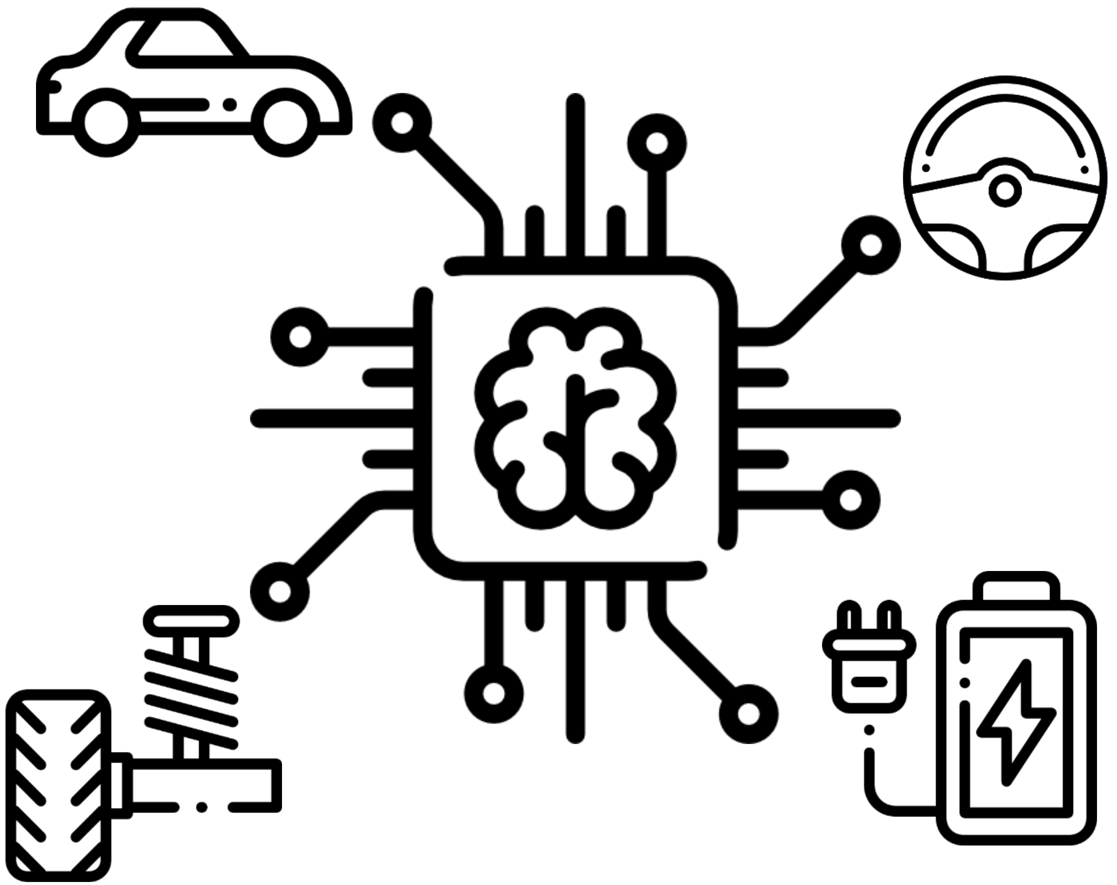
Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Artificial Intelligence in Automotive Technology

Johannes Betz / Prof. Dr.-Ing. Markus Lienkamp / Prof. Dr.-Ing. Boris Lohmann



Lecture Overview

Lecture 16:15 – 17:45	Practice 17:45 – 18:30
1 Introduction: Artificial Intelligence 17.10.2019 – Johannes Betz	**Practice 1** 17.10.2019 – Johannes Betz
2 Perception 24.10.2019 – Johannes Betz	**Practice 2** 24.10.2019 – Johannes Betz
3 Supervised Learning: Regression 31.10.2019 – Alexander Wischnewski	**Practice 3** 31.10.2019 – Alexander Wischnewski
4 Supervised Learning: Classification 7.11.2019 – Jan Cedric Mertens	**Practice 4** 7.11.2019 – Jan Cedric Mertens
5 Unsupervised Learning: Clustering 14.11.2019 – Jan Cedric Mertens	**Practice 5** 14.11.2019 – Jan Cedric Mertens
6 Pathfinding: From British Museum to A* 21.11.2019 – Lennart Adenaw	**Practice 6** 21.11.2019 – Lennart Adenaw
7 Introduction: Artificial Neural Networks 28.11.2019 – Lennart Adenaw	**Practice 7** 28.11.2019 – Lennart Adenaw
8 Deep Neural Networks 5.12.2019 – Jean-Michael Georg	**Practice 8** 5.12.2019 – Jean-Michael Georg
9 Convolutional Neural Networks 12.12.2019 – Jean-Michael Georg	**Practice 9** 12.12.2019 – Jean-Michael Georg
10 Recurrent Neural Networks 19.12.2019 – Christian Dengler	**Practice 10** 19.12.2019 – Christian Dengler
11 Reinforcement Learning 09.01.2020 – Christian Dengler	**Practice 11** 09.01.2020 – Christian Dengler
12 AI-Development 16.01.2020 – Johannes Betz	**Practice 12** 16.01.2020 – Johannes Betz
13 Guest Lecture: VW Data:Lab 23.01.2020 –	

Feedback from last week

Objectives for Lecture 4: Classification

After the lecture you are able to…

… understand the concept of classification, its association to pattern recognition and the urge for machine learning.

… acquire labeled training data and prepare it for the training and validation phase.

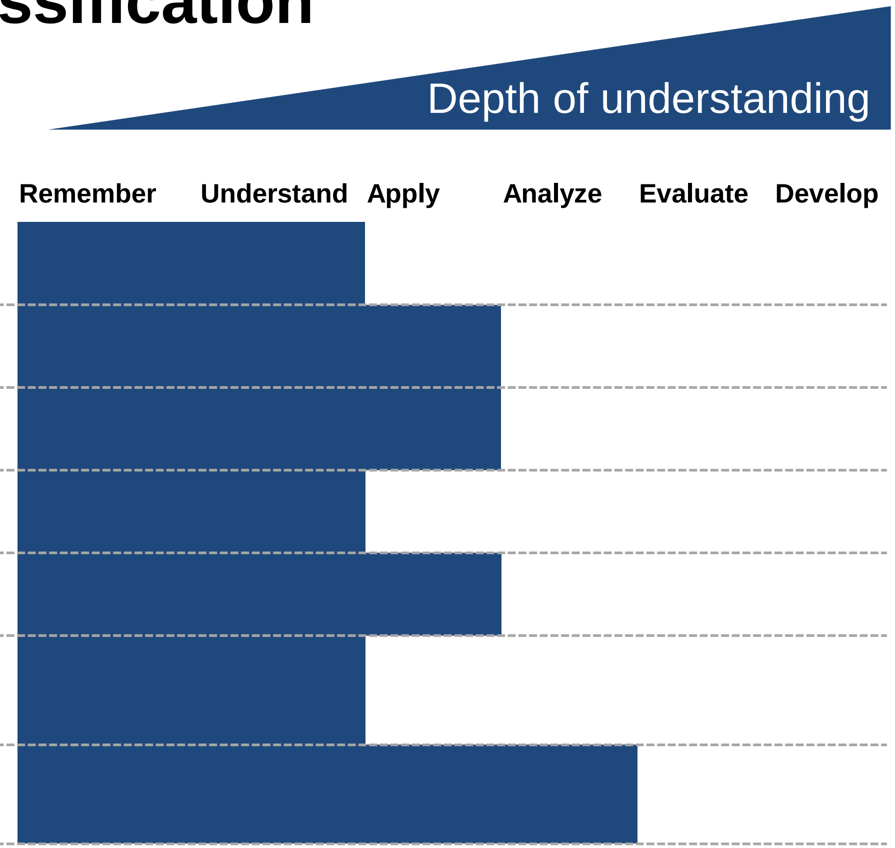
… plan the basic workflow for an arbitrary supervised learning problem.

… understand the concepts of different classification methods together with their pros and cons.

… implement, train and use a classification method with python libraries.

… understand how classification can be used in the perception for automated vehicles.

… analyze the quality of a given classifier regarding to different criteria.

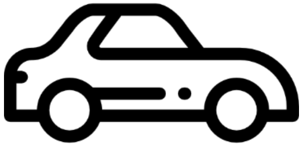


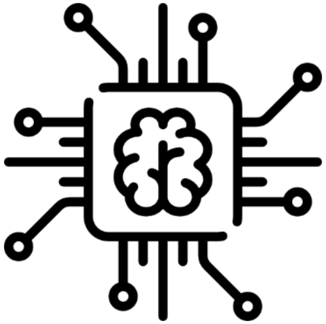
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

Agenda





Classification

"Systematic arrangement in groups or categories according to established criteria" [13]

[3]

Classification

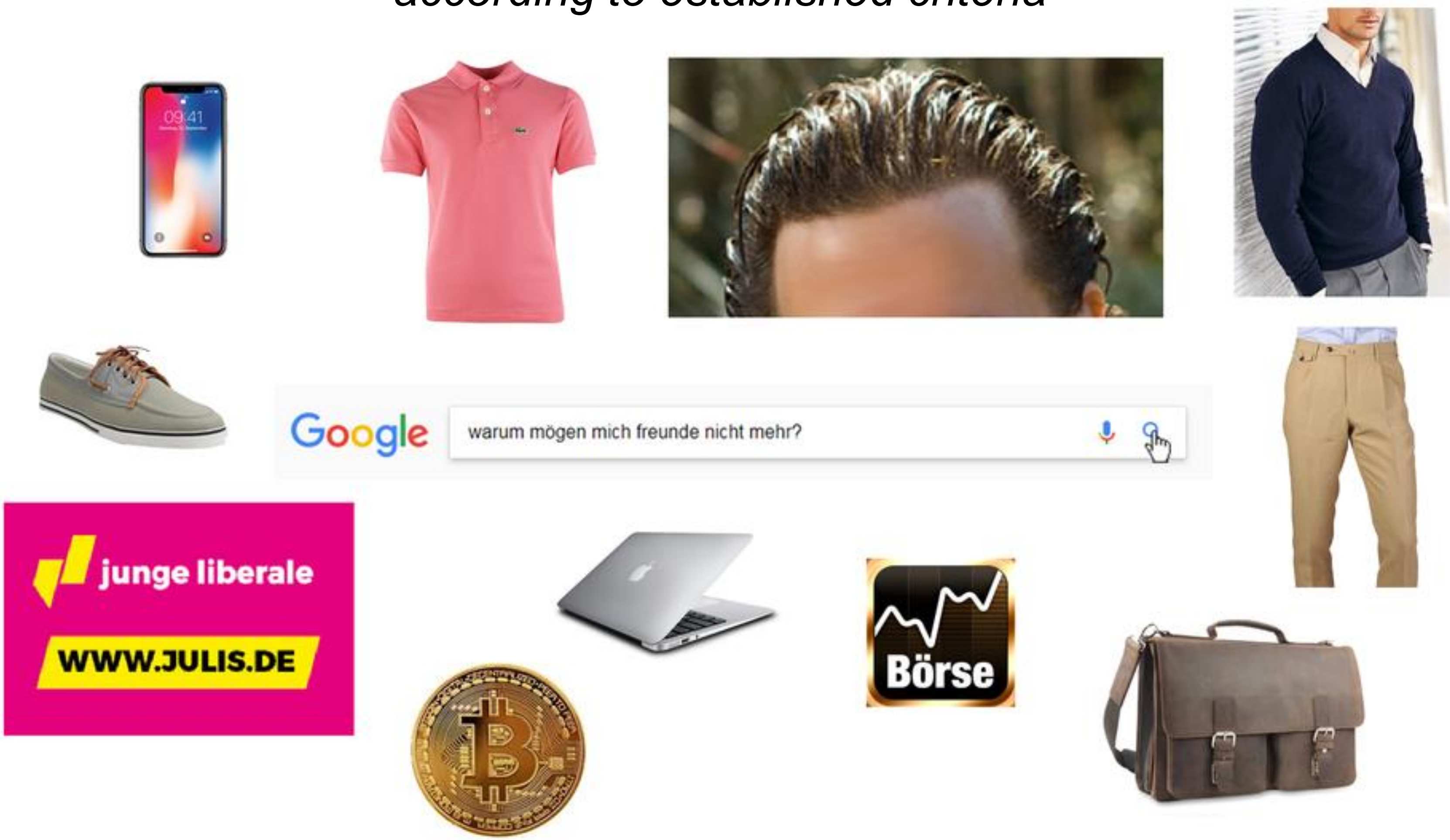
"Systematic arrangement in groups or categories according to established criteria" [13]

[14]

Classification

"Systematic arrangement in groups or categories according to established criteria" [13]

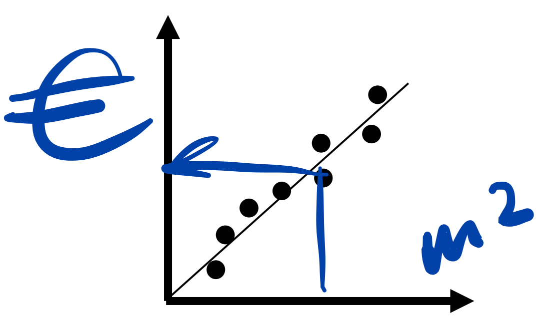
[15]

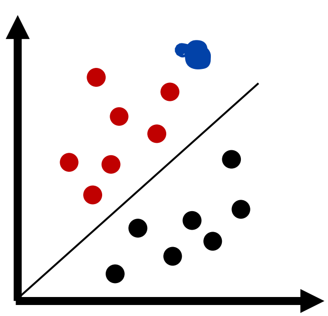


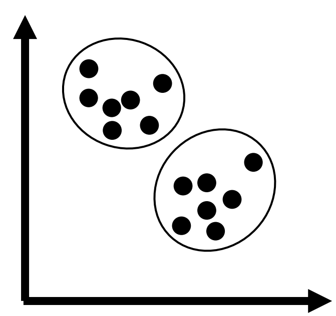
Method Overview

Pattern Recognition

Regression	**Classification**	Clustering
• Predict **continuous** valued output	• Predict **discrete** valued output	• Predict discrete valued output
• Supervised	• Supervised	• **Unsupervised**

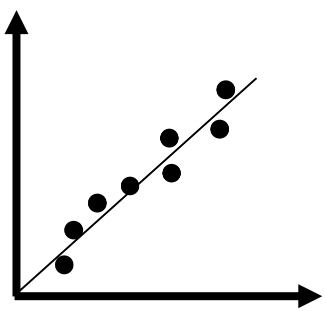






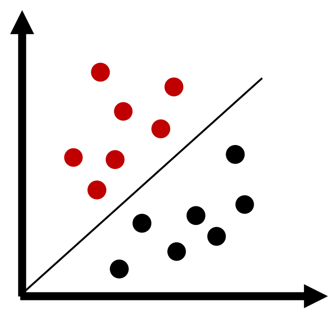
Method Overview

Regression



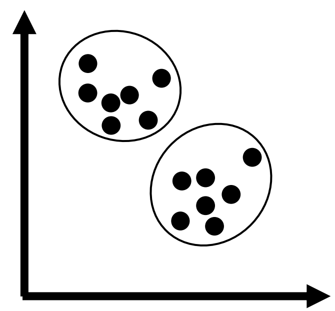
- House pricing
- Number of sales
- Persons weight

Classification



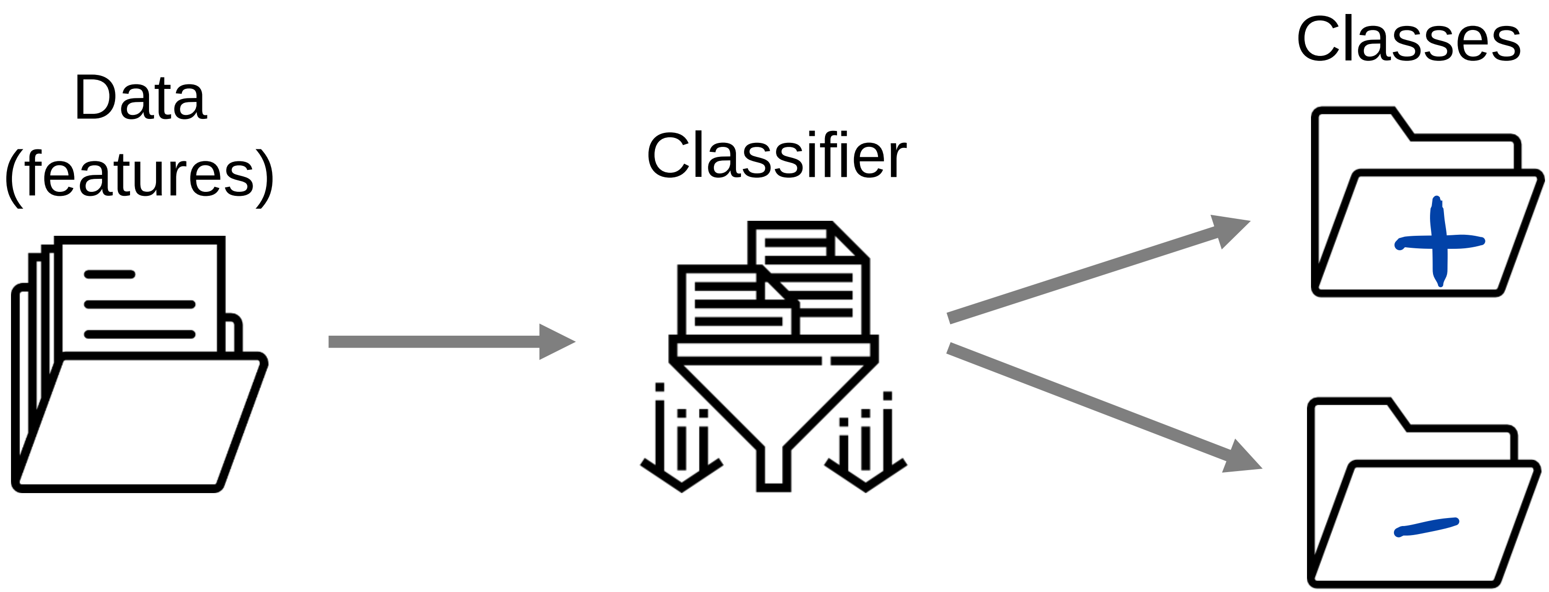
- Object detection
- Spam detection
- Cancer detection

Clustering



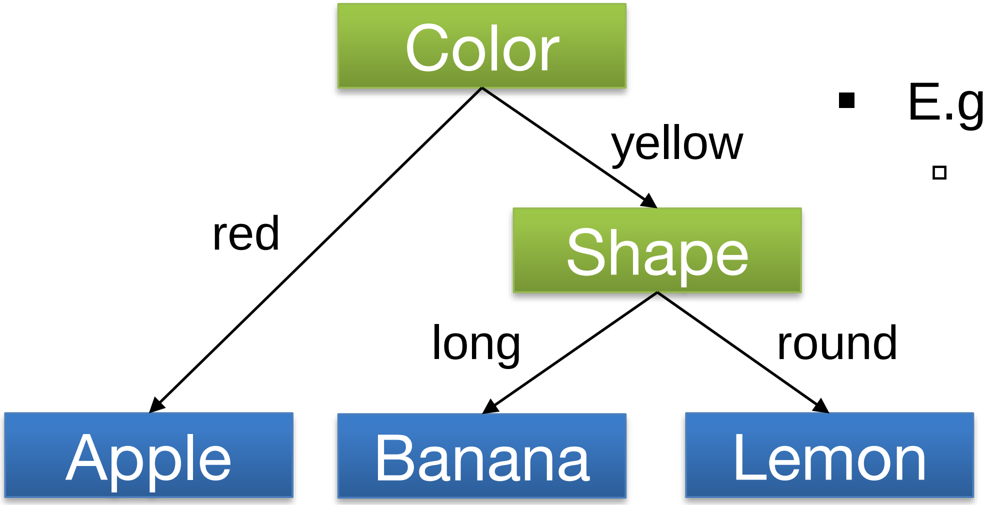
- Genome patterns
- Google news
- Point cloud (lidar) processing

General Approach



Email (keywords, ...)	Spam?	Yes/no
Tumor (size, …)	Malignant?	Yes/no
Object (color, …)	What type?	Cat/car/fruit/…

Classic Method vs. Machine Learning Method



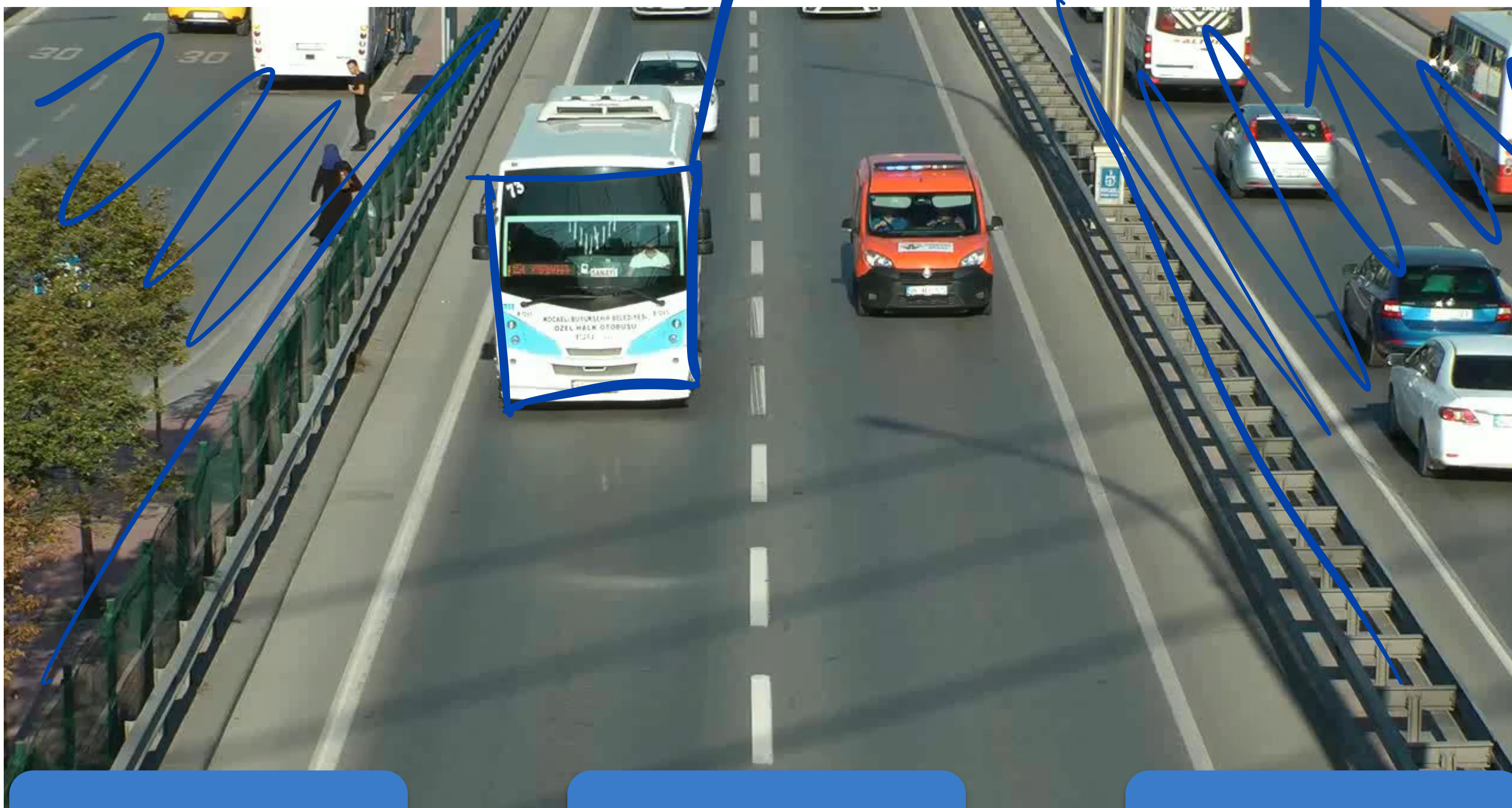
- E.g., decision tree
  - Use a priori knowledge to formulate classification rules

- Advantages of machine learning
  - Automatic generation of a priori knowledge
  - Automatic generation of complex classification rules
  - → Suitable for extreme large datasets

JUST CARFRONT

NO BACKS

Classification - Example



[4]

Object classification → Object detection → Object tracking

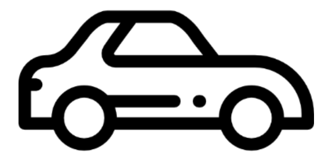
Supervised Learning: Classification

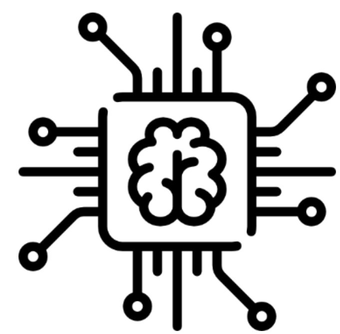
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

Agenda

1. Chapter: Introduction
   - 1.1 Overview
   - **1.2 Training and Validation**
2. Chapter: Methods
   - 2.1 Logistic Regression
   - 2.2 Nearest Neighbors
   - 2.3 Support Vector Machine
3. Chapter: Application
4. Chapter: Summary





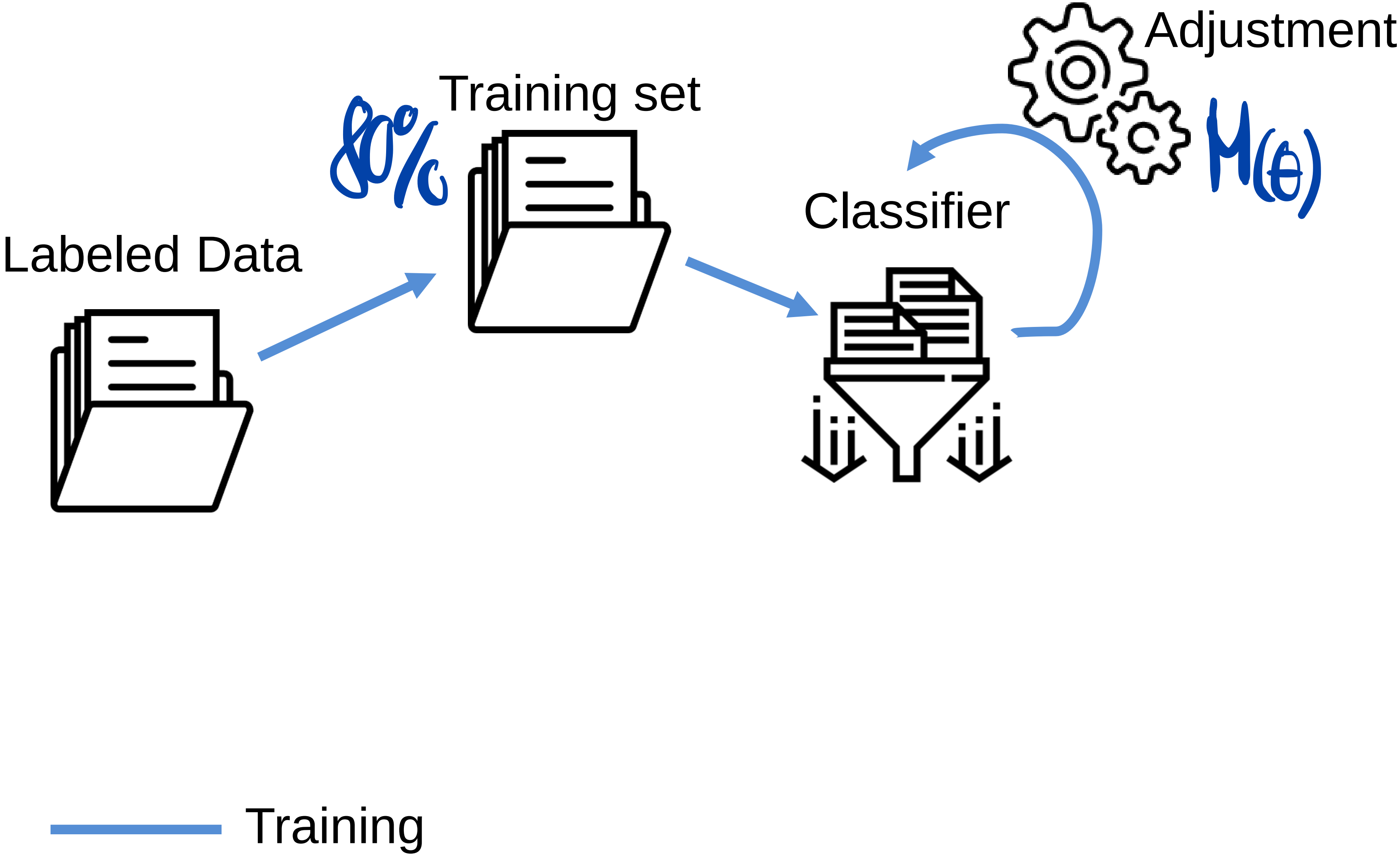
Formal Definition - Classification

$$C_{M(\theta)}: D \rightarrow Y$$

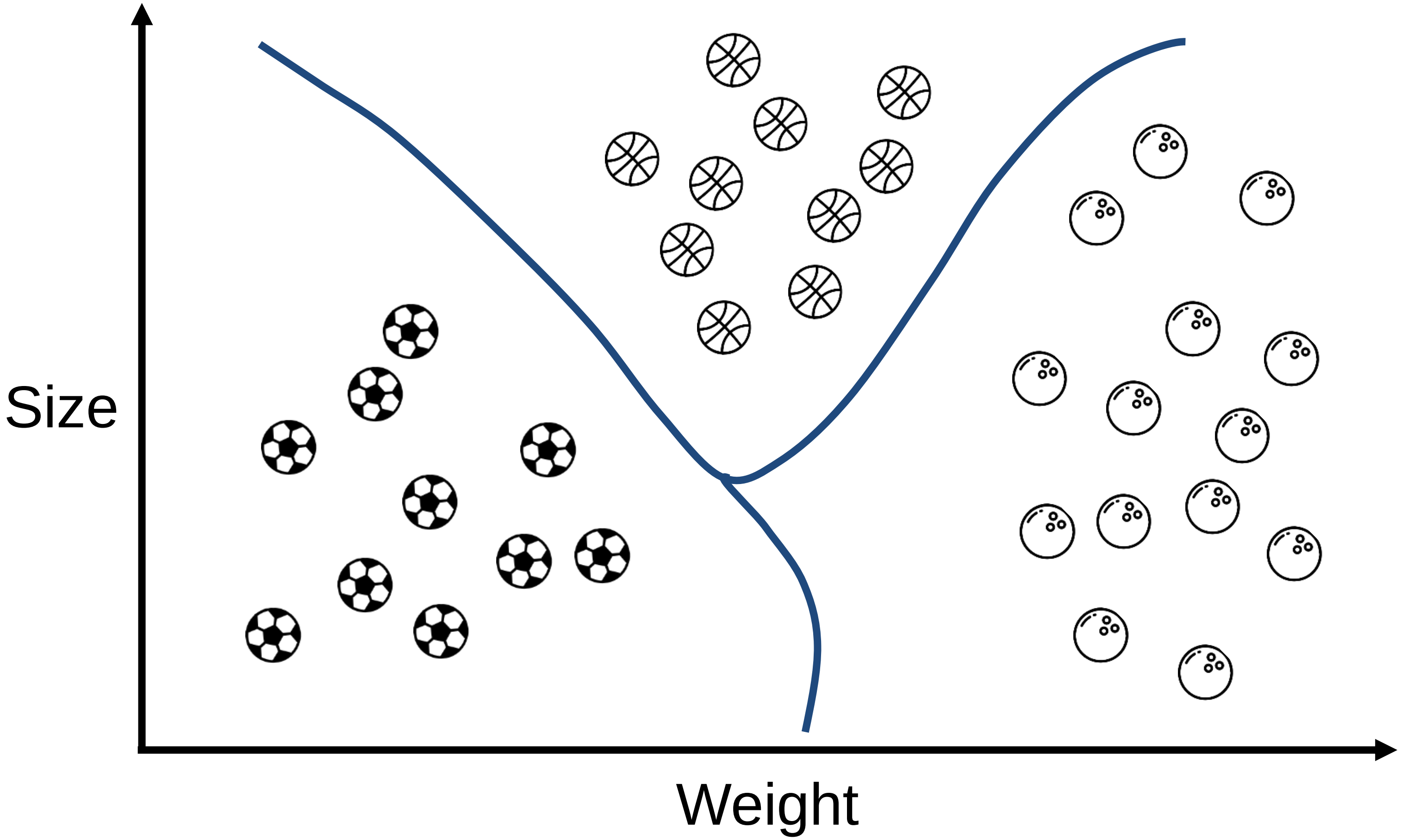
- Classifier C
- Model M with parameter θ
- Dataspace D
- Labels Y
- Training Data $O \subseteq D$ with known labels

- Training: Given O, find optimal parameter θ
- Classification: Apply $C_{M(\theta)}$ on objects from D

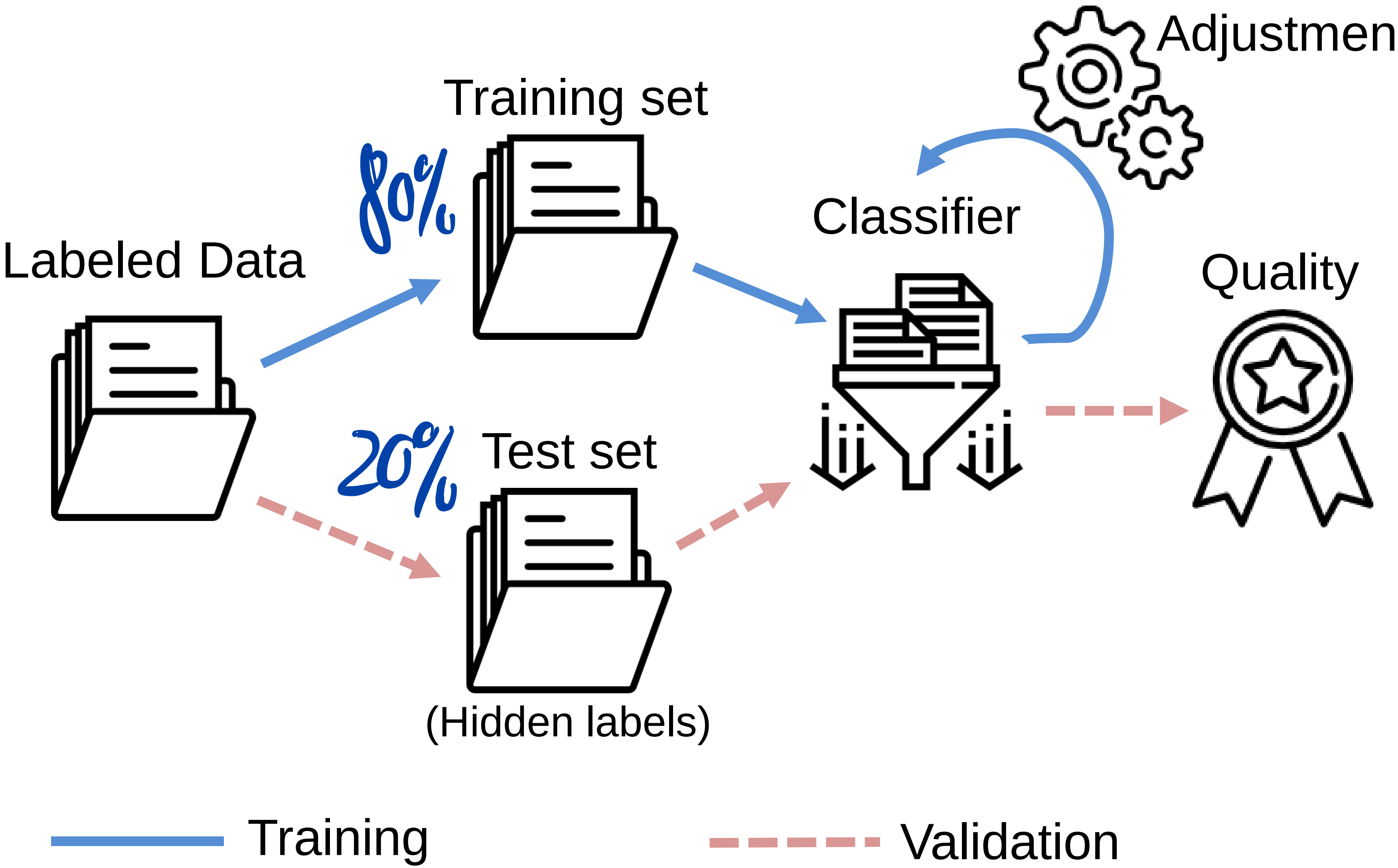
Supervised Learning - Classification



Classifier Training



Supervised Learning - Classification



Quality Measures for Classifiers

Scalability

Compactness

Accuracy

Interpretability

Efficiency

Robustness

Evaluation of Classifiers

- k-fold cross-validation
  - Decompose data set evenly into k subsets of (nearly) equal size.
  - Iteratively use k–1 partitions as training data and the remaining single partition as test data.
- Additional requirement: stratified folds
  - Class distributions in training and test set should represent the class distribution in D (or at least in O).
- Standard: 10-fold stratified cross-validation

Confusion Matrix

Correct Label \ Classified as	Class 1	Class 2	Class 3	Class 4
Class 1	45 TP	0	2 FN	1
Class 2	3	44	0	1
Class 3	0 FP	0	67 TN	0
Class 4	8	5	6	37

Positive — Negative

- Recall: $\frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{45}{45+3}$
- Precision: $\frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{45}{45+11}$
- Specificity: $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$

True Positives: TP

False Positives: FP

True Negatives: TN

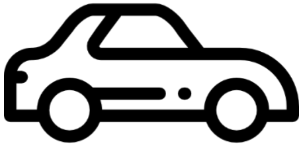
False Negatives: FN

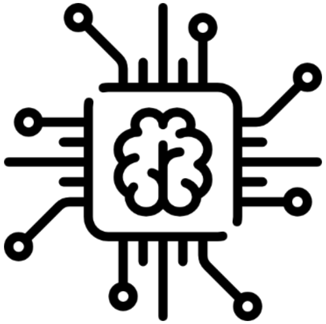
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

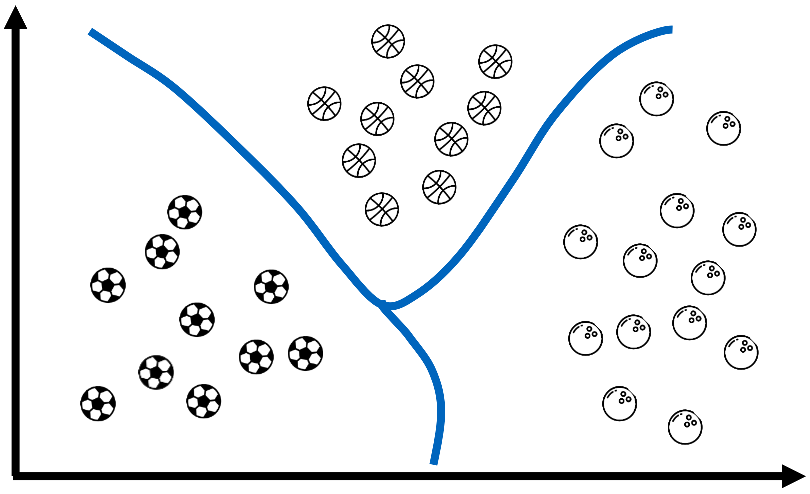
Agenda





Methods

- Descision tree
- **Logistic regression**
- **Nearest neighbors**
- **Support vector machine**
- Neural networks
- Etc.



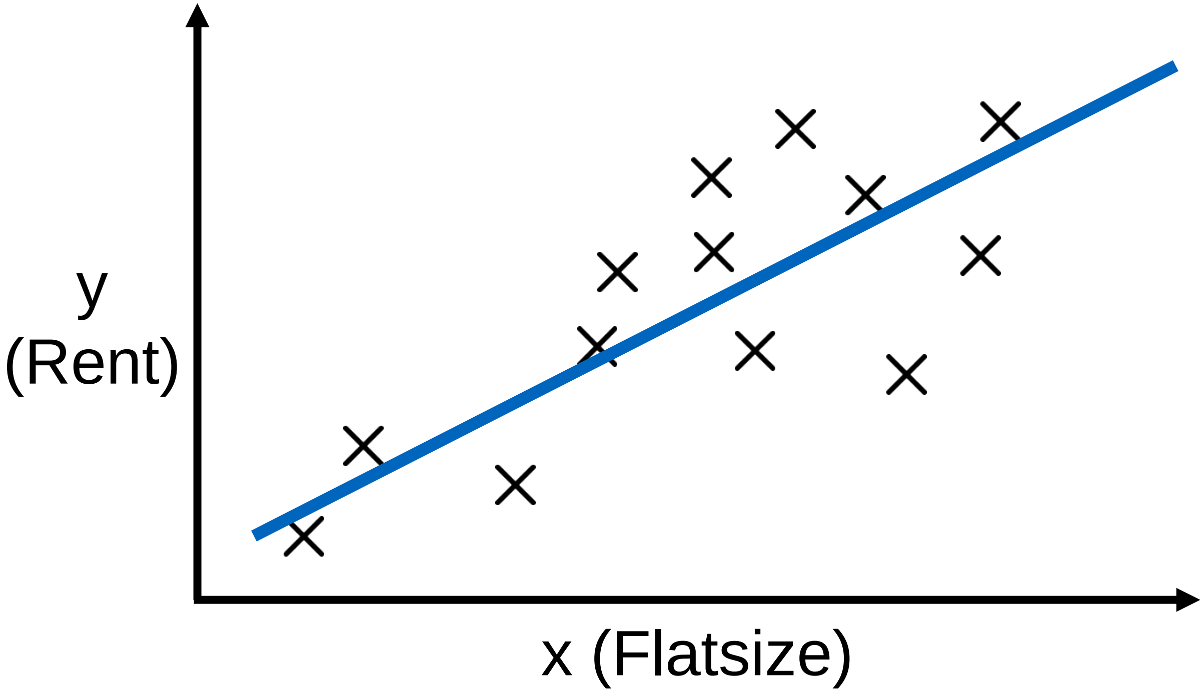
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

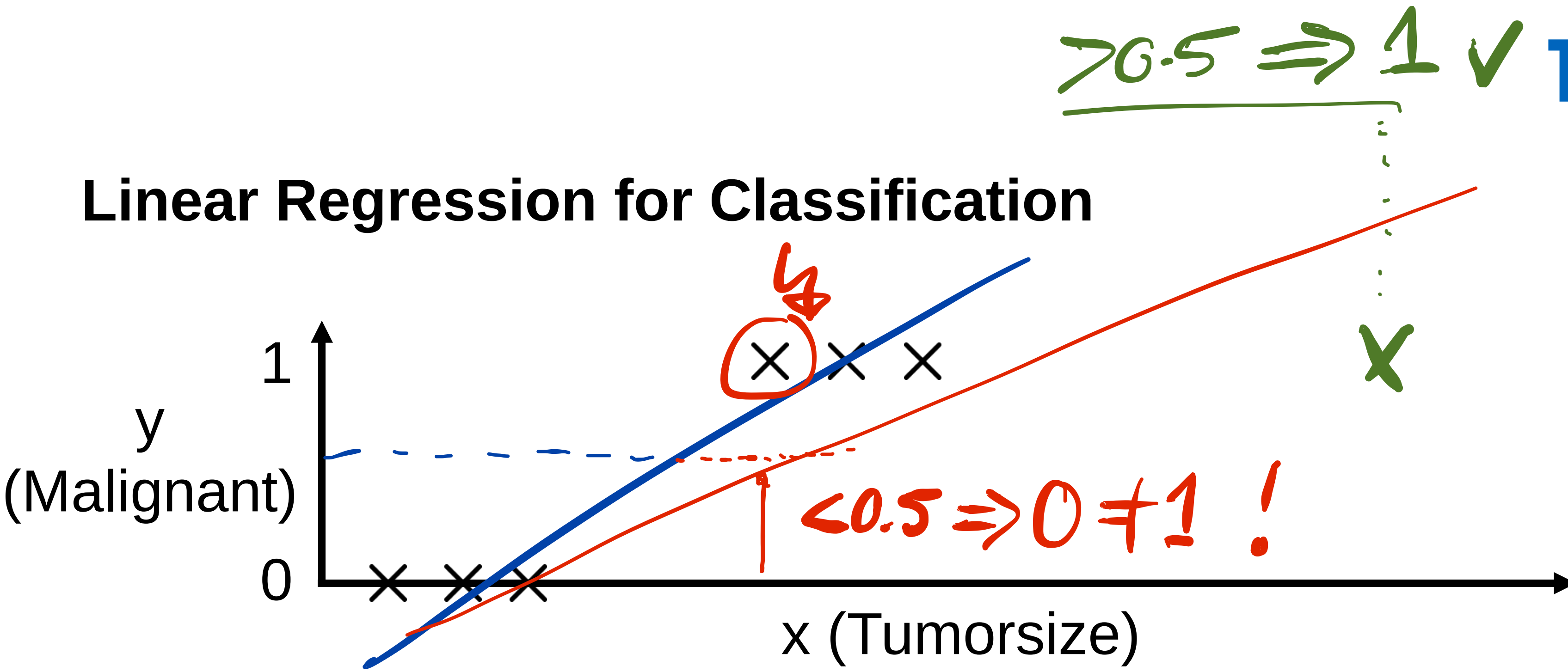
Agenda

Recap Linear Regression



$$y = h_\theta(x), \qquad y \epsilon \mathbb{R}$$

Linear Regression for Classification



$$y = h_\theta(x), \qquad y \epsilon \mathbb{R}$$

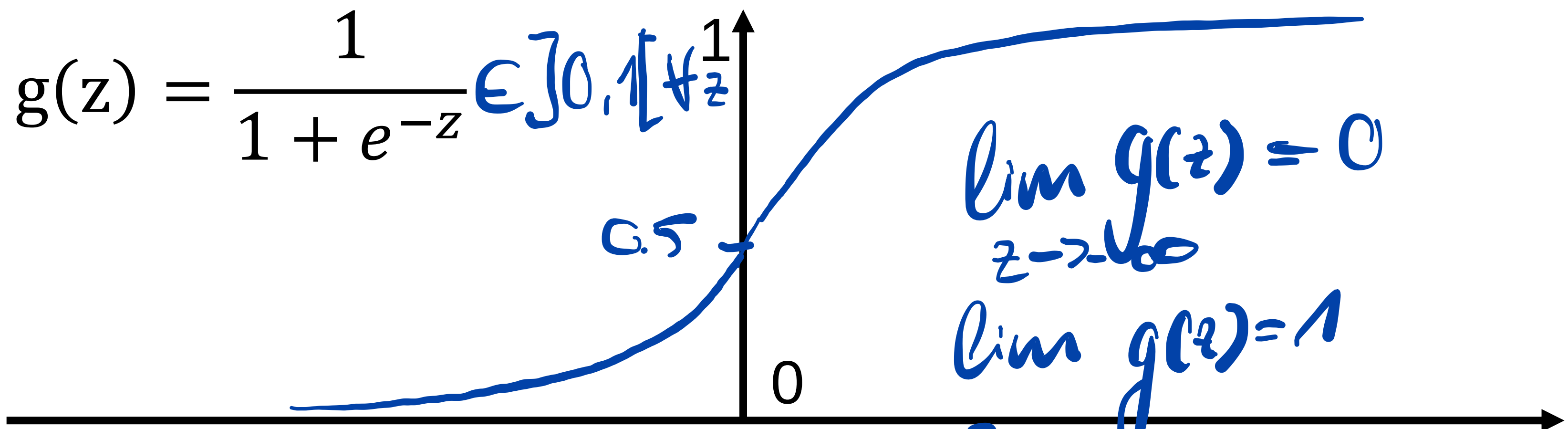
$y < 0.5$: PREDICT 0

$y \geq 0.5$: — " — 1

„S Function"

Sigmoid Function

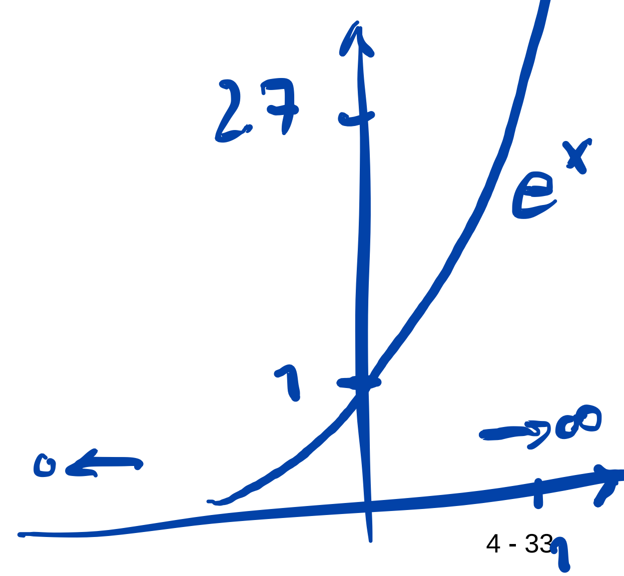
$$g(z) = \frac{1}{1+e^{-z}} \in \,]0,1[\;\forall z$$



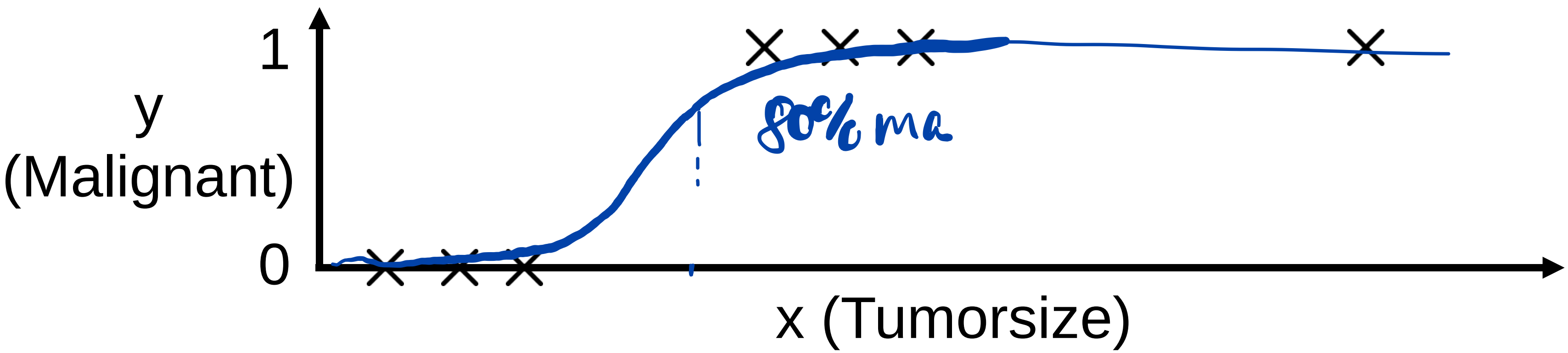
$\lim\limits_{z \to -\infty} g(z) = 0$

$\lim\limits_{z \to \infty} g(z) = 1$

z	e^{-z}	$1+e^{-z}$	$g(z)$
$\to -\infty$	$\to \infty$	$\to \infty$	$\to 0$
0	1	2	0.5
$\to \infty$	$\to 0$	$\to 1$	$\to 1$



Logistic Regression



Probabilistic classification:

$$y = g_{\theta}(h_{\theta}(x)) \qquad y \in \,]0,1[$$

Discussion Logistic Regression

- Pro:
  - **Implementation**: Easy to use
  - **Probabilistic**: Probability of an object being in a certain class
  - **Computation**: Quick training phase
  - **Insights**: Produces understandable models

- Contra:
  - **Linearity**: Hard to adopt to non-linear problems
  - **Overfitting**: Training data has to be well chosen

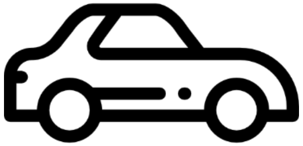
Supervised Learning: Classification

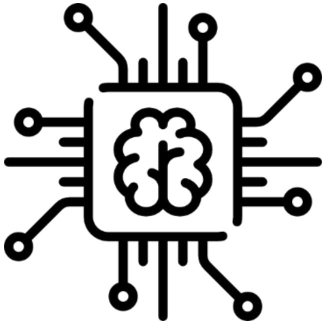
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

Agenda

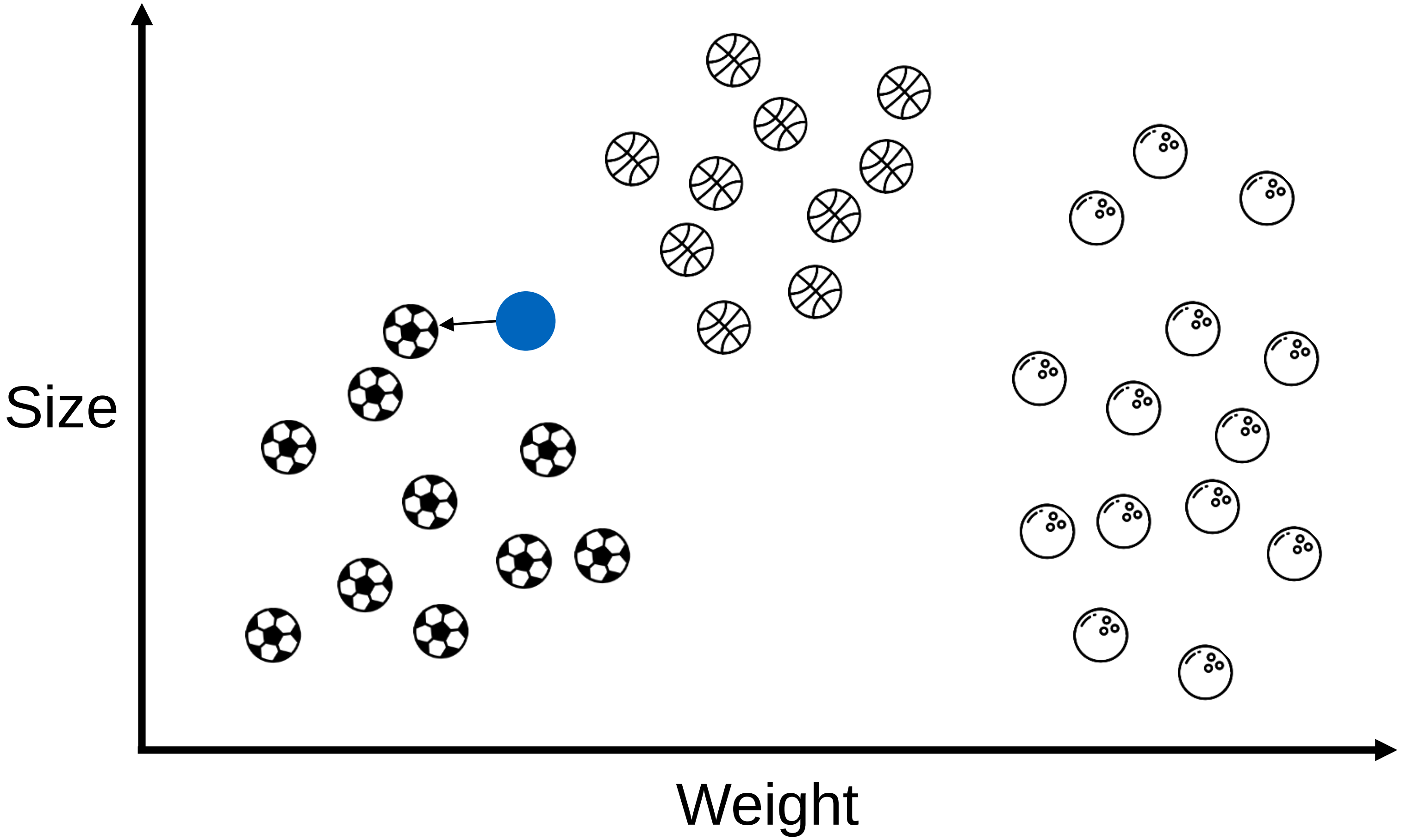
1. Chapter: Introduction
    1.1 Overview
    1.2 Training and Validation
2. Chapter: Methods
    2.1 Logistic Regression
    2.2 Nearest Neighbors
    2.3 Support Vector Machine
3. Chapter: Application
4. Chapter: Summary





Nearest Neighbor

Classify a new object based on it's nearest neighbor



Nearest Neighbor – Instance-based Learning

- No training and test phase
    - No generated model
- Store labeled training data
    - Points in a metric space
- Process training data when a new object should be classified
    - "lazy evaluation"
- Tradeoff between time and complexity
    - Hard to build a model based on a large dataset but it is easy to use
    - Easy to just store the large dataset but hard to search

Nearest Neighbor Variants

- NN classifier
  - Consider only the nearest neighbor

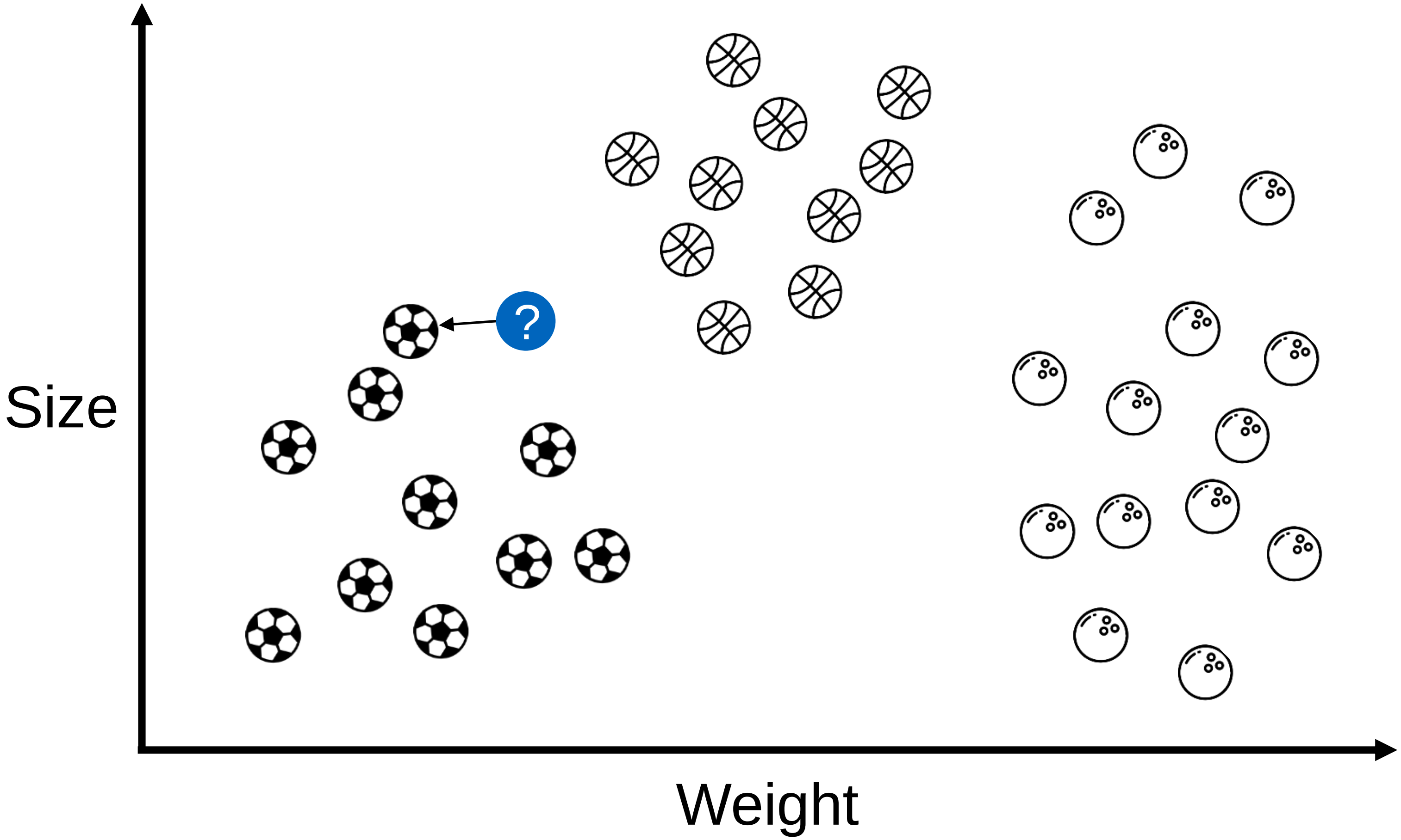
- k-NN classifier
  - Consider k nearest neighbors (k>1)

- Weighted k-NN classifier
  - Consider the weighted distances to the k nearest neighbors

- Mean-based NN classifier
  - Consider the closest mean position of a class

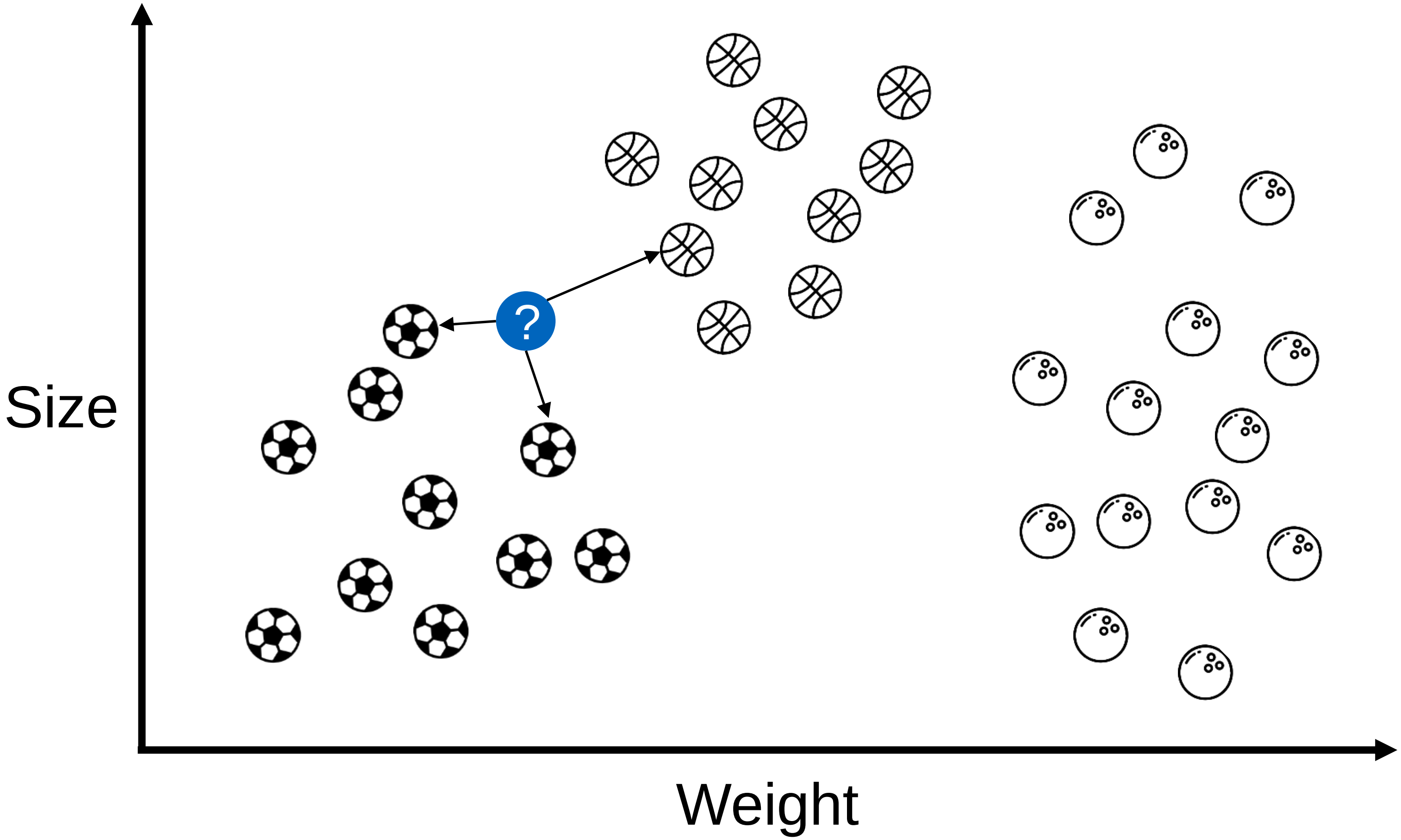
Nearest Neighbor Variants

- NN classifier
  - Consider only the nearest neighbor



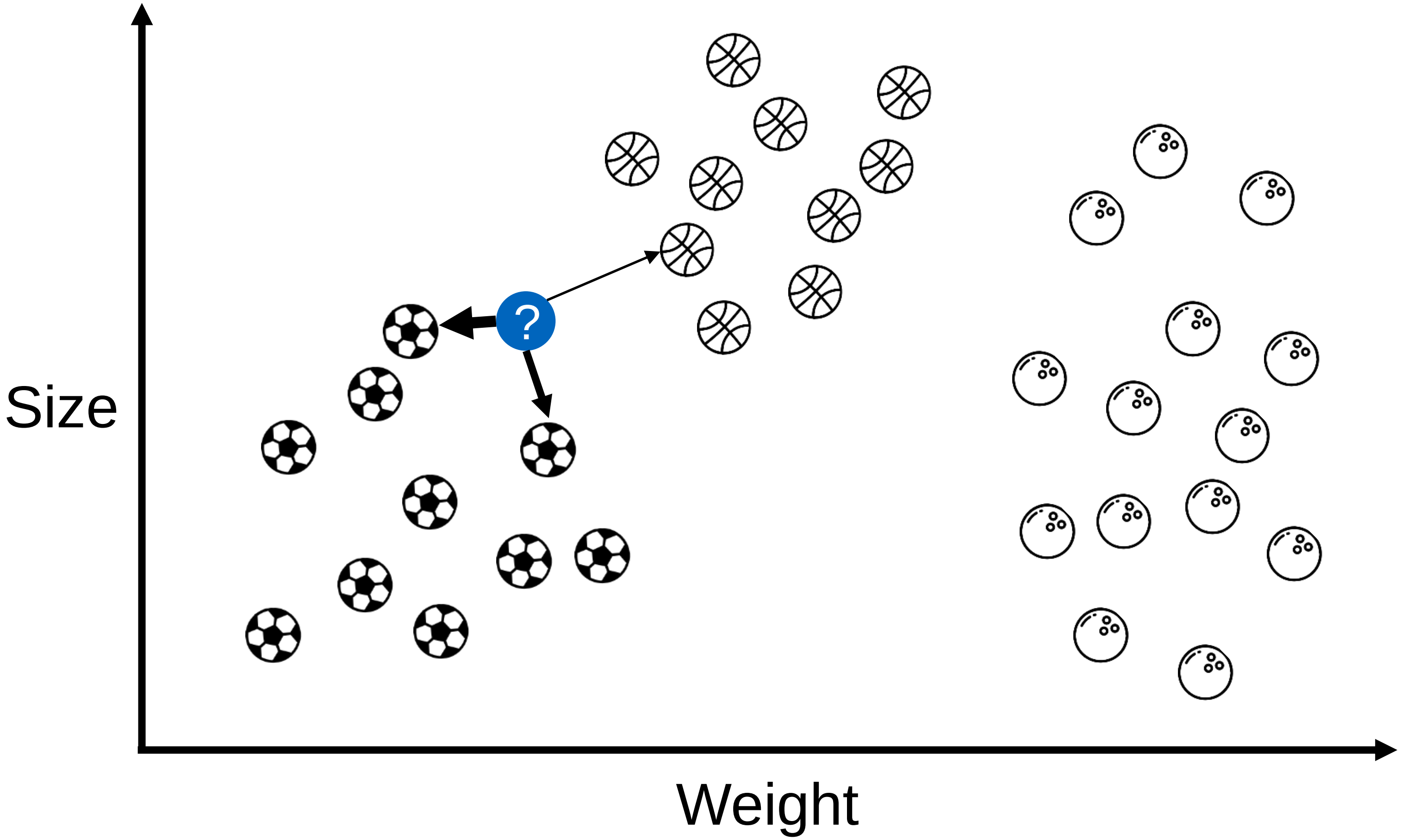
Nearest Neighbor Variants

- k-NN classifier
  - Consider k nearest neighbors (k>1)



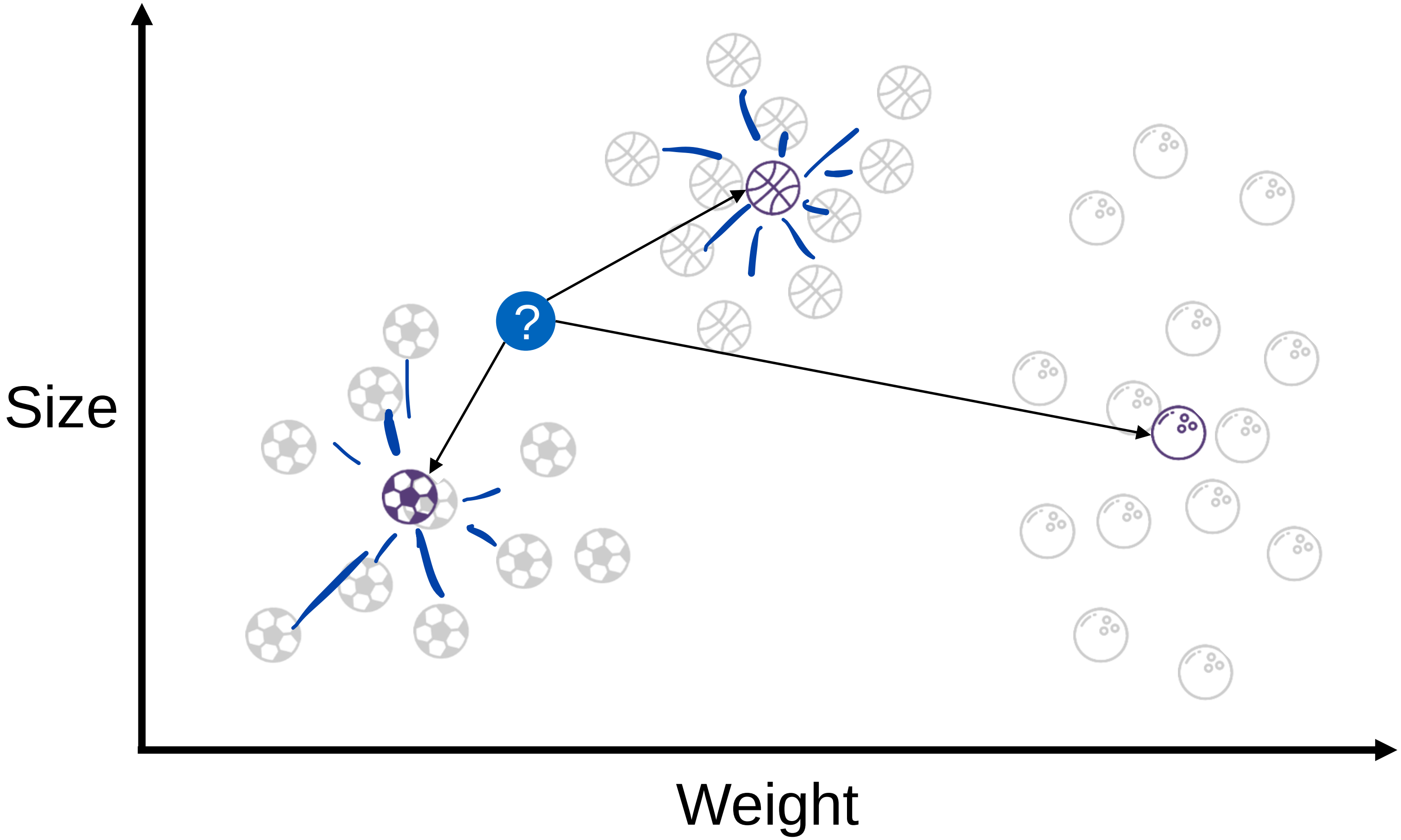
Nearest Neighbor Variants

- Weighted k-NN classifier
  - Use weights for the classes of the k nearest neighbors



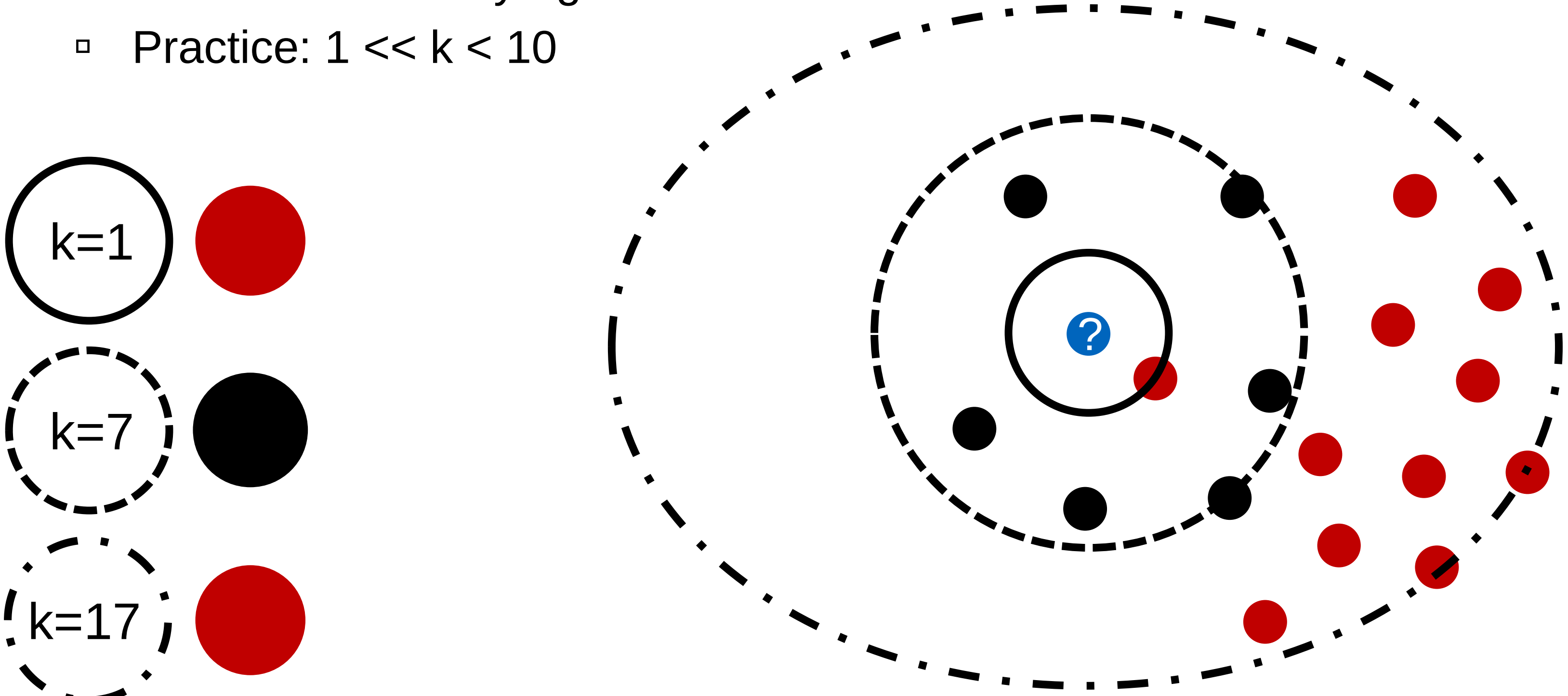
Nearest Neighbor Variants

- Mean-based NN classifier
  - Consider the closest mean position of a class



k-NN Classifier

- How to choose k?
  - Generalization vs. overfitting
  - Large k: Many objects from different classes
  - Small k: Sensitivity against outliers
  - Practice: 1 << k < 10



Weighted k-NN Classifier

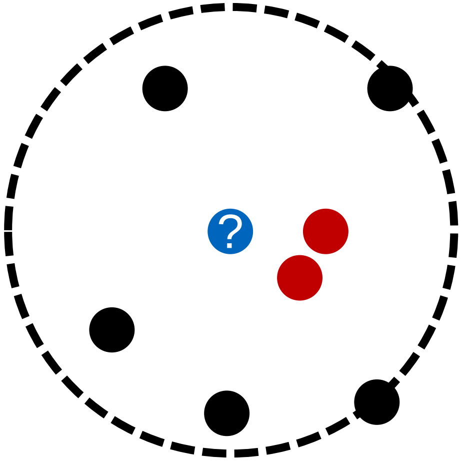
- How to weight the neighbors?
  - Frequency of neighbors class
    - $w_i = \frac{1}{frequency_i}$
  - Distance to neighbor
    - $w_i = \frac{1}{{distance_i}^2}$

k=7

Normal

Weighted (Frequency)

Weighted (Distance)



Discussion NN Classifier

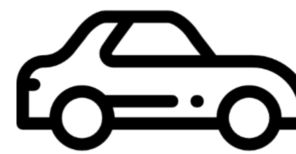
- Pro:
  - **Applicability**: Easy to calculate distances
  - **Accuracy**: Great results for many applications
  - **Incremental**: Easy adoption of new training data
  - **Robust**: Scopes with noise by averaging (k-NN)
- Contra:
  - **Efficiency**: Processing grows with training data $\mathcal{O}(n)$
    - Can be reduced to $\mathcal{O}(\log n)$ with an index structure (requires training phase)
  - **Dimensionality**: Not every dimension is relevant
    - Weight dimensions (scale axes)
- Neutral
  - Does not produce explicit knowledge about classes

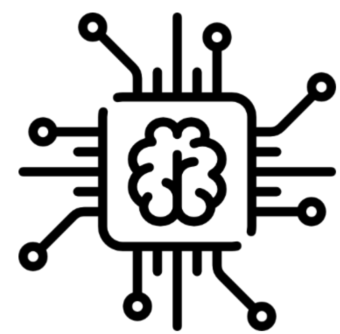
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

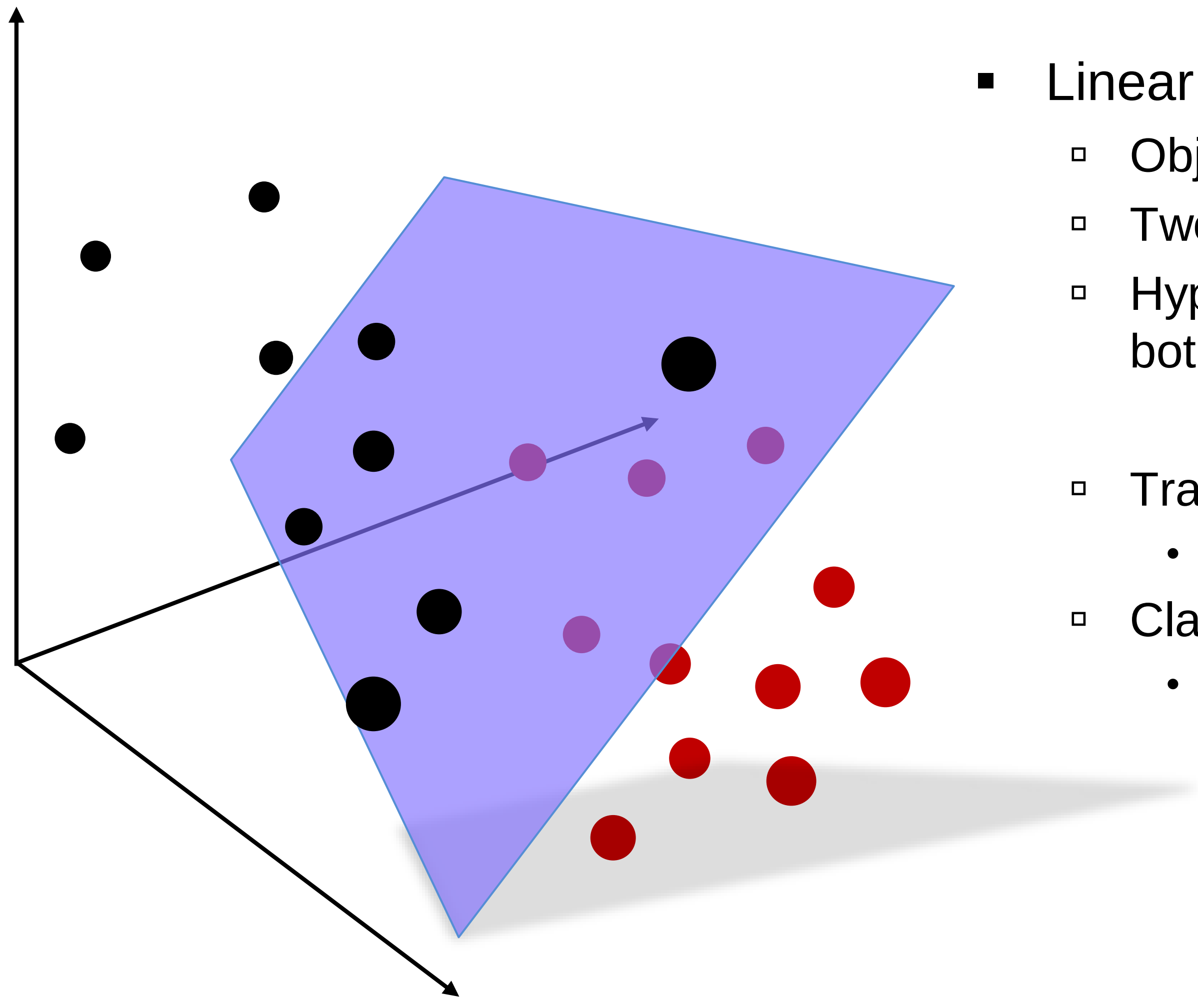
(Jan Cedric Mertens, M.Sc.)

Agenda





Support Vector Machine (SVM)

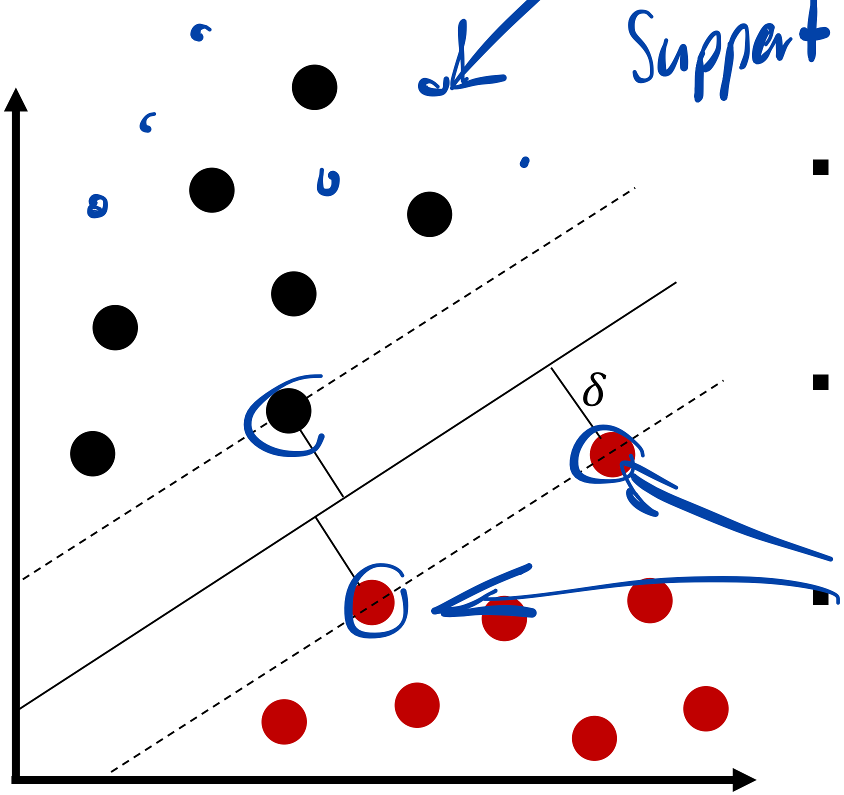


- Linear separation
  - Objects in $\mathbb{R}^d$
  - Two classes
  - Hyperplane separates both classes

  - Training
    - Compute Hyperplane
  - Classification
    - Distance to Hyperplane

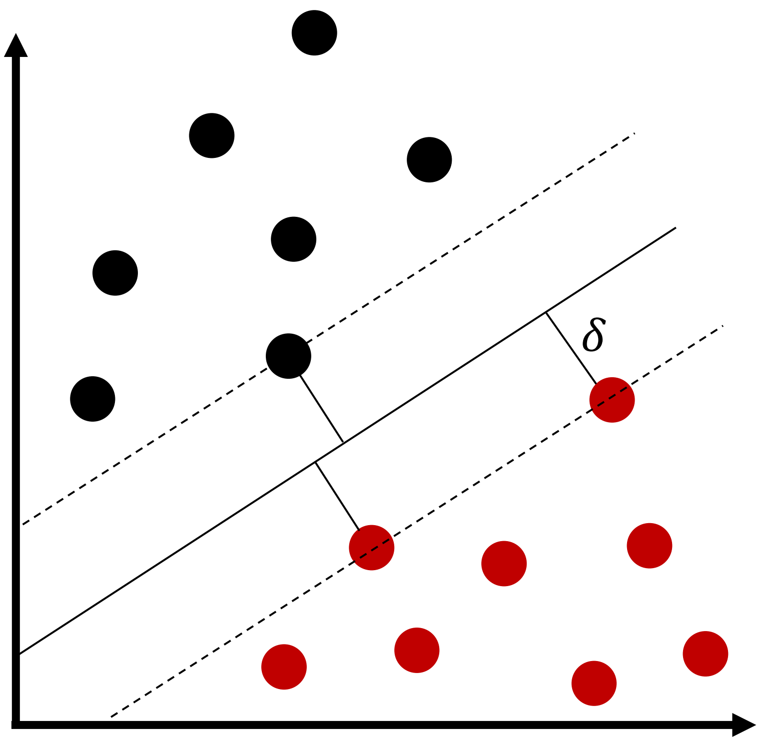
SVM - Maximum Margin Hyperplane (MMH)

More Points do not influence Support Vectors ⇒ Robust against overfitting



- Max. distance to hyperplane
  - At least δ (Margin)
- High generalization
  - Maximal stable
- Support Vector
  - Only depends on objects with distance δ

SVM – Formal Definition

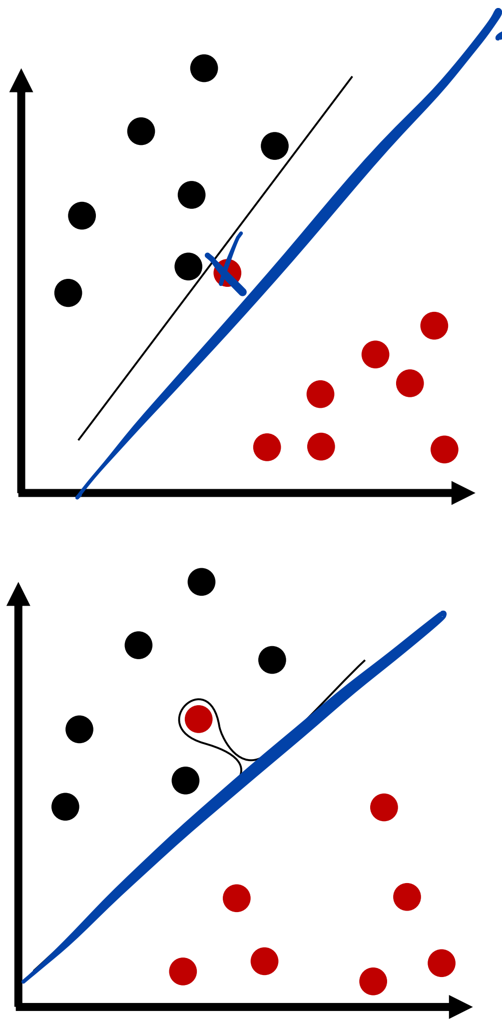


- Training data: $(x_1, y_1) \dots (x_n, y_n)$
  with $x \in \mathbb{R}^d, y \in \{-1,1\}$
- Hyperplane: $w \cdot x - b = 0$
  with w normalvector, $\frac{b}{\|w\|}$ offset from origin,
- Margin: $\delta = \frac{1}{\|w\|}$
- Training: Minimize $\| w \|$
  with $y_i(w \cdot x_i - b) \geq 1$ for $i = 1 \dots n$
- Classification:
  if $(w \cdot x - b) \geq 0$, $y = 1$; else $y = -1$
  with Data $x \in \mathbb{R}^d$

dot product is important for SVM!

SVM - Soft Margin

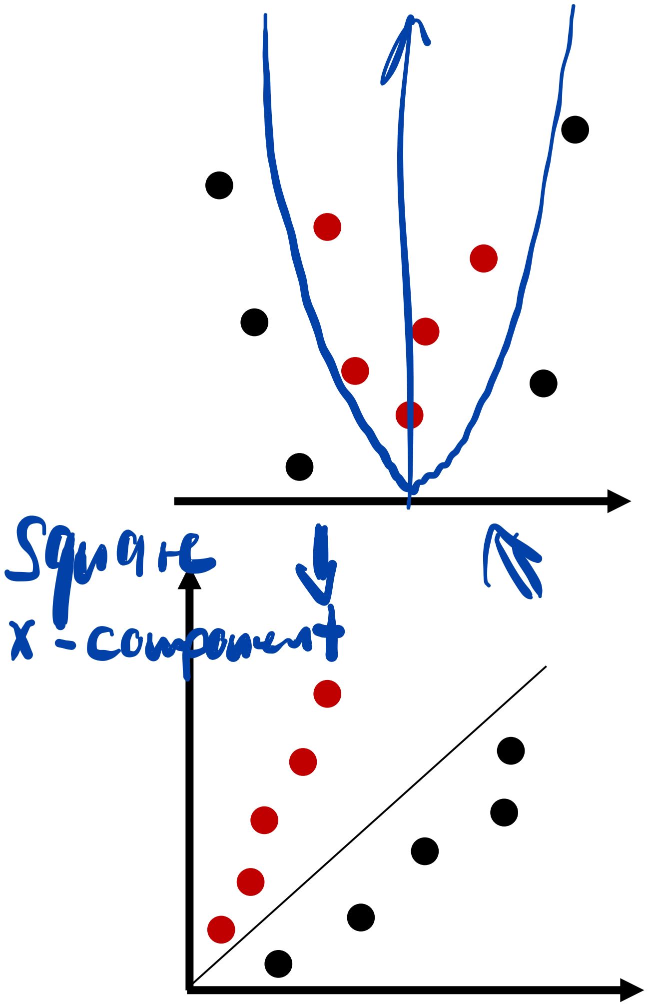
Better to have big margin and allow a small error



- Linear separation
  - Not always possible
  - Not always optimal

- Tradeoff between error and margin
  - Allow classification error to maximize margin

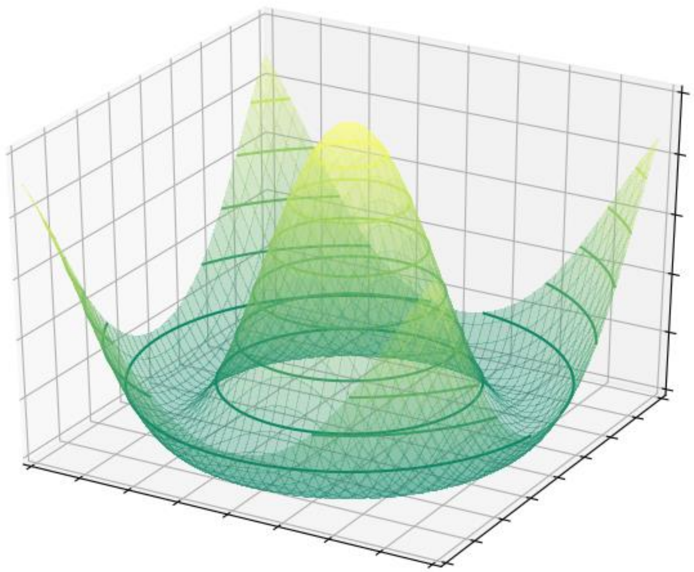
SVM - Space Transformation



- Non linear data
  - Too many errors with soft margin
- Use higher dimensional space
  - Increase dimensions until linear separation is possible
  - Transform hyperplane back to lower dimensions
  - Hyperplane becomes non-linear
- Example: Quadratic transformation
  - Hyperplane becomes polynomial of degree 2

[5]

SVM - Kernel Machines



- Space transformations
  - Lower to higher dimensions
  - Computational complex
- Hyperplane transformation
  - Higher to lower dimension
  - Feasibility not guaranteed
  - Computational complex
- Kernel
  - Computational elegant
  - Calculate dot product without full space transformation

SVM - Kernel Machines

- Replace the dot product with a non-linear kernel function

- Polynomial:

$$k(x_i, x_j) = (x_i \cdot x_j)^d$$

- Gaussian radial bias function (RBF):

$$k(x_i, x_j) = \exp(-\gamma \parallel x_i - x_j \parallel^2) \text{ for } \gamma > 0$$

- Linear, sigmoid, hyperbolic, …

SVM - Kernel Example

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \longrightarrow \begin{pmatrix} x_1 \cdot x_1 \\ x_1 \cdot x_2 \\ \vdots \\ x_3 \cdot x_3 \end{pmatrix} \in \mathbb{R}^9$$

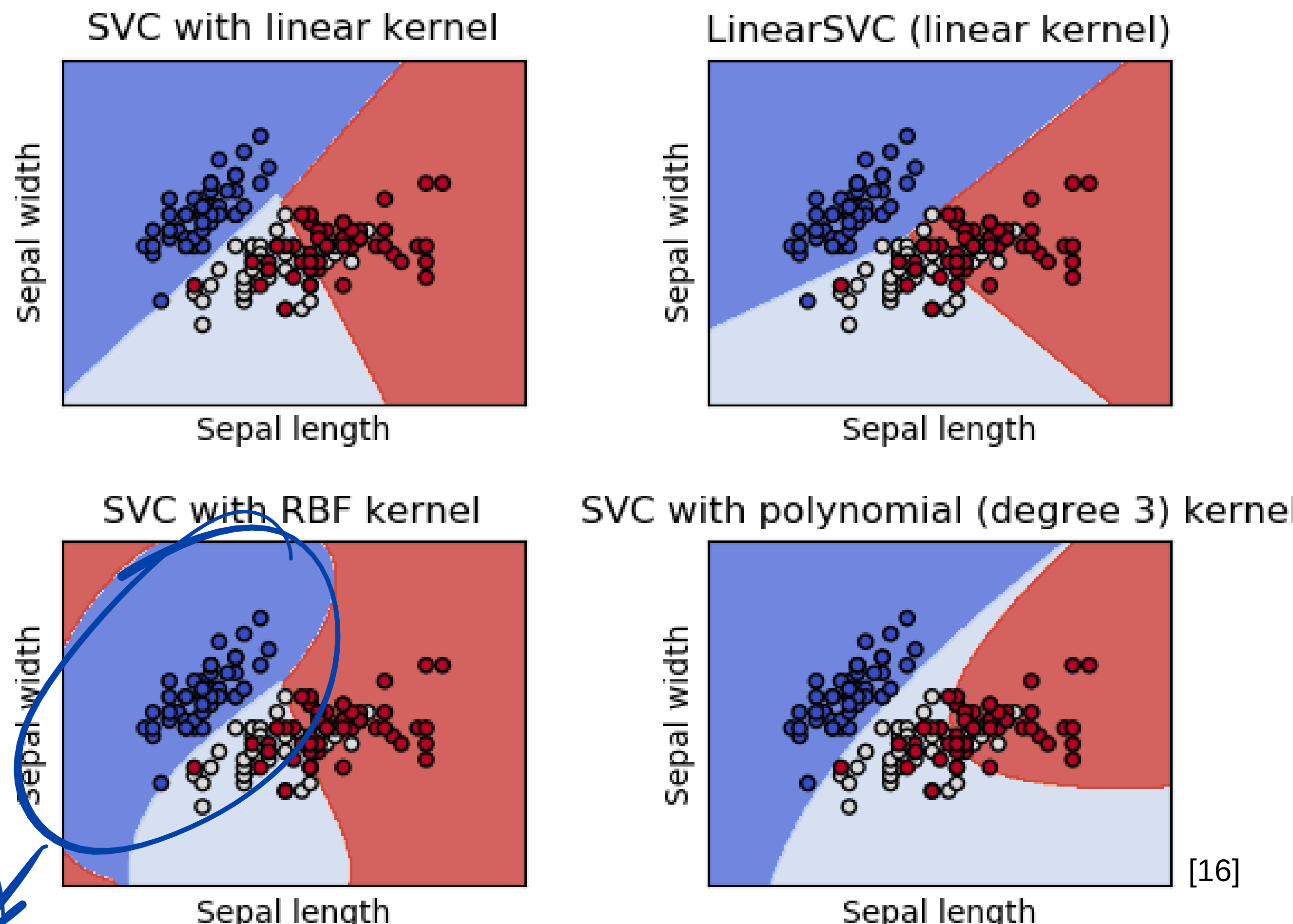
- $f\colon \mathbb{R}^3 \to \mathbb{R}^9$
  $f(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$
  $k(x, y) = (x \cdot y)^2$

- $x = (1,2,3), \quad y = (4,5,6)$
  $f(x) = (1,2,3,2,4,6,3,6,9)$ $\in \mathbb{R}^9$
  $f(y) = (16,20,24,20,25,30,24,30,36)$ $\in \mathbb{R}^9$
  $f(x) \cdot f(y) = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$

- $k(x, y) = (4 + 10 + 18)^2 = 32^2 = 1024$

→ no transformation to $\mathbb{R}^9$ required

SVM - Kernel Machines



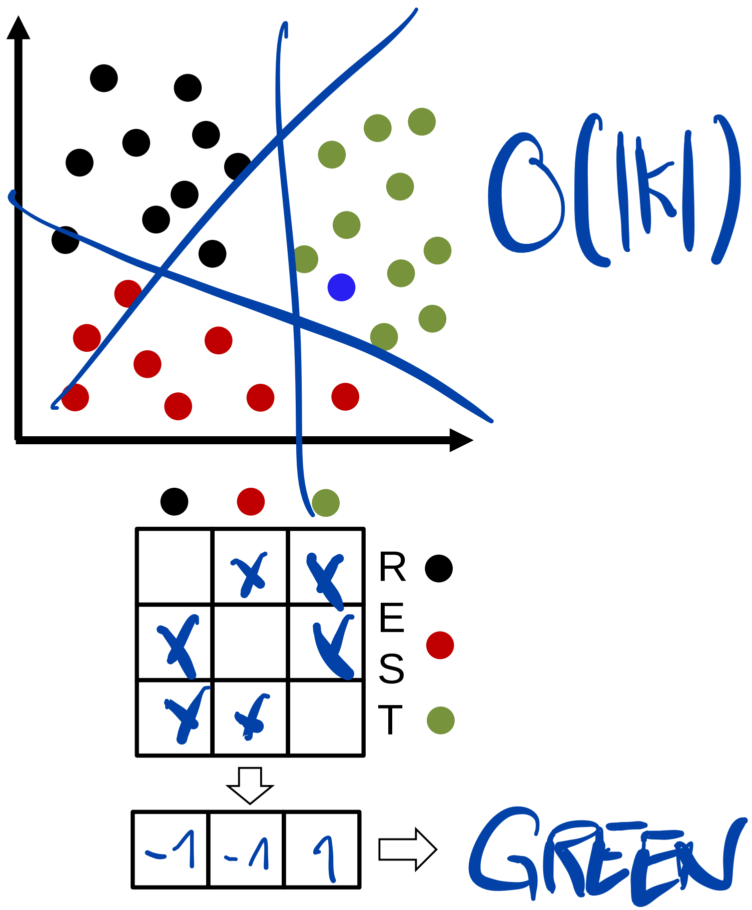
[16]

RBF: Good to extract a class that is surrounded

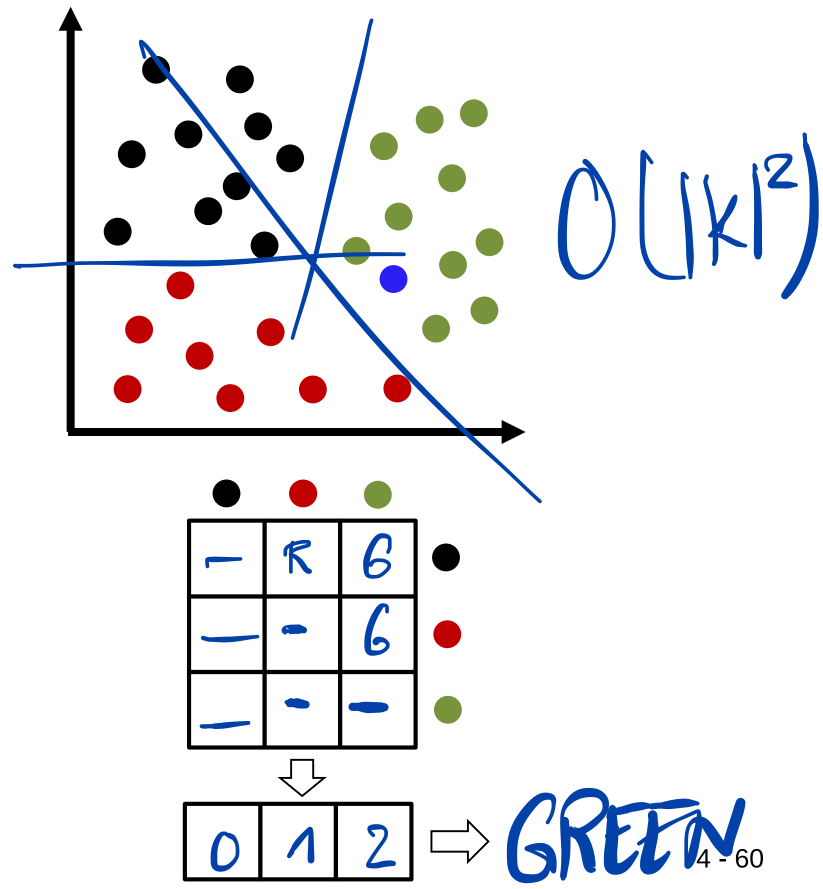
Multi Class SVM

Combination of SVMs

1 vs. rest



1 vs. 1



Discussion SVM

- Pro:
  - **Accuracy**: High classification rate
  - **Effective**: Even when number of dimensions > number of samples
  - **Robust**: Low tendency to overfitting
  - **Compact Models:** “Plane in space”
  - **Versatile**: Different kernel function

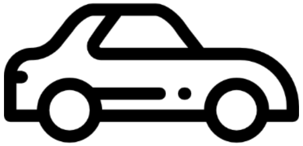
- Contra:
  - **Efficiency**: Long training phase
  - **Complexity**: High implementation effort
  - **Black-Box:** Hard to interpret models

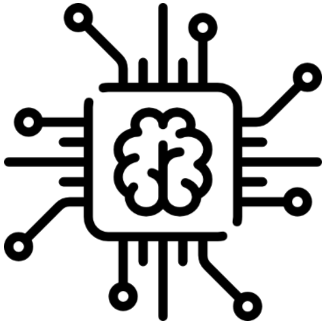
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

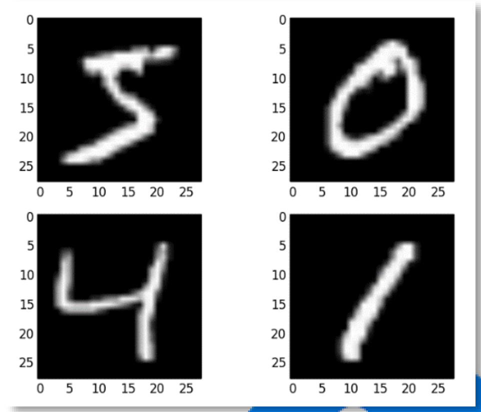
Agenda





Classification Problems

- Big data
  - Find patterns
  - Make data usable
- Image classification
  - Handwritten digits
  - X-rays
- Music classification
  - Shazam
- Speech/Language classification
  - Siri/Alexa/Echo
- Fault detection
  - Quality control during production



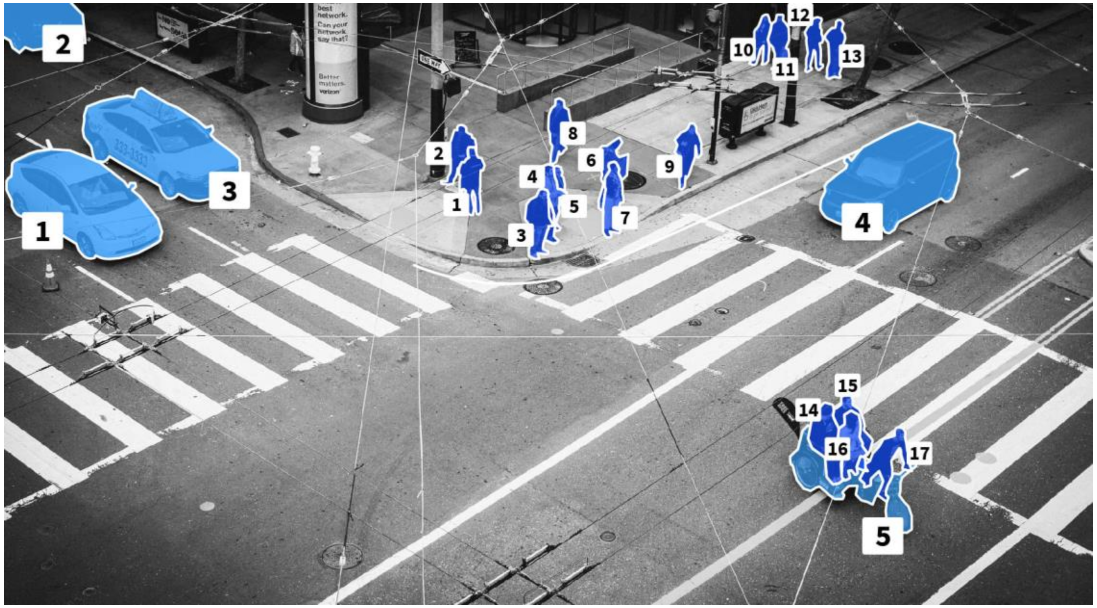
[6]

[7]

[8]

Classification for automotive technology

- Example: **Perception**
  - Camera outputs pixel array
  - Classification adds value to each pixel
  - Pixel segmentation
  - Object detection
  - Object tracking



[9]

Vehicle Detection and Tracking

[10]

Vehicle Detection and Tracking

- Get training data
- Extract features from images
- Generate a model based on the features
- Take one video frame and classify the features of the sub-images
- Merge classified areas

Training Data

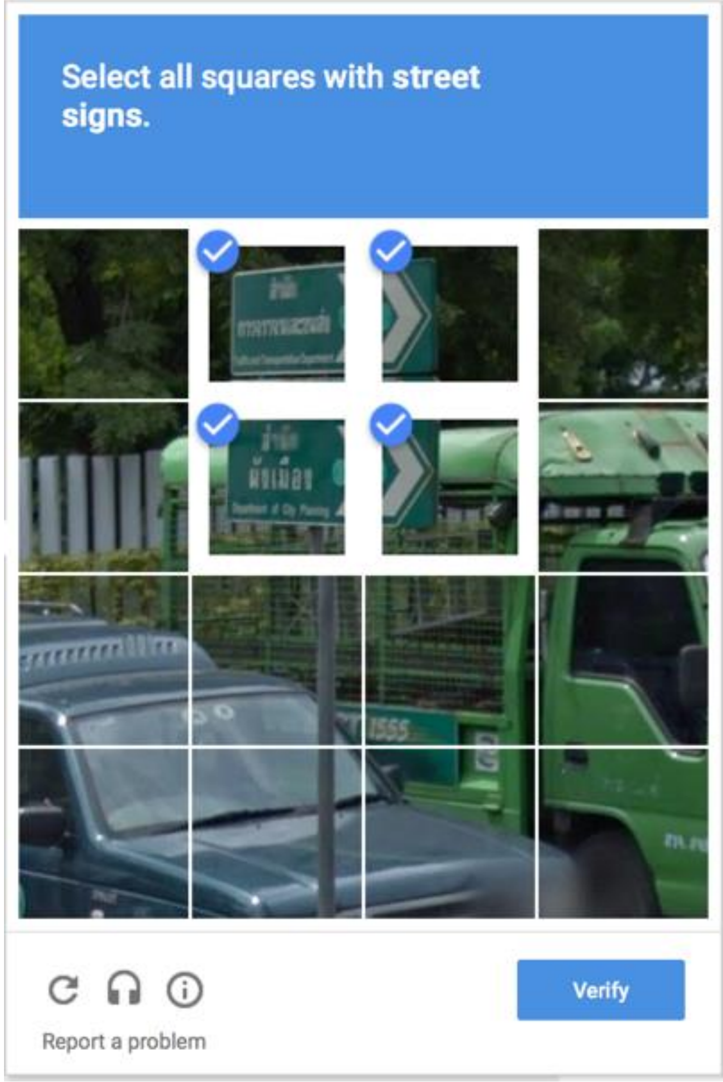
- Required label: „car“ or „no car“
- Required Images:
  - Same format used for classification
  - Represenative for what we expect to find in the videostream
  - 8000 images (90 % training and 10 % test)

[11]

Training Data

- How to get labeled data?
  - Label data by yourself
  - Pay someone else to label your data
  - Let other label your data for free

- Collection of labeled data
  - Digits: MNIST
    - http://yann.lecun.com/exdb/mnist/
    - 70k images
  - Cars: KITTI
    - www.cvlibs.net/datasets/kitti/
    - 80k images



[12]

Training Data

„It's terrifying that both of these things are true at the same time in this world:

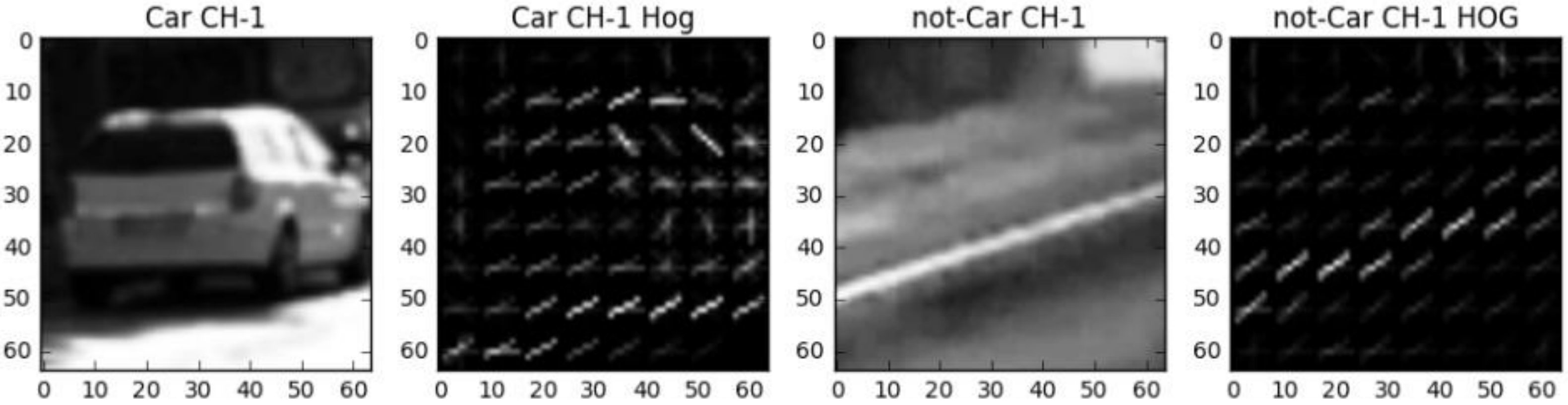
1. *Computers drive cars around*

2. *The state of the art test to check that you're not a computer is whether you can successful identify stop signs in pictures“*

- Anonym

Feature Extraction

- Histogram of oriented gradients (HOG)
  - Compressed and encoded version of an image



[10]

Build SVM Classifier

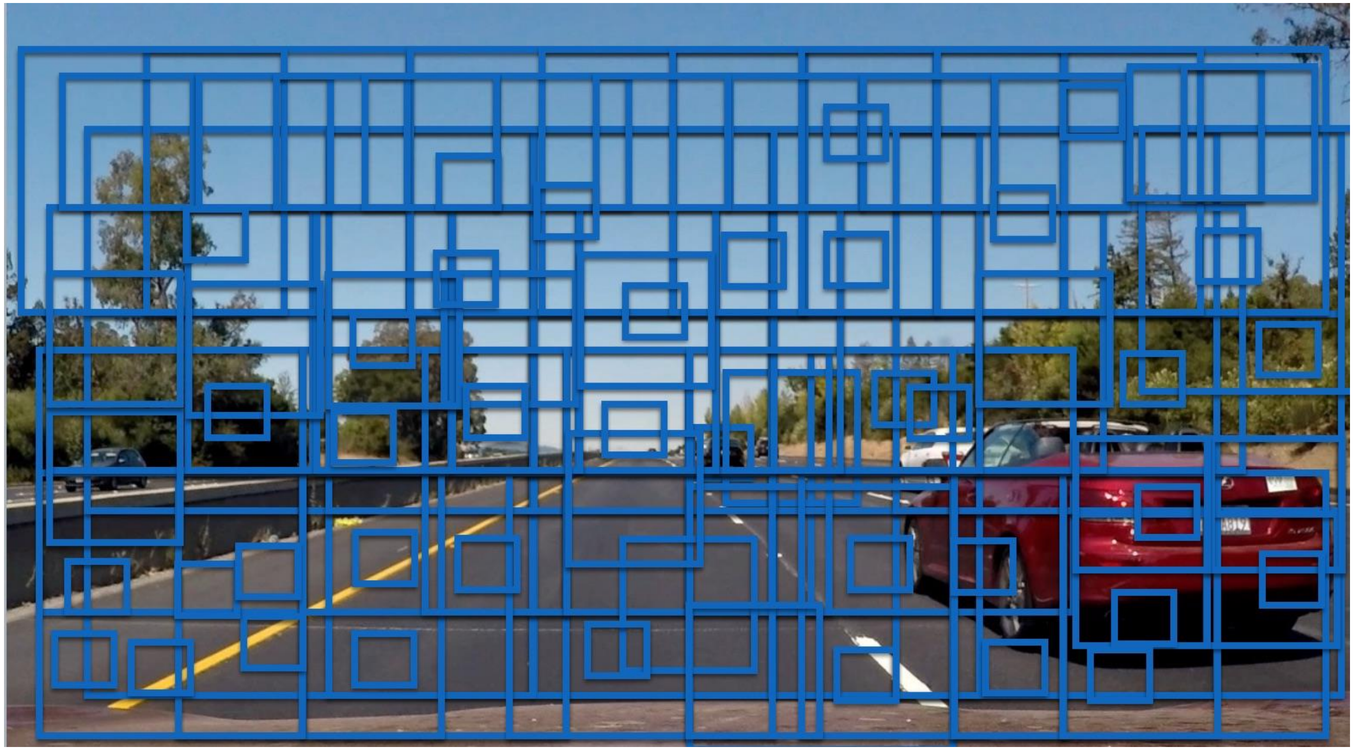
- Machine learning libraries (python)
  - scikit-learn (http://scikit-learn.org/)

```
>>> from sklearn import svm
>>> clf = svm.SVC()
>>> clf.fit(training_features, training_labels)
>>> clf.score(test_features, test_labels)
>>> clf.predict(new_feature)
```

  - Training: 1.44 Seconds
  - Test: Accuracy = 0.9848
  - Prediction: 0 or 1

Classifiy sub-images

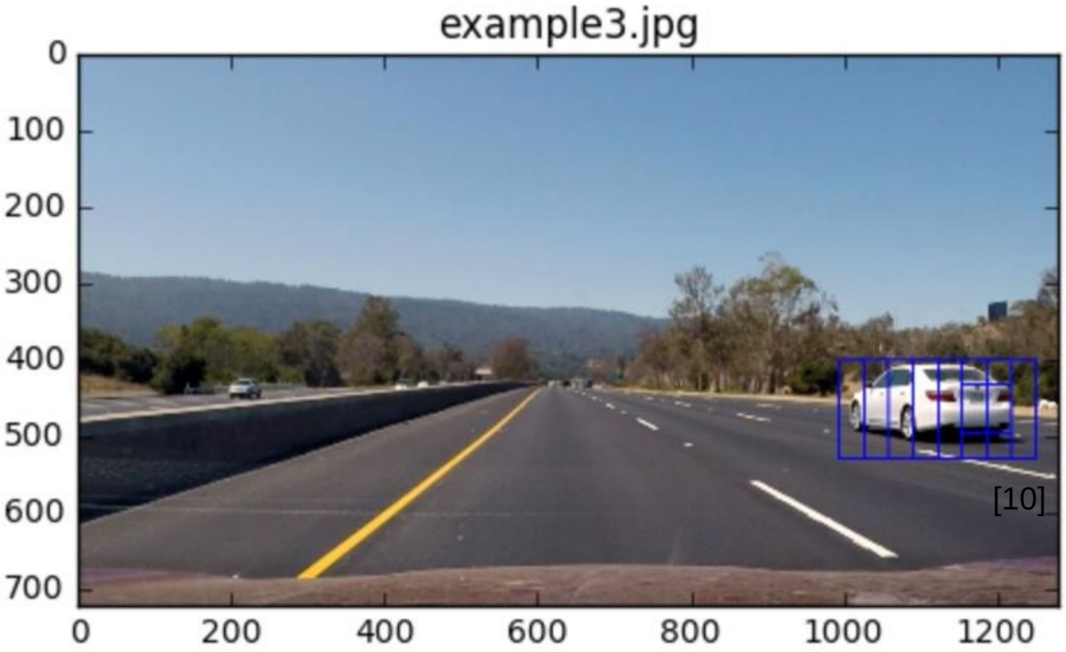
- Produce sub-images of each frame for classification

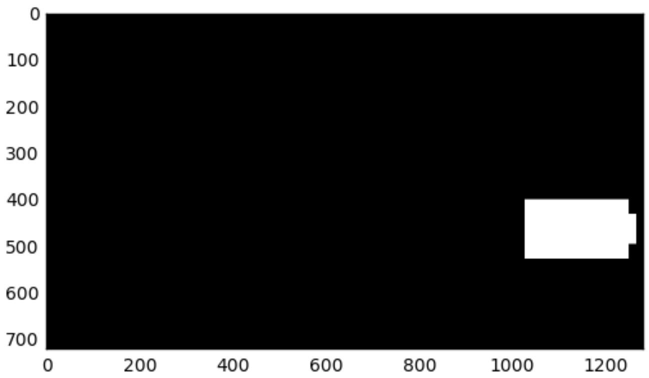


[10]

Merge classified areas

- Merge classes of sub-images





[10]

Final Output

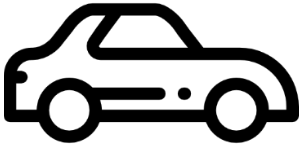
[10]

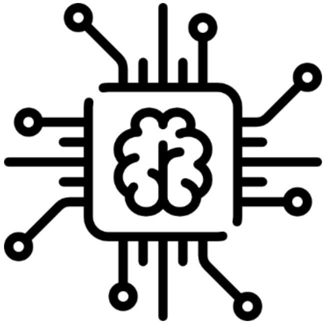
Supervised Learning: Classification

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Jan Cedric Mertens, M.Sc.)

Agenda





Summary

What did we learn today:

- **Classification** is about assigning given classes to data.
- We need lots of **training data** to build a model for classification.
- **Machine learning** can extract knowledge from huge datasets.
- Classification is a **supervised learning** problem.
- We need labeled data for training and validation (hidden label).
- We have several criteria to measure the **quality of a classifier**.
- The concepts of **Logistic regression, nearest neighbor and SVM**
- We can use linear regression together with a sigmoid function as classification method.
- Nearest neighbor is an instance based learning method, no training is required.

Summary

What did we learn today:

- **SVMs** are linear classifier using a maximum margin hyperplane.
- With the **kernel trick**, SVMs can be used for non-linear classification.
- Classification is very important for the **perception**, e.g. in cars.
- Acquiring lots of **labeled data** is a problem.
- We have access to good and easy to use **python libraries** for classification.
- We have access to many **open source datasets** (e.g., KITTI for car images).
- Training with big datasets can take **a long time**.
- We have to **partition, classify and then merge** images.
- We have to **extract features** from images for the classification.

Sources

- [1] https://9gag.com/gag/anXDLVn/the-russian-gangster-starter-kit
- [2] https://funnyjunk.com/My+neighbours+like+this/funny-pictures/6231925/
- [3] http://creepypasta.wikia.com/wiki/Alien_Life
- [4] https://www.youtube.com/watch?v=QopUtQobWJ0
- [5] https://www.youtube.com/watch?v=9NrALgHFwTo
- [6] https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/
- [7] https://play.google.com/store/apps/details?id=com.shazam.android&hl=de
- [8] https://en.wikipedia.org/wiki/Siri
- [9] https://playment.io/semantic-segmentation-tool/
- [10] https://eu.udacity.com/course/self-driving-car-engineer-nanodegree--nd013
- [11] www.cvlibs.net/datasets/kitti/
- [12] https://samsclass.info/120/proj/captchas-021916.htm
- [13] https://www.merriam-webster.com/dictionary/classification
- [14] https://microage.com/blog/machine-learning-new-cure-all/
- [15] https://www.reddit.com/r/de/comments/8ammn4/schnellesgeldmachentxt/
- [16] https://scikit-learn.org/stable/auto_examples/svm/plot_iris.html

Acknowledgment

- **Machine Learning (Stanford/Coursera)**
  - Andrew Ng

  https://www.coursera.org/learn/machine-learning

- **Knowledge Discovery in Databases I (LMU)**
  - Prof. Dr. Peer Kröger

  http://www.dbs.ifi.lmu.de/cms/studium_lehre/lehre_master/kdd1718/index.html