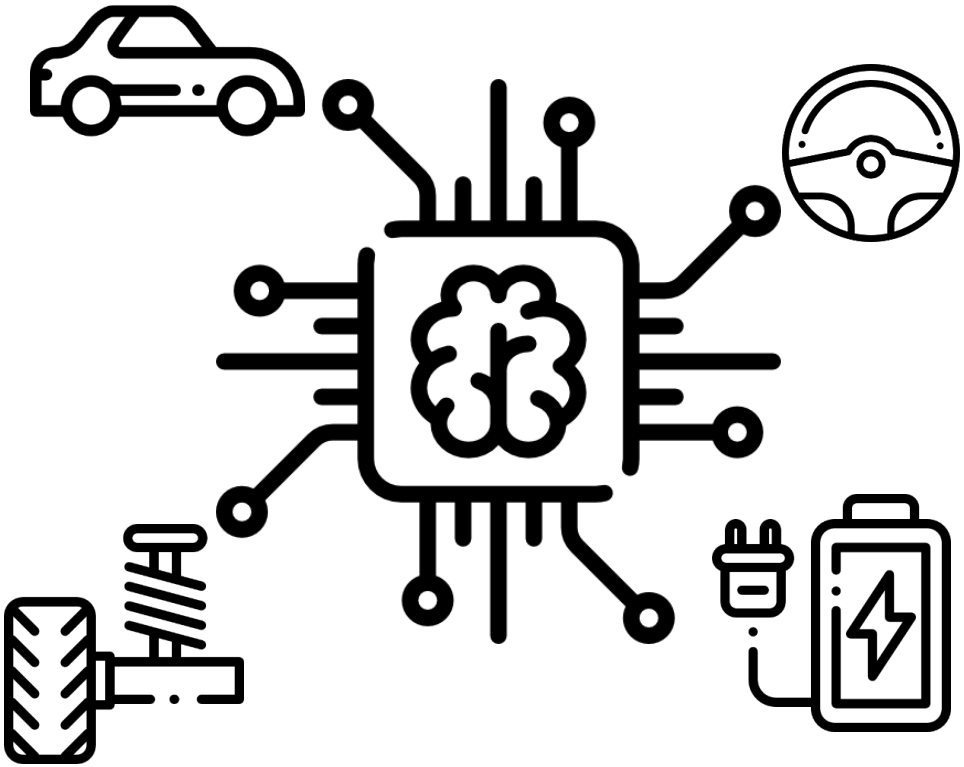
Institute of Automotive Technology
TUM School of Engineering and Design
Technical University of Munich

# Artificial Intelligence in Automotive Technology

Maximilian Geißlinger / Fabian Netzler

Prof. Dr.-Ing. Markus Lienkamp

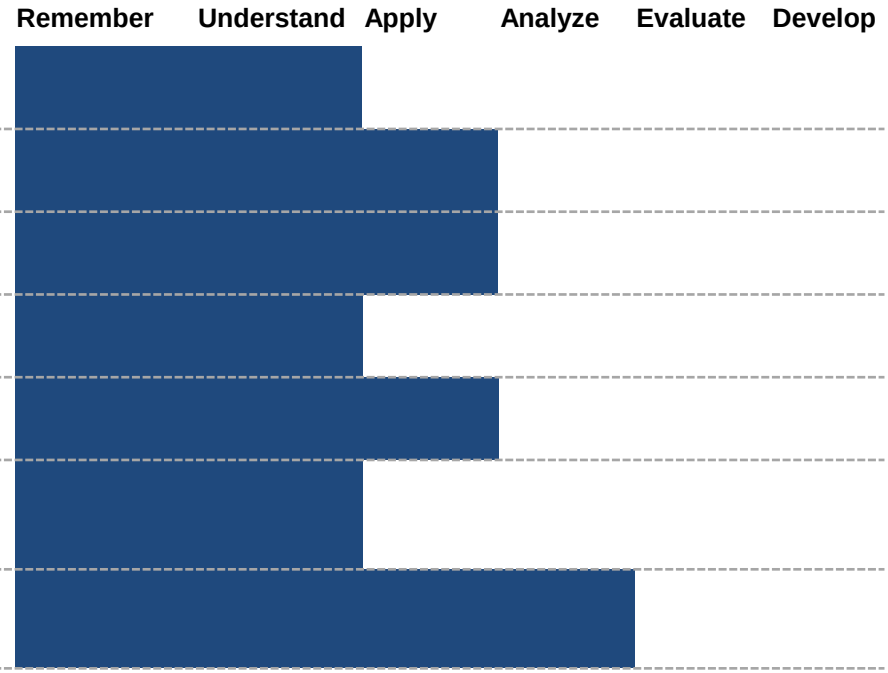


# Lecture Overview

| Lecture 16:15-17:45 \| Practice 17:45-18:30 | |
|---|---|
| **1 Introduction: Artificial Intelligence** | 20.10.2022 – Maximilian Geißlinger |
| **2 Perception** | 27.10.2022 – Sebastian Huber |
| **3 Supervised Learning: Regression** | 03.11.2022 – Fabian Netzler |
| **4 Supervised Learning: Classification** | 10.11.2022 – Andreas Schimpe |
| **5 Unsupervised Learning: Clustering** | 17.11.2022 – Andreas Schimpe |
| **6 Introduction: Artificial Neural Networks** | 24.11.2022 – Lennart Adenaw |
| **7 Deep Neural Networks** | 08.12.2022 – Domagoj Majstorovic |
| **8 Convolutional Neural Networks** | 15.12.2022 – Domagoj Majstorovic |
| **9 Knowledge Graphs** | 12.01.2023 – Fabian Netzler |
| **10 Recurrent Neural Networks** | 19.01.2023 – Matthias Rowold |
| **11 Reinforcement Learning** | 26.01.2023 – Levent Ögretmen |
| **12 AI-Development** | 02.02.2023 – Maximilian Geißlinger |
| **13 Guest Lecture** | 09.02.2023 – to be announced |

# 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 pro and cons.

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

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

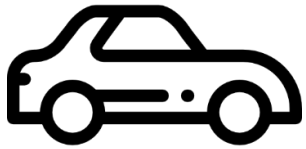
… analyze the quality of a given classifier regarding to different criteria.

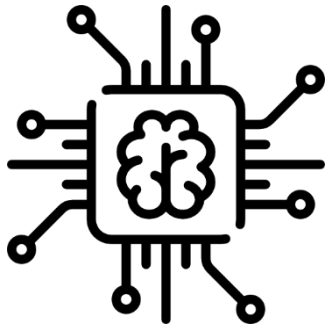
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

## Agenda







# Classification

*“Systematic arrangement in groups or categories according to established criteria”* [13]

[3]

**Additional Slide**

Classification, independent of machine learning and AI, is about assigning something perceived to a known class. The perceived can be an object, a process, a behavior or similar. The classes in our everyday life can be arbitrarily generic or specific, and can also overlap. If we blindly drink a beer we can classify it as "liquid", but we can also go much further and classify it as a "drink", a "beer“ or "bitter“. In dependence of our training (if we have often drunk a beer of the same brand), we can also classify the brand although it may not be written on the bottle.

A core element of classification is that we have to learn rules before we can classify. If someone, who has drunk wine all his life and has never heard or seen anything about beer, drinks the first beer, he will not be able to recognize it as beer. Only when he learns what characteristics a beer has (appearance, taste,...), he will be able to classify a drink as beer.

The next thing to emphasize is that classification is not about memorizing something, but about building an abstract model. When we drink the first beer, we remember properties (features) and build rules that have to be met in order to classify a drink as beer. This way we can later classify, what we perceived even though we had never perceived it before.

# Classification

*“Systematic arrangement in groups or categories according to established criteria”* [13]

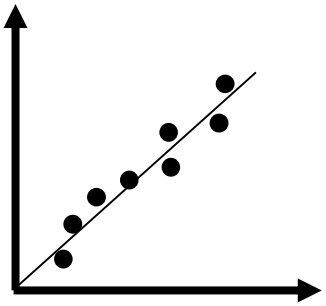
[14]

## Method Overview

# Pattern Recognition

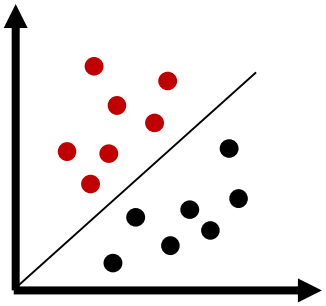
### Regression

- Predict **continuous** valued output
- Supervised



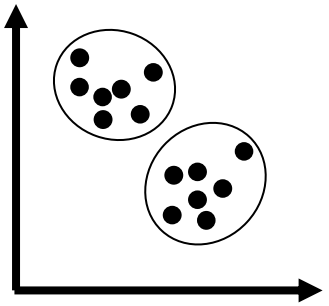
### **Classification**

- Predict **discrete** valued output
- Supervised



### Clustering

- Predict discrete valued output
- **Unsupervised**



**Additional Slide**

Patterns are distinctive structures in data. The data can have an arbitrary origin, so patterns can be searched in images, in behavior, in processes, in measurement signals etc., in principle everywhere, where data (or only metadata) is available.

The patterns can indicate a trend (regression).

The patterns can refer to a certain known type (classification).

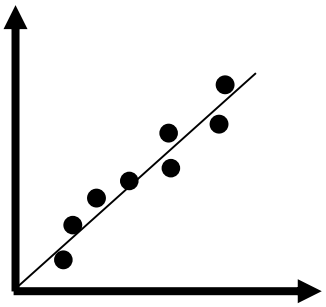
The patterns can indicate similarities between different data points (clustering).

Pattern recognition in machine learning is based on the ability to generate abstract rules for patterns from a large amount of data. With the help of these rules, it is possible to gain information about the data structure (clustering), forecasts (regression) and to better categorize new data (classification).

It is important that many data have to be available in good quality for pattern recognition to work with machine learning

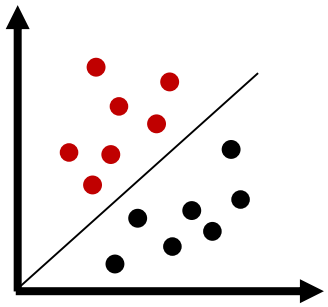
# Method Overview – Examples

## Regression



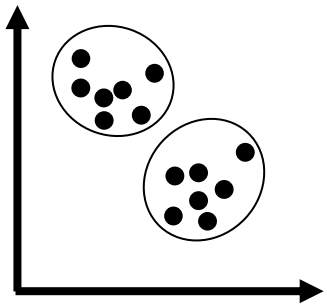
- House pricing
- Number of sales
- Persons weight

## Classification



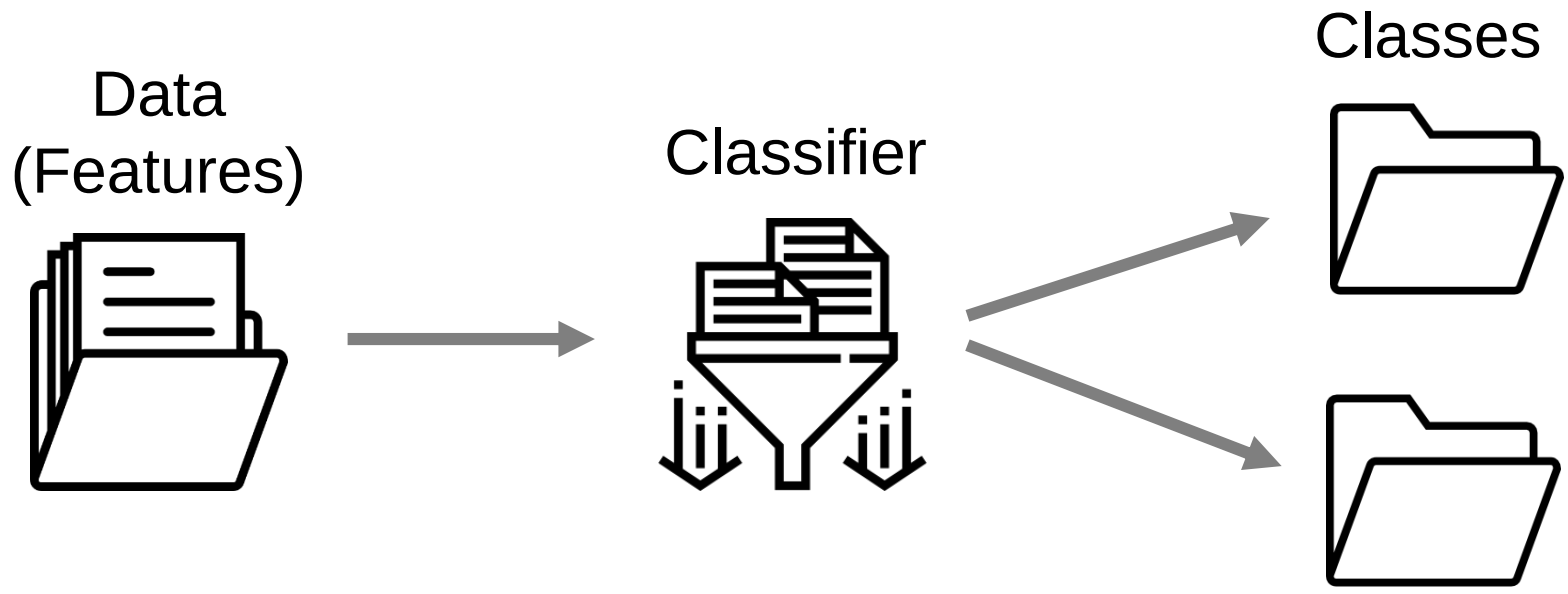
- Object detection
- Spam detection
- Cancer detection

## Clustering



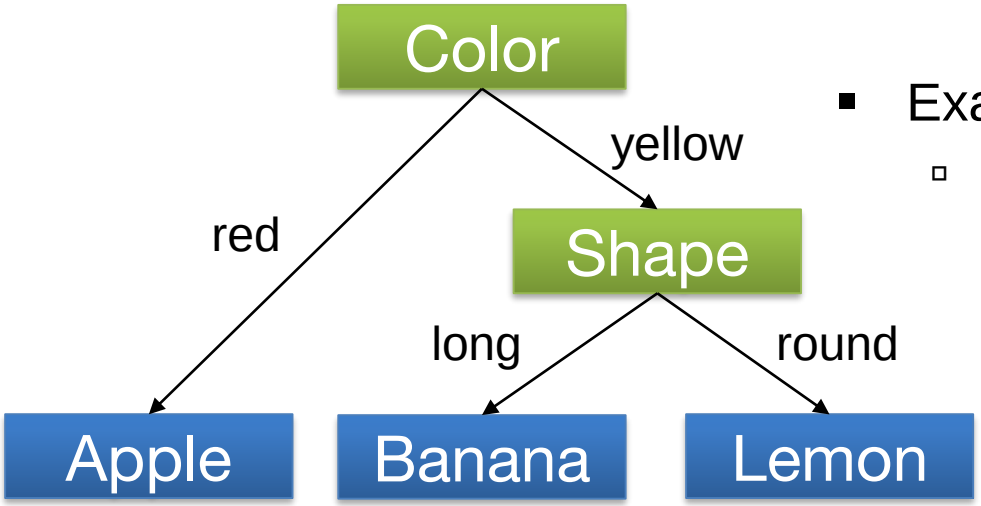
- Genome patterns
- Google news
- Pointcloud (Lidar) processing

# 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

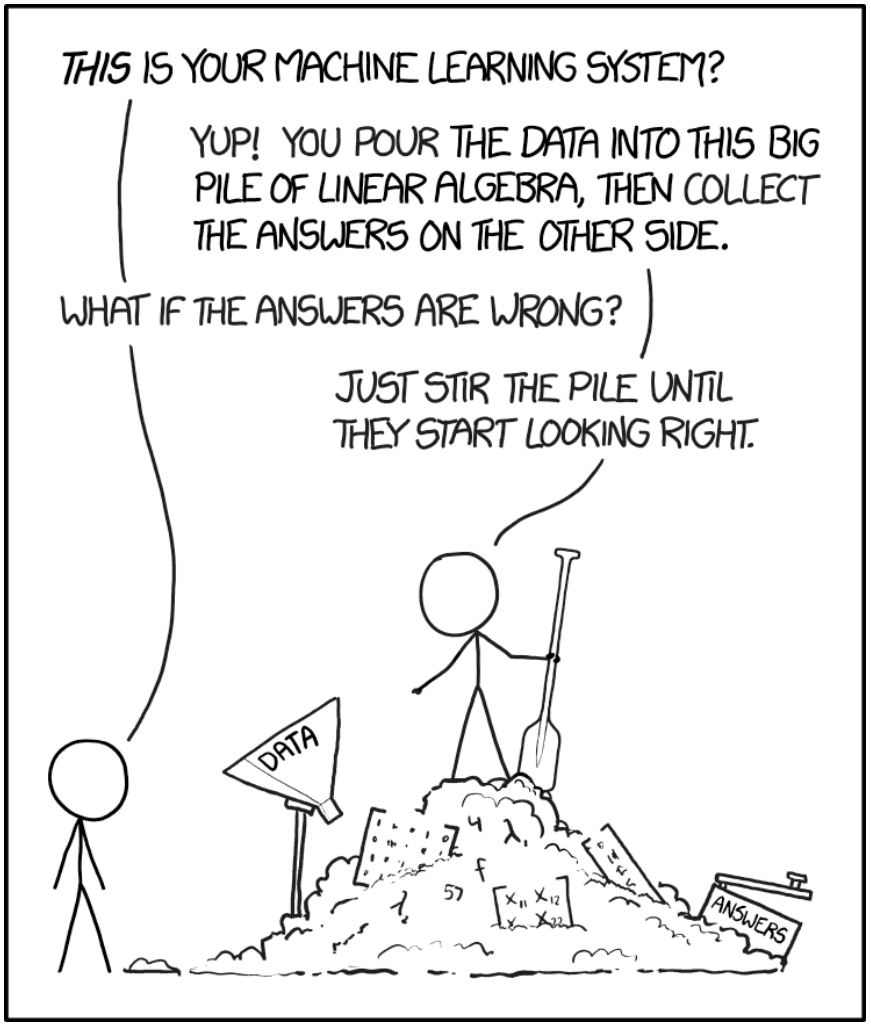



- Example: Decision tree
  - Use a-priori knowledge to formulate classification rules

- Advantages of machine learning
  - Automatic generation of a-priori knowledge
  - Automatic generation of complex classification rules

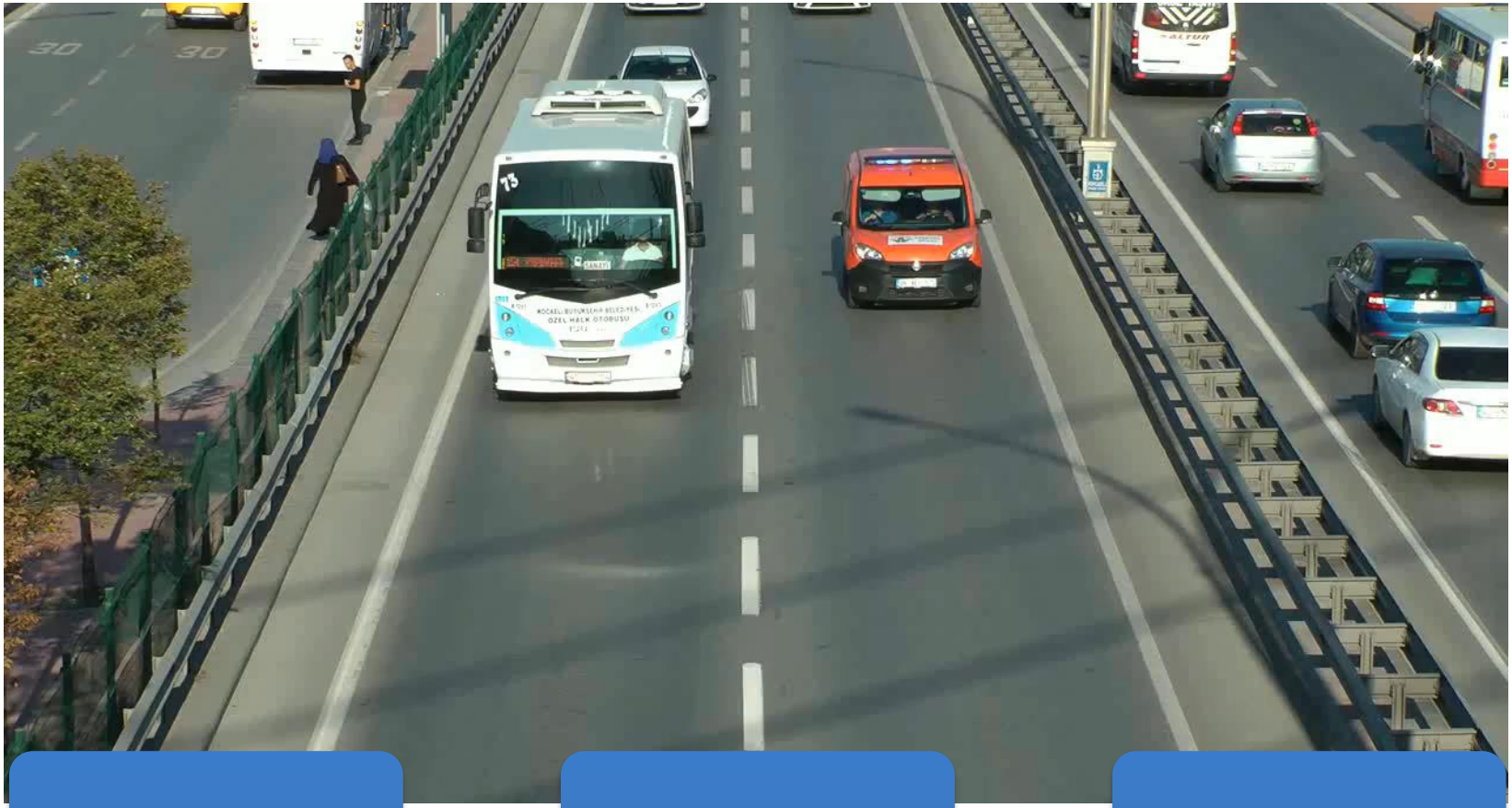
  → Suitable for extreme large datasets

# Machine Learning in Reality




[16]

# Classification – Example



[4]

Object Classification → Object Detection → Object Tracking

**Additional Slide**

In this video, different vehicle types were detected. However, it is also noticeable that no vehicles are detected on the opposite lane. Two potential reasons for this could be that

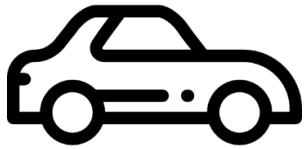
- the image area that is classified is limited to the center area of the camera images
- or only vehicle fronts and no rear of vehicles can be detected as training data only included labels for vehicle fronts.

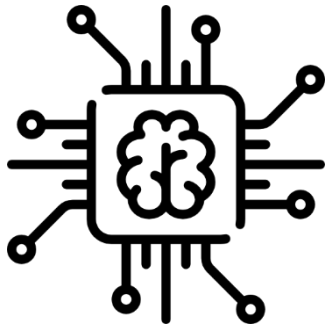
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

## Agenda





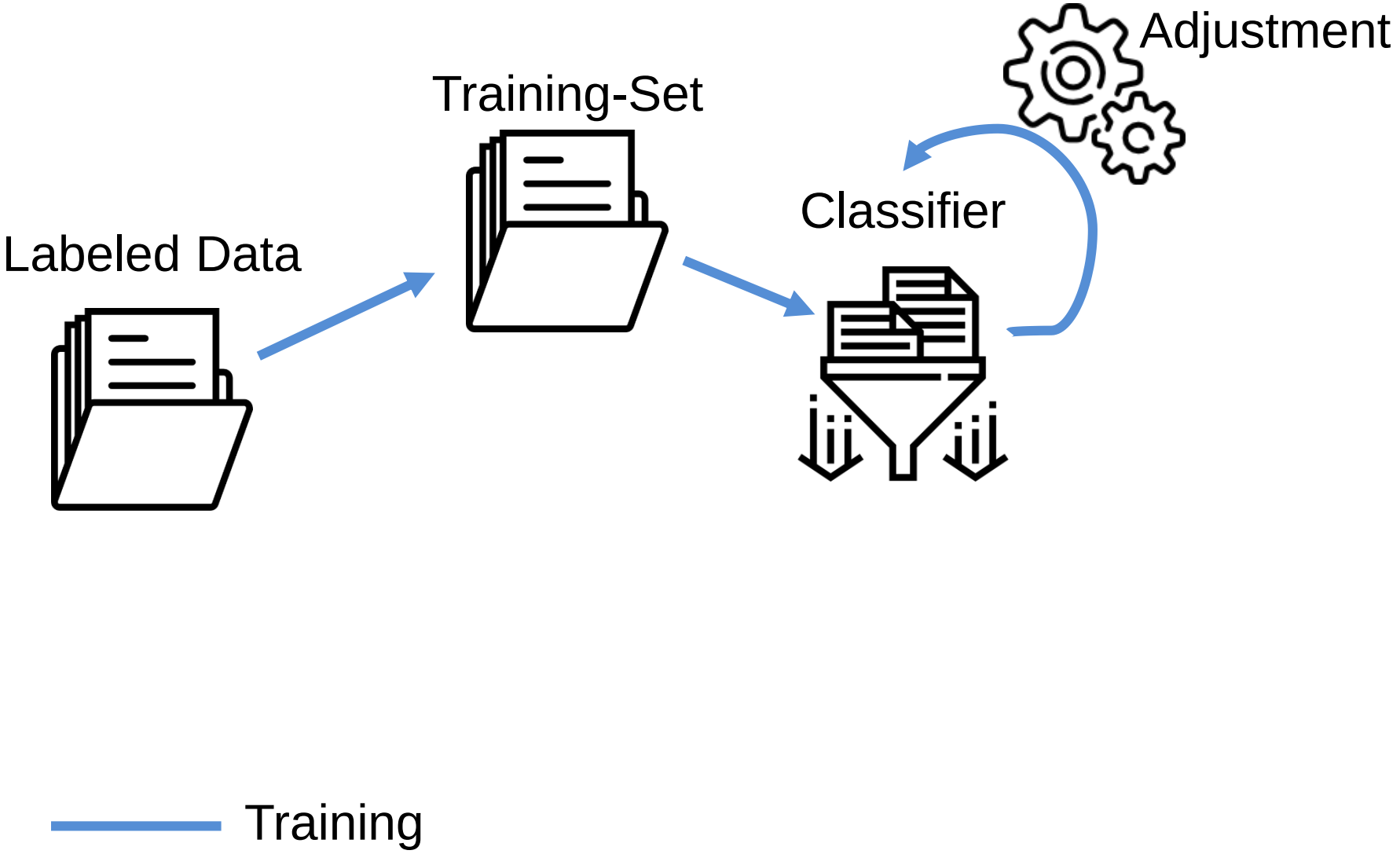


# Formal Definition - Classification

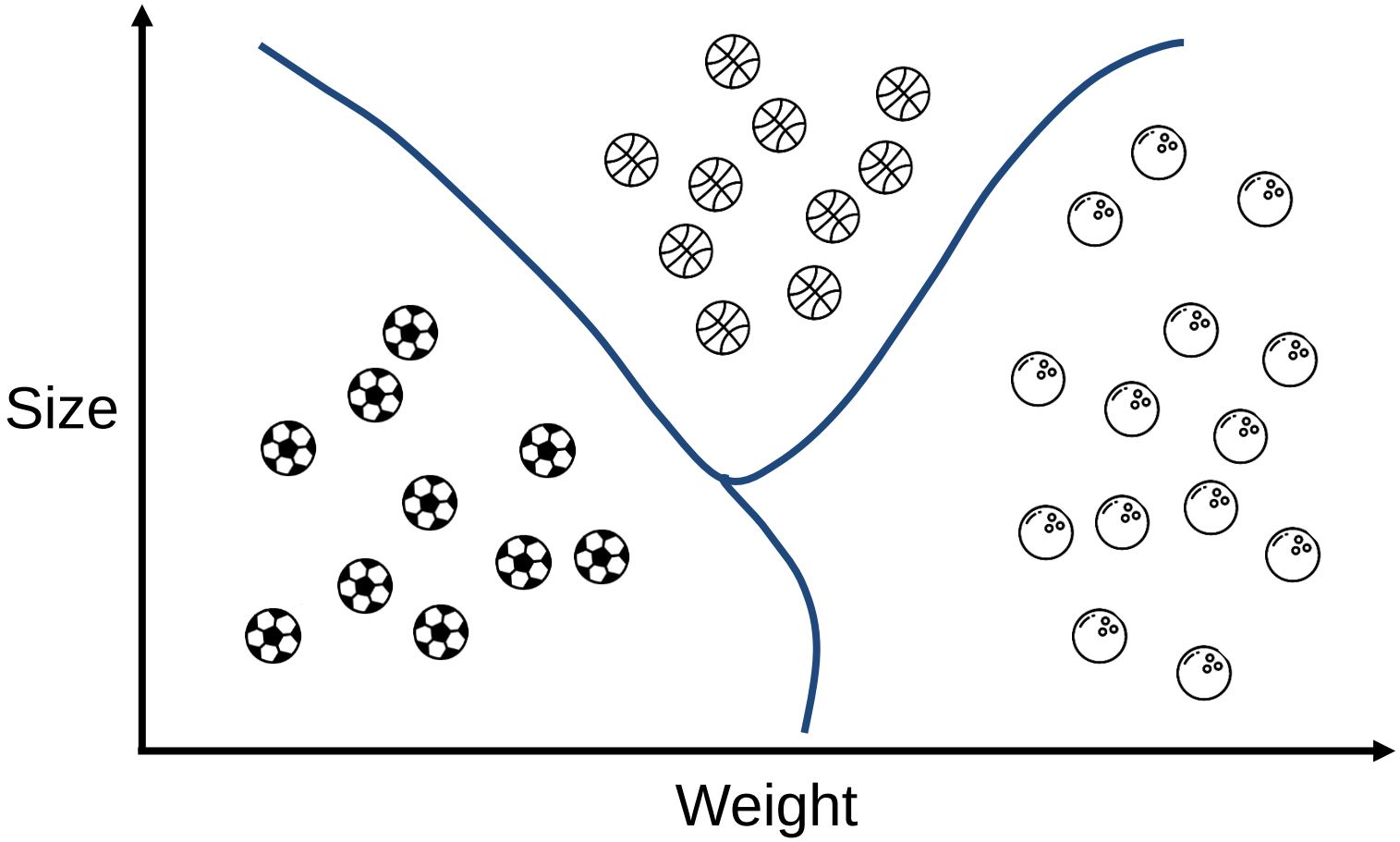
$$C_{M(\theta)}: X \rightarrow Y$$

- **Classifier:** $C$
- **Model:** $M$ with parameters $\theta$
- **Input:** Feature Space $\mathrm{X} = [x_1, x_2, \ldots x_N]^T$
- **Output:** (Predicted) Labels / Classes: $Y$
- **Dataset:** $D = \{(X_1, Y_1), (X_2, Y_2), \ldots (X_M, Y_M)\}$
- **Training:**
  - Given training data set $O \subseteq D$ with known labels
  - Optimize model parameters $\theta$ by minimizing prediction errors
- **Classification:**
  - Evaluate $C_{M(\theta)}$ at (unknown/new) element $\mathrm{X}$
  - Receive class prediction $Y$

# Supervised Learning - Classification

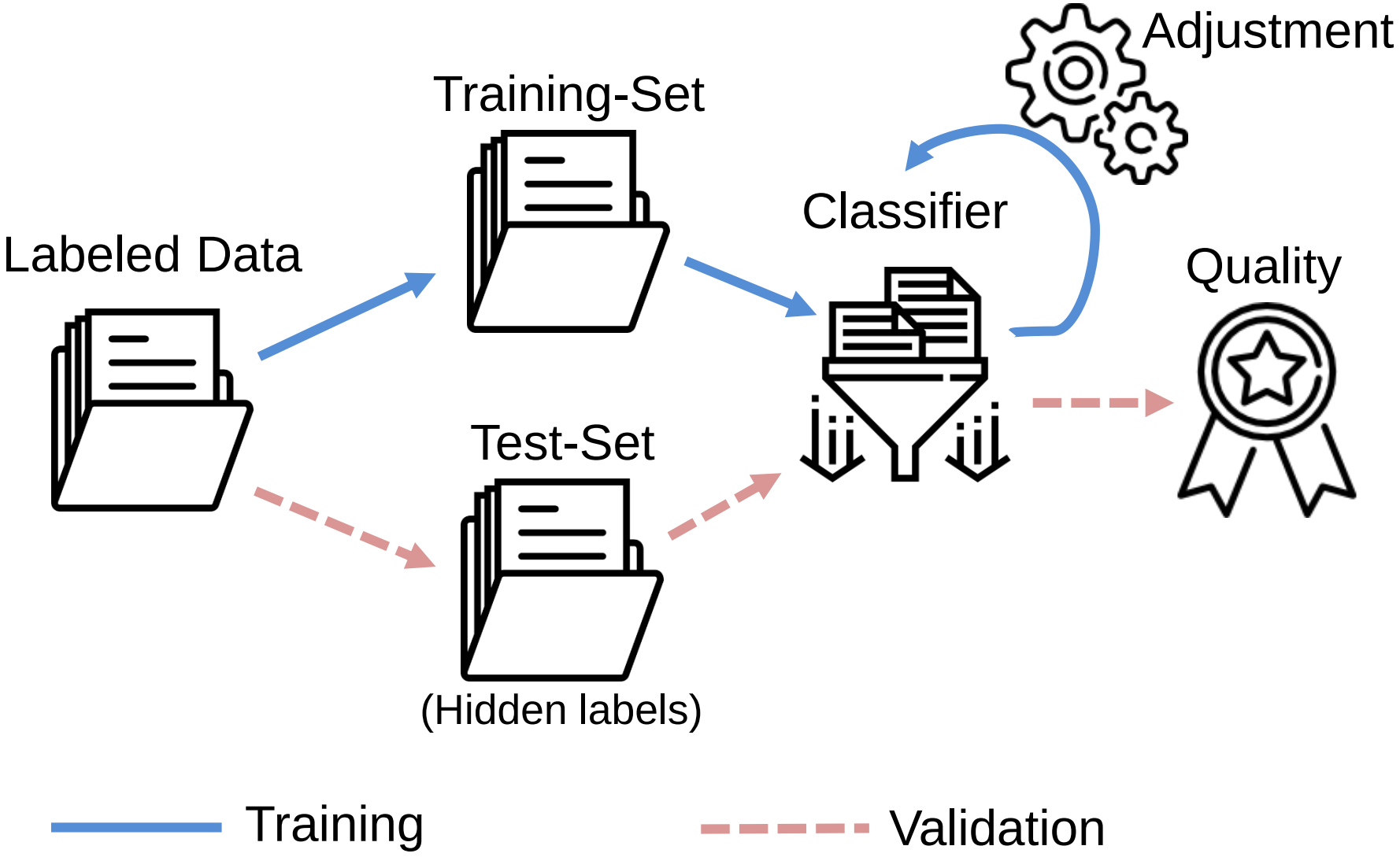


# Classifier Training Result




Features: Size, Weight  Data Points:  Decision Boundaries:

# Supervised Learning - Classification



## Additional Slide

For supervised learning, labelled data must be available. These are divided into a training set (approx. 80 %) and a test set (approx. 20 %). With the help of the training data, a machine learns, optimizing its model parameters. Afterwards, the (unseen) test data is used to check the quality of the model.

## Quality Measures for Classifiers

# Quality Measures for Classifiers

- Classification accuracy or classification error (complementary)
  - Loss Functions
- Compactness of the model
  - Decision tree size; number of decision rules
- Interpretability of the model
  - Insights and understanding of the data provided by the model
- Efficiency
  - Time to generate the model (training time)
  - Time to apply the model (prediction time)
- Scalability for large databases
  - Efficiency in disk-resident databases
- Robustness
  - Robust against noise or missing values

# Validation 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.


(k=3)

- Additional requirement: stratified folds
  - Class distributions in training and test set should represent the class distribution in whole data set (or at least in training set)

- Standard: 10-fold stratified cross validation

# Confusion Matrix

Classified as

Correct Label

| | 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 |

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

True Positives: $\mathrm{TP}$

False Positives: $\mathrm{FP}$

True Negatives: $\mathrm{TN}$

False Negatives: $\mathrm{FN}$

**Additional Slide**

There is always a Trade Off between two values of FN TN TP TN

Recall - **Sensitivität**: Fraction of test objects of class X, which have been identified correctly

Precision - **Genauigkeit**: Fraction of test objects assigned to class X, which have been identified correctly

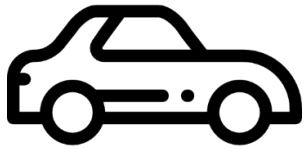
Specificity - **Spezifität**: Fraction of test objects assigned to other classes, which actually belong to other classes

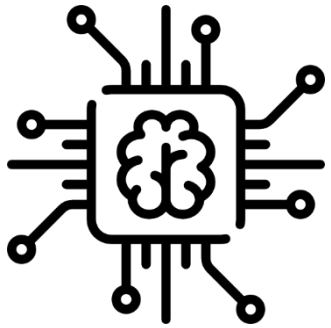
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

## Agenda

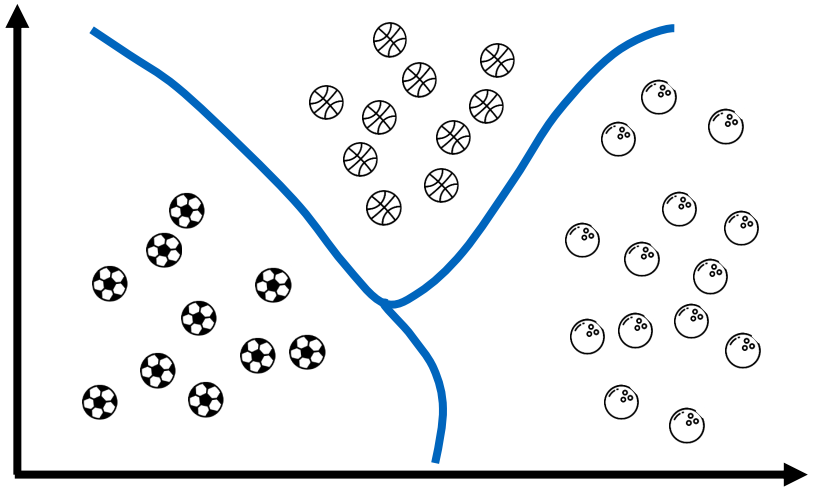
1. Chapter: Introduction
    - 1.1 Overview
    - 1.2 Training and Validation
2. **Chapter: Methods**
    - 2.1 Logistic Regression
    - 2.2 Nearest Neighbors
    - 2.3 Support Vector Machine
3. Chapter: Application
4. Chapter: Summary





# Methods

- Decision Trees
- **Logistic Regression**
- **Nearest Neighbors**
- **Support Vector Machine**
- Neural Networks
- …

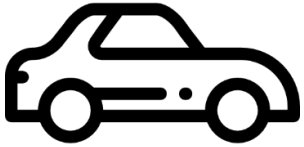


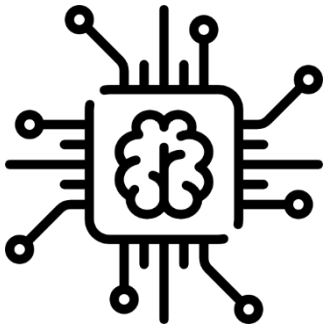
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

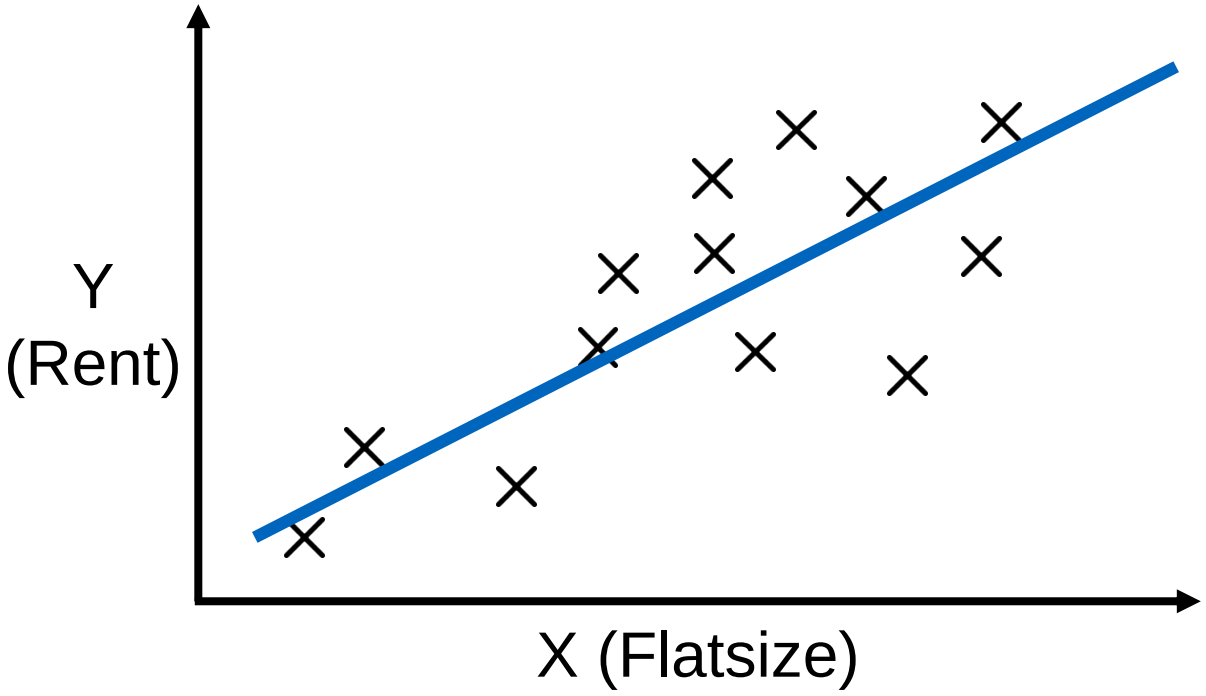
## Agenda

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



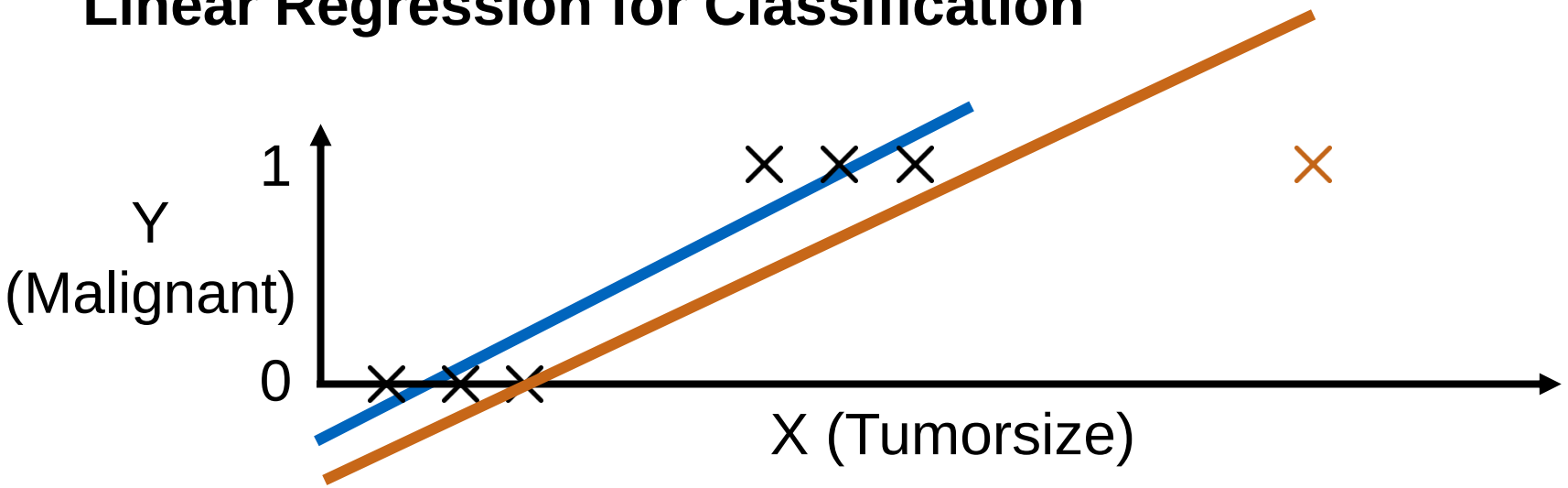


# Recap Linear Regression




$$Y = h_\theta(X), \qquad Y \in \mathbb{R}$$

## Linear Regression for Classification

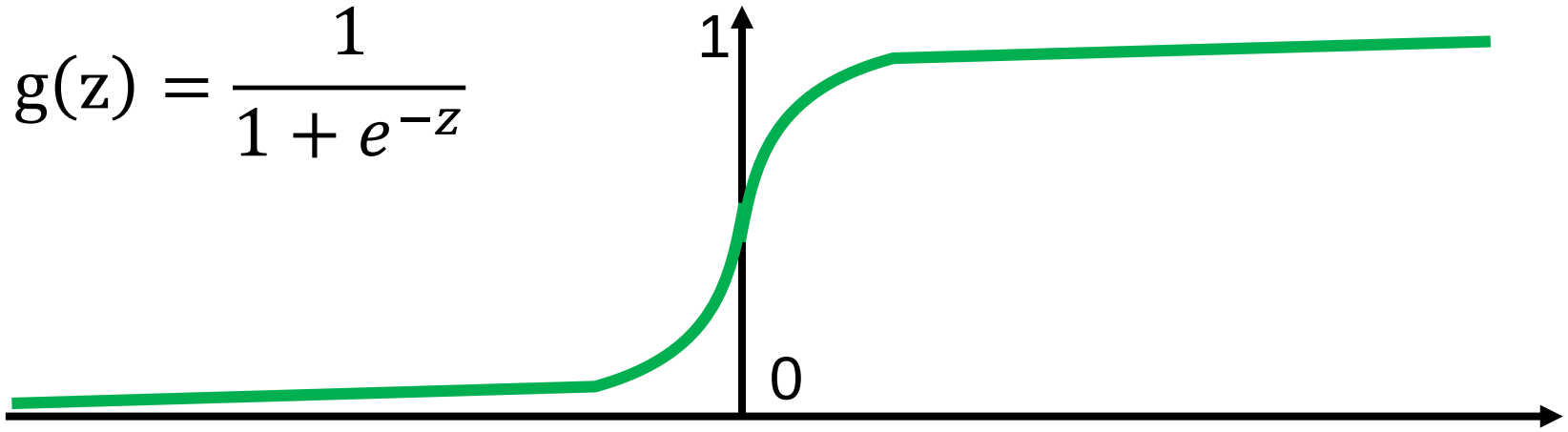



$$Y = h_\theta(X), \qquad Y \in \mathbb{R}$$

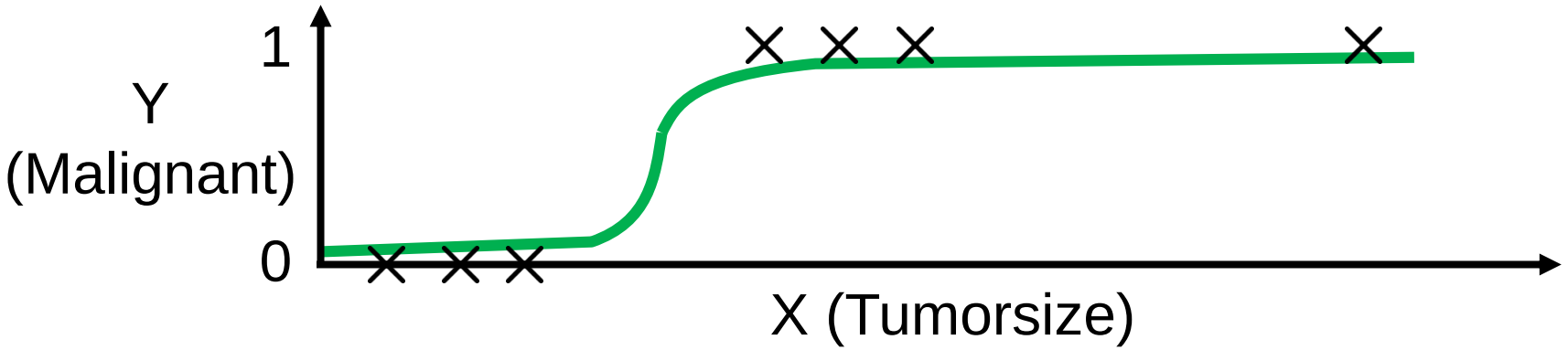
Predict malignant if $Y \geq 0.5$

## Sigmoid Function

$$g(z) = \frac{1}{1 + e^{-z}}$$



## Logistic Regression




Probabilistic classification:

$$\mathrm{Y} = g_\theta(h_\theta(X)) \qquad Y \in ]0,1[$$

Predict malignant if $Y \geq 0.5$

# 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

## Additional Slide

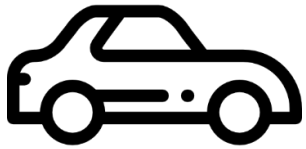
Linear regression alone is not suitable for classifying data because a continuous output must be mapped to discrete values. This blurs the class boundaries, the most relevant element for classification. By using the Sigmoid function, a S function limited between 0 and 1, the logistic regression is obtained. Despite the name, this is a classification method. The rather steep flank of the S function makes it possible to build up a clear class boundary.

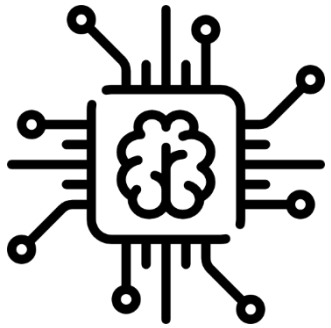
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

## Agenda

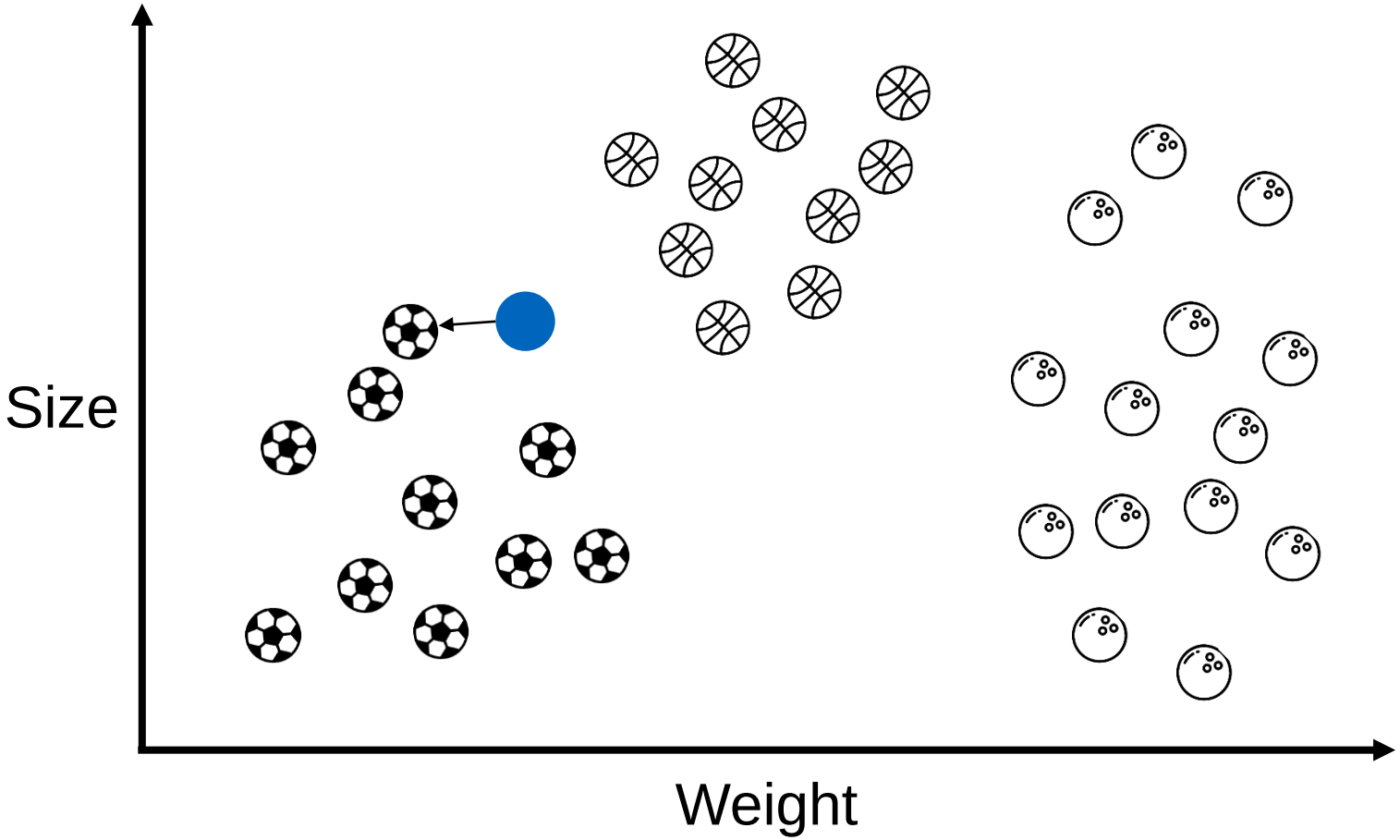
1. Chapter: Introduction
    - 1.1 Overview
    - 1.2 Training and Validation
2. Chapter: Methods
    - 2.1 Logistic Regression
    - **2.2 Nearest Neighbors**
    - 2.3 Support Vector Machine
3. Chapter: Application
4. Chapter: Summary





# Nearest Neighbors

Classify a new object based on its nearest neighbor(s)



# Nearest Neighbors

**An instance-based (memory-based) „learning“ method**

- No training and test phase → No model is generated
- Instead: Store labeled training data as points in metric space
- Process training data when new object should be classified
  → „lazy evaluation“

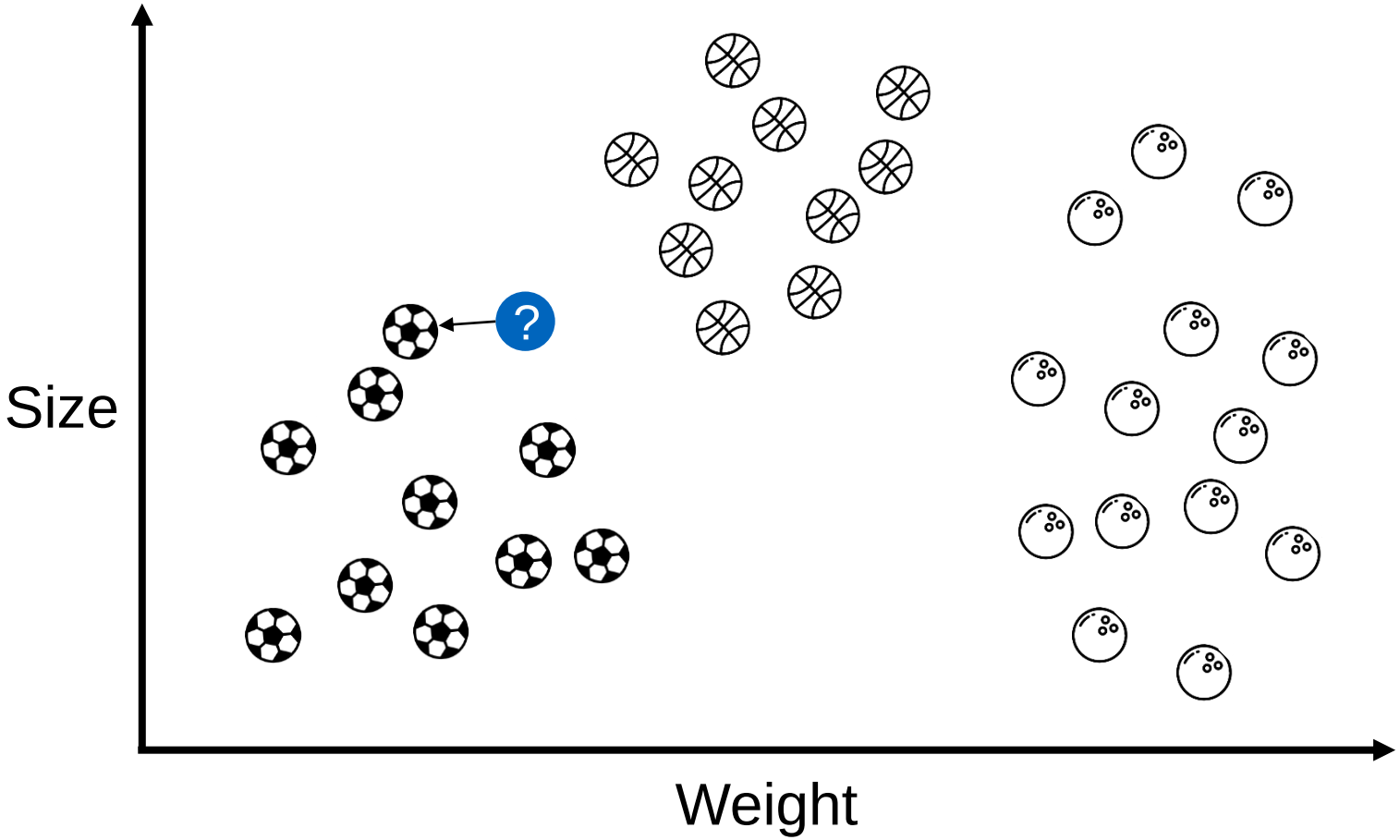
→ Easy to use, but memory and time inefficient for large data sets

# Nearest Neighbors Variants

- NN Classifier
  - Classify based on nearest neighbor only
- k-NN Classifier
  - Classify based on k nearest neighbors (k>1)
- Weighted k-NN Classifier
  - Classify based on k nearest neighbors, weighted by their distances
- Mean-based NN Classifier
  - Classify based on distance to mean positions of classes

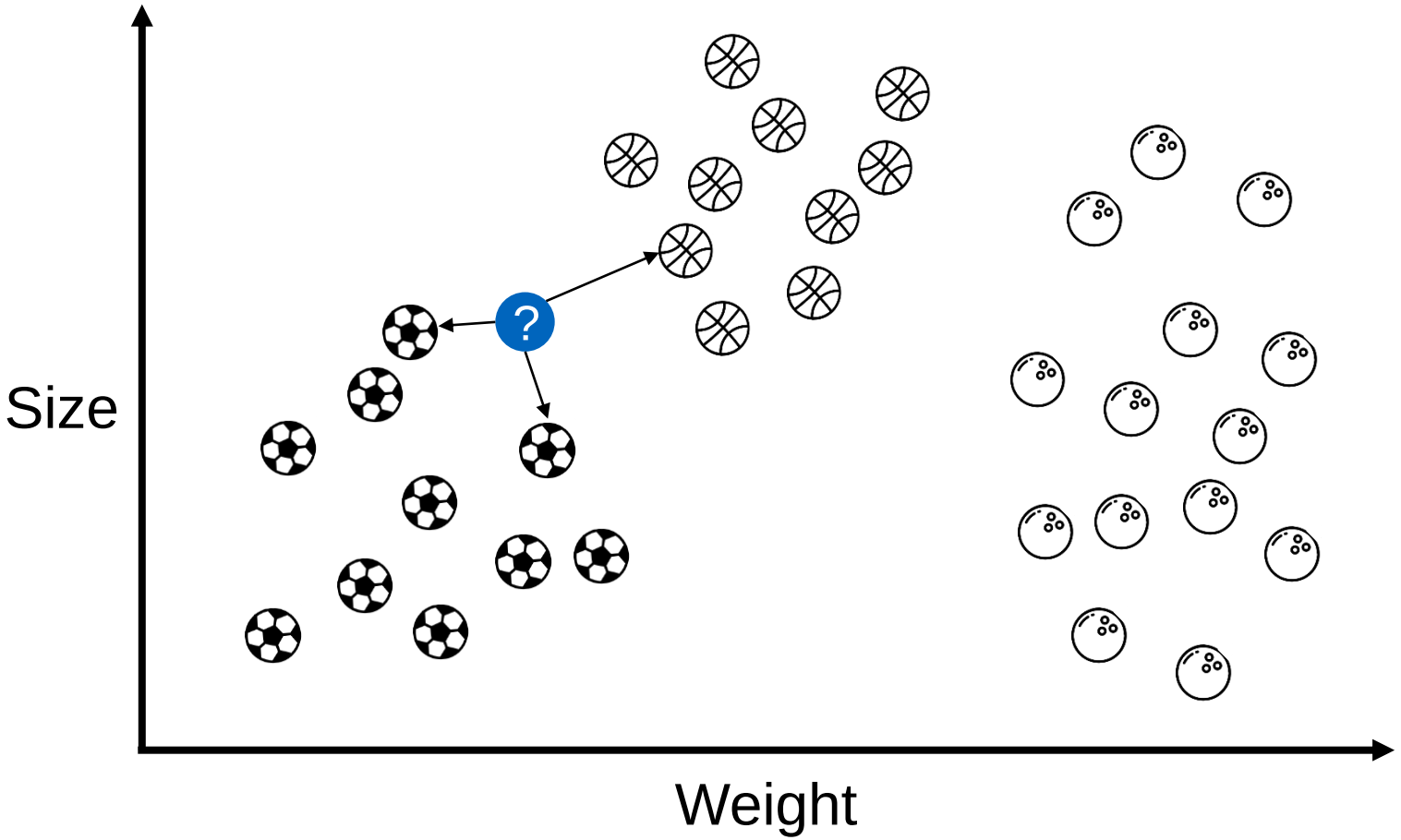
# Nearest Neighbors Variants

- NN Classifier
  - Classify based on nearest neighbor only



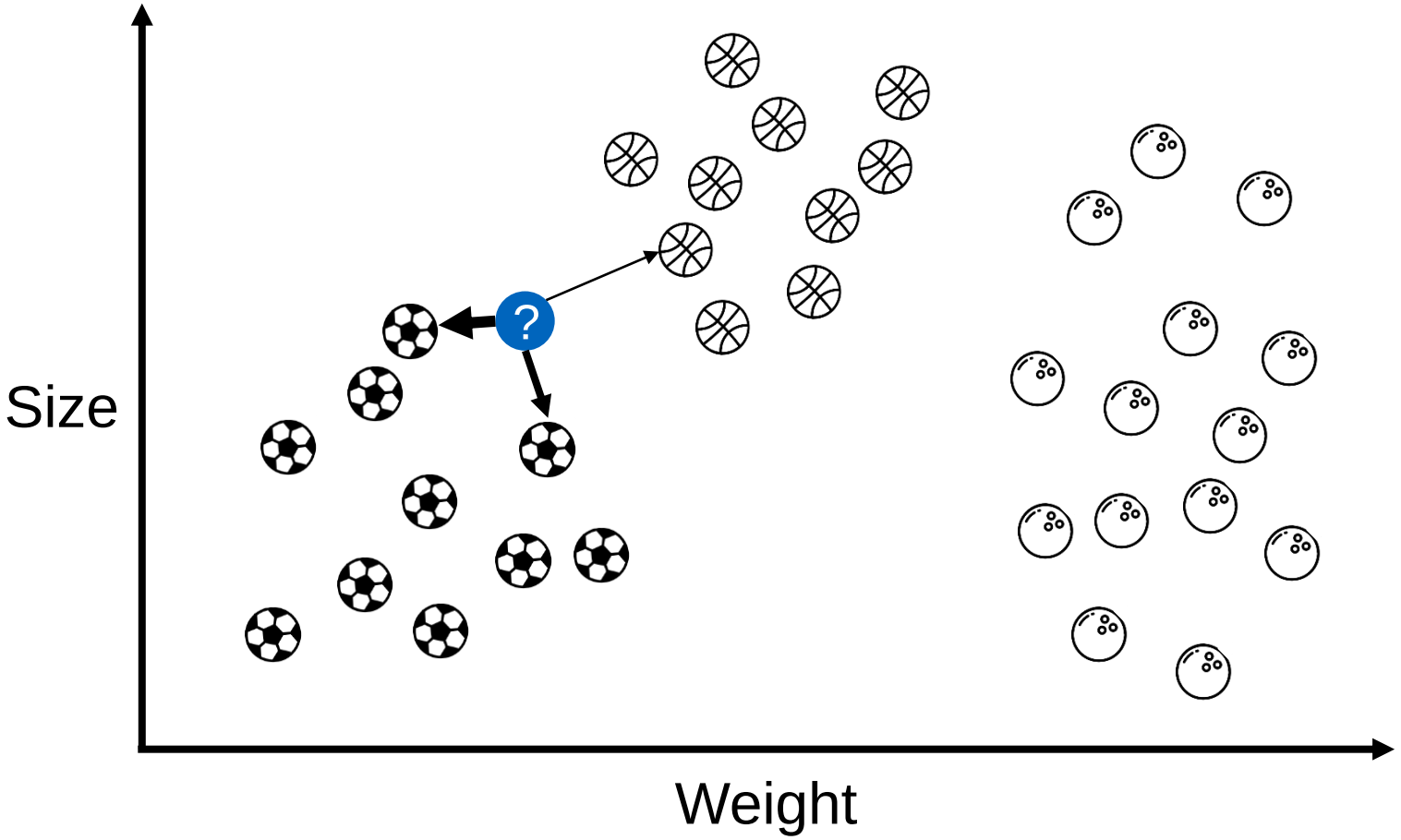
# Nearest Neighbors Variants

- k-NN Classifier
  - Classify based on k nearest neighbors (k>1)



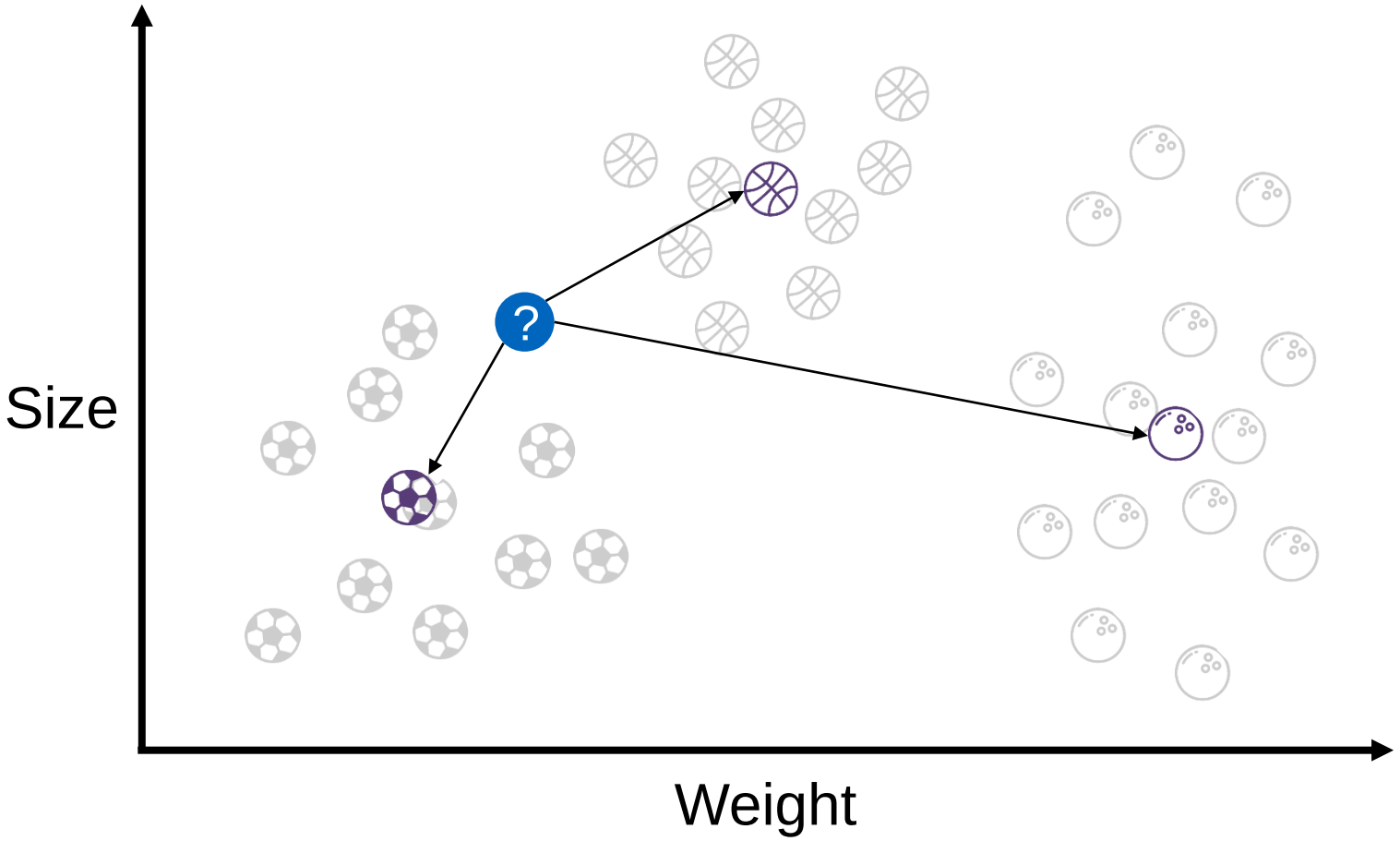
# Nearest Neighbors Variants

- Weighted k-NN Classifier
  - Classify based on k nearest neighbors, weighted by their distances



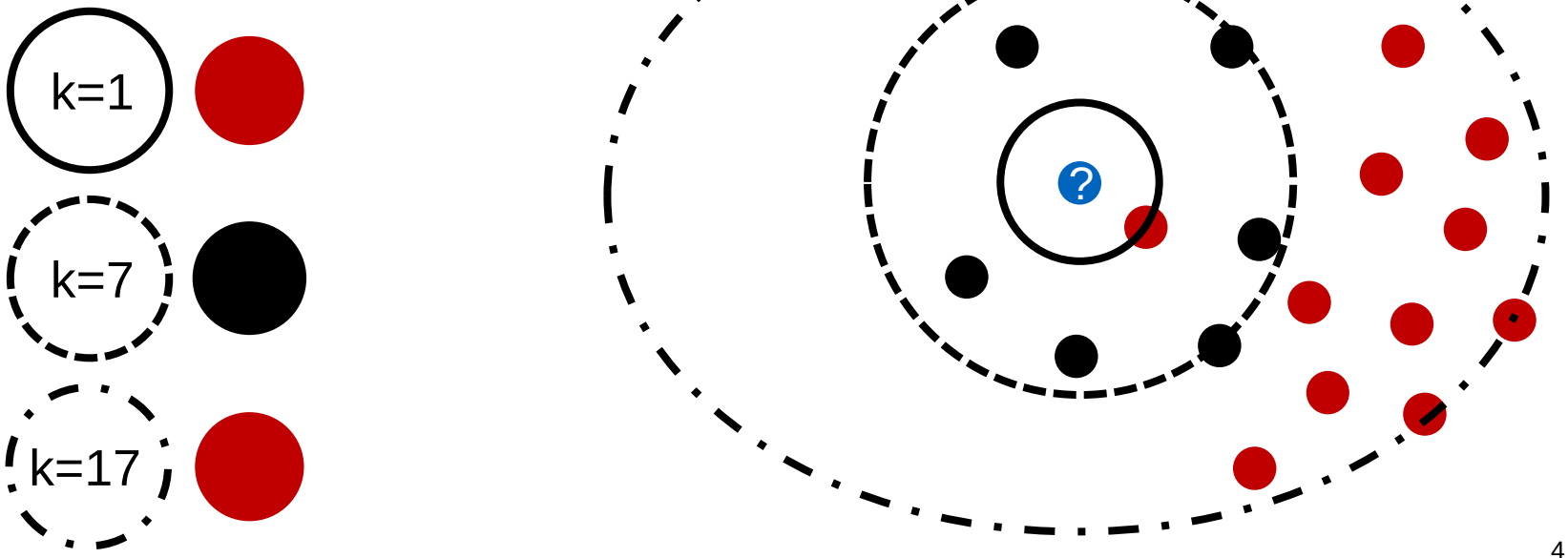
# Nearest Neighbors Variants

- Mean-based NN Classifier
  - Classify based on distance to mean positions of classes



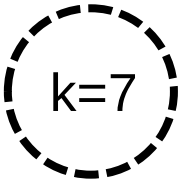
# k-NN Classifier

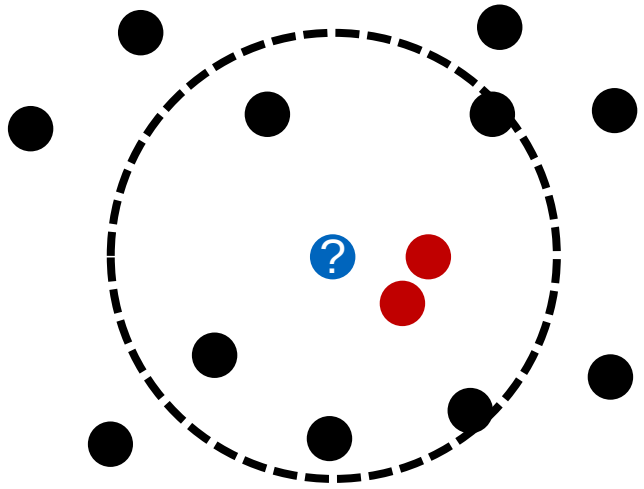
- How to choose k?
  - Generalization vs Overfitting
  - Large k: Many objects from different classes
  - Small k: Sensitivity against outliers
  - Practice: 1 << k < 10



# Weighted k-NN Classifier

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




**Classification Result**

Normal

Weighted (Distance)

Weighted (Frequency)

**Additional Slide**

Nearest Neighbors classification does not build an explicit model compared to the other classification methods, but uses all existing data to check how the elements near the new data point look like for each new classification. Beside the direct proximity to the next element there are several other variants, especially k-Nearest Neighbors, where the k data points with the smallest distance to the new data point are considered. The choice of k is relevant, so a value must be found that balances between overfitting and generalization.

It is also possible to make a model from the Nearest Neighbors method. For k = 1, the results would be a Voronoi diagram. However in most cases, the effort to construct this is greater than the effort to compute the k nearest neighbors directly.

# 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

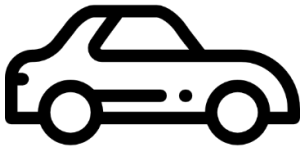
- Does not produce explicit knowledge about classes

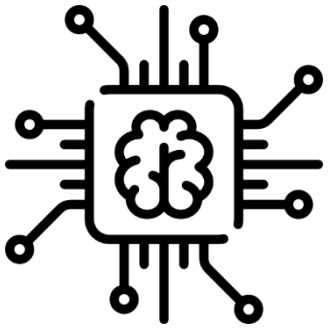
## Supervised Learning: Classification
## Prof. Dr.-Ing. Markus Lienkamp

## (Andreas Schimpe, M.Sc.)

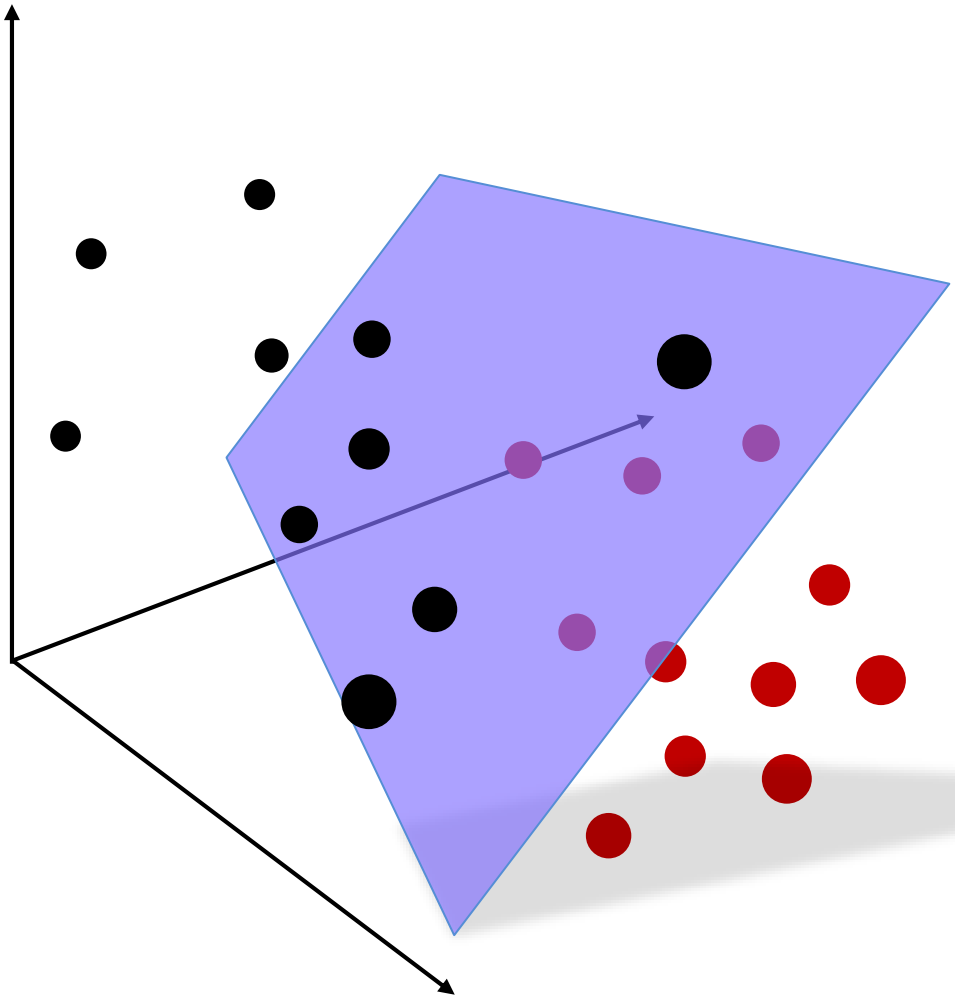
## Agenda

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



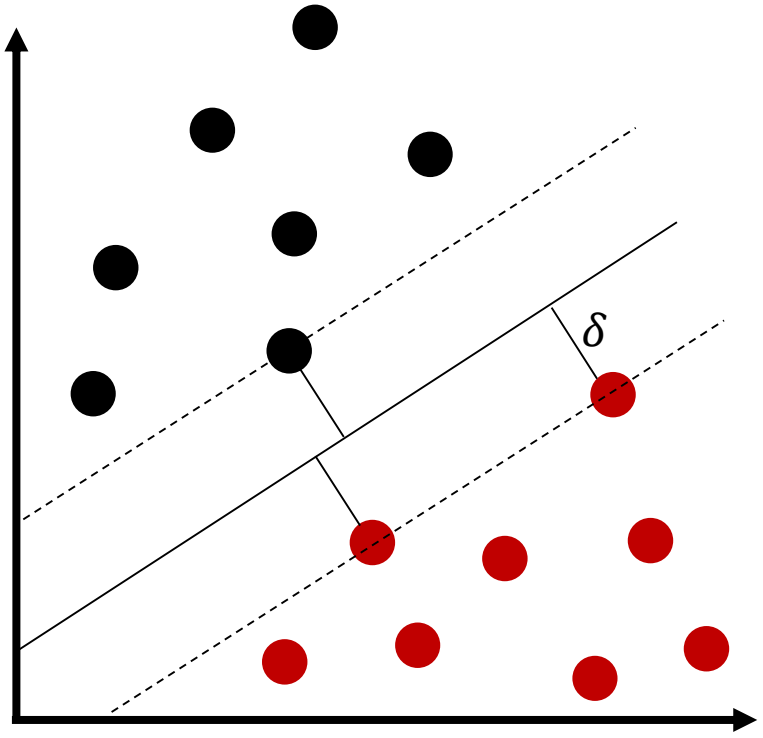


# Support Vector Machine (SVM)



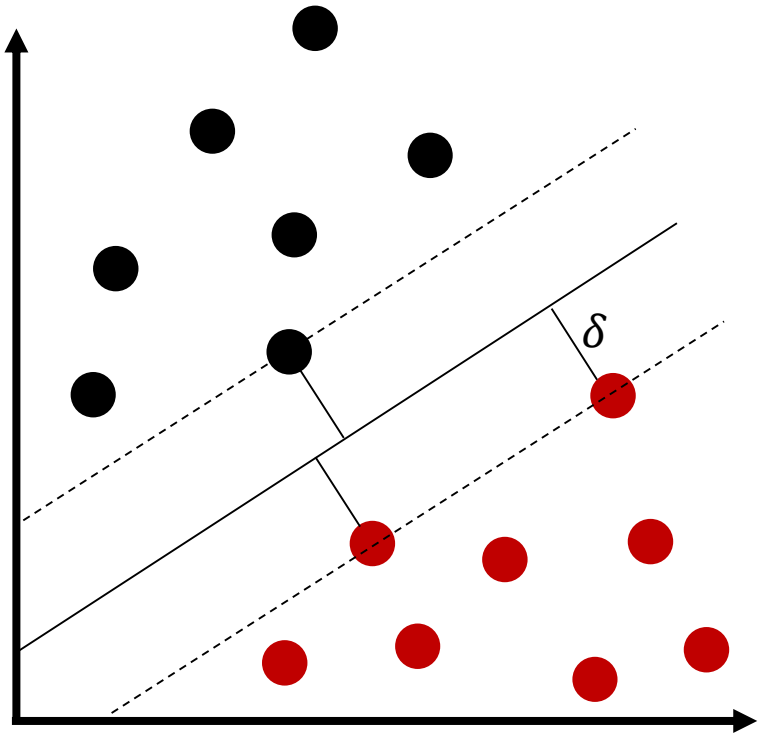
- Linear separation
  - Objects: $X \in \mathbb{R}^N$
  - Two classes: $Y \in \{-1,1\}$
  - Hyperplane separates both classes
  - Training
    - Compute hyperplane
  - Classification
    - Compute sign / distance to hyperplane

# SVM – Maximum Margin Hyperplane (MMH)




- **Objective**
  Maximize distance to Hyperplane
  - Distance to hyperplane at least $\delta$ (Margin) for all data points
- High generalization
  - Maximal stable
- Small number of support vectors
  - Only depends on objects with distance $\delta$

# SVM – Formal Definition



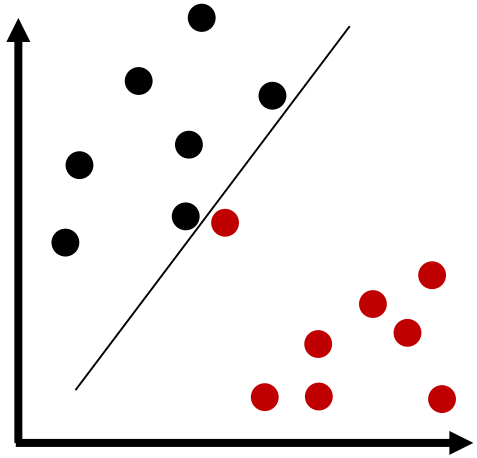

- Trainingdata: $(X_1, Y_1) \dots (X_M, Y_M)$
  with $X \in \mathbb{R}^N, Y \in \{-1,1\}$
- Hyperplane: $w^T X + b = 0$
  with $w \in \mathbb{R}^N$: normalvector,
  $\frac{b}{\|w\|}$: offset from origin
- Margin: $\delta = \frac{1}{\|w\|}$
- Training: Minimize $\| w \|$
  subject to $Y_i(w^T X_i + b) \geq 1$, for $i = 1 \dots M$
- Classification of new sample $X$:

$$Y = \begin{cases} +1, & if \quad w^T X + b \geq 0 \\ -1, & else \end{cases}$$

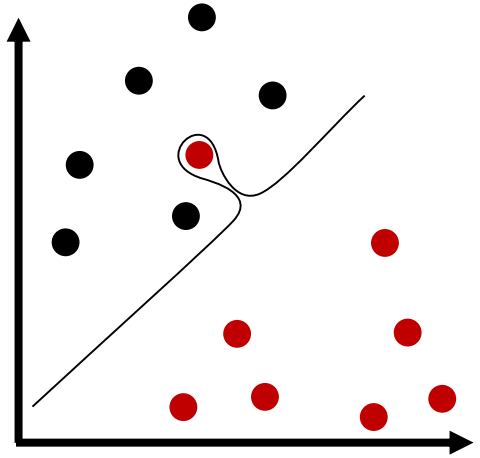
# SVM – Soft Margin

## Linear Separation

**Not always optimal**



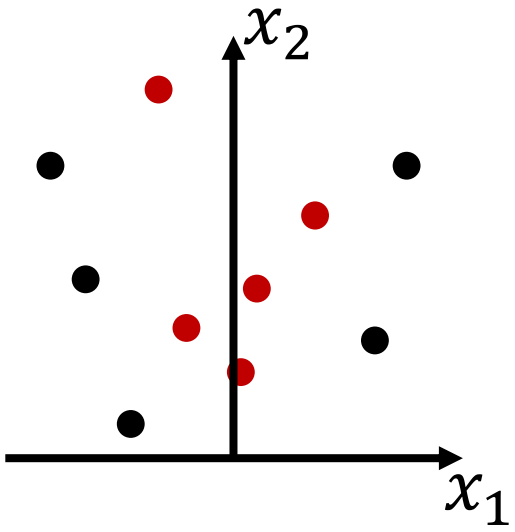
**Not always possible**



→ Tradeoff between error and margin

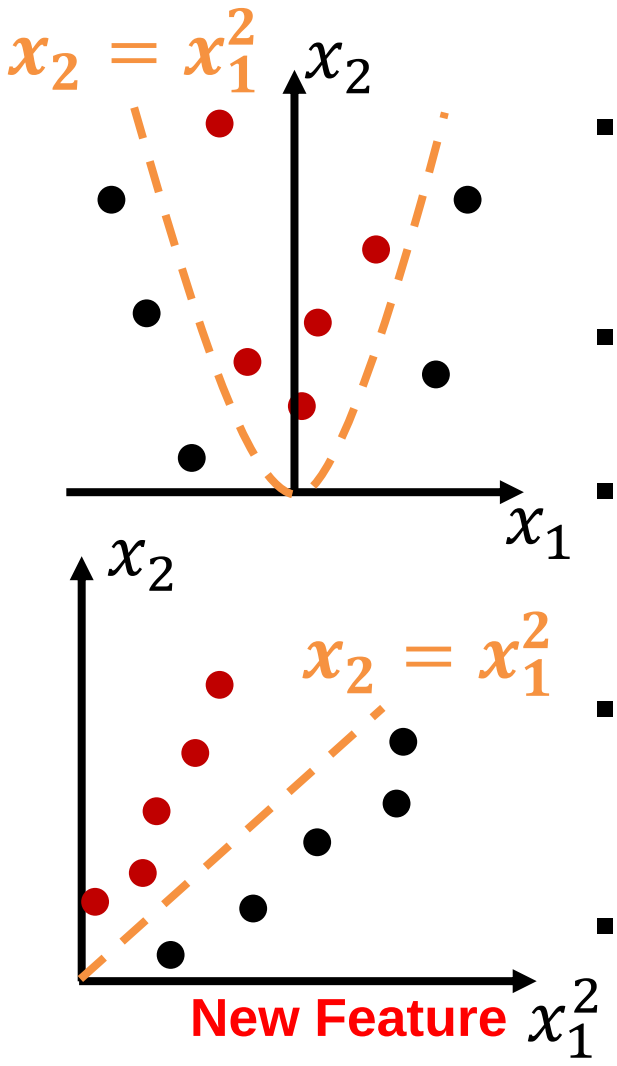
→ Allow classification error to maximize margin (soft margin)

# SVM – Space Transformation




- Non-linear data
  → Too many errors with soft margin

# SVM – Space Transformation

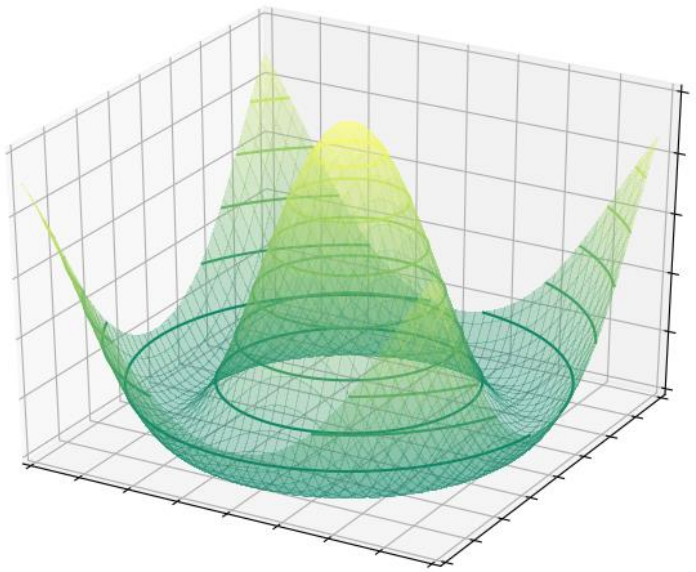



- Non-linear data
  → Too many errors with soft margin
- Transform in higher dimensional space $\Phi(X)$
- Separation with linear hyperplane
  $$w^T \Phi(X) + b = 0$$
- Inverse transform yields non-linear hyperplane in original space
- Example: Quadratic transformation
  - Hyperplane becomes polynomial of degree 2

# SVM – Space Transformation Visualisation

[5]

# SVM – Space Transformations



1. Transform Data Points
   - Lower to higher dimension
   - Computational complex

2. SVM Training → Get Hyperplane

3. Transform Hyperplane
   - Higher to lower dimension
   - Computational complex

# SVM – Kernel Machines

- SVM training and classification:
  both functions of dot product of samples

- Kernel Trick: Replace dot product with non-linear kernel function
  → **Space transformation no longer required**

- Kernel functions:
  - Polynomial: $k(X_i, X_j) = (X_i^T X_j)^d$
  - Gaussian radial bias function (RBF):
    $k(X_i, X_j) = \exp(-\gamma \parallel X_i - X_j \parallel^2)$ for $\gamma > 0$
  - Linear, sigmoid, tanh, …

# SVM – Kernel Machines

**Example: Polynomial space transformation (2nd degree)**

- Original space (example: 2D): $X = [x_1,\ x_2] \in \mathbb{R}^2$
- Transformed space: $\Phi(X) = [x_1^2,\ x_1 x_2,\ x_2 x_1,\ x_2^2] \in \mathbb{R}^4$

- Scalar product of two samples in transformed space

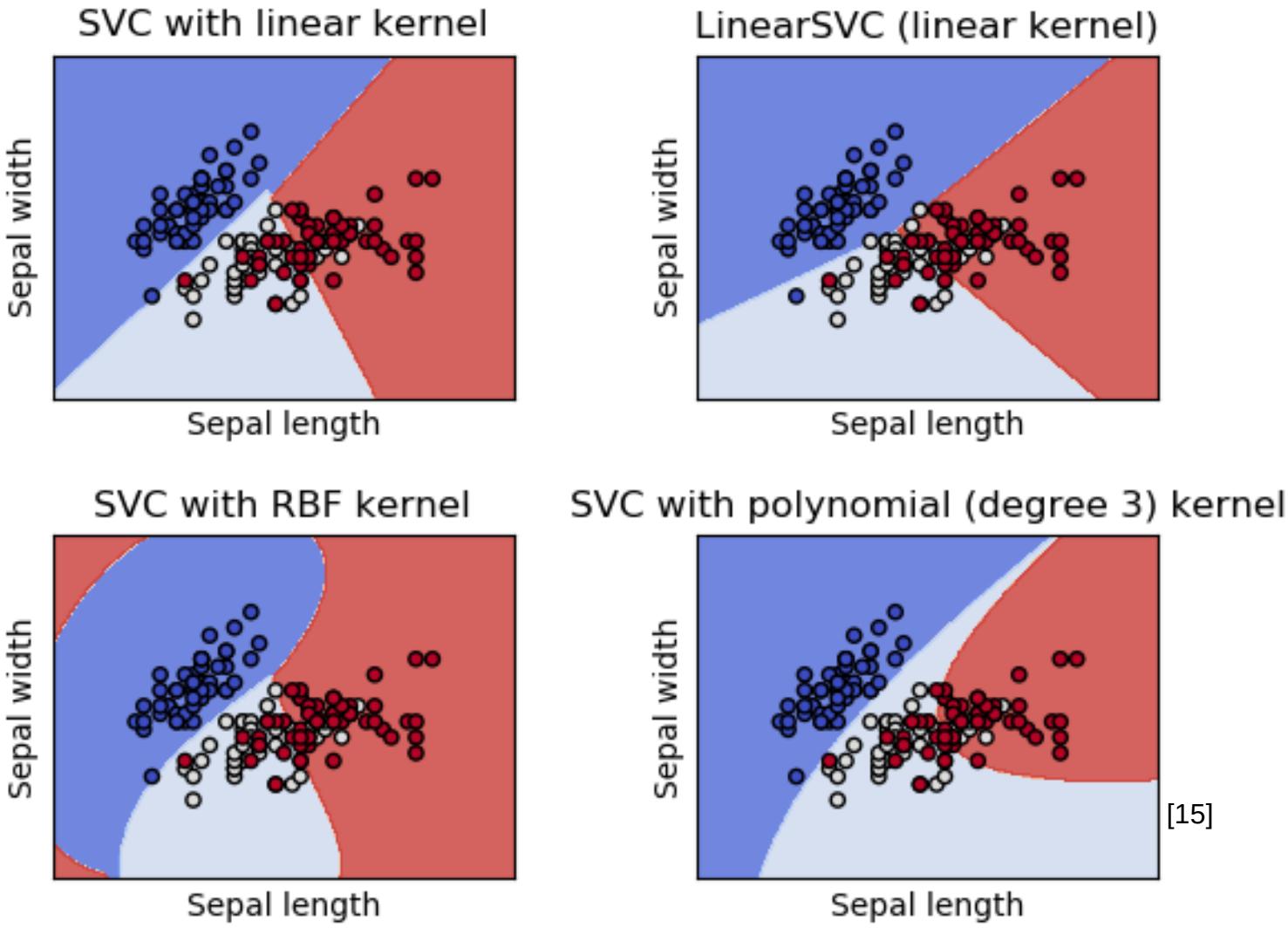
$$\Phi(X_1)^T \Phi(X_2) = x_{1,1}^2 x_{2,1}^2 + 2x_{1,1}x_{2,1}x_{1,2}x_{2,2} + x_{1,2}^2 x_{2,2}^2$$
$$= \left(x_{1,1}x_{2,1} + x_{1,2}x_{2,2}\right)^2$$

- Polynomical kernel function (example: 2nd degree)

$$k(X_1, X_2) = (X_1^T X_2)^2 = \left(x_{1,1}x_{2,1} + x_{1,2}x_{2,2}\right)^2 = \Phi(X_1)^T \Phi(X_2)$$

**→ No transform in higher dimensional space required**

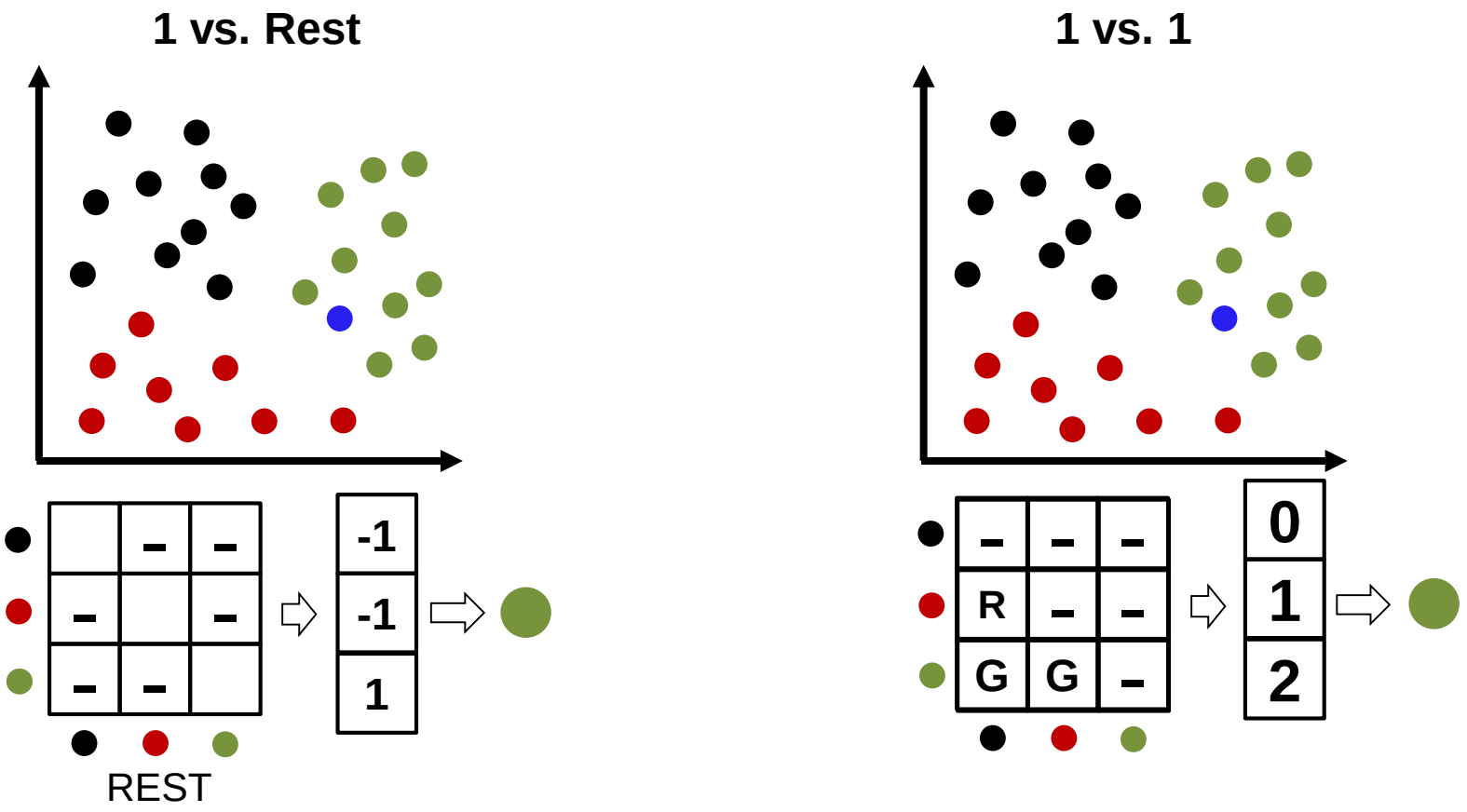
# SVM – Kernel Machines




[15]

# Multi-Class SVM

Combination of SVMs



## Additional Slide

SVM are very powerful classifiers that are still widely used today. The basic idea is to separate the classes by a clear boundary. The border should have the maximum distance to both classes, i.e. run through the middle of the lane between the classes. The points of the different classes closest to the class boundaries are the support vectors and "span" the class boundary. The naive idea is that, for example, in 2D a straight line can be drawn between the classes, and in 3D a plane separates the classes. This is often not the case, so a curved line or plane would be necessary. The approach of SVM is to transform the data points into a higher dimensional space until a linear separation by a hyperplane is possible. Since a transformation into a higher space is very computationally expensive, the kernel trick is used, which allows to calculate relations between two data points in a highly transformed space efficiently. Therewith, the whole transformation into a higher dimension can be omitted. There is a handful of established kernel functions with different properties. The right choice requires detailed knowledge about the own data and the kernel function, which is why it is recommended to try out different kernel functions.

# 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**: Applicability of different kernel functions

**Contra**

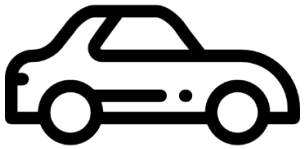
- **Efficiency**: Long training phase
- **Complexity**: High implementation effort
- **Black-Box:** Hard to interpret models

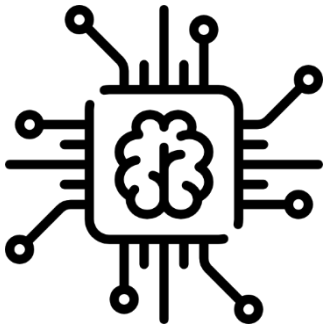
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, M.Sc.)

## Agenda

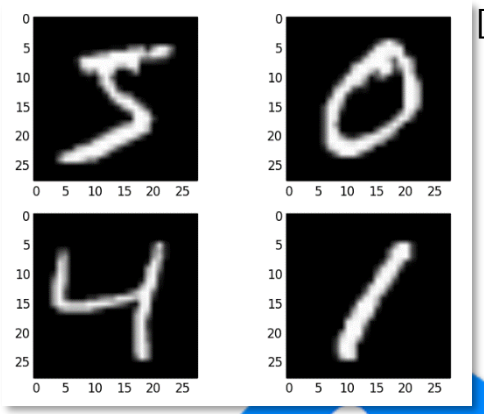
1. Chapter: Introduction
    - 1.1 Overview
    - 1.2 Training and Validation
2. Chapter: Methods
    - 2.1 Logistic Regression
    - 2.2 Nearest Neighbors
    - 2.3 Support Vector Machine
3. **Chapter: Application**
4. Chapter: Summary





# Classification Applications

- 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



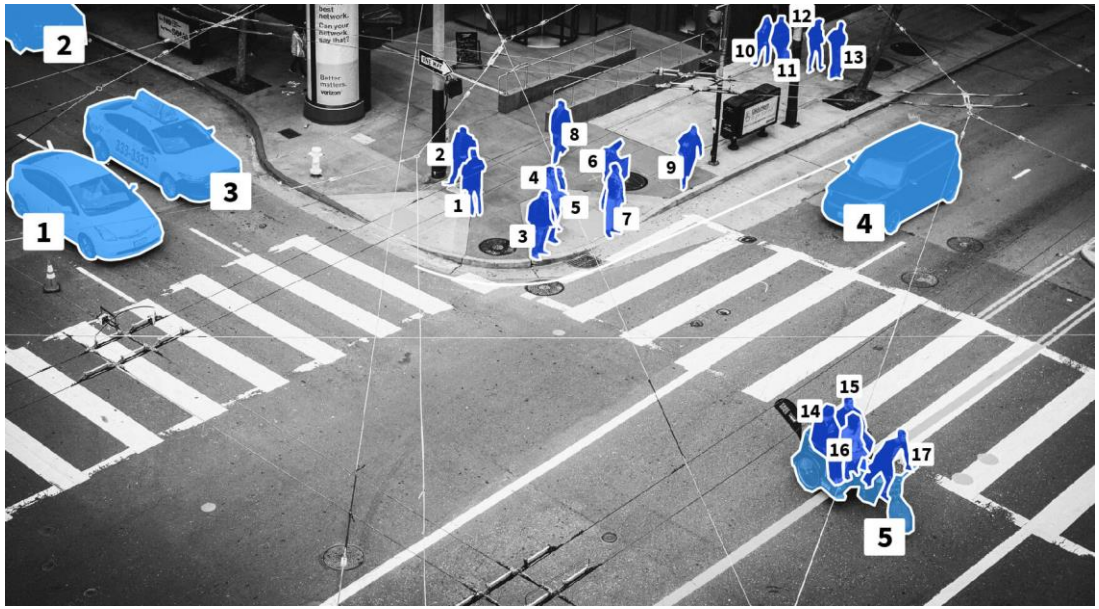
[6]

[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
- Train classification model based on features
- Take one video frame and classify features of sub-images
- Merge classified areas and create bounding box

# Training Data

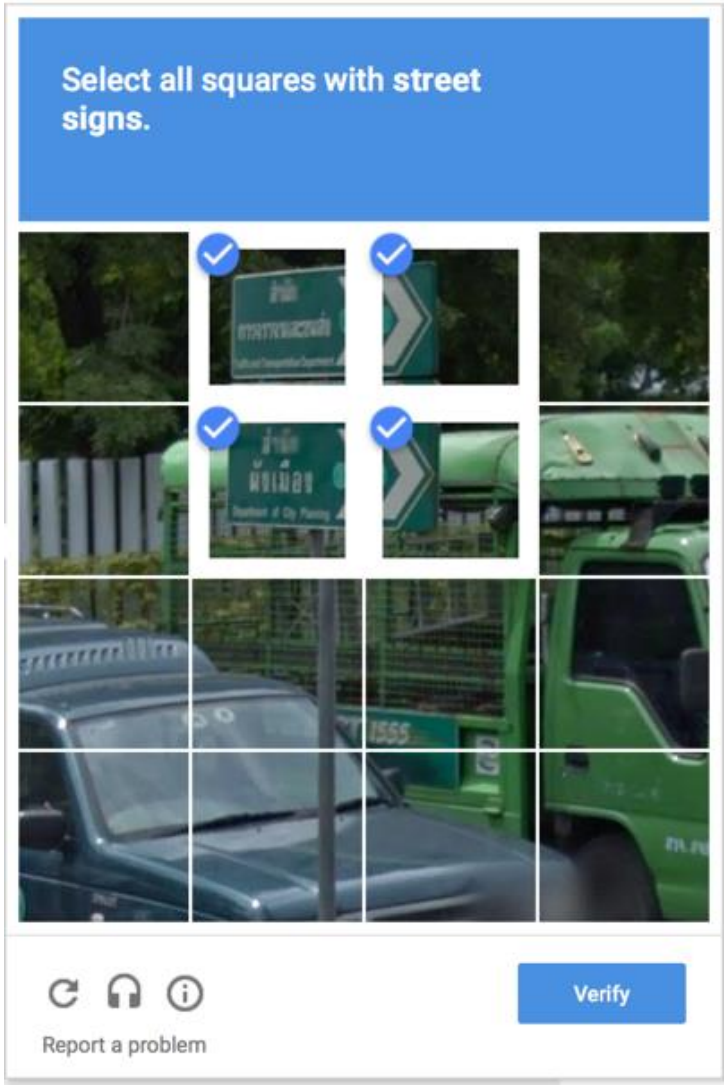
- Required label: „car“ or „no car“
- Required images:
  - Same format used for classification
  - Representative for what is expected to be found in videostream
  - 8000 images (90 % training and 10 % test)

[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:*

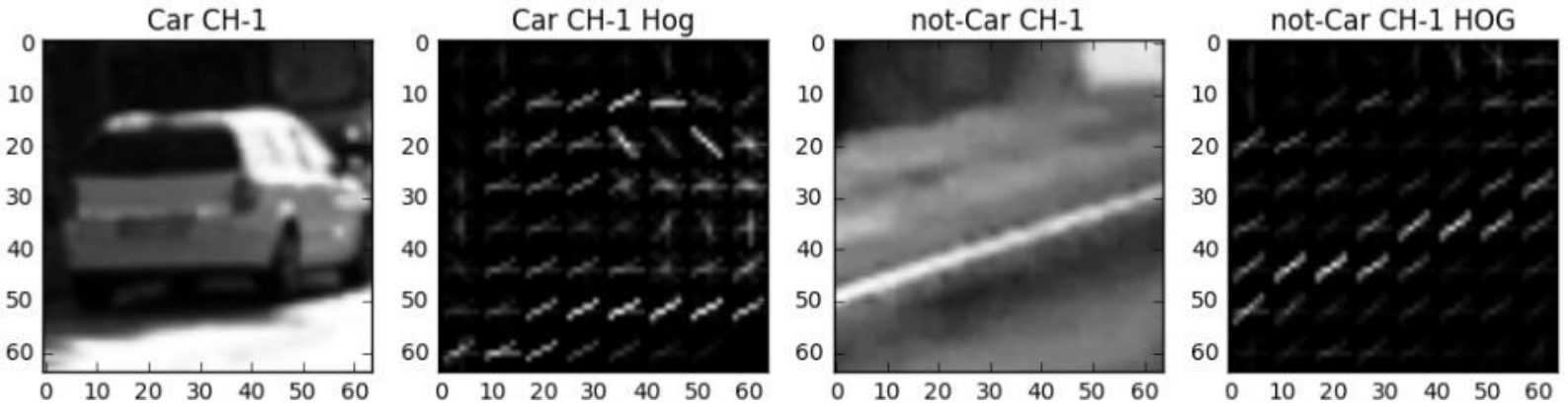
*On the one hand, computers are driving cars.*

*On the other, the state of the art test to check that you are not a computer is whether you can successfully identify stop signs in pictures.“*

- *Anonym*

# Feature Extraction

- Histogram of Oriented Gradients (HOG)
  - compressed & encoded version of the image




[10]

# Build SVM Classifier

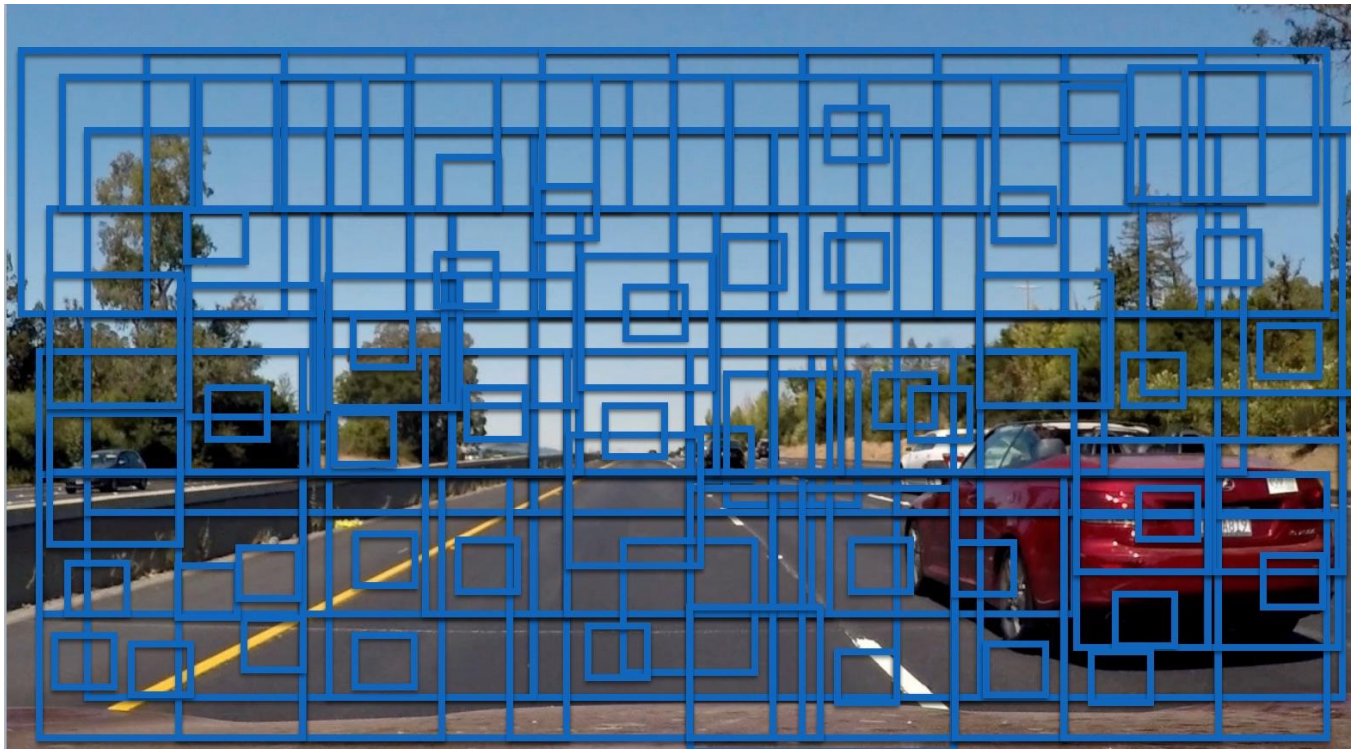

- Machine learning libraries (python)
  - scikit-learn (http://scikit-learn.org/)

```
>>> 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

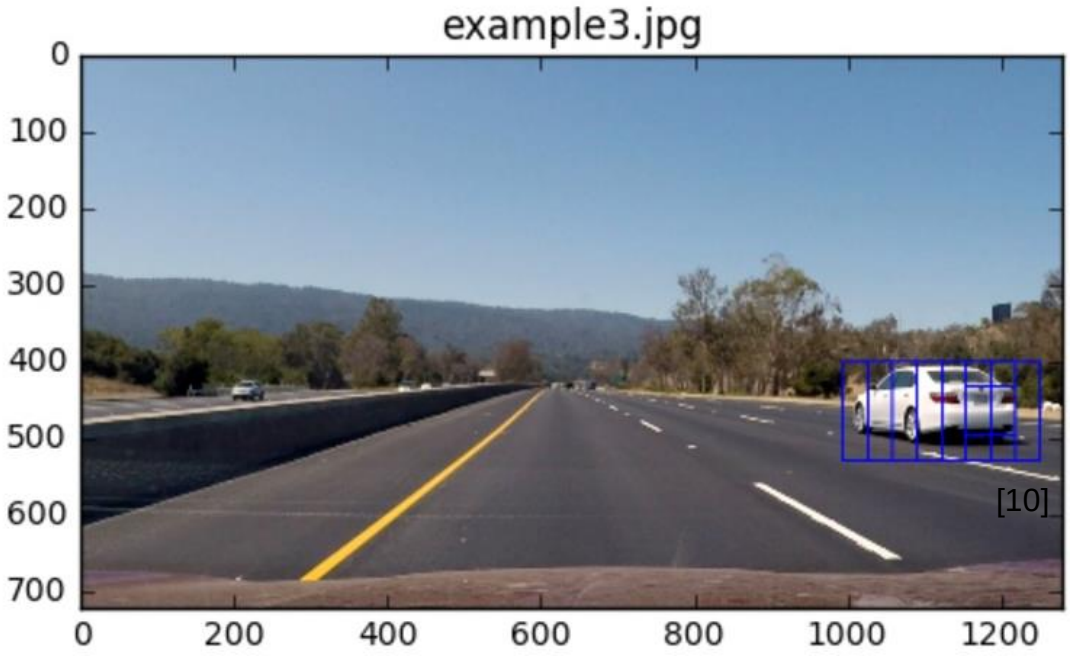
- Produce sub-images of each frame for the classification

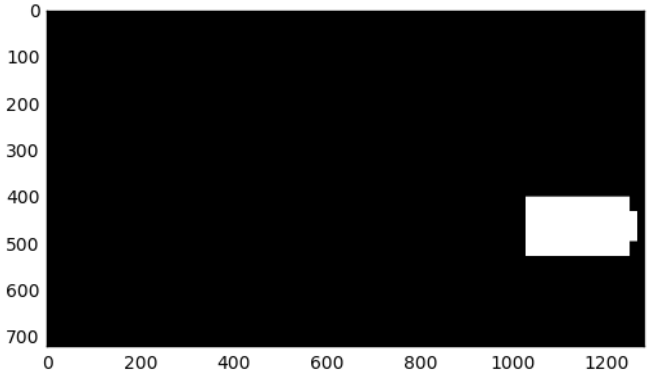


[10]

# Merge classified areas

- Merge classes of sub-images



# Final Output

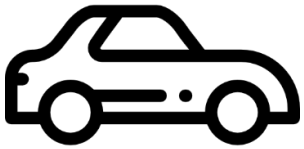
[10]

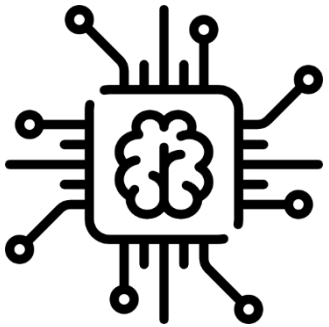
Supervised Learning: Classification
Prof. Dr.-Ing. Markus Lienkamp

(Andreas Schimpe, 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**





# Summary

What did we learn today:

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

# Summary

What did we learn today:

- **SVMs** are linear classifiers using a maximum margin hyperplane.
- With the **kernel trick**, SVMs can be used for non-linear classification.
- Classification is important for the **perception**, e.g., of 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**.
- For image classification, an approach is to **partition, classify (based on features) and then merge** the sub-images.

# 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://scikit-learn.org/stable/auto_examples/svm/plot_iris.html
- [16] https://xkcd.com/1838/

# 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