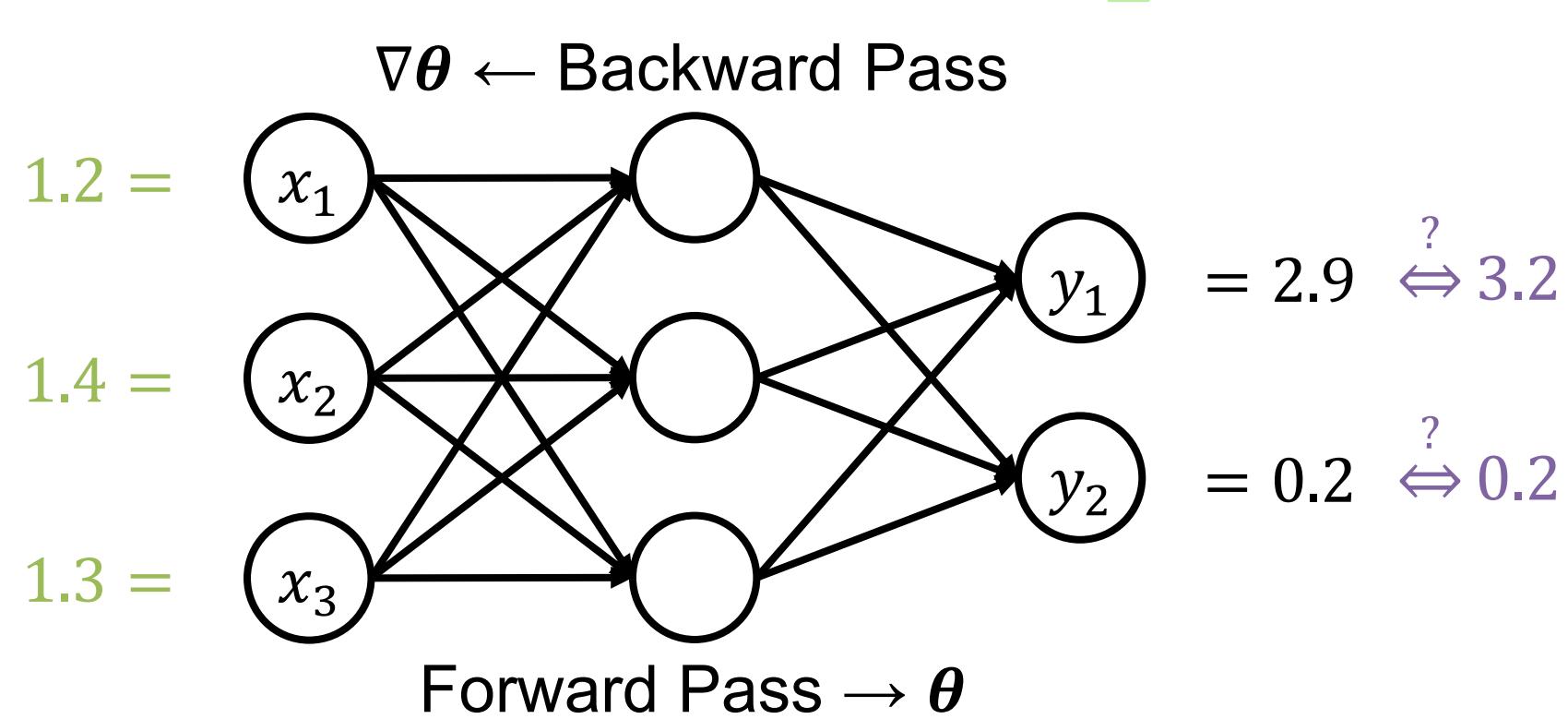
- After the transformation, each  $\bar{y}_1, \dots, \bar{y}_n$  represents a class probability
- The negative log –  $-\log \bar{y}_i$  for the target class index  $i$  tells us how far off we are, with the cross entropy calculated as

$$\begin{aligned} L(\hat{y}, y) &= -\log \bar{y}_i \\ &= -\log \frac{\exp(\hat{y}_i)}{\sum_{j=1}^n \exp(\hat{y}_j)} \end{aligned}$$



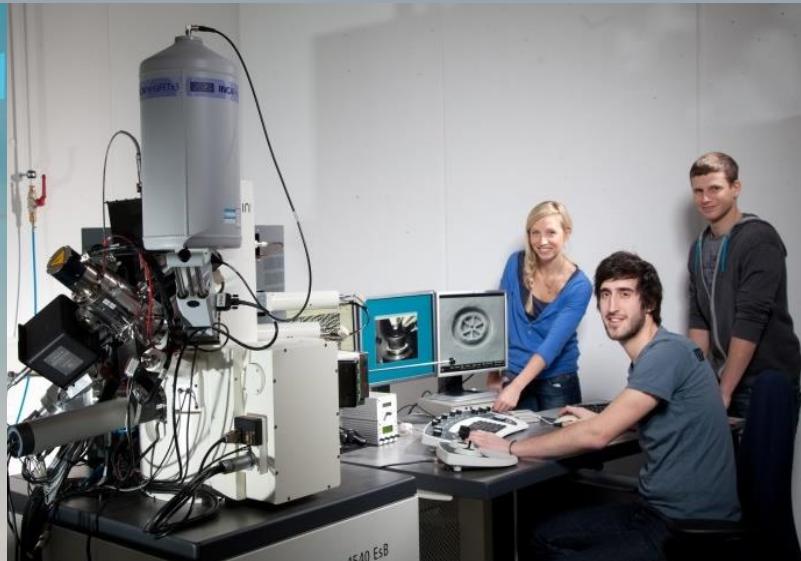
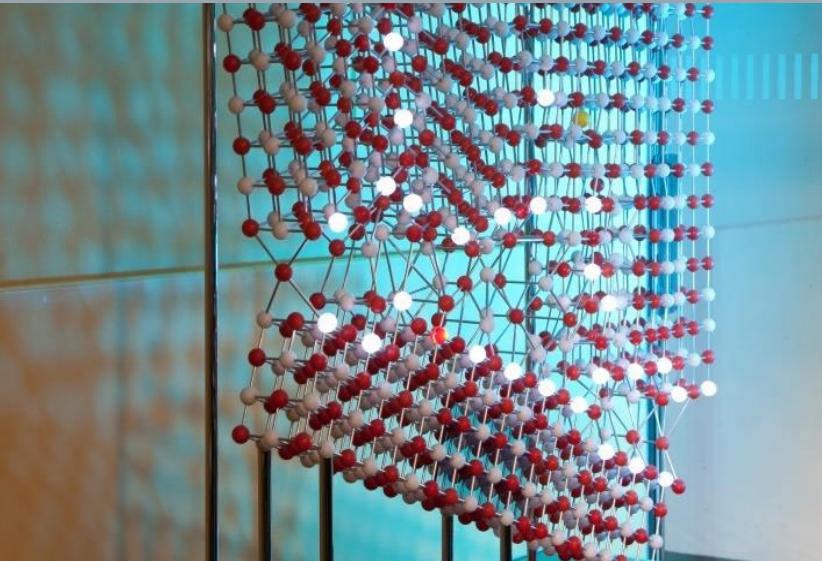
# How do our models learn? – A Summary

- How wrong is the current set of parameters  $\theta$ ? **Forward Pass**
- How should we change the set of parameters by  $\nabla\theta$ ? **Backward Pass**



# Machine Learning for Engineers

Deep Learning – Gradient Descent



Bilder: TF / Malter

# Parameter Optimization

- At this stage, we can compute the gradient of the parameters  $\nabla\theta$
- The gradient tells us how we need to change the current parameters  $\theta$  in order to make fewer errors on the given data
- We can use this in an iterative algorithm called gradient descent, with the central equation being

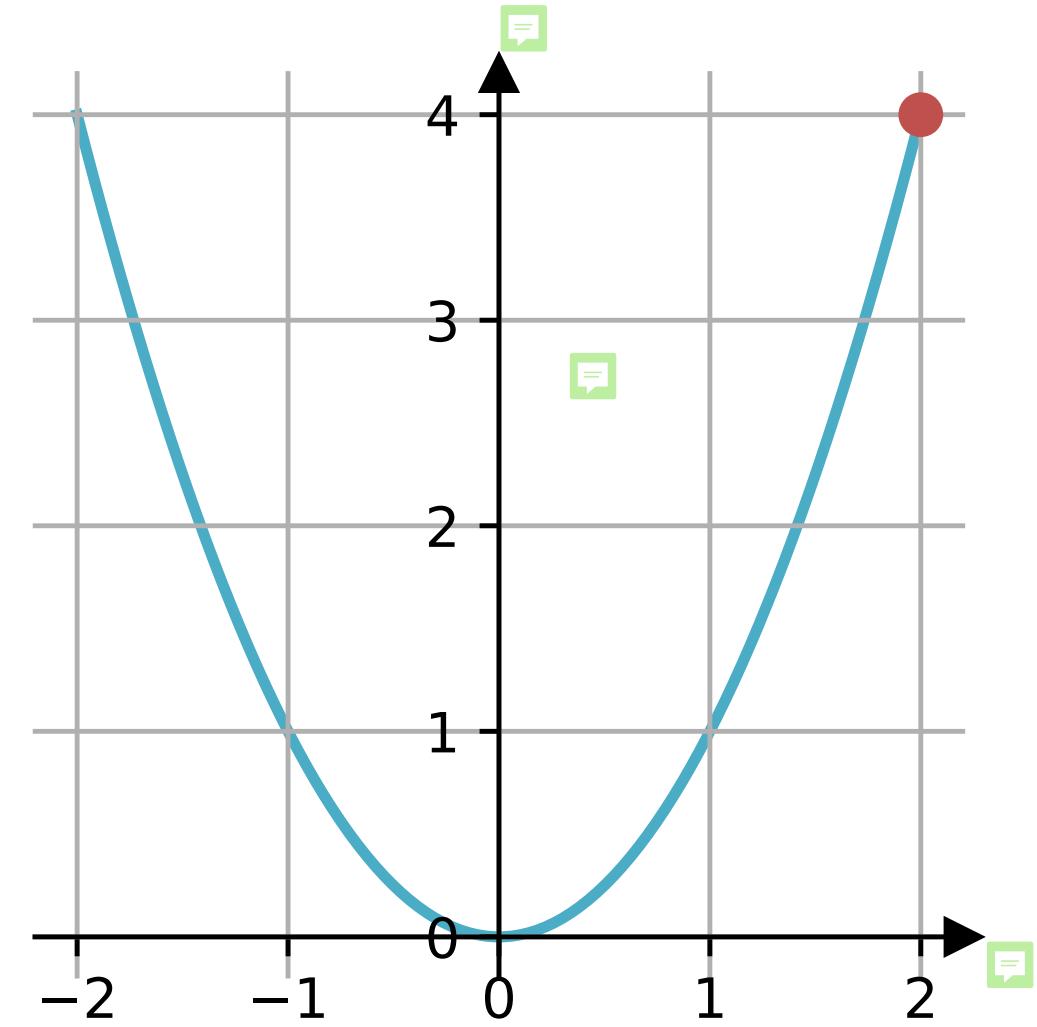
$$\theta^{i+1} = \theta^i - \mu \cdot \nabla\theta^i$$

- The learning rate  $\mu$  tells us how quickly we should change the current parameters  $\theta^i$

→ Let us have a closer look at an illustrative example 

# Gradient Descent – An Example

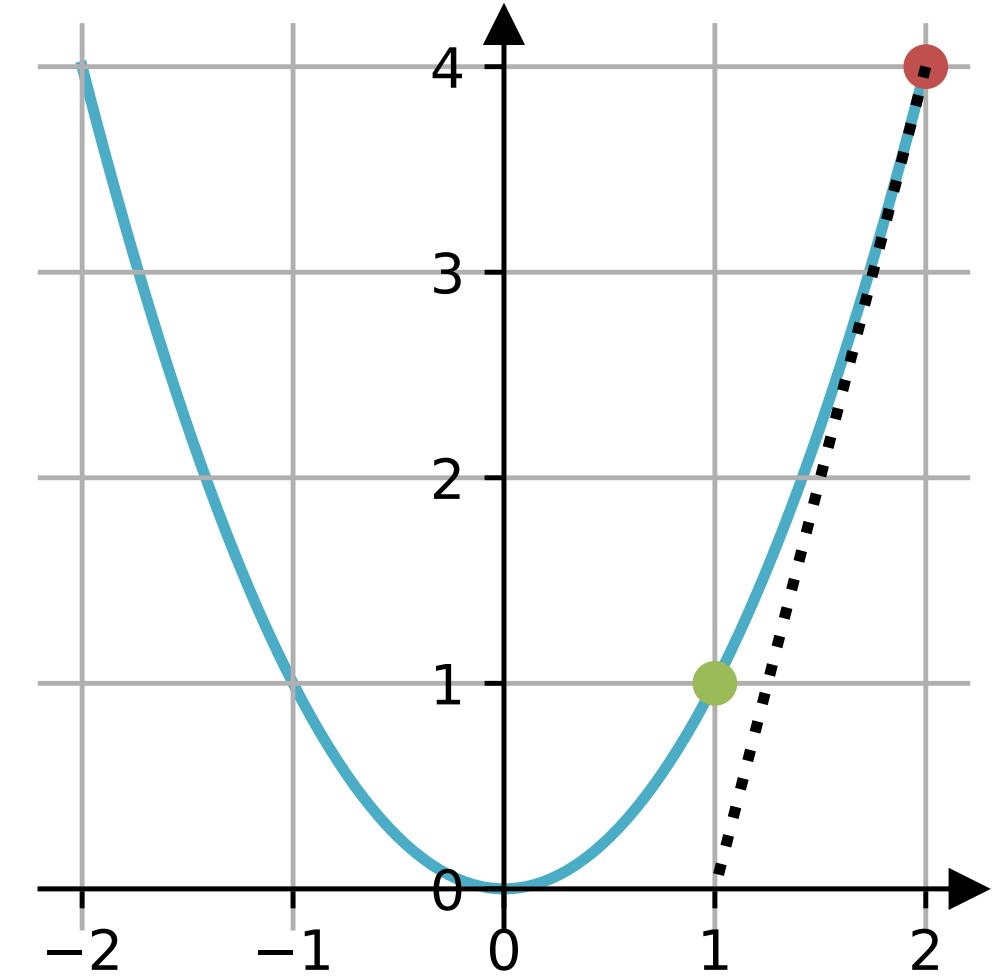
- Let us assume that the error function is  $f(\theta) = \theta^2$  
- The gradient of the error function is the derivative  $f'(\theta) = 2 \cdot \theta$  
- We aim to find the minimum error at the location  $\theta = 0$
- Our initial estimate for the location is  $\theta^1 = 2$  
- The learning rate is  $\mu = 0.25$



# Gradient Descent – An Example

- 1. Step:  $\theta^1 = 2$ ,  $\nabla\theta^1 = 4$

$$\theta^2 = \theta^1 - \mu\nabla\theta^1 = 2 - 0.25 \cdot 4 = 1$$



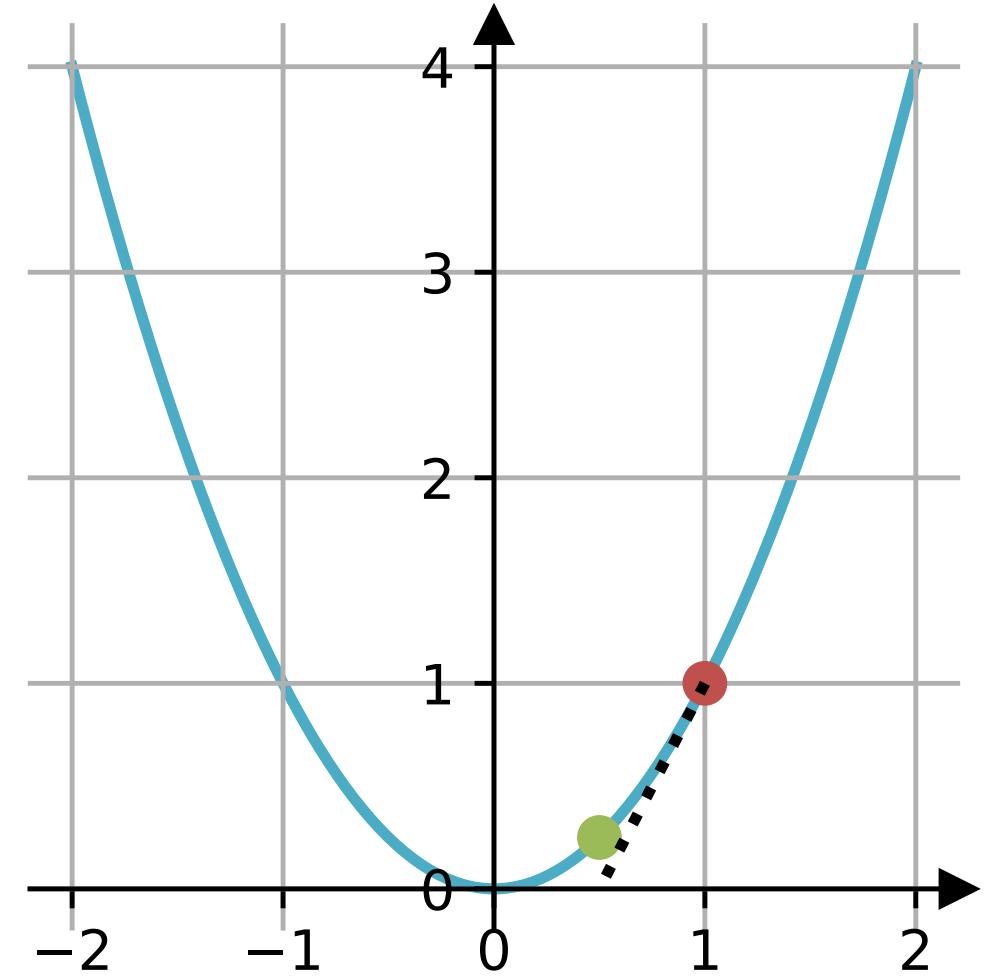
# Gradient Descent – An Example

- 1. Step:  $\theta^1 = 2, \nabla\theta^1 = 4$

$$\theta^2 = \theta^1 - \mu\nabla\theta^1 = 2 - 0.25 \cdot 4 = 1$$

- 2. Step:  $\theta^2 = 1, \nabla\theta^2 = 2$

$$\theta^3 = \theta^2 - \mu\nabla\theta^2 = 1 - 0.25 \cdot 2 = 0.5$$



# Gradient Descent – An Example

- 1. Step:  $\theta^1 = 2, \nabla\theta^1 = 4$

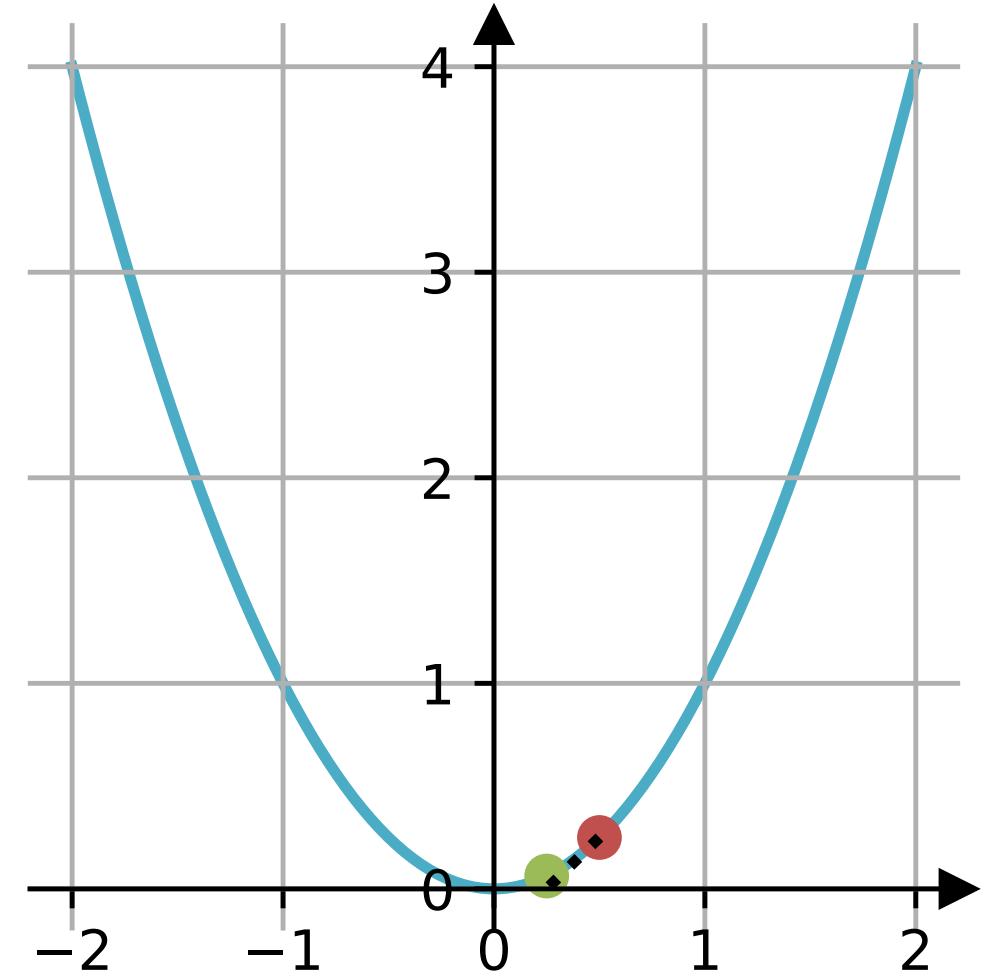
$$\theta^2 = \theta^1 - \mu\nabla\theta^1 = 2 - 0.25 \cdot 4 = 1$$

- 2. Step:  $\theta^2 = 1, \nabla\theta^2 = 2$

$$\theta^3 = \theta^2 - \mu\nabla\theta^2 = 1 - 0.25 \cdot 2 = 0.5$$

- 3. Step:  $\theta^3 = 0.5, \nabla\theta^3 = 1$

$$\theta^4 = \theta^3 - \mu\nabla\theta^3 = 0.5 - 0.25 \cdot 1 = 0.25$$



# Gradient Descent – An Example

- 1. Step:  $\theta^1 = 2, \nabla\theta^1 = 4$

$$\theta^2 = \theta^1 - \mu\nabla\theta^1 = 2 - 0.25 \cdot 4 = 1$$

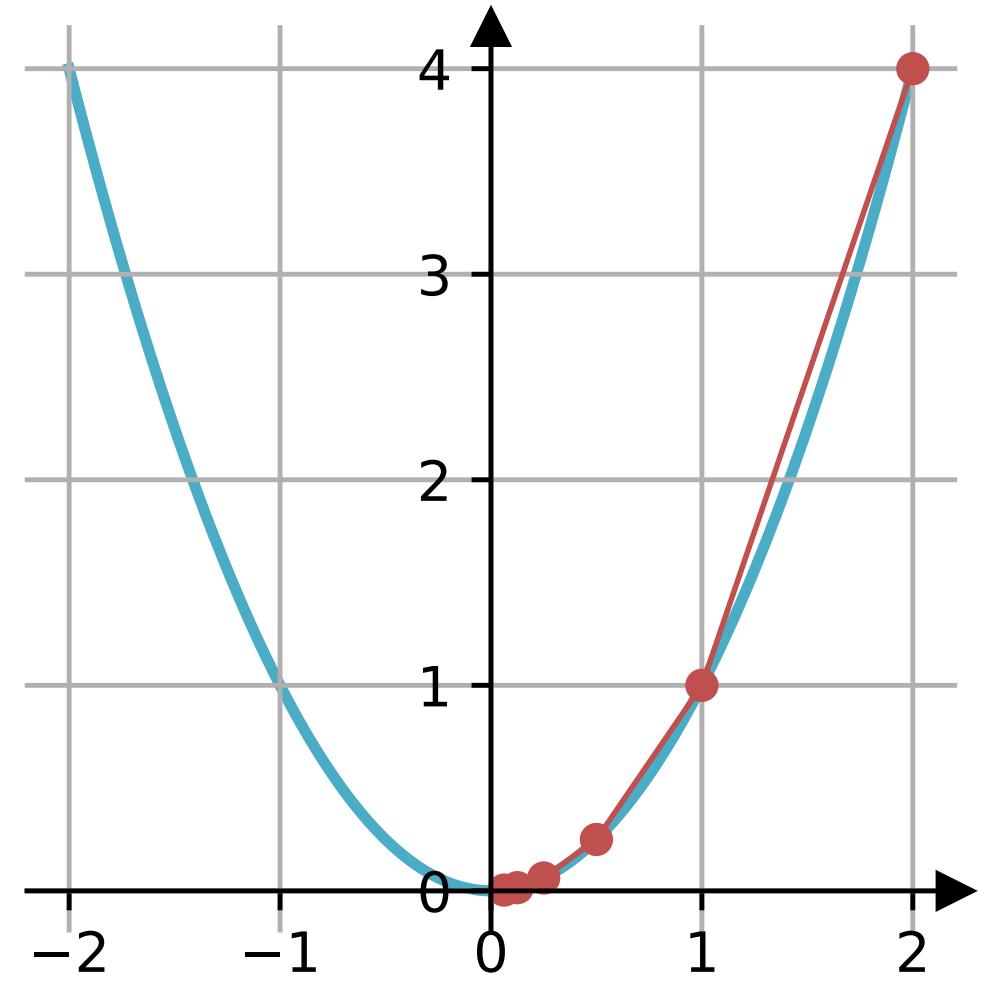
- 2. Step:  $\theta^2 = 1, \nabla\theta^2 = 2$

$$\theta^3 = \theta^2 - \mu\nabla\theta^2 = 1 - 0.25 \cdot 2 = 0.5$$

- 3. Step:  $\theta^3 = 0.5, \nabla\theta^3 = 1$

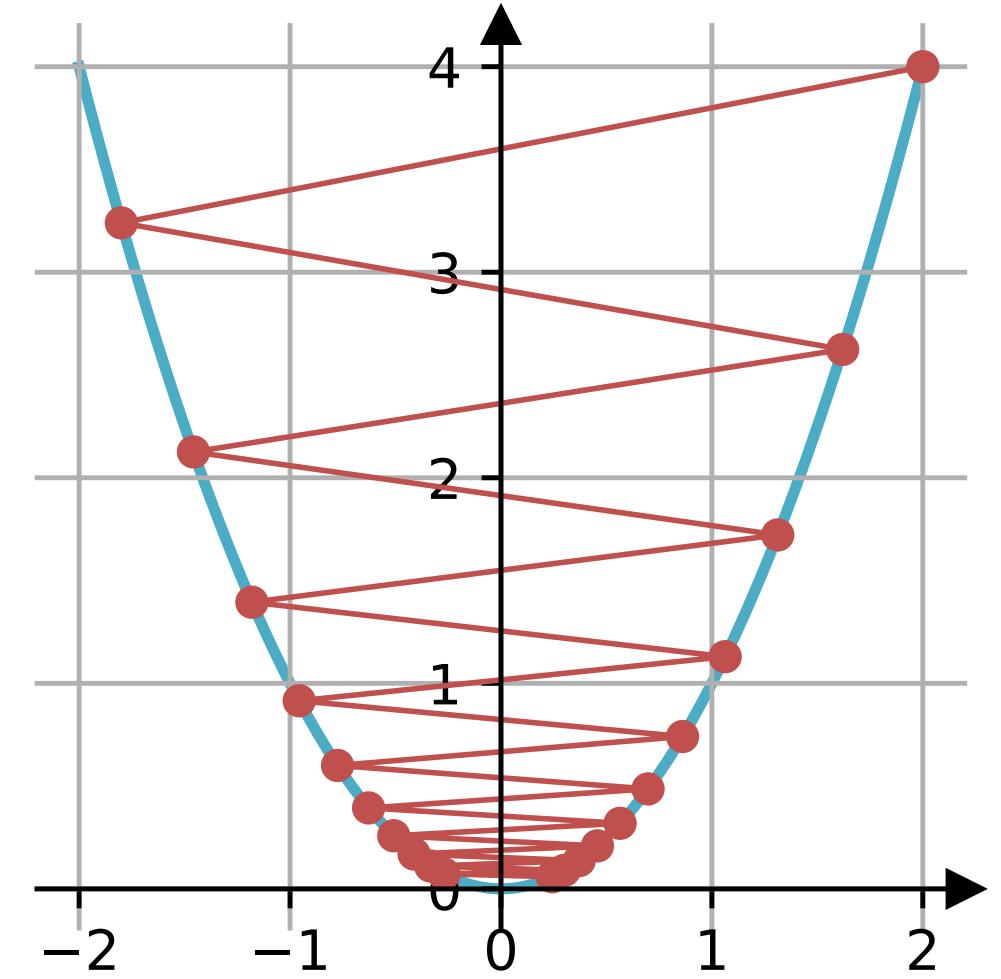
$$\theta^4 = \theta^3 - \mu\nabla\theta^3 = 0.5 - 0.25 \cdot 1 = 0.25$$

- We incrementally approach the expected target  $\theta = 0$  



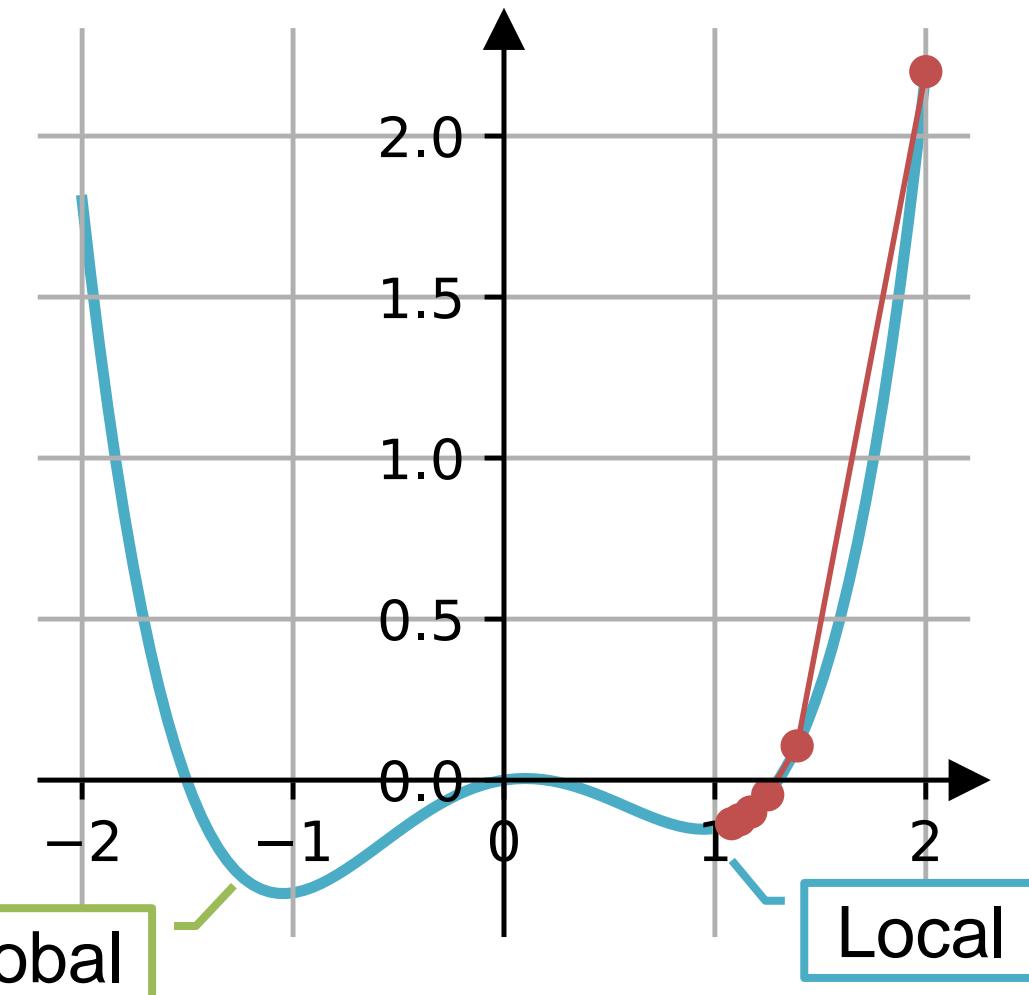
# Gradient Descent – Limitations

- The learning rate  $\mu$  can lead to slow convergence if not properly configured (see example for  $\mu = 0.95$ ) 



# Gradient Descent – Limitations

- The learning rate  $\mu$  can lead to slow convergence if not properly configured (see example for  $\mu = 0.95$ )
- The starting parameters  $\theta^1$  can lead to a solution stuck in local minimum (see example for  $f(\theta) = 0.5\theta^4 - 0.25\theta^2 + 0.1\theta$ )
- There's no guarantee to finding an optimal solution (a global minimum)

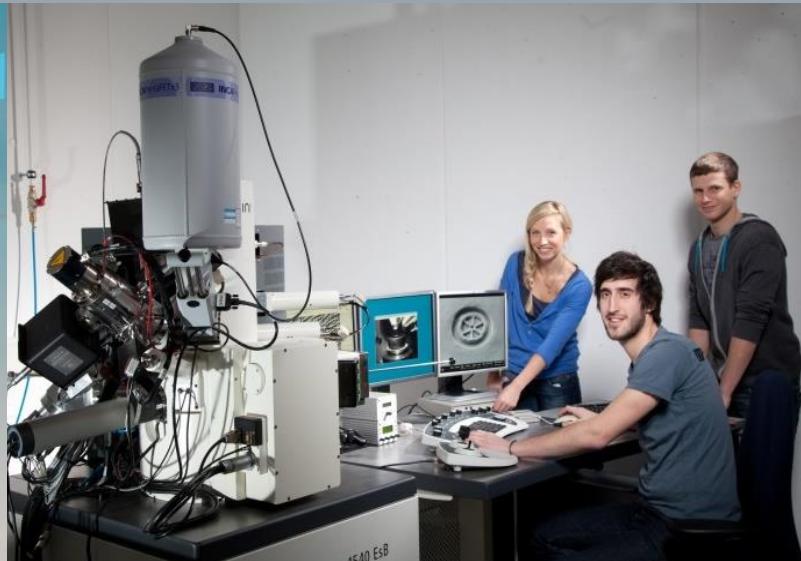
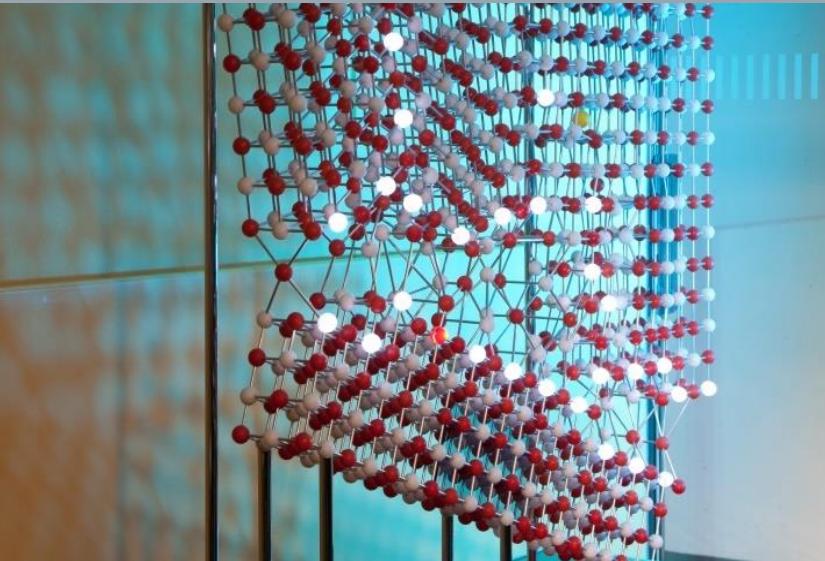


# Gradient Descent – A Summary

- Gradient Descent incrementally adjusts the parameters  $\theta$  based on the gradient  $\nabla\theta$  of the parameters
  1. For each iteration
  2. Compute the error of the parameters  $\theta$
  3. Compute the gradient of the parameters  $\nabla\theta$
  4. Update parameters using  $\theta^{i+1} = \theta^i - \mu \cdot \nabla\theta^i$
- There is no guarantee that the algorithm converges to the global optimum (e.g., due to learning rate  $\mu$  or initial parameters  $\theta^1$ )

# Machine Learning for Engineers

## Deep Learning – Learning Process



Bilder: TF / Malter

# How do our models learn?

- We've just talked about the concept of gradient descent, but largely avoided the details for the function  $f(\theta)$ <sup>1</sup>
- The goal is to minimize the error or loss function across the complete dataset  $(x_1, y_1), \dots, (x_n, y_n)$

$$f(\theta) = \sum_{i=1}^n L(\hat{y}_i, y_i) = \sum_{i=1}^n L(g(x_i, \theta), y_i)$$

- The function  $g(x_i, \theta)$  encapsulates the computations of a multilayer perceptron with parameters  $\theta$

1) In the previous section the function was  $f(\theta) = \theta^2$ . For multilayer perceptron, the function is quite obviously different.

# How do our models learn?

- We thus need to loop over the training data  $(x_1, y_1), \dots, (x_n, y_n)$  to compute sums over individual data points
- This results in a slight modification to the gradient descent algorithm
  1. For each epoch – Loop over training data
  2. For each batch – Loop over pieces of training data
  3. Compute the error of the parameters  $\theta$
  4. Compute the gradient of the parameters  $\nabla\theta$
  5. Update parameters using  $\theta^{i+1} = \theta^i - \mu \cdot \nabla\theta^i$

# The Concept of Epochs

- Gradient descent requires multiple iterations to reach a minimum
  - The optimization function  $f(\theta)$  sums the loss functions  $L(\hat{y}_i, y_i)$  of each individual data point  $(x_i, y_i)$  in the training data
  - We need to go through the training data multiple times to find suitable parameters  $\theta$  for our problem
- Each such iteration is called an **epoch**

→ Requires the computation of the full sum, which is expensive 

# The Concept of Batches

- Instead of computing the full sum of one epoch in one go, we instead break it apart into multiple smaller pieces
  - These pieces are usually of a specified size (e.g., 64), which tells us how many training data points to use for each piece
  - One epoch then consists of processing all pieces
- Each such piece of the training data is called a **batch**

Batch<sup>1)</sup> (30 samples)

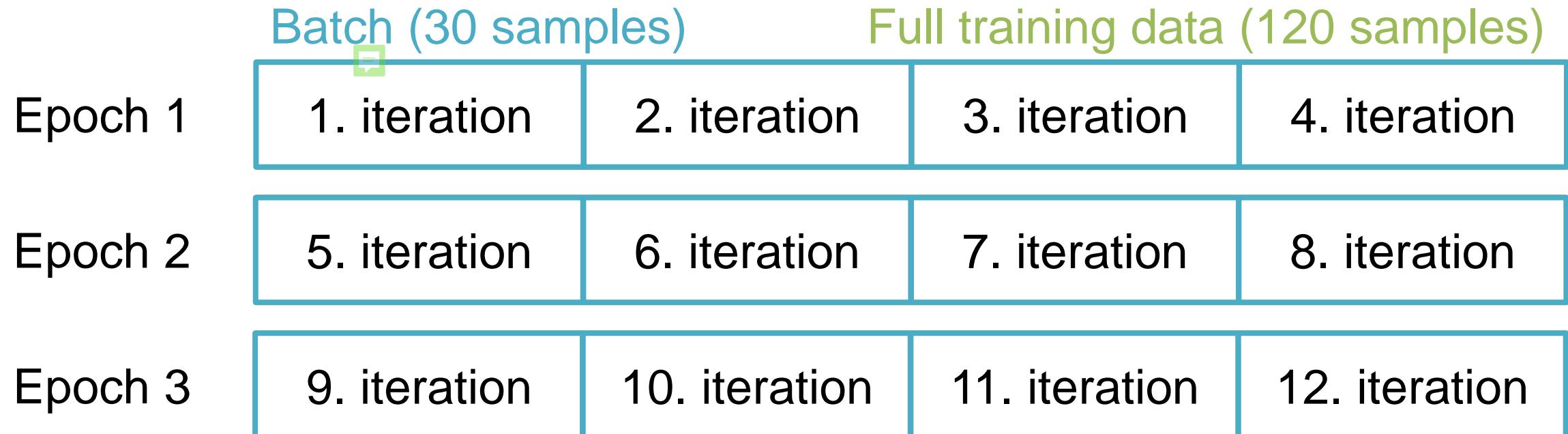
Full training data (120 samples)



1) Note how we need 4 batches to go through all of the training data in this specific example.

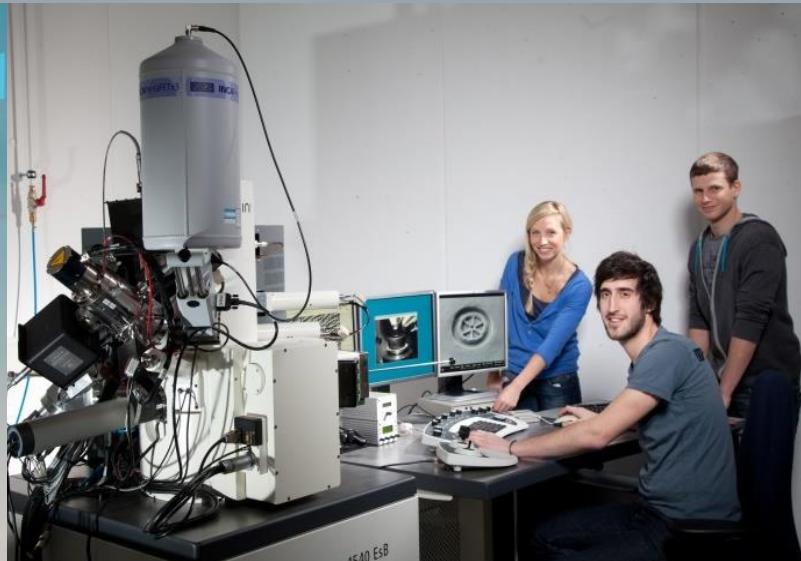
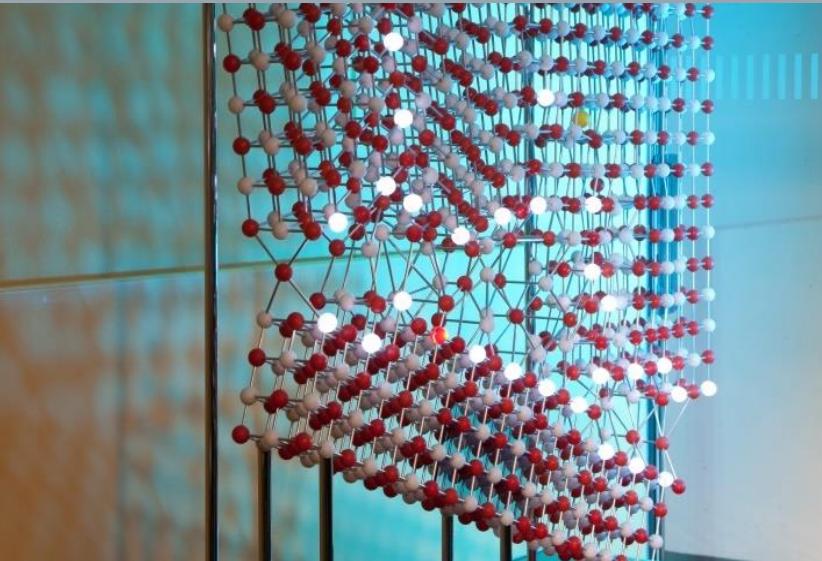
# The Learning Process – A Summary

- The parameters  $\theta$  are optimized by iterating through the training data 
- For practical reasons, this is split into epochs and batches



# Machine Learning for Engineers

## Deep Learning – Convolution

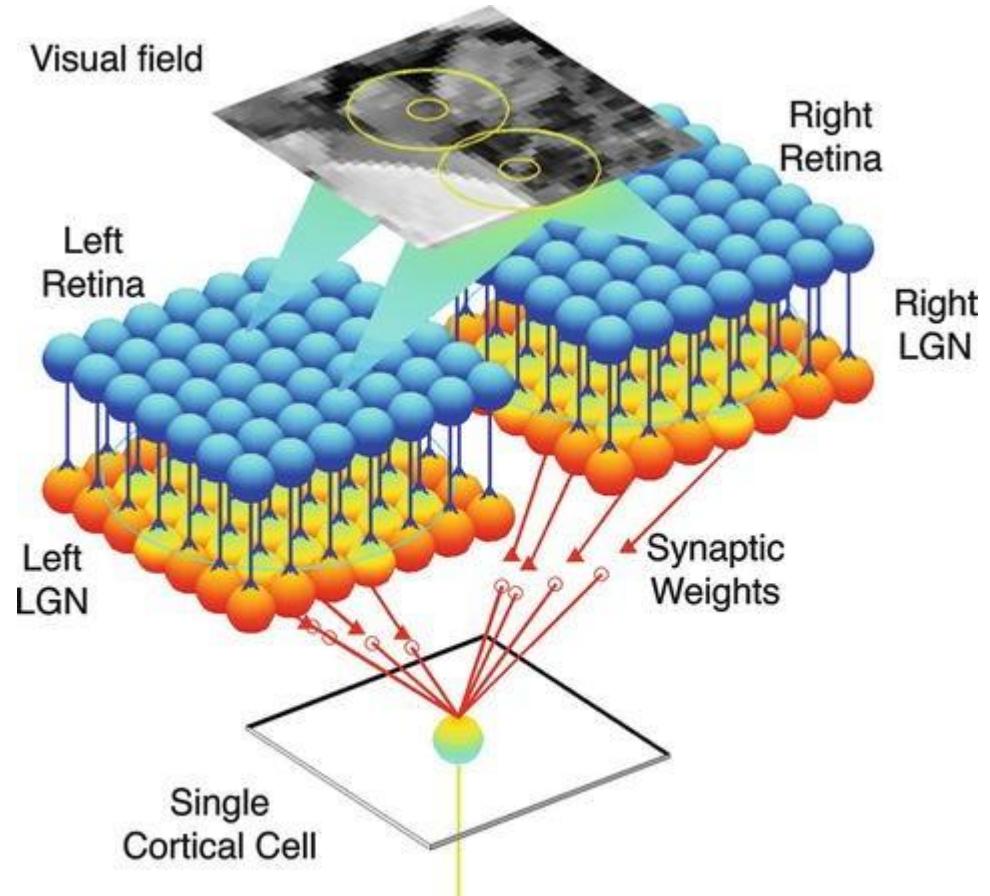


Bilder: TF / Malter

# The Visual Cortex

We've previously learned that the brain has specialized regions.

- The visual cortex is in charge of processing visual information collected from the retinae 
- However, cortical cells only respond to stimuli of a small receptive fields 

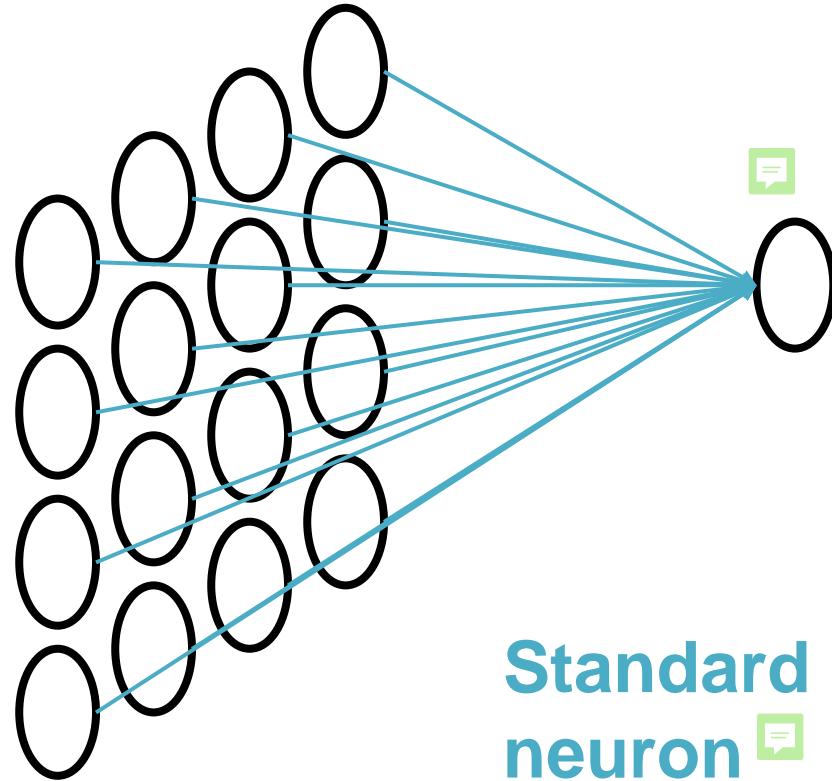


→ Let's us look at the implications this has on neural networks 

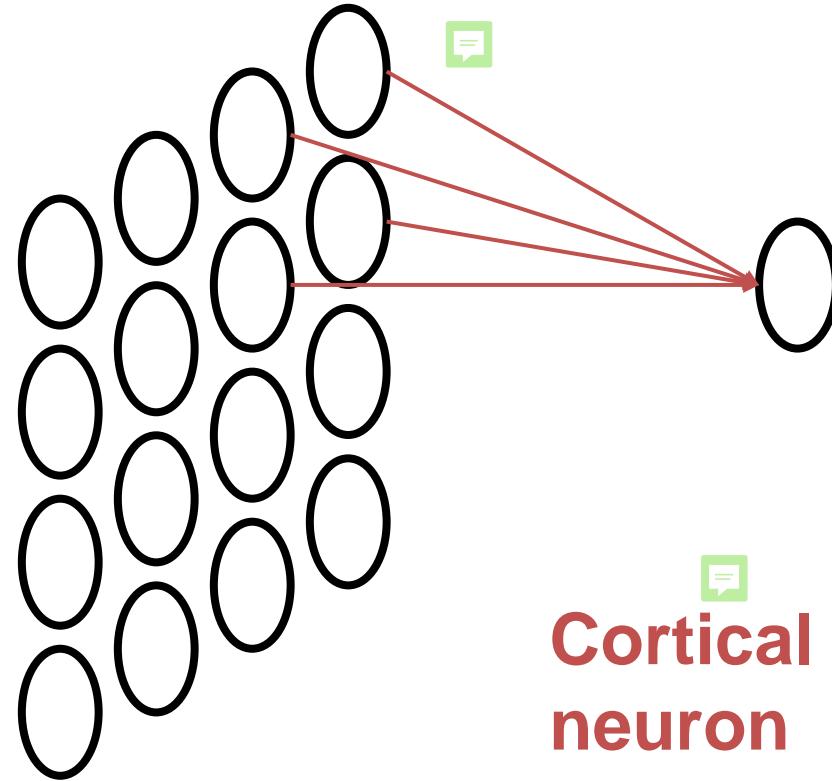
Image from [https://link.springer.com/referenceworkentry/10.1007/978-1-4614-7320-6\\_360-2](https://link.springer.com/referenceworkentry/10.1007/978-1-4614-7320-6_360-2)

# Multilayer Perceptron versus Cortical Neuron

- Multilayer perceptron's neurons are connected to all previous neurons
- Cortical neurons are only connected to a small receptive field



Standard  
neuron 



Cortical  
neuron 

# The Convolution Operation

The output computation now only depends on a subset of inputs: 

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13} + w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23} + w_{31}x_{31} + w_{32}x_{32} + w_{33}x_{33}$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$	$x_{25}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$	$x_{35}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$	$x_{45}$
$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	$x_{55}$

★

$w_{11}$	$w_{12}$	$w_{13}$
$w_{21}$	$w_{22}$	$w_{23}$
$w_{31}$	$w_{32}$	$w_{33}$

=

$y_{11}$		

# The Convolution Operation



The output computation now only depends on a subset of inputs:

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{13}x_{14} + w_{21}x_{22} + w_{22}x_{23} + w_{23}x_{24} \\ + w_{31}x_{32} + w_{32}x_{33} + w_{33}x_{34}$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$	$x_{25}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$	$x_{35}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$	$x_{45}$
$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	$x_{55}$

★

$w_{11}$	$w_{12}$	$w_{13}$
$w_{21}$	$w_{22}$	$w_{23}$
$w_{31}$	$w_{32}$	$w_{33}$

=

$y_{11}$	$y_{12}$	

# The Convolution Operation



The output computation now only depends on a subset of inputs:

$$y_{13} = w_{11}x_{13} + w_{12}x_{14} + w_{13}x_{15} + w_{21}x_{23} + w_{22}x_{24} + w_{23}x_{25} \\ + w_{31}x_{33} + w_{32}x_{34} + w_{33}x_{35}$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$	$x_{25}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$	$x_{35}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$	$x_{45}$
$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	$x_{55}$

★

$w_{11}$	$w_{12}$	$w_{13}$
$w_{21}$	$w_{22}$	$w_{23}$
$w_{31}$	$w_{32}$	$w_{33}$

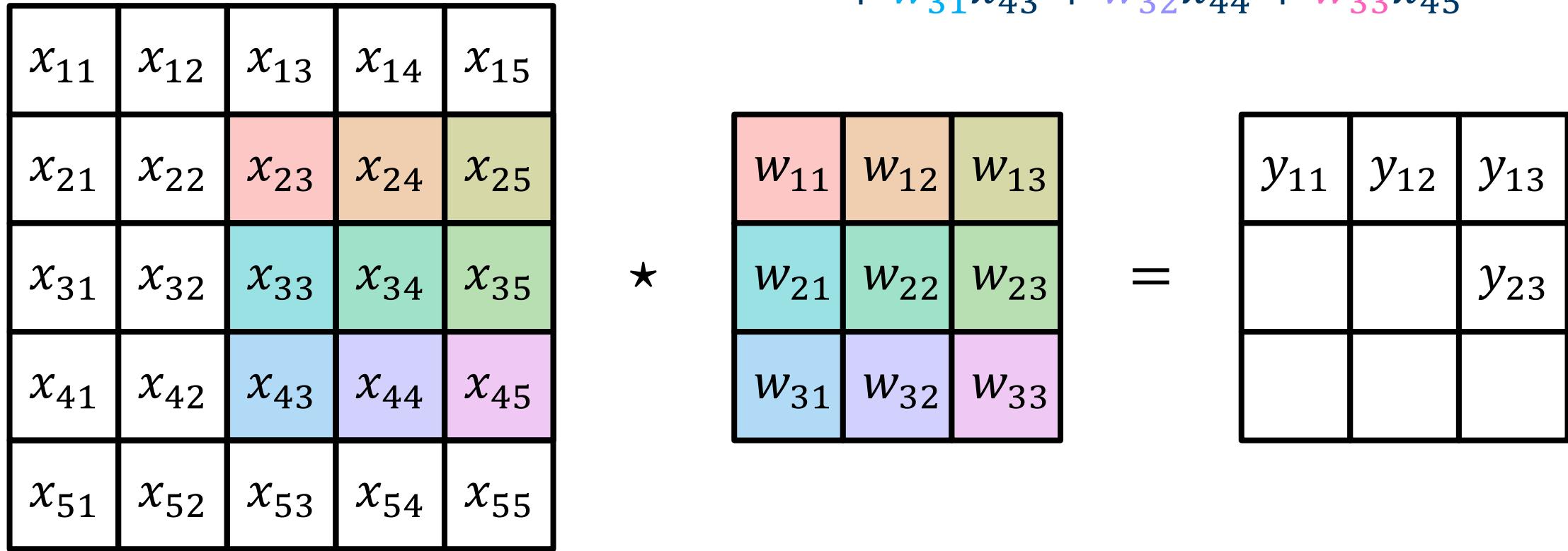
=

$y_{11}$	$y_{12}$	$y_{13}$

# The Convolution Operation

The output computation now only depends on a subset of inputs:

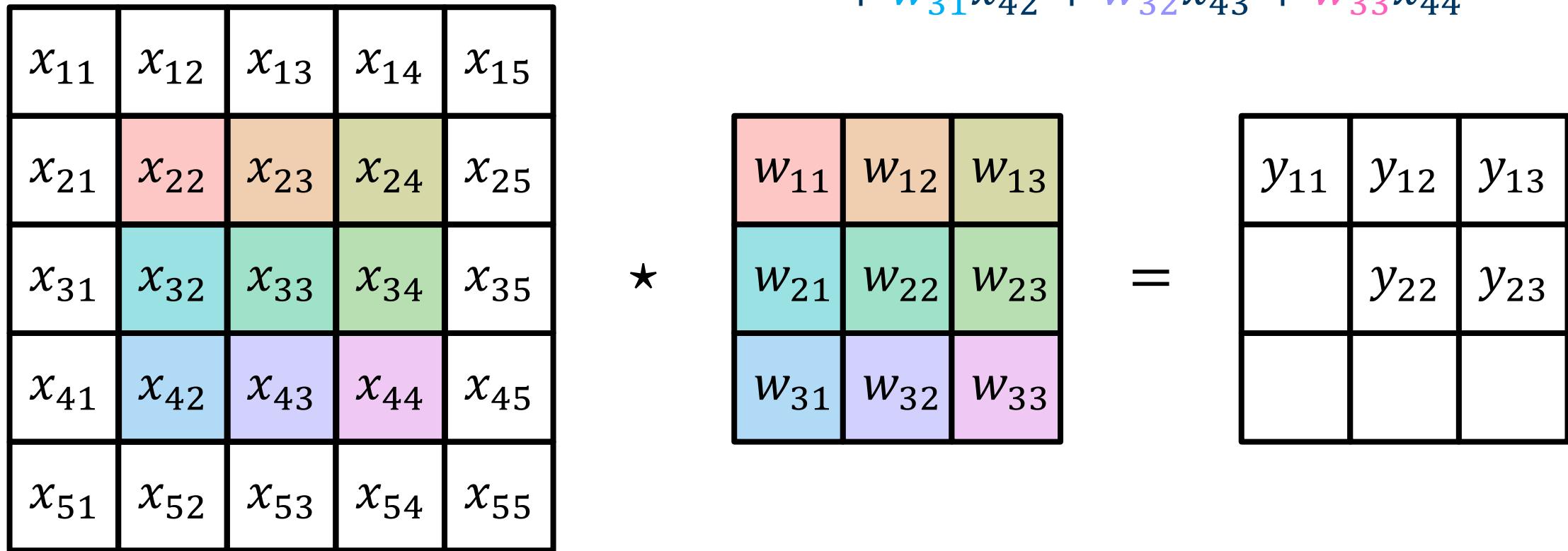
$$y_{23} = w_{11}x_{23} + w_{12}x_{24} + w_{13}x_{25} + w_{21}x_{33} + w_{22}x_{34} + w_{23}x_{35} \\ + w_{31}x_{43} + w_{32}x_{44} + w_{33}x_{45}$$



# The Convolution Operation

The output computation now only depends on a subset of inputs:

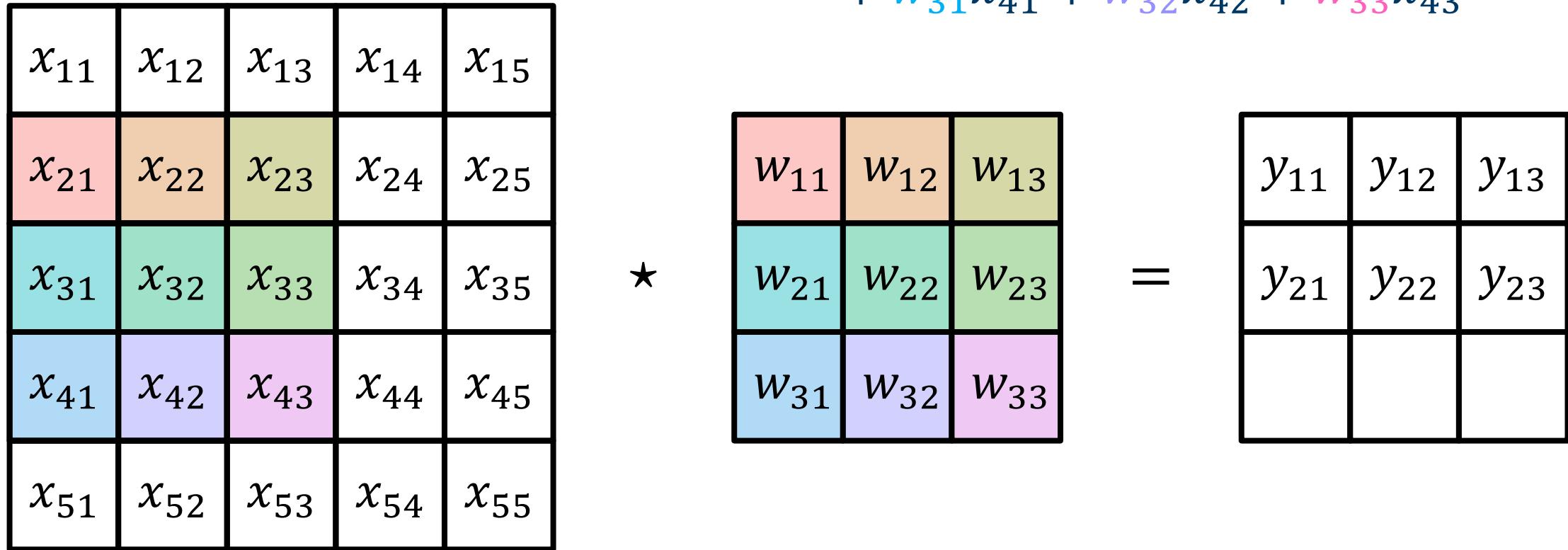
$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{13}x_{24} + w_{21}x_{32} + w_{22}x_{33} + w_{23}x_{34} \\ + w_{31}x_{42} + w_{32}x_{43} + w_{33}x_{44}$$



# The Convolution Operation

The output computation now only depends on a subset of inputs:

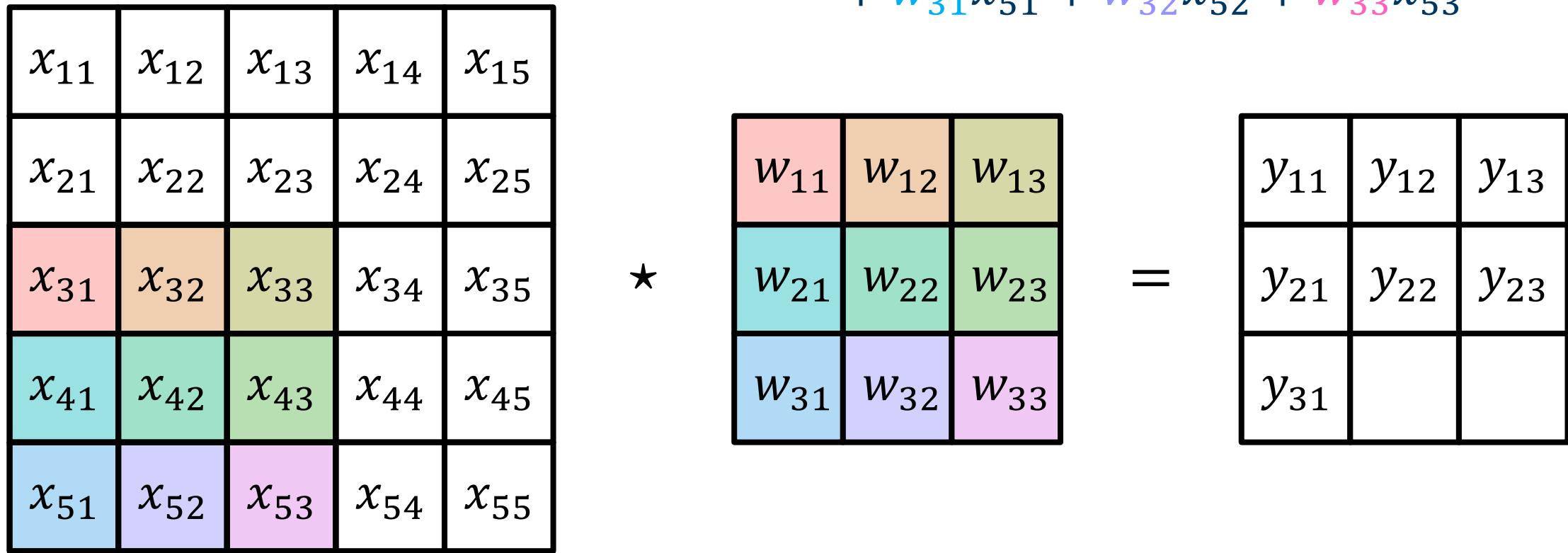
$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{13}x_{23} + w_{21}x_{31} + w_{22}x_{32} + w_{23}x_{33} \\ + w_{31}x_{41} + w_{32}x_{42} + w_{33}x_{43}$$



# The Convolution Operation

The output computation now only depends on a subset of inputs:

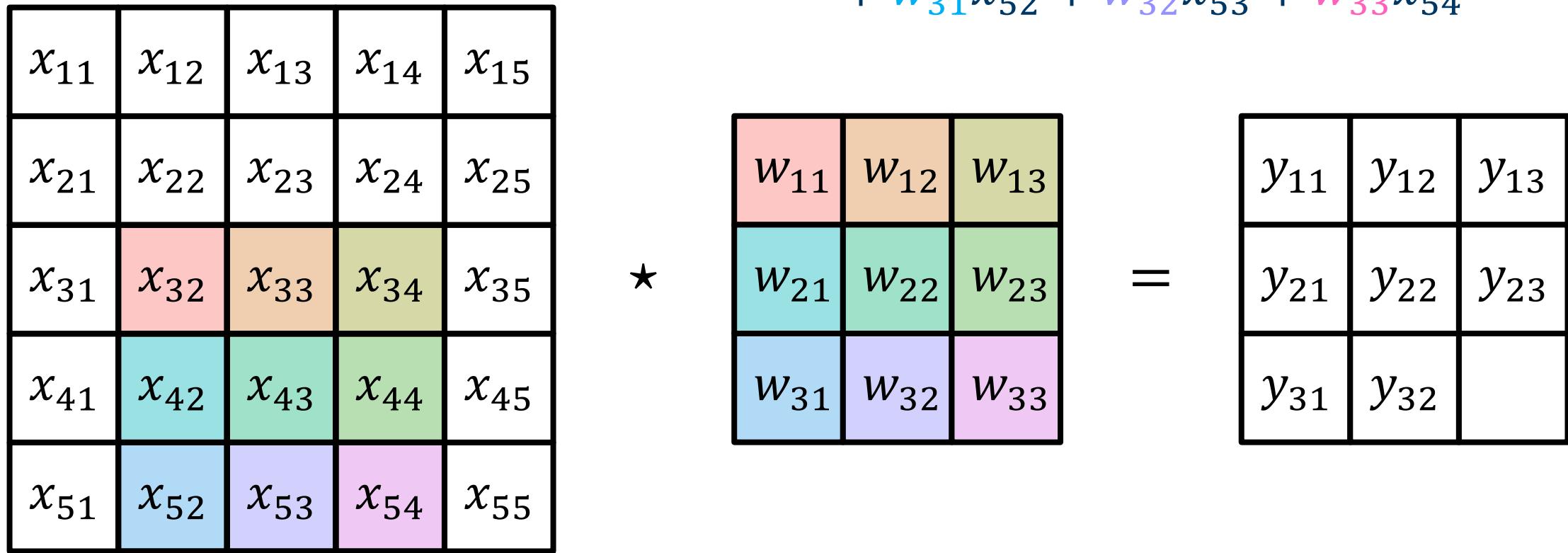
$$y_{31} = w_{11}x_{31} + w_{12}x_{32} + w_{13}x_{33} + w_{21}x_{41} + w_{22}x_{42} + w_{23}x_{43} \\ + w_{31}x_{51} + w_{32}x_{52} + w_{33}x_{53}$$



# The Convolution Operation

The output computation now only depends on a subset of inputs:

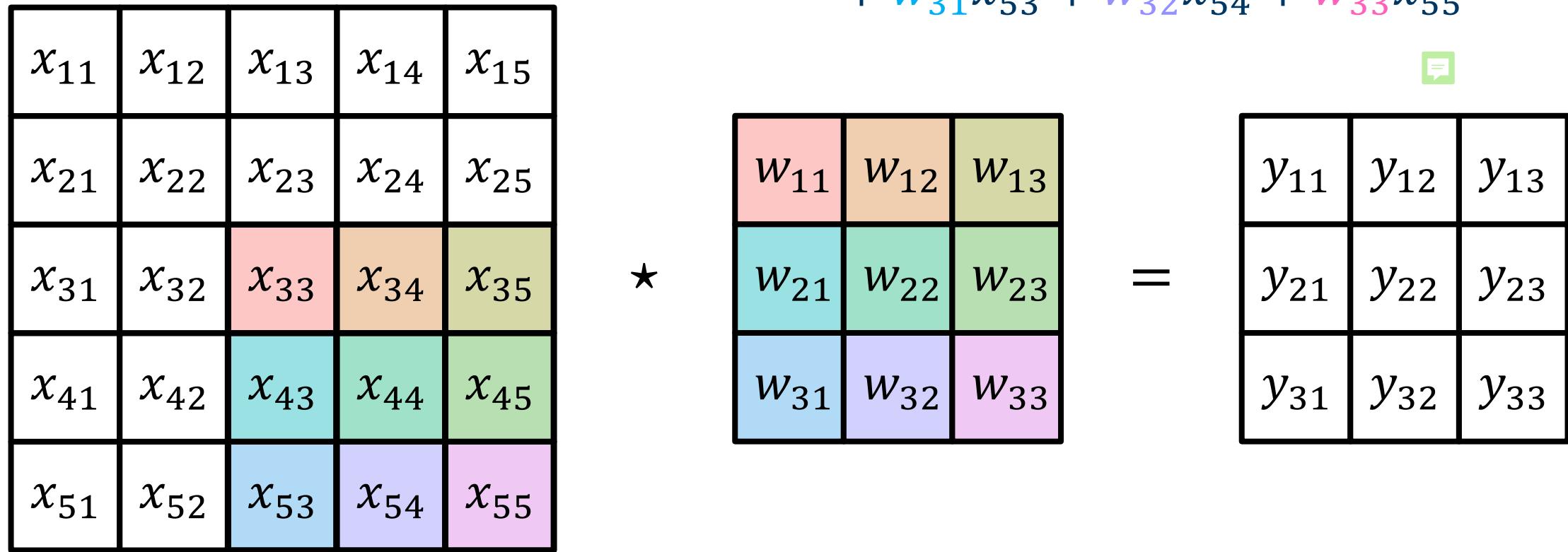
$$y_{32} = w_{11}x_{32} + w_{12}x_{33} + w_{13}x_{34} + w_{21}x_{42} + w_{22}x_{43} + w_{23}x_{44} \\ + w_{31}x_{52} + w_{32}x_{53} + w_{33}x_{54}$$



# The Convolution Operation

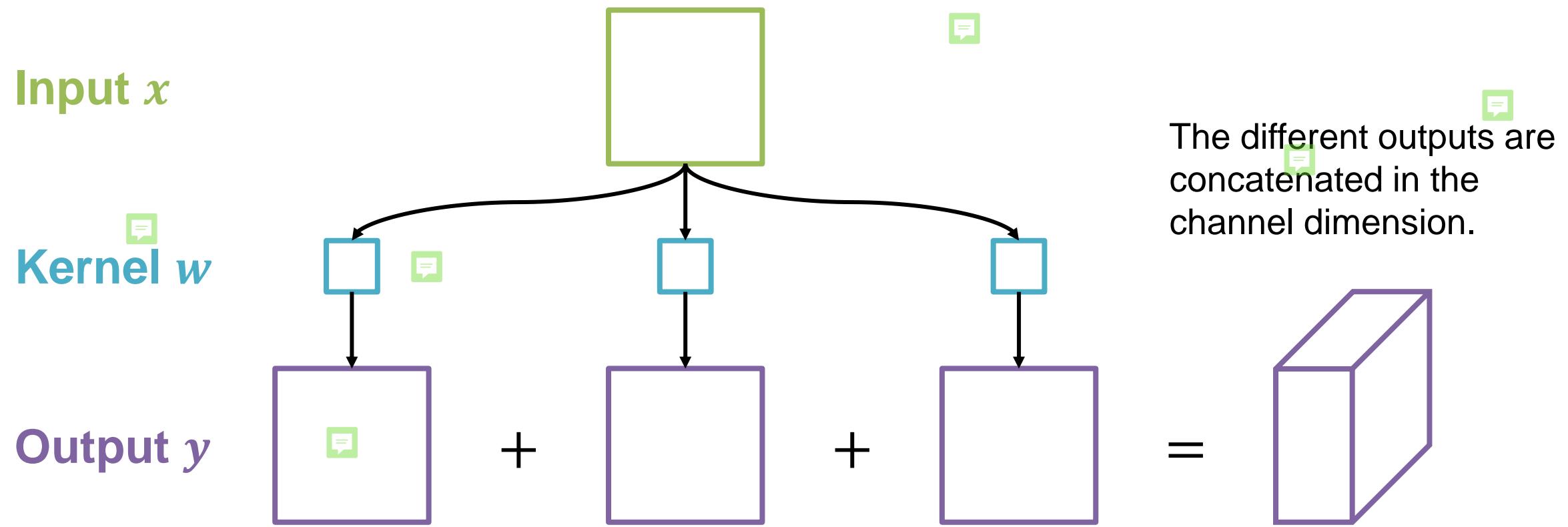
The output computation now only depends on a subset of inputs:

$$y_{33} = w_{11}x_{33} + w_{12}x_{34} + w_{13}x_{35} + w_{21}x_{43} + w_{22}x_{44} + w_{23}x_{45} \\ + w_{31}x_{53} + w_{32}x_{54} + w_{33}x_{55}$$



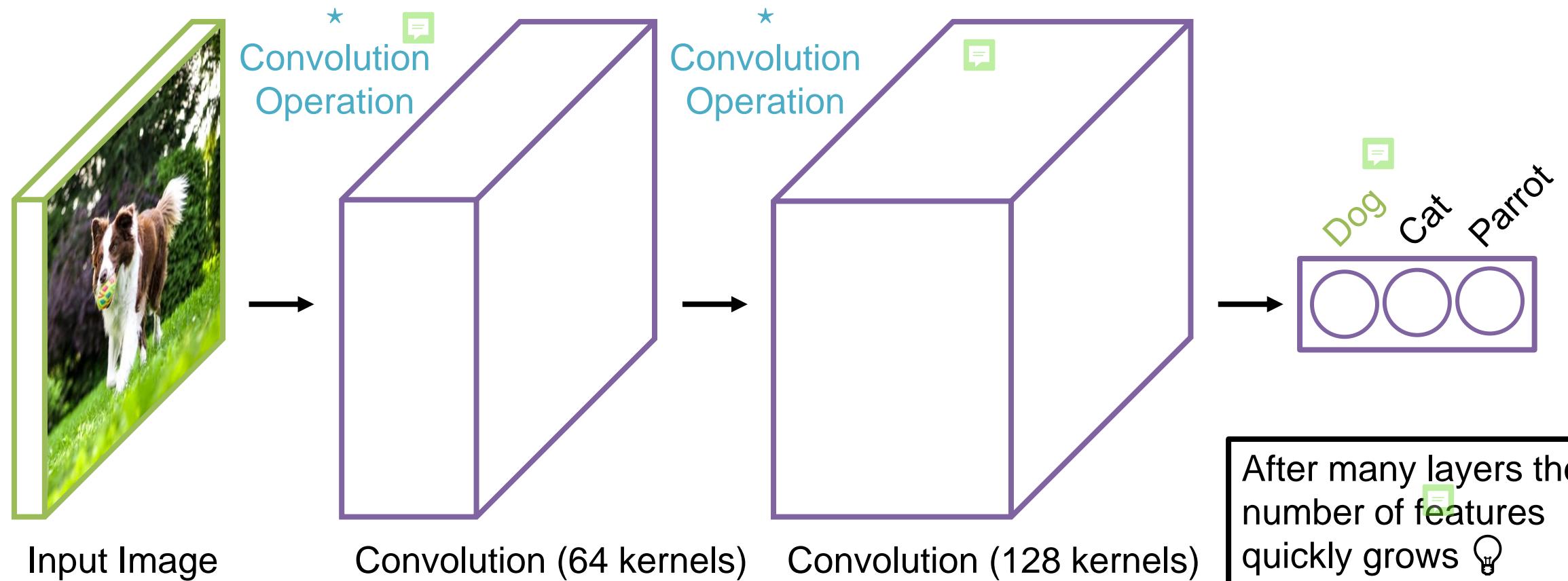
# The Convolutional Neural Network – Multiple Filters

Instead of a single convolution operation we can also apply multiple convolution operations in parallel on the same input. 



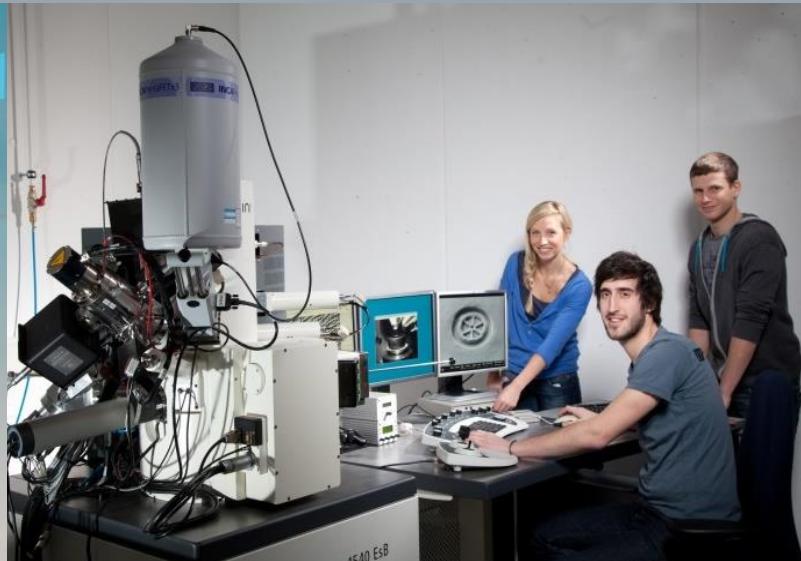
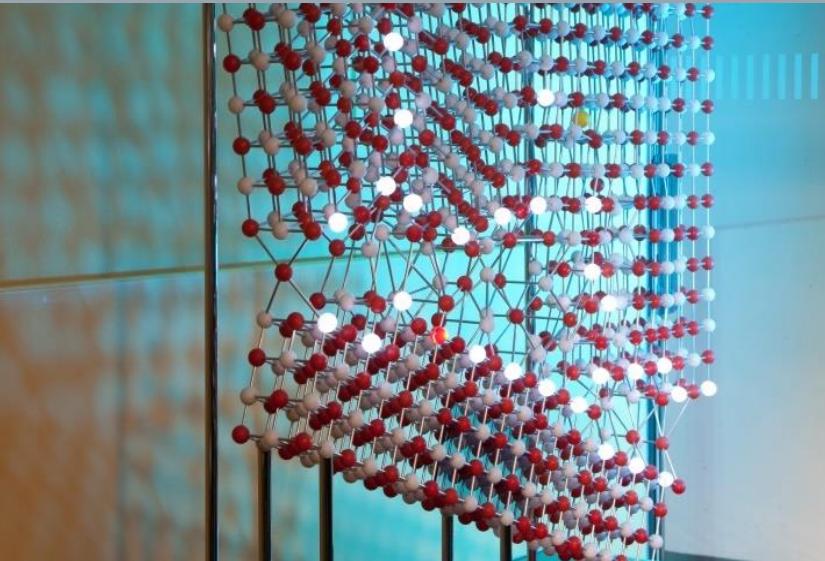
# The Convolutional Neural Network – Multiple Layers

To incrementally process the input to relevant features, we again apply multiple layers of convolutions.



# Machine Learning for Engineers

## Deep Learning – Pooling



Bilder: TF / Malter

# Summarizing Features via Pooling

- Deeper in the network the number of features quickly grows 
- It makes sense to summarize these features deeper in the network
  - From a practical perspective, this reduces the number of computations 
  - From a theoretical perspective, these higher-level (i.e., global scale) features don't require a high spatial resolution 
- This process is called pooling and works like a convolution, with the kernel replaced by a pooling operation 

# The Pooling Operation

The output computation again only depends on a subset of inputs, but the stride (spacing between subsets) is commonly larger:

$$y_{11} = \max(x_{11}, x_{12}, x_{21}, x_{22})$$



$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$

$\max x$   





$y_{11}$	

# The Pooling Operation

The output computation again only depends on a subset of inputs, but the stride (spacing between subsets) is commonly larger:

$$y_{12} = \max(x_{13}, x_{14}, x_{23}, x_{24})$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$

$\max x$   
→

$y_{11}$	$y_{12}$

# The Pooling Operation

The output computation again only depends on a subset of inputs, but the stride (spacing between subsets) is commonly larger:

$$y_{21} = \max(x_{31}, x_{32}, x_{41}, x_{42})$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$

$\max x$   
→

$y_{11}$	$y_{12}$
$y_{21}$	

# The Pooling Operation

The output computation again only depends on a subset of inputs, but the stride (spacing between subsets) is commonly larger:

$$y_{22} = \max(x_{33}, x_{34}, x_{43}, x_{44})$$

$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$
$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$
$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$

$\max x$   
→



$y_{11}$	$y_{12}$
$y_{21}$	$y_{22}$

# The Pooling Operation

There are multiple methods to summarize the features which are commonly used in Convolutional Neural Networks (CNN):

## Example for $n = 4$

- Maximum  $y = \max(x_1, x_2, x_3, x_4)$

- Average  $y = \frac{1}{4}(x_1 + x_2 + x_3 + x_4) = \frac{1}{4} \sum_{i=1}^4 x_i$

- L<sup>2</sup>-Norm  $y = x_1^2 + x_2^2 + x_3^2 + x_4^2 = \sum_{i=1}^4 x_i^2$

## General case

$$y = \max(x_1, \dots, x_n)$$

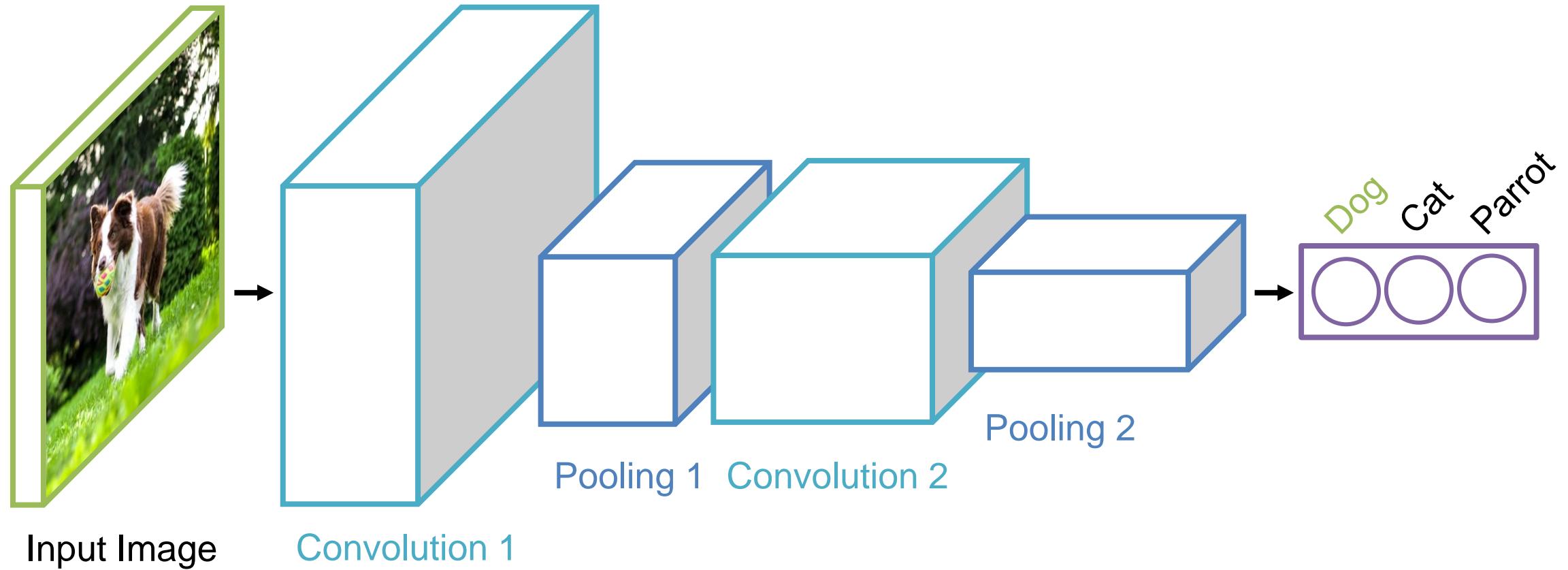
$$y = \frac{1}{n} \sum_{i=1}^n x_i$$

$$y = \sum_{i=1}^n x_i^2$$

→ There's no general consensus what the “best” method is 

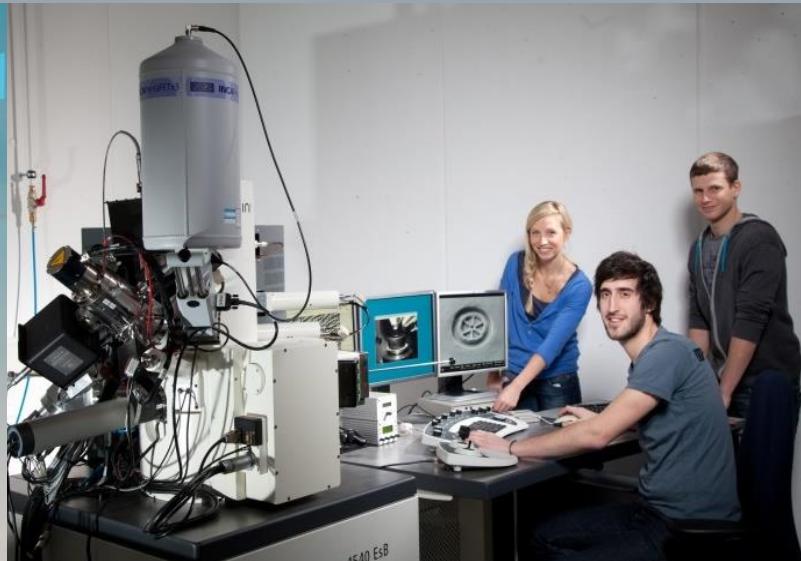
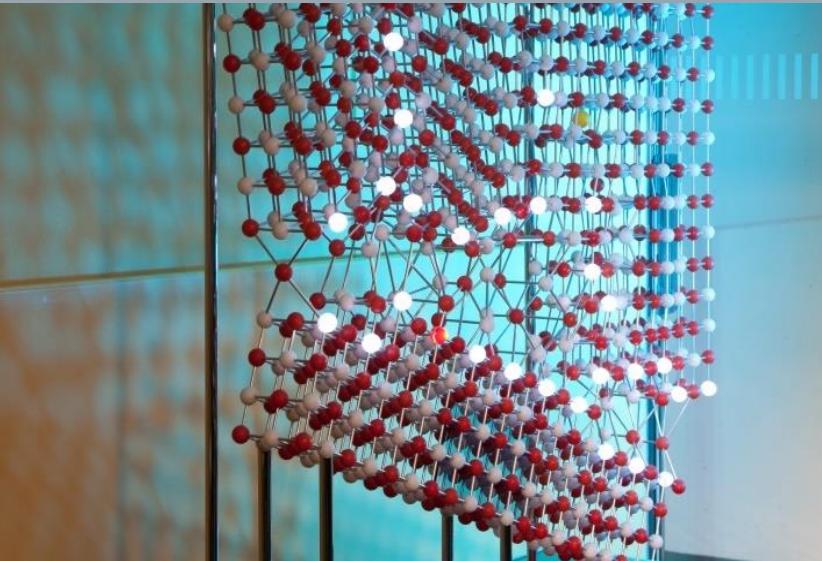
# The Convolutional Neural Network – Pooling

Adding pooling to the network reduced the number of features deeper in the network using a (relatively) inexpensive method.



# Machine Learning for Engineers

## Deep Learning – Applications



Bilder: TF / Malter

# Full Image Classification

- **Task:** Plant leaf disease classification of 14 different species
- **Input:** Image of plant leaf
- **Output:** Plant and disease class
- **Model:** EfficientNet
- Ü. Atilaa, M. Uçarb, K. Akyolc, and E. Uçarb, “Plant leaf disease classification using EfficientNet deep learning model”, 2020.



grape with black measles



potato with late blight



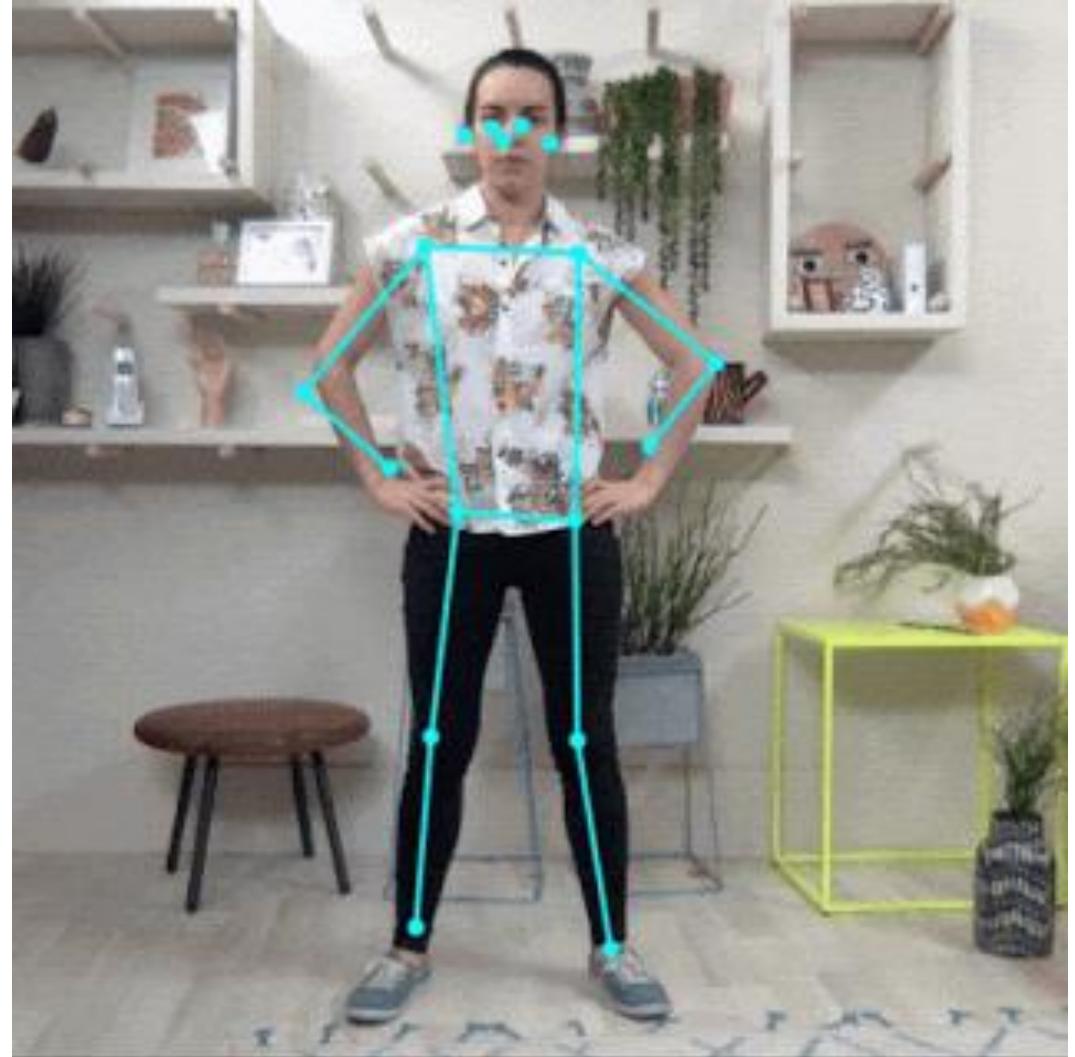
healthy strawberry



healthy tomato

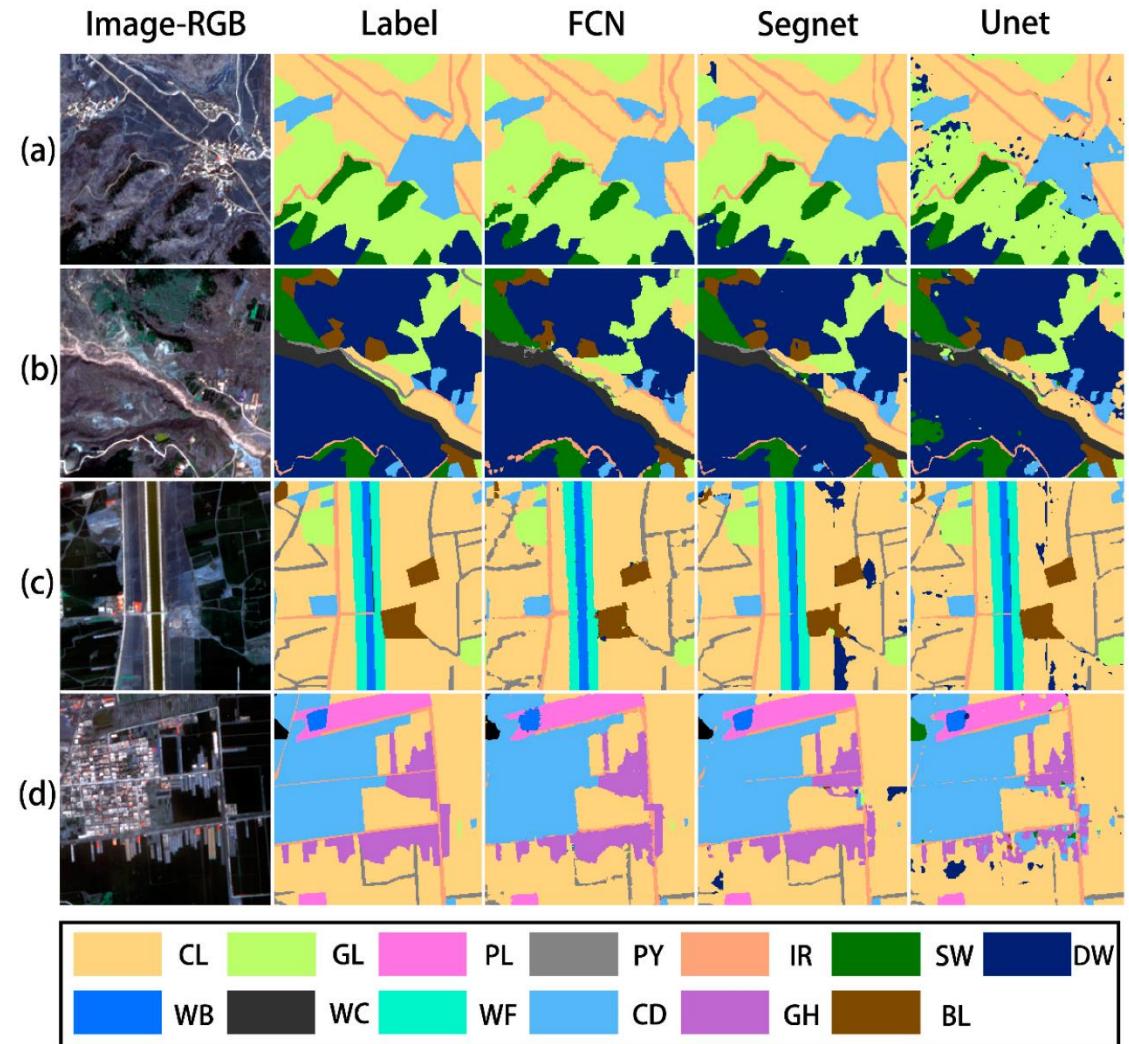
# Full Image Regression

- **Task:** Pose estimation based on human key points (eyes, joints, ...)
- **Input:** Image of human
- **Output:** Coordinates of 17 key points
- **Model:** PoseNet
- G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, “PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model”, 2018.
- Animated image taken from TensorFlow Lite documentation at [https://www.tensorflow.org/lite/examples/pose\\_estimation/overview](https://www.tensorflow.org/lite/examples/pose_estimation/overview)



# Pixel-wise Image Classification (Segmentation)

- **Task:** Land cover classification in satellite imagery
- **Input:** Satellite image
- **Output:** Land cover class per individual pixel
- **Model:** Segnet, Unet
- Z. Han, Y. Dian, H. Xia, J. Zhou, Y. Jian, C. Yao, X. Wang, and Y. Li, “Comparing Fully Deep Convolutional Neural Networks for Land Cover Classification with High-Spatial-Resolution Gaofen-2 Images”, 2020.



# Pixel-wise Image Regression

- **Task:** Relative depth estimation from a single image
- **Input:** Any image
- **Output:** Depth value per individual pixel
- **Model:** MiDaS
- R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer", 2019.



# 5. Deep Learning

## ◀ Applications

### Questions

What are the characteristics of a neuron?

- Output of a neuron is produced by an activation function and it is non-linear
- Both are correct
- Input to a neuron is linearly combined with the neuron's weights



Correct!

Submit

In a natural neuron computes a decision using Voltage and an Actionpotential. Please order the computation steps, described in the lecture.

Excitatory stimuli reach the neuron

Threshold is reached

Neuron fires and triggers action potential



Correct!

Submit

Please name at least three activation functions used in Deep Networks!

1.
2.
3.



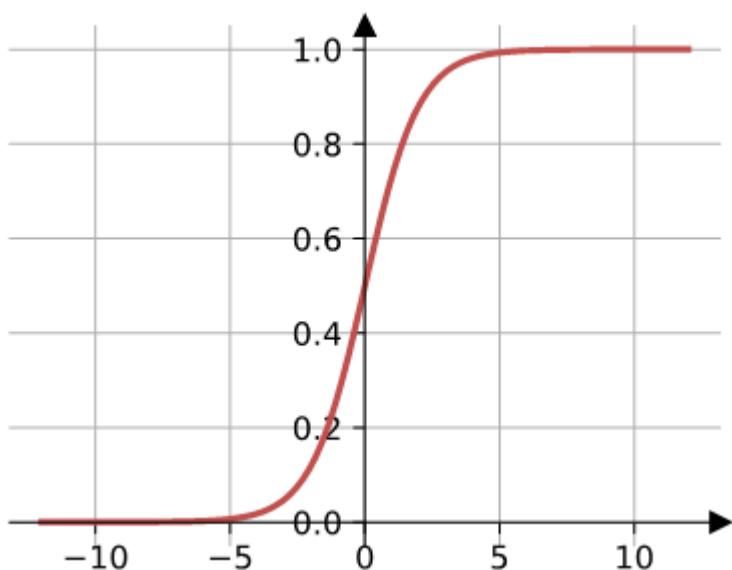
Also correct are:

- Sigmoid
- tanh
- ReLU
- Hyperbolic Tangent
- Rectified Linear Unit

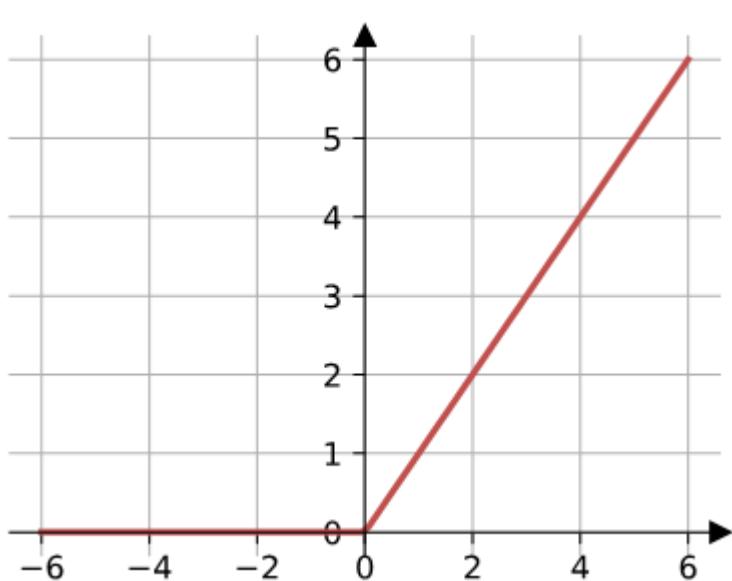
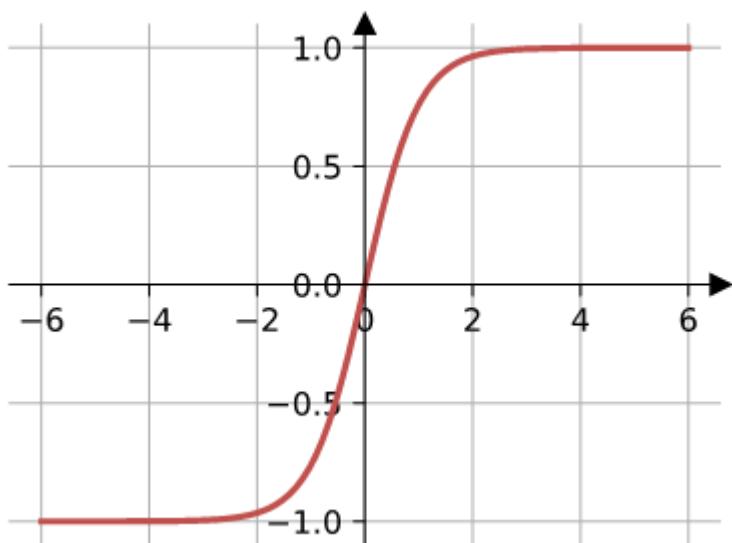


Correct!

### Graph 1



### Graph 2

**Graph 3**

Please assign each function shape, the according activation function.

[Reset Ordering](#)

Graph 1

Graph 3

Graph 2

$1 / (1 + e^{-x})$

tanh(x)

max(x,0)



Correct!

Submit

Please assign each problem the according error metric from the lecture!

Reset Ordering

Classification

Regression

cross entropy

mean squared error



Correct!

Submit

Fill out the following text about parameter optimization from the deep learning lecture.

In the lecture, we learned how we can improve the current parameters of the network. To this end, we can compute the

✓ [gradient] of the parameters  $\nabla\theta$ .

This is done, using the  ✓ [gradient], which tells us how we need to change the current parameters

$\theta$  in order to make fewer errors on the given data. We can use this in an  ✓ [iterative] algorithm called  ✓ [gradient descent], with the central equation being  $\theta^{i+1} = \theta^i - \mu \cdot \nabla \theta^i$ . The learning rate  $\mu$  tells us how  ✓ [fast | quickly] we should  ✓ [change] the current parameters  $\theta^i$



**Correct!**

**Submit**

Name at least three limitations of Gradient Descend!

Limitation 1: The learning rate can lead to

✓ [slow convergence], if not configured properly!

Limitation 2: The starting parameters lead to solutions

✓ [stuck in local minima | stuck in local minimum]!

Limitation 3: There is

✓ [no guarantee] to finding an optimal solution!



**Correct!**

**Submit**

Assign the correct order of the following steps in the rough learning process for a Deep Network.

For each epoch – Loop over training data

For each batch – Loop over pieces of training data

Compute the error of the parameters

Compute the gradient of the parameters

Update parameters



**Correct!**

Submit

What does regularization do?

- Make the parameters' value large
- It depends on the data set specification
- Keep the parameters' value small



**Correct!**

Submit

What does overfitting mean in the context of Deep Learning? How can you avoid overfitting?

- The training and test error is high / Making DNNs larger
- The training error is high and test error is low / Regularisation, Dropout
- The training error is low but the test error is high / Regularisation, Dropout



**Correct!**

Submit

What may we change in DNNs architecture to model more massive data sets?

- Increase the number of layers
- Both are correct
- Increase the number of nodes in one layer



Correct!

Submit



## ◀ Applications



What are the characteristics of a neuron?

Multilayer Perceptron pdf 3/7. slide + az activation function-ök nem lineárisak

In a natural neuron computes a decision using Voltage and an Actionpotential. Please order the computation steps, described in the lecture.

Perceptron 3.slide

Please name at least three activation functions used in Deep Networks!

Perceptron 10. slide

Please assign each function shape, the according activation function.

Perceptron 10. slide

Please assign each problem the according error metric from the lecture!

Loss Function 4. slide

Fill out the following text about parameter optimization from the deep learning lecture.

Gradient Descent egész kb

Name at least three limitations of Gradient Descend!

Gradient Descent 9. slide

Assign the correct order of the following steps in the rough learning process for a Deep Network.

Learning process 3.slide

What does regularization do?

Keep the parameters' value small

NEM VOLT BENNE AZ ELŐADÁSBAN, slideokban sem találom  
jelentése: szabályozás

Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain.

In the context of machine learning, regularization is the process which regularizes (szabályoz) or

shrinks the coefficients towards zero. In simple words, regularization discourages learning a more complex or flexible model, to prevent overfitting

What does overfitting mean in the context of Deep Learning? How can you avoid overfitting?

The training error is low but the test error is high / Regularisation, Dropout

Tehát a modell túl komplex, ezt egyszerűsítve regularisation-nel lehet megoldani (előző kérdésre a válasz), másrészről Dropout-tal (gondolom hogy kidobunk pár dolgot, hogy minél kevesebb dolog alapján építse a modellt, hogy minél egyszerűbb legyen)

What may we change in DNNs architecture to model more massive data sets?

konkrétnak nem volt benne a slideokban, de logikusnak tűnik a válasz

nodes: perceptronokat érti alatta