Christian-Albrechts-Universität zu Kiel

Lecture
”Intelligent Systems“

Chapter 8: Classification of Time-Series

Prof. Dr.-Ing. habil. Sven Tomforde / Intelligent Systems
Winter term 2020/2021

Contents

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings

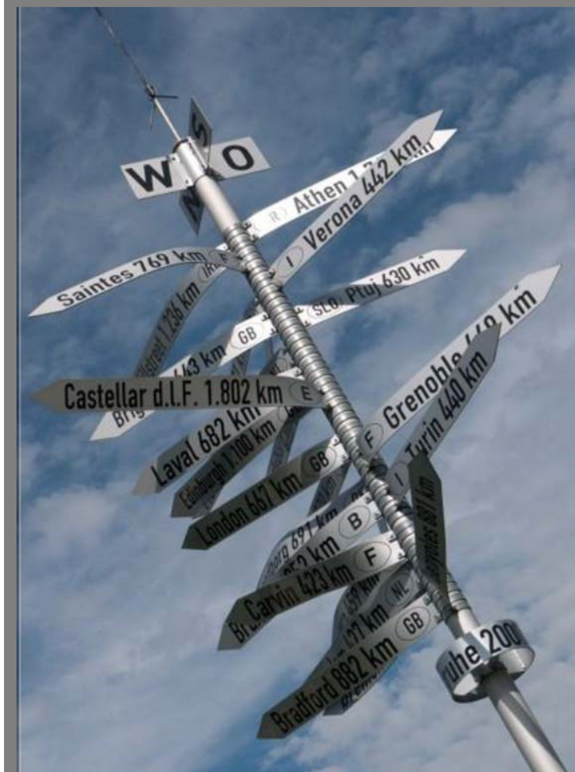
Goals

Students should be able to:

- define what classification of time-series is and why it is necessary.
- explain the differences between the different algorithms and their applicability.
- decide which technique to apply in a certain scenario.

Agenda

- **Introduction to classification**
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings

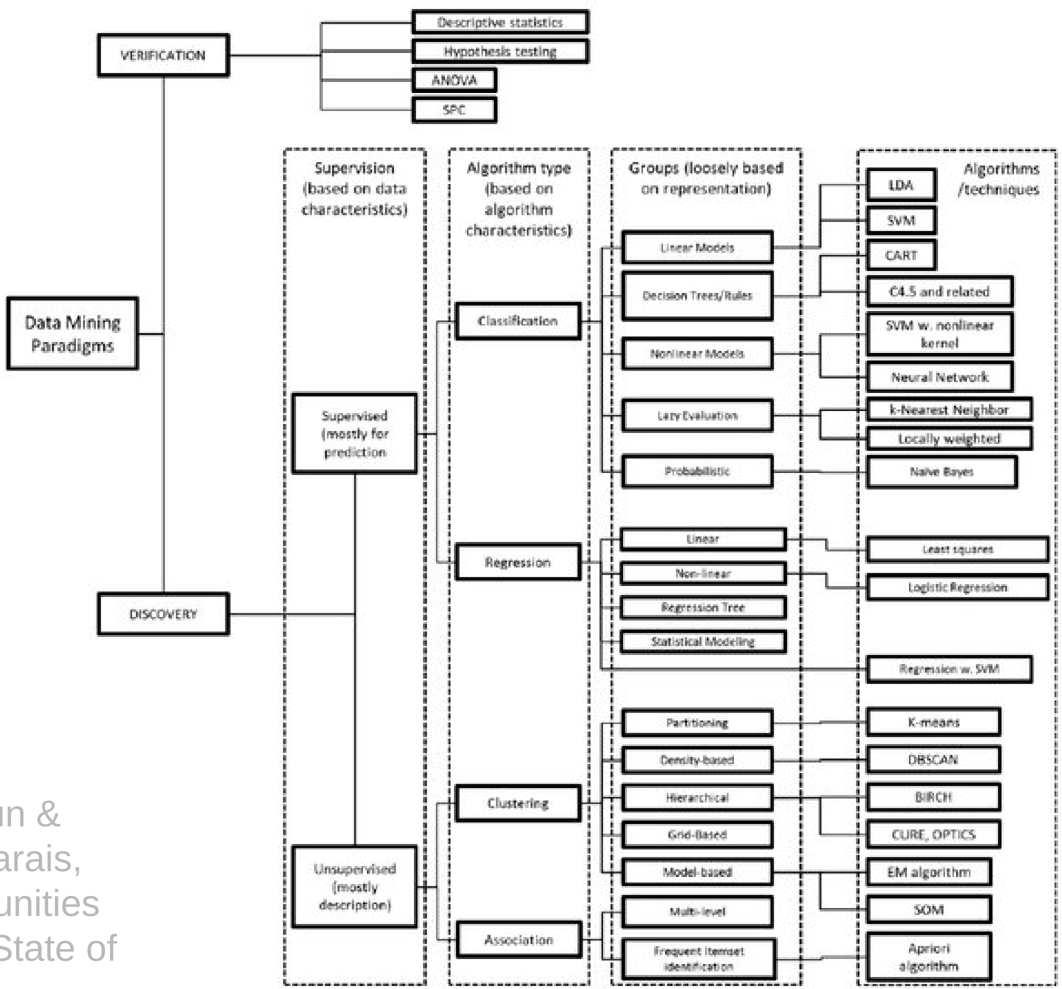


Classification

- Goal:
  - Find a method to predict the class of observations (samples).
- Learning:
  - Done based on samples of known class (i.e. labelled samples)
  - Training samples of the form $(x_1, \ldots, x_D, C_i)$
- In contrast to regression, labels are discrete classes $(C_1, \ldots, C_c)$.
- Different methods available in the literature:
  - Decision Trees
  - Classification Rule Sets
  - Neural Networks
  - …

Introduction to classification (2)

Christian-Albrechts-Universität zu Kiel



Example of a taxonomy

Source: Gavrilovski, Alek & Jimenez, Hernando & Mavris, Dimitri & Rao, Arjun & Shin, San-Hyun & Hwang, Inseok & Marais, Karen. (2016). Challenges and Opportunities in Flight Data Mining: A Review of the State of the Art. 10.2514/6.2016-0923.

Introduction to classification (3)

In this lecture, the focus is on simple and basic classification methods.

- Occasionally:
  - the assumption holds that the samples are distributed identically and independently (iid)
  - one feature / a simple set of rules / a linear combination of features is enough for solving the classification problem

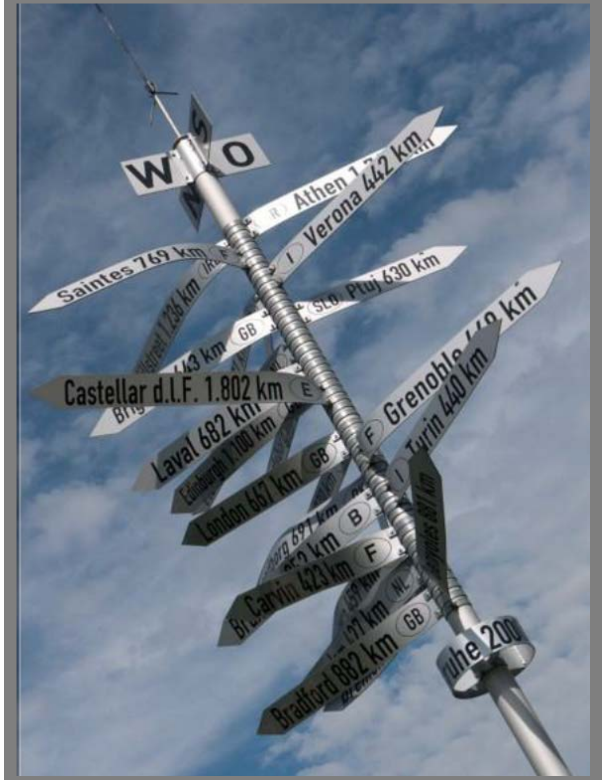
Ockham's razor

Entia non sunt multiplicanda praeter necessitatem.

→ ”There should not be made any assumptions beside the necessary.”
(William of Ockham, 1287–1347)

Agenda

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings



The 1-R Classifier

- Suitable for nominal features
  - Nominal features are in a discrete and finite value range and have no inherent ordering of preference structure
  - For example: gender (male or female), subject (economy, computer science, medicine, …), nationality (German, Italian, Austrian, British, …)
- Goal: Find a set of rules applied to one feature only
  - Set of rules correspond to a Decision Tree (see later) with one layer
- Inventor: Holte (University of Ottawa) 1993
  - Introduced in the paper: Comparison of 16 benchmark data sets – similar performance as more complex Decision Trees
- Possible extension for ordinal features:
  - Ordinal features are in a finite value range with an ordering structure

Algorithm 1-R Classifier:

- For all possible values of a feature:
  - Count the occurrences of every class.
  - Find the most frequent class.
  - Produce a rule assigning the class to the feature value
- Calculate the failure rate of rules.
- Choose the rule of lowest failure rate

The 1-R Classifier (3)

Example: Playing golf?

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

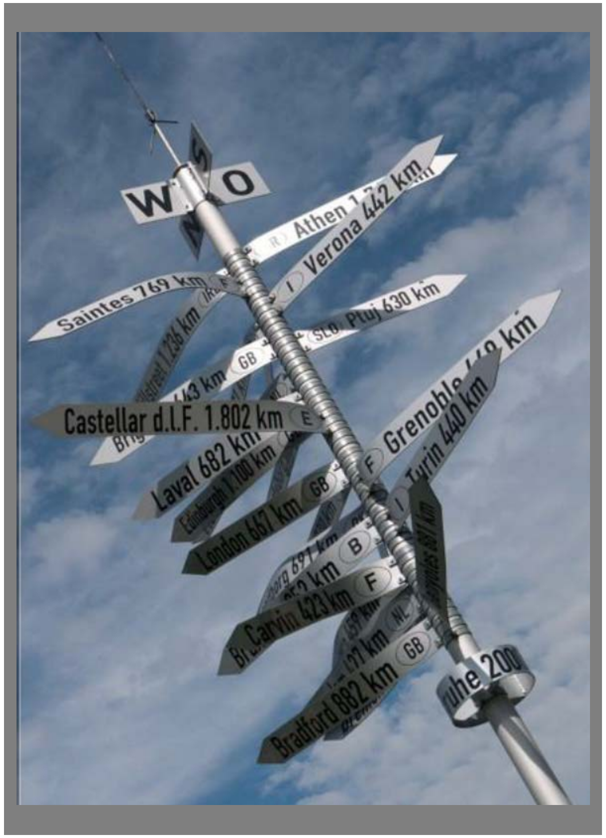
* Represents a preference in case of ties

Here, the rules of the features *humidity* or *outlook* are chosen.

Agenda

Christian-Albrechts-Universität zu Kiel

- Introduction to classification
- 1-R Classifier
- **k-Nearest Neighbour**
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Conclusion and further readings



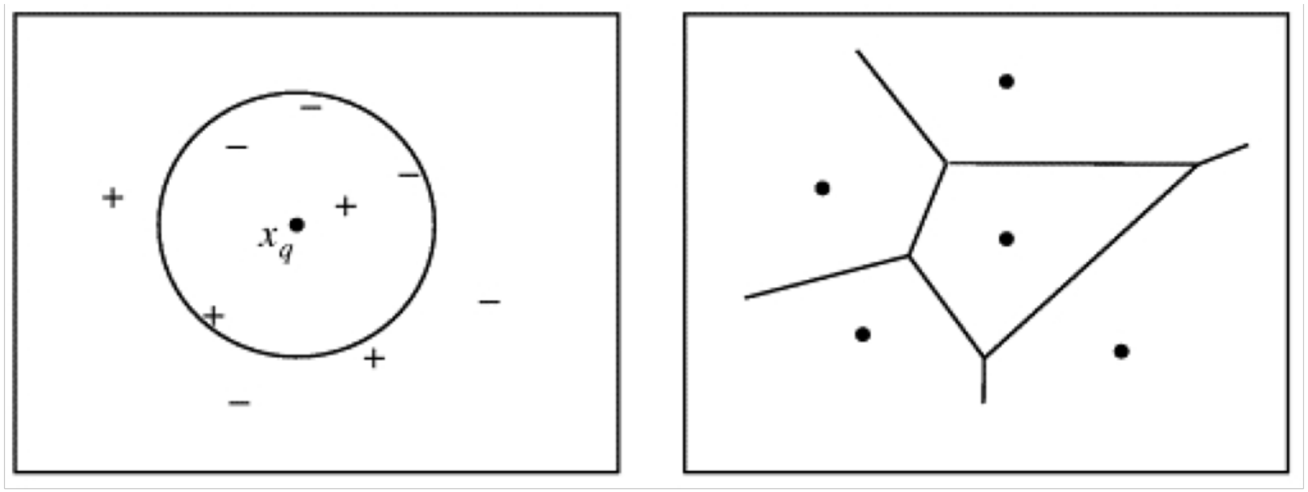
k-NN

- Very simple classification technique
- Approach:
  - Use all training samples as a model
    - No selection
    - No training
  - Classify an unknown sample by observing ist k nearest neighbours
  - Application of a well-known distance metric for samples
  - Discrimination of the class via majority decision
- Decision
  - Only parameter k was taken into account
  - k determines the number of nearest neighbours
  - Typical values for k: 1, 3 or 5 (for a two-class problem)
  - For more classes: Higher values, ideally allow for a majority decision

k-Nearest Neighbour Classifier (2)

Christian-Albrechts-Universität zu Kiel

Example



5-NN assigns the sample to the class " – "

1-NN is represented by a Voronoi diagram (cmp. decision boundary)

Choice of parameter k

- Properties of large k
  - Process becomes more resistant against noise
  - But: Also not relevant samples will be taken into consideration
- Properties of small k
  - Nuances of the class distribution can be modelled
  - But: Sensitive against noise
- Challenge: Find an appropriate trade-off
- Alternative
  - Weighting according to class affiliations of neighbours (and their neighbours)
  - Restriction to k neighbours is obsolete since all training samples will be considered

k-Nearest Neighbour Classifier (4)

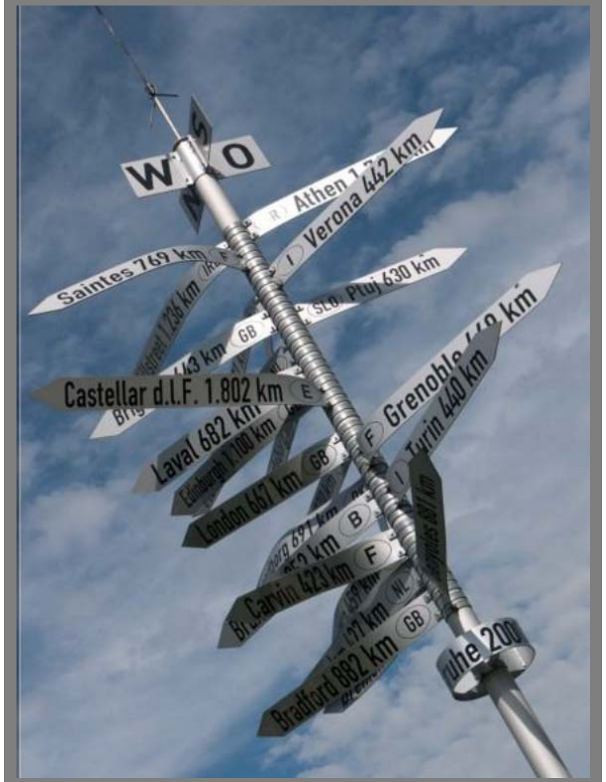
Evaluation

- Training is not required
- Fast, but storing of all the training samples is necessary
- Classification process: Expensive, because all samples have to be taken into account (search nearest neighbours)
- Model of class distribution of the known training samples
  → Only local approximation (for every sample to be classified)

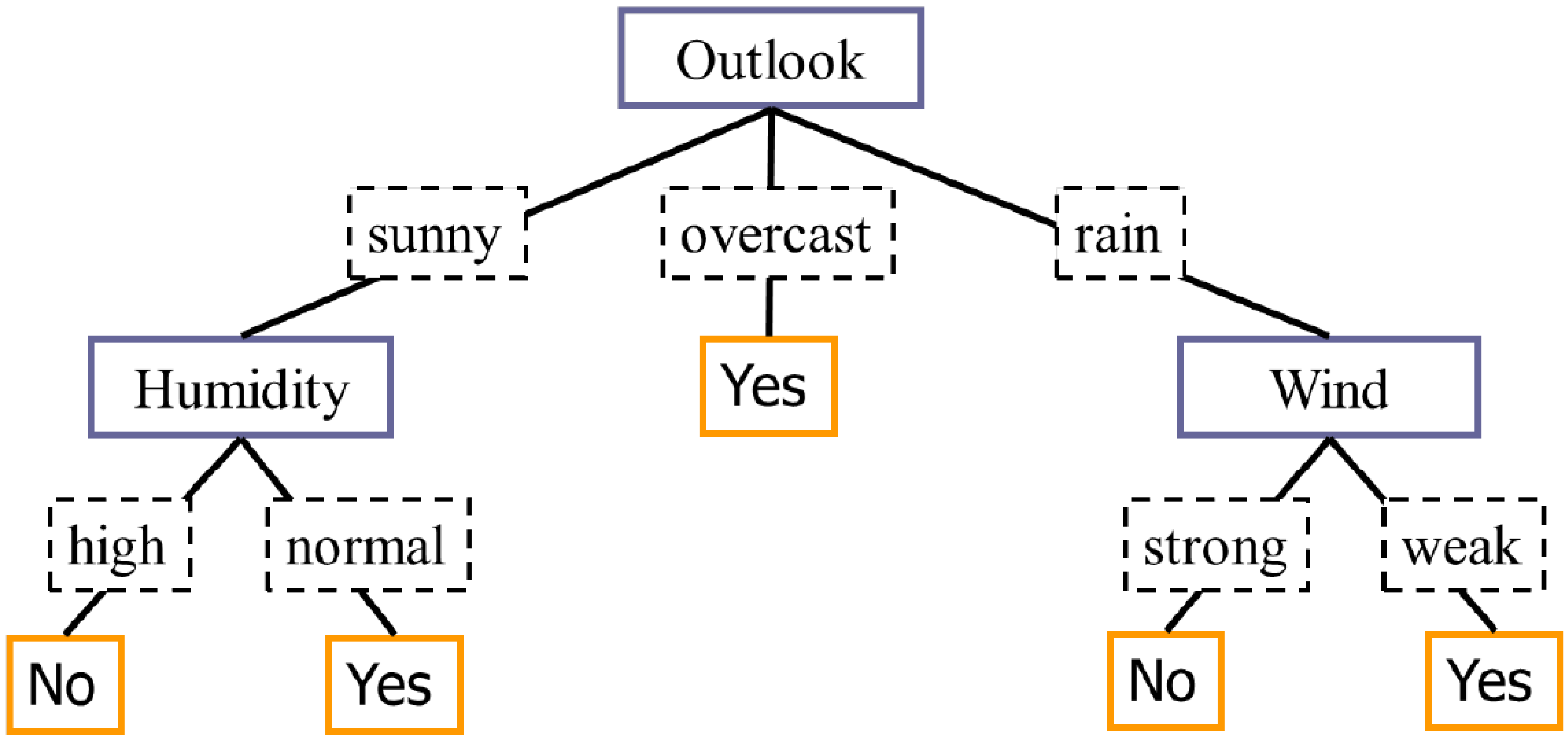
- Good for reference values of the classification performance

Agenda

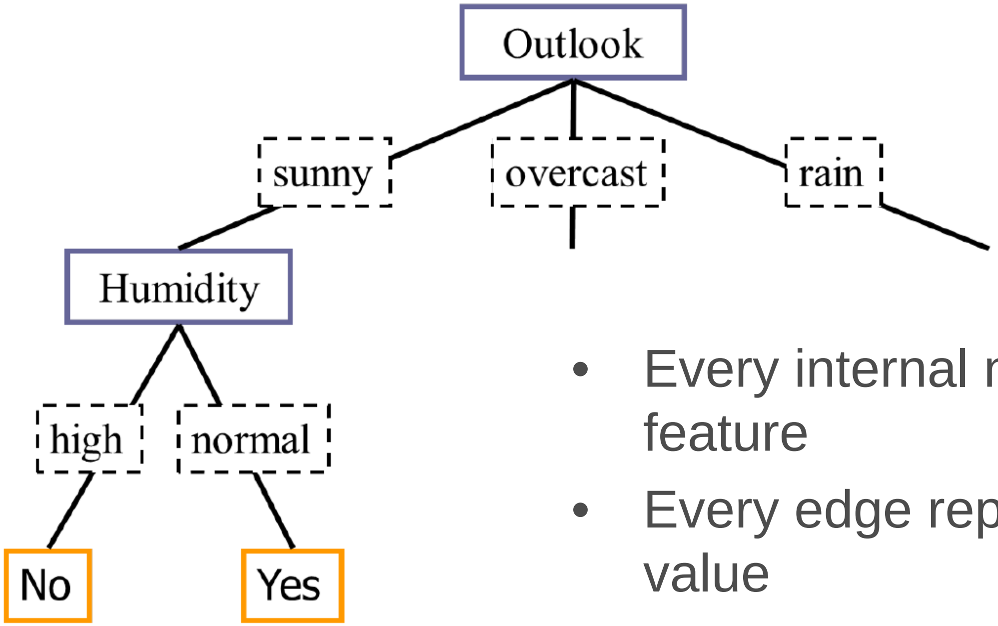
- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- **Decision Trees**
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings



Playing golf: Yes/No?



Internal nodes, edges, leaf nodes



- Every internal node checks a feature
- Every edge represents a feature value
- Every leaf node represents a class assignment

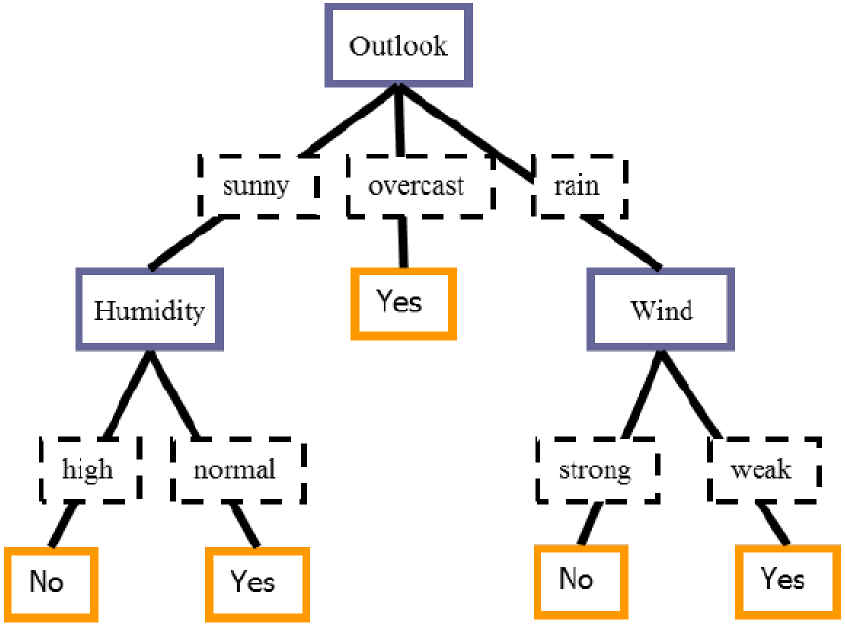
Traversing decision trees

Algorithm:

1) Begin with the root node
2) While the current node is no leaf node
   - Answer the question of the current node
   - Follow edge with observed feature value to the next current node
3) Result can be read from the leaf node

Traversing – Example

- 4 features: outlook, temperature, humidity, wind
- Sample: [rain, cool, normal, strong]
- Outlook = rain → choose right edge
- Wind = strong → choose left edge
- Decision: No



Decision Trees (5)

Properties

- Discrete and continuous features
- Noisy data
- The classification process shall be interpretable (rule extraction)

Construction of Decision Trees

- Manually: developing Decision Trees with the help of experts
  - Rules are often redundant, incomplete, or inefficient
  - Time-consuming and expensive process
- Induction: derive Decision Trees automatically from sample data (training data)

Decision Trees (6)

Christian-Albrechts-Universität zu Kiel

Induction-based methods:

- Enumerative approach:
  - Produce all possible Decision Trees
  - Choose the tree with a minimal number of nodes
  - → Optimal tree will be found
  - → But: Very inefficient process
- Heuristical approach:
  - Extend an existing tree with additional internal nodes
  - Terminate when the stop criterion is fulfilled
  - → More efficient
  - → Optimal tree is not found generally

Christian-Albrechts-Universität zu Kiel

Decision Tree construction

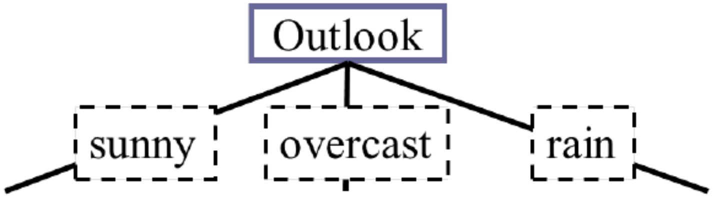
Simple algorithm:

1) Begin with an empty tree
2) Partition the training set recursively by selecting a single feature step by step
3) Stop when no more features are available or another stop criterion is fulfilled

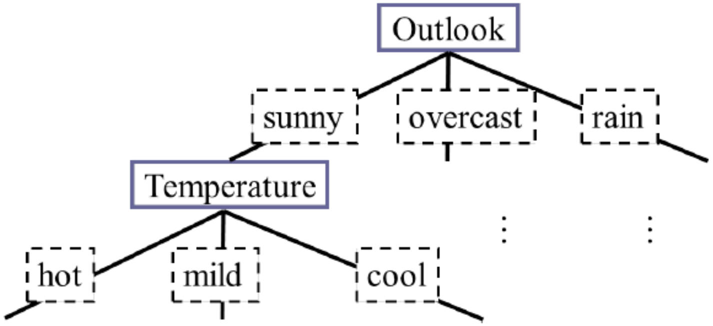
Decision Trees (8)

Christian-Albrechts-Universität zu Kiel

Application of the algorithm:



1) Feature: Outlook
   Possible feature values:
   sunny, overcast, rain



2) Feature: Temperature
   Possible feature values:
   hot, mild, cool

Order of the feature selection

- What happens if one chooses a different order?

  → You get a different tree.

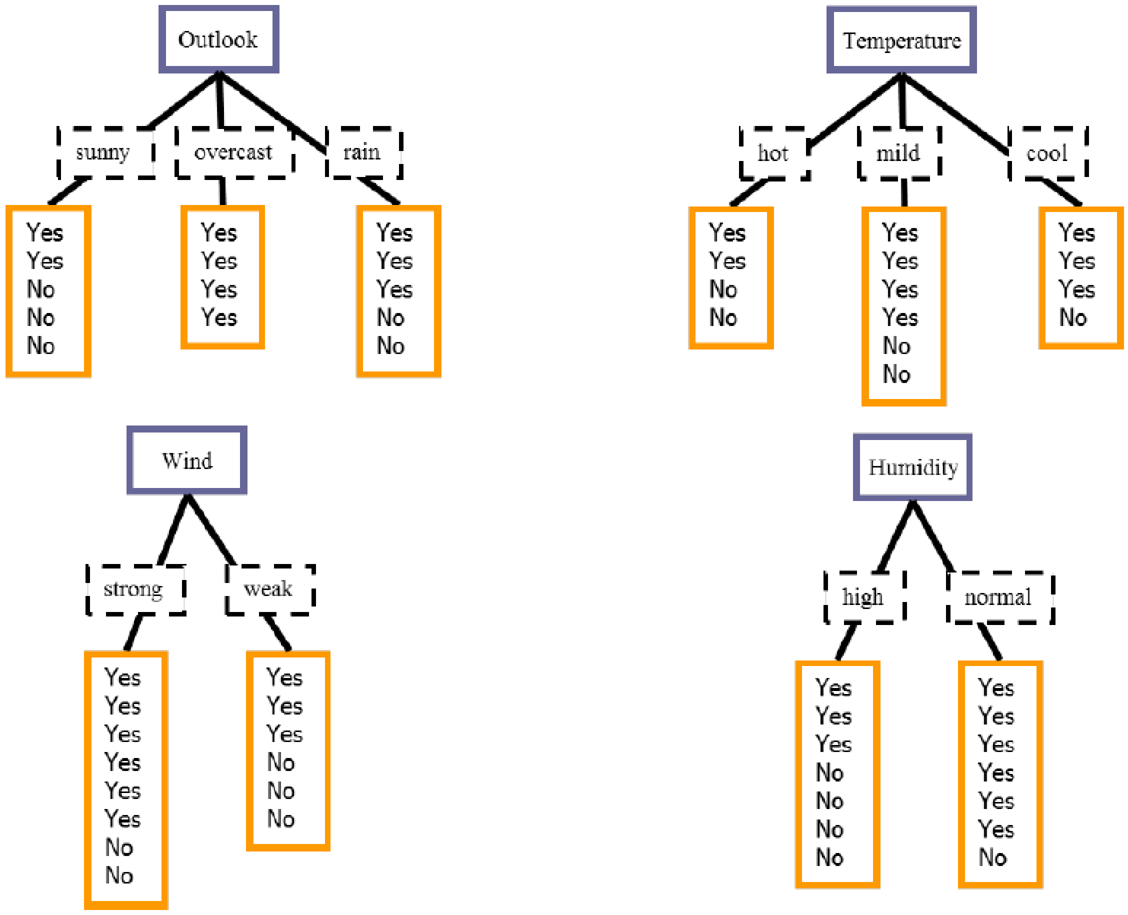
- Which order is the best? Which feature shall be selected next?
- Splitting strategies:
  - Information gain
  - Gain ratio
  - Gini index

Decision Trees (10)

Christian-Albrechts-Universität zu Kiel

Splitting strategies

- Which feature shall be selected next?



Christian-Albrechts-Universität zu Kiel

What is the best feature?

- The feature producing a minimal tree
- Heuristical: Choose the feature which produces the “purest“ class distribution (e.g. only Yes or only No)

Splitting strategy: Information Gain (IG)

- Popular method
- Already known; feature selection
- Property: The more average “purity“ the partitioned subsets have, the higher is the IG
- Strategy: Choose the feature with the highest IG

Decision Trees (12)

Reminder: Measure of Information

- Partitioning a data set X by a feature d in L subsets X_d with $l = 1, \ldots, L$.
- There are C classes corresponding to the manifestations (possible instances) of the feature.
- Information Gain (IG) of a feature d:

$$IG(d) := I(X) - \sum_{l=1}^{L} \frac{|x_{d_l}|}{|X|} I(x_{d_l})$$

$$I(x_{d_l}) := -\sum_{c=1}^{C} px_{d_l}(c) \cdot \log_2 px_{d_l}(c)$$

$$I(X) := -\sum_{c=1}^{C} px\,(c) \cdot \log_2 px\,(c)$$

Christian-Albrechts-Universität zu Kiel

Example: Feature Outlook

- Outlook = sunny: 5 samples – 3 times ‘No‘, 2 times ‘Yes‘
  - $E(outlook = sunny) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$
- Outlook = overcast: 4 samples – 0 times ‘No‘, 4 times ‘Yes‘
  - $E(outlook = sunny) = -\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4} = 0$
- Outlook = rain: 5 samples – 2 times ‘No‘, 3 times ‘Yes‘
  - $E(outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$

Example: Feature Outlook

- Entropy of the whole data set: 14 samples – 5 times ‘No‘, 9 times ‘Yes‘
  - $E(X) = -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14} = 0.940$

- Hence:

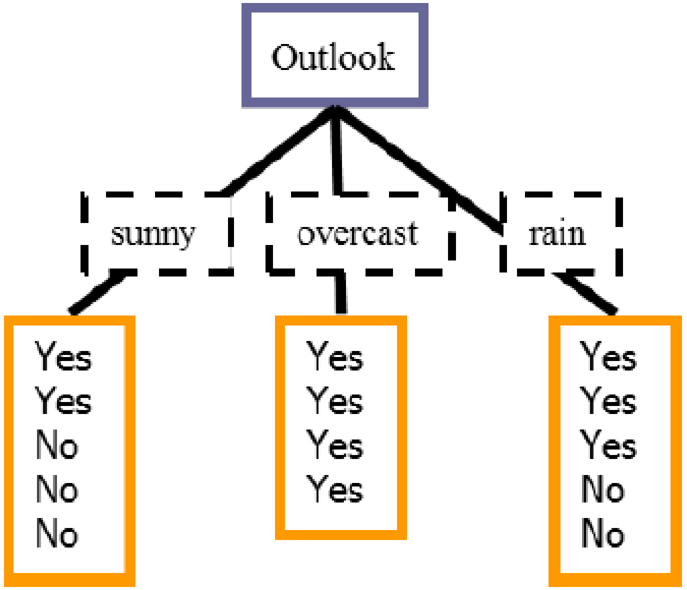
$$IG(Outlook) = E(X) - \frac{5}{14}E(Outlook = sunny) - \frac{4}{14}E(Outlook = overcast) - \frac{5}{14}E(Outlook = rain) = 0.247$$

Christian-Albrechts-Universität zu Kiel

Results for all features:

- $IG(Outlook) = 0.247$
- $IG(Temperature) = 0.029$
- $IG(Humidity) = 0.152$
- $IG(Wind) = 0.048$

- Insight: Outlook is the first partitioning feature!



What comes next?

- Consider left branch: Outlook = sunny
- New data set D:

Outlook	Temperature	Humidity	Wind	Play
sunny	hot	high	weak	No
sunny	hot	high	strong	No
sunny	mild	high	weak	No
sunny	cool	normal	weak	Yes
sunny	mild	normal	strong	Yes

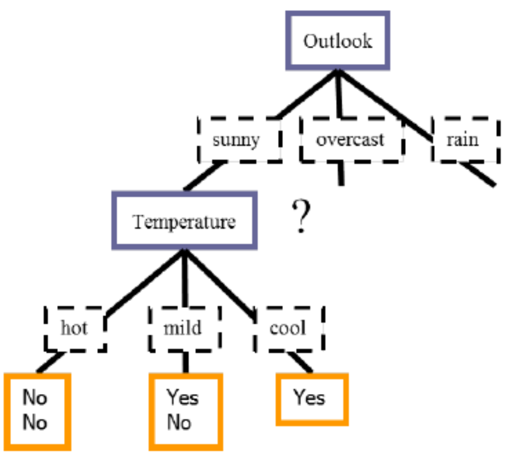
- The entropy of the new data set:

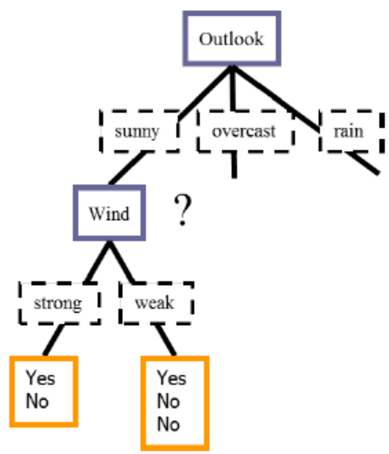
$$E(D) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$
$$= E(Outlook = sunny)$$

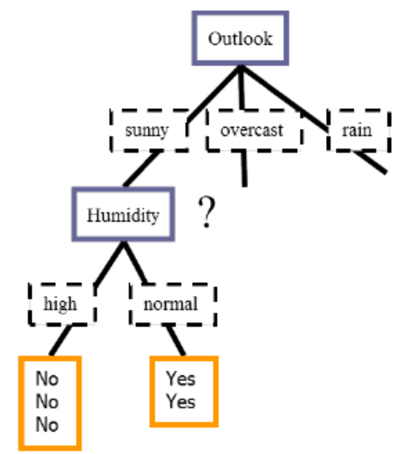
Decision Trees (17)

Other possible partitions

- Only three features: temperature, humidity, wind







$IG(Temperature) = 0.571$ $\quad$ $IG(Wind) = 0.020$ $\quad$ $IG(Humidity) = 0.971$

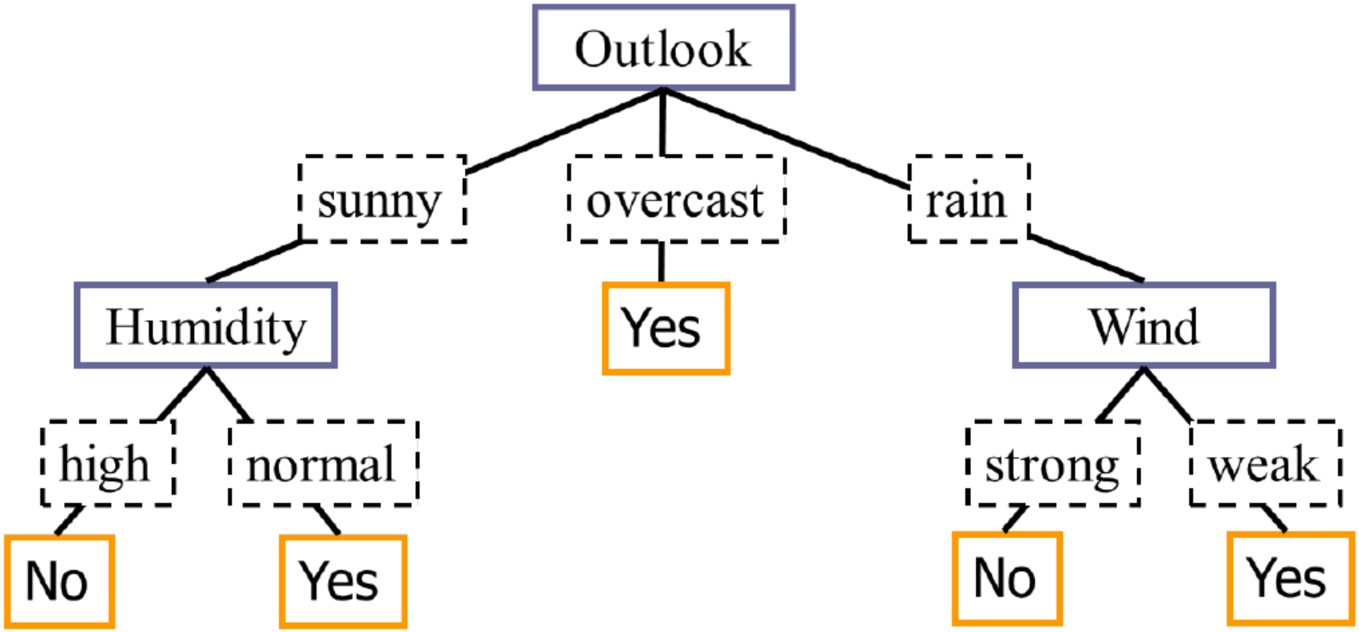
Remark

- Select feature humidity, because it corresponds to the largest IG value (0.971)
- No further separation of this branch necessary (entropy is zero)

Next steps

- Analogously with Outlook = overcast and Outlook = rain
- Recursive steps until all features are treated or IG=0.

Result:



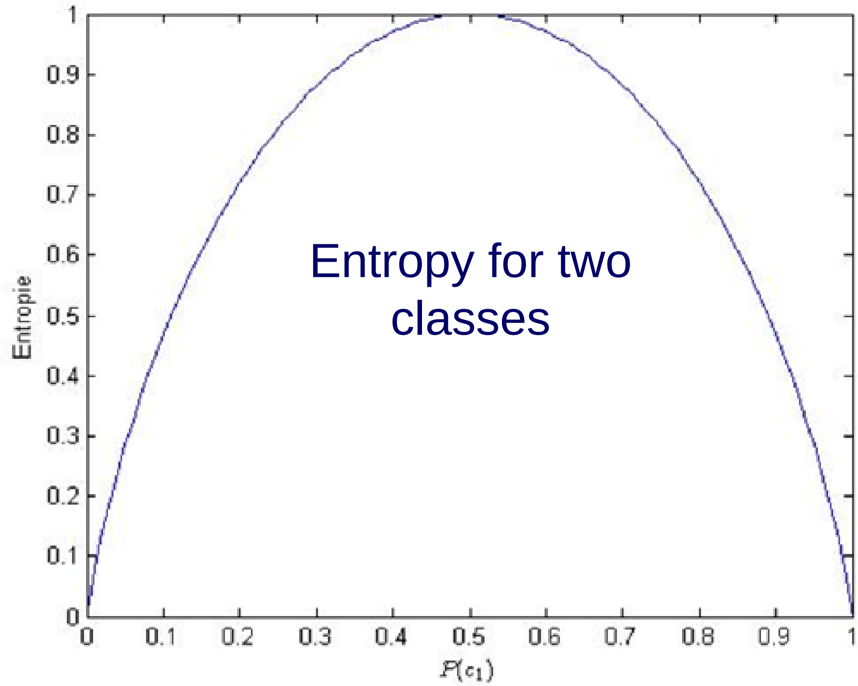
Remarks:

- At the end, there might be nodes left containing more than one class (e. g. same samples with the different class assignment)

Decision Trees (20)

Properties of entropy

- Entropy is maximal if the classes are equally frequent
- If only one class is left, the entropy is zero
- Example: $E(Outlook =$



Extreme case: Column of indices

Index	Outlook	Temperature	Humidity	Wind	Play
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes
D5	rain	cool	normal	weak	Yes
D6	rain	cool	normal	strong	No
D7	overcast	cool	normal	strong	Yes
D8	sunny	mild	high	weak	No
D9	sunny	cool	normal	weak	Yes
D10	rain	mild	normal	weak	Yes
D11	sunny	mild	normal	strong	Yes
D12	overcast	mild	high	strong	Yes
D13	overcast	hot	normal	weak	Yes
D14	rain	mild	high	strong	No

Christian-Albrechts-Universität zu Kiel

Compute IG for feature index

$$E(Index = D1) = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$$

…

$$E(Index = D14) = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$$

- Means:

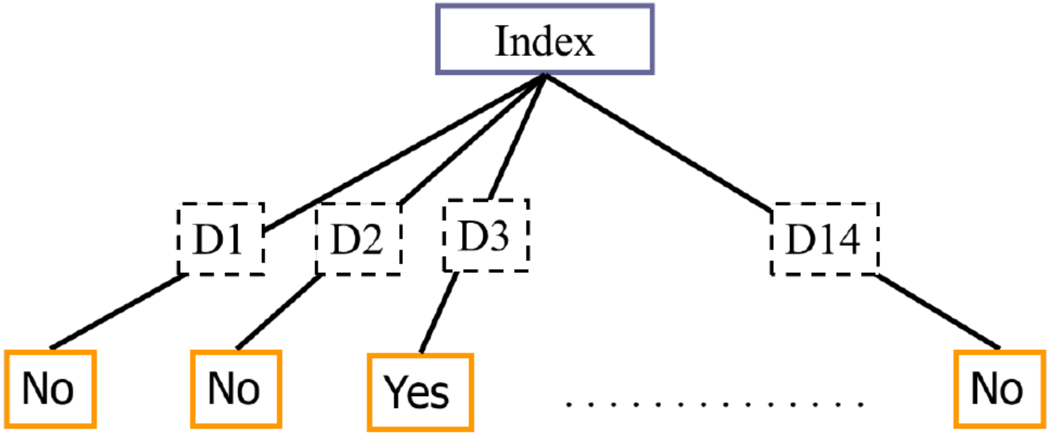
$$\begin{aligned} &IG(Index) = E(D) \\ &- \frac{1}{14}E(Index = D1) \\ &\vdots \\ &- \frac{1}{14}E(Index = D14) \\ &= 0.940 \end{aligned}$$

Results for all features:

- $IG(Index) = 0.940$
- $IG(Outlook) = 0.247$
- $IG(Temperature) = 0.029$
- $IG(Humidity) = 0.152$
- $IG(Wind) = 0.048$

- Insight: Index is always selected!

Corresponding tree:



- Bias: Features with a high number of distinct values are always selected
- Is this useful?

→ No! Good classification for training data but worse for unknown samples (new index values).

Overfitting

- A tree b is overfitted if there is another tree b‘ with

$$error_{train}(b) < error_{train}(b')$$

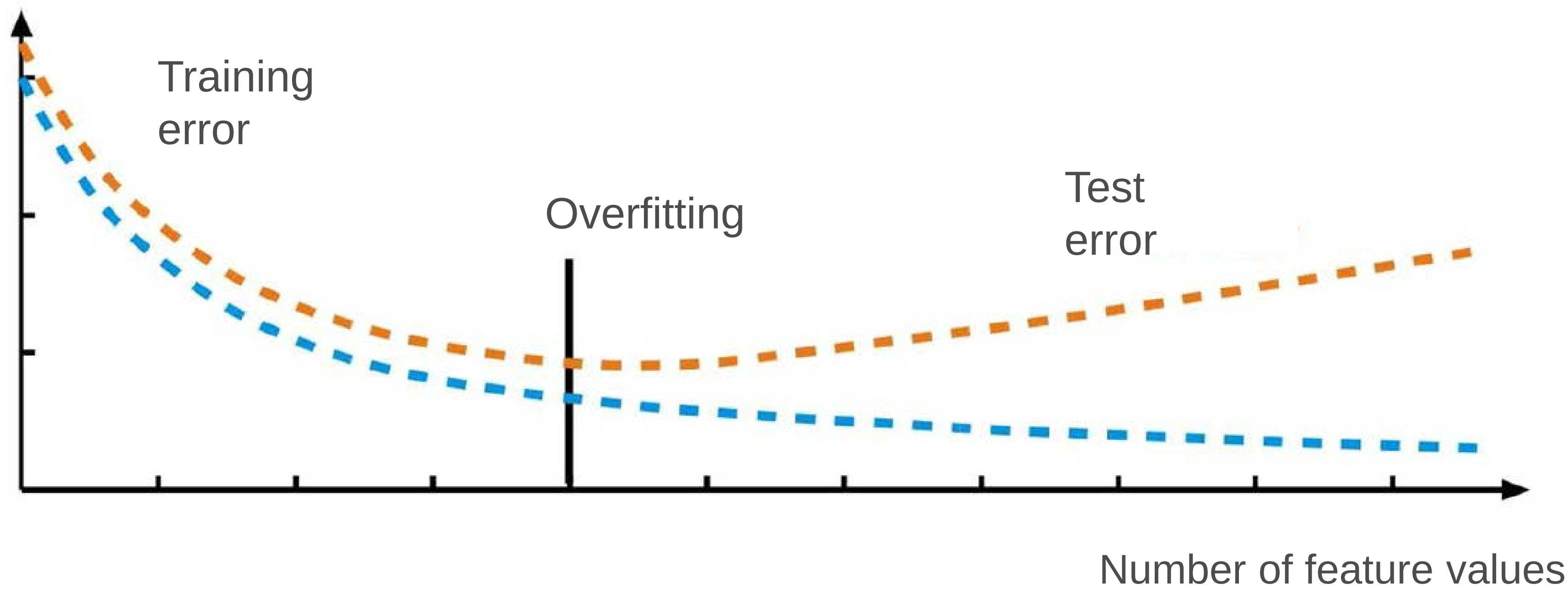
- Furthermore:

$$error_{test}(b) > error_{test}(b')$$

- Where:
  - $error_{train}(b)$ is the classification error of tree b on training data
  - $error_{train}(b')$ is the classification error of tree b′ on training data
  - $error_{test}(b)$ is the classification error of tree b on test data
  - $error_{test}(b')$ is the classification error of tree b‘ on test data

Christian-Albrechts-Universität zu Kiel

Overfitting – Example



Christian-Albrechts-Universität zu Kiel

Gain Ratio

- Modification to reduce bias provoked by features with lots of distinct values
- Gain Ratio (GR) concerns the number and size of branches of a node

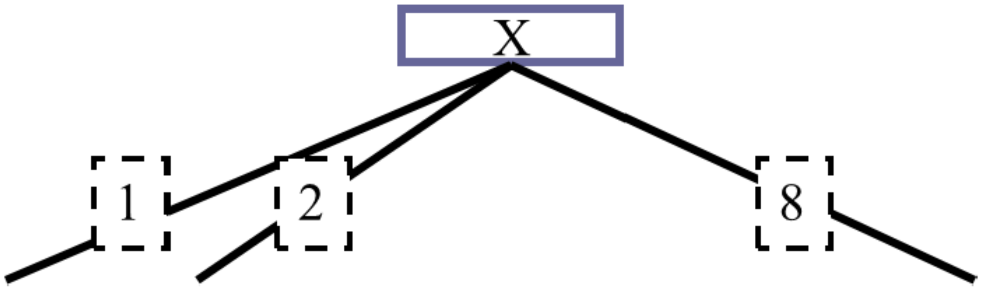
Concrete

- IG is corrected by regarding the information of the branching itself (how much information is necessary to say to which branch a sample belongs?)
- Intrinsic Information (II)

Christian-Albrechts-Universität zu Kiel

Intrinsic information for example:

- Simple tree
  - 1 feature
  - 1 node
  - 8 samples
  - 8 possible feature vectors



- How much information is necessary to encode the feature value of such a sample?

$$\Pi(X) = -\sum_{j=1}^{8} \frac{1}{8} \log_2 \frac{1}{8} = 3\ bit$$

Christian-Albrechts-Universität zu Kiel

Example: Π for feature ‘index‘

- Partitioning into 14 subsets
- Subset size: 1

$$\Pi(Index) = -\sum_{j=1}^{14} \frac{1}{14} \log_2 \frac{1}{14}$$
$$= 14 \cdot \left(-\frac{1}{14} \log_2 \frac{1}{14}\right)$$
$$= 3.807$$

Example 2: Π for feature ‘outlook‘

- Partitioning into 3 subsets
- Subset size: 5 (sunny), 4 (overcast), 5 (rain)

$$
\begin{aligned}
\Pi(Outlook) = &-\frac{5}{14}\log_2\frac{5}{14} \\
&-\frac{4}{14}\log_2\frac{4}{14} \\
&-\frac{5}{14}\log_2\frac{5}{14} \\
&= 1.577
\end{aligned}
$$

Definition: Gain Ratio

- Correction (i.e. normalisation) of IG
- Gain Ratio of a feature X

$$GR(X) = \frac{IG(X)}{\Pi(X)}$$

- Strategy: Select feature with the highest Gain Ratio!

Decision Trees (32)

Gain Ratio for the golf example:

- $GR(Index) = \frac{0.940}{3.807} = 0.247$
- $GR(Outlook) = \frac{0.247}{1.577} = 0.157$
- $GR(Temp.) = \frac{0.029}{1.362} = 0.021$
- $GR(Humidity) = \frac{0.152}{1} = 0.152$
- $GR(Wind) = \frac{0.048}{3.958} = 0.050$

Observations:

- Original data set (without index): Outlook is still the best feature
- With index: despite the correction, the feature index has the largest GR
- Solution: Test procedure detecting special features like index
- Then, why at least GR?

  → Works for features with lots of distinct values – only struggles in the extreme case of index columns

Extension to numerical features

- So far only nominal and discrete features were taken into account
- Not applicable for a practical use case

  → E. g. sensor data such as length [m], weight [kg], speed [km/h]
- Extension required: Numerical features have to be processed differently

Christian-Albrechts-Universität zu Kiel

Example: weather data with numerical feature

- So far: Temperature values categorisable into (hot, mild, cold)
- Now: Integer values representing degrees Celsius

Outlook	Temperature	Humidity	Wind	Play
sunny	85	high	weak	No
sunny	80	high	strong	No
overcast	83	high	weak	Yes
rain	75	high	weak	Yes
rain	68	normal	weak	Yes
rain	65	normal	strong	No
overcast	64	normal	strong	Yes
sunny	72	high	weak	No
sunny	69	normal	weak	Yes
rain	70	normal	weak	Yes
sunny	75	normal	strong	Yes
overcast	72	high	strong	Yes
overcast	81	normal	weak	Yes
rain	71	high	strong	No

Approach: Formation of intervals

- Sorting of values
- Formation of “new features“ introducing interval borders
- Then: Apply Splitting Strategy

Example: feature ‘temperature‘

1. Sort feature values
2. Determine interval border, e.g. at 71.5
   1. Temperature < 71.5: 2x ‘No‘, 4x ‘Yes‘
   2. Temperature > 71.5: 3x ‘No‘, 5x ‘Yes‘
3. Calculate IG for the interval border, i.e. $split = 71.5$

Temp.	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Play?	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

IG for interval border $split = 71.5$

- $E(Temperature < 71.5) = -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6} = 0.918$
- $E(Temperature > 71.5) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = 0.954$

Hence:

$$IG(split = 71.5) = E(D) \\ -\frac{6}{14}\cdot 0.918 \\ -\frac{8}{14}\cdot 0.954 \\ = 0.940 - 0.939 = 0.001$$

Christian-Albrechts-Universität zu Kiel

IG for all possible interval borders

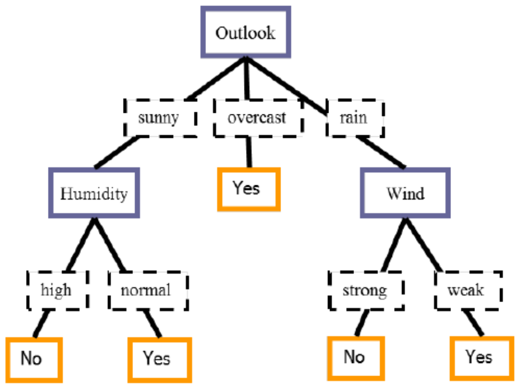
64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- Calculate IG for all possible borders
- Define border with maximal IG
- Maximal IG corresponds to IG for feature temperature

Optimisation

- Do we really need to score all borders?
- No, cause borders within a target class cannot be possible
  - Only 7 interval borders left instead of 13

Rule extraction from trees

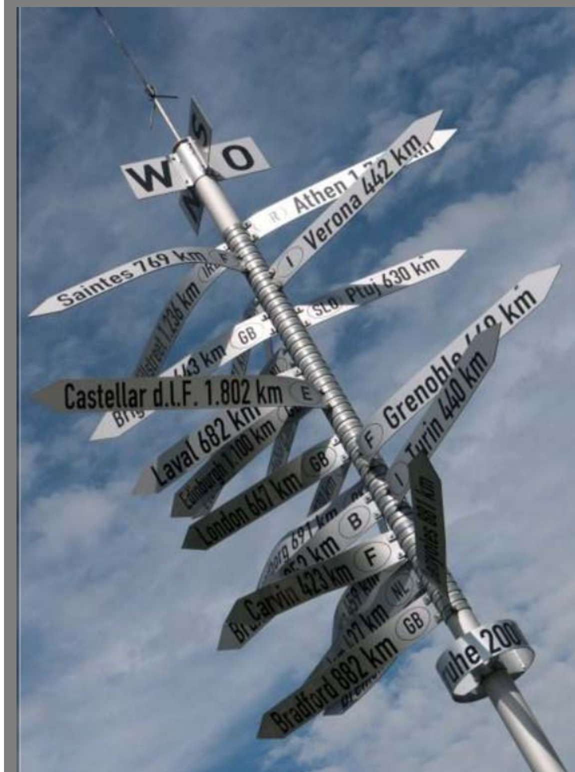


- **IF** … **THEN** … rules
- General:
- **IF** $test_1$ **AND** … **AND** $test_n$ **THEN** Decision C
- One rule per leaf

- **IF** Outlook=sunny **AND** Humidity=high **THEN** Decision No
- **IF** Outlook=sunny **AND** Humidity=normal **THEN** Decision Yes
- **IF** Outlook=overcast **THEN** Decision Yes
- **IF** Outlook=rain **AND** Wind=strong **THEN** Decision No
- **IF** Outlook=rain **AND** Wind=weak **THEN** Decision Yes

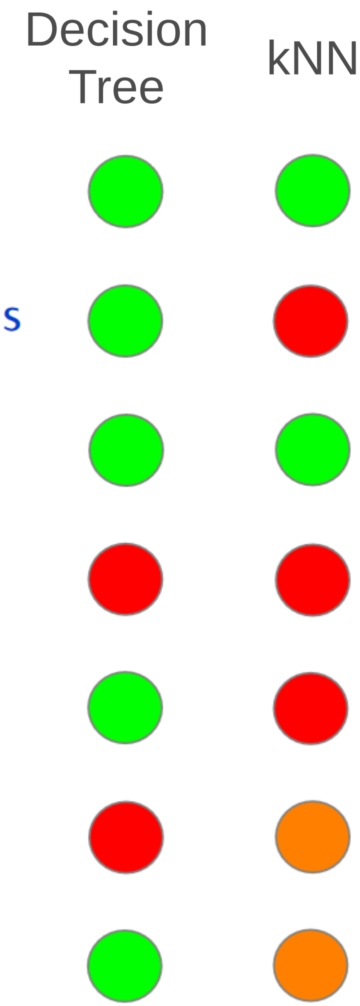
Agenda

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- **Random Forest**
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings



Christian-Albrechts-Universität zu Kiel

- Intrinsically multiclass
- Handles Apple and Orange features
- Robustness to outliers
- Works w/ "small" learning set
- Scalability (large learning set)
- Prediction accuracy
- Parameter tuning



Random Forest

- Definition
  - Collection of unpruned Decision Trees
  - Rule to combine individual tree decisions
- Purpose
  - Improve prediction accuracy
- Principle
  - Encouraging diversity among the tree
- Solution: randomness
  - Bagging
  - Random decision trees

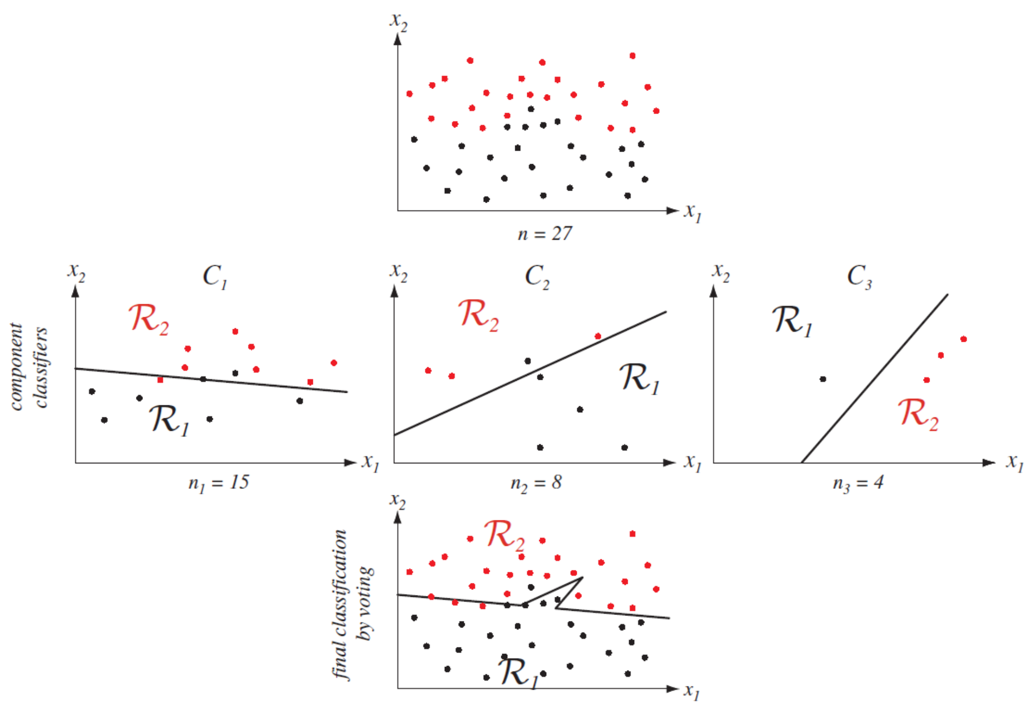
Bagging

- Bootstrap aggregation
- Technique from the domain of ensemble learning
  - To avoid overfitting
    → Important since trees are not pruned!
  - To improve stability and accuracy
- Two steps
  - Bootstrap sample set
  - Aggregation

Christian-Albrechts-Universität zu Kiel

Combination of classifiers

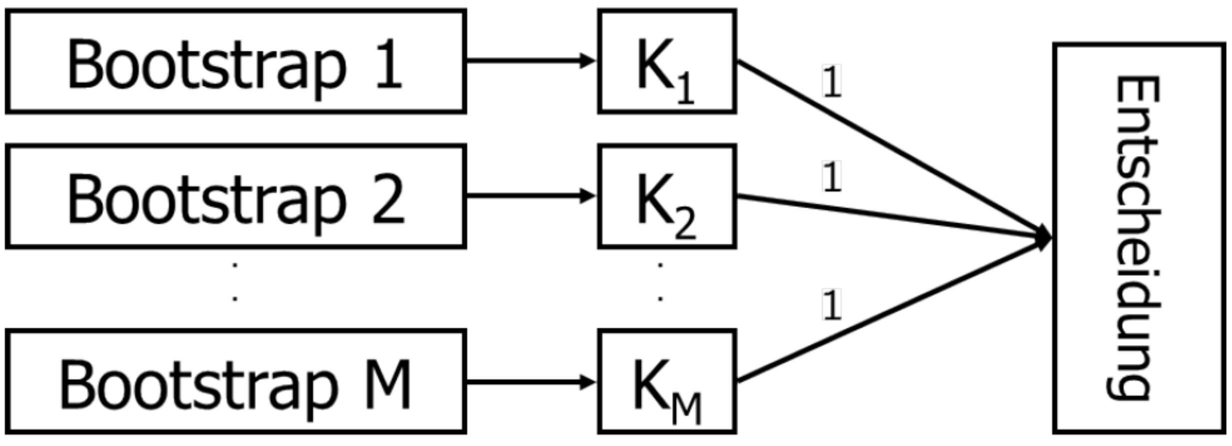
- Why to combine multiple classifiers?
  - → Possible improvement of the classification performance
  - → Example:



Christian-Albrechts-Universität zu Kiel

Bagging

- Bootstrap aggregation: for every classifier, a new/own training set ("bootstrap") will be generated
  - Random draws with placing back
- Combination of all classifiers via majority decision



Boosting

- The probabilities for selection of a sample are not constant (as in bagging), but will be recalculated in every Bootstrap iteration
- All classifiers are generated step by step
- Sample which has been misclassified will be selected more likely
- For the total decision the classification performance of every single classifier is taken into account
- Alternative name: ARCing (Adaptive Reweighting and Combining)

Bootstrap

- L: original learning set composed of p samples
- Generate K learning sets L_K
  - composed of q samples with $q \leq p$
  - obtained by uniform sampling with replacement from L
  - In consequences, L_K may contain repeated samples
- Random forest: $q = p$
  - Asymptotic proportion of unique samples in L_K
    I.e. $L_K = 100\left(1 - \frac{1}{e}\right) \sim 63\%$
  - The remaining samples can be used for testing

Random Forest (8)

Aggregation

- Learning:
  - For each L_K, one classifier C_K (random Decision Tree) is learned

- Prediction
  - S: a new sample
  - Aggregation = majority vote among the K predictions/votes $C_K(S)$

Random decision tree

- Algorithm:

All labelled samples initially assigned to the root node
N ← root node
With node N do
1) Find the feature F among a random subset of features + threshold value T
* that splits the samples assigned to N into 2 subsets S_{left} and S_{right}
* so as to maximise the label purity within these subsets
2) Assign (F,T) to N
3) If S_{left} and S_{right} too small to be splitted
* Attach child leaf nodes S_{left} and S_{right} to N
* Tag the leaves with the most present label in S_{left} and S_{right}
4) Else
* Attach child nodes N_{left} and N_{right} to N
* Assign S_{left} and S_{right} to them, resp.
* Repeat procedure for $\mathrm{N} = N_{left}$ and $\mathrm{N} = N_{right}$

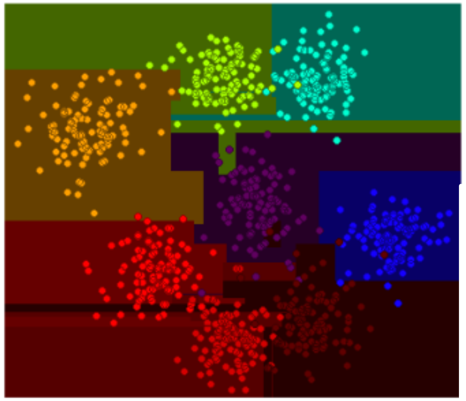
Remarks:

- Random subset of features
  - Random drawing repeated at each node
  - For D-dimensional samples, typical subset size = round(sqrt(D))
    → also round(log2(x))
  - Increases diversity among the random decision trees
  - Also reduces the computational load
- Purity
  - Typical purity measure: Gini index

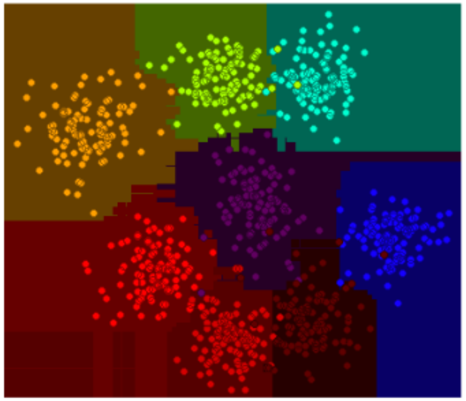
Random Forest (11)

	RF	Decision Tree	kNN
• Intrinsically multiclass	🟢	🟢	🟢
• Handles Apple and Orange features	🟢	🟢	🔴
• Robustness to outliers	🟢	🟢	🟢
• Works w/ "small" learning set	🔴	🔴	🔴
• Scalability (large learning set)	🟢	🟢	🔴
• Prediction accuracy	🟢	🔴	🟠
• Parameter tuning	🟢	🟢	🟠

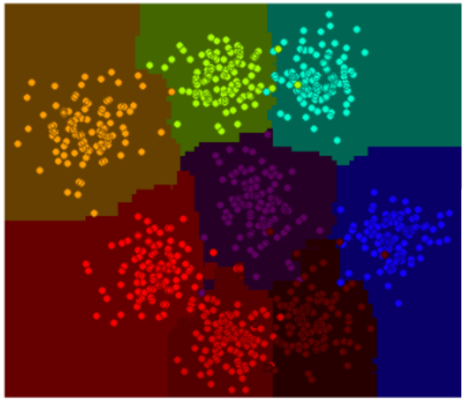
Example illustration



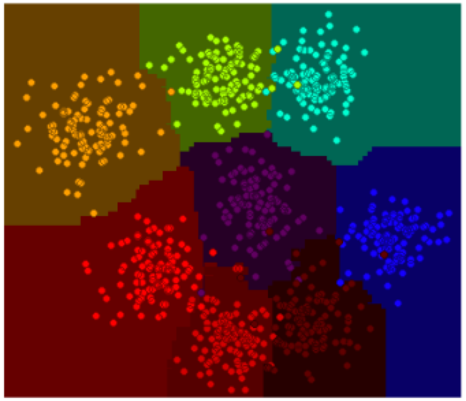
1 Decision Tree



10 Decision Trees



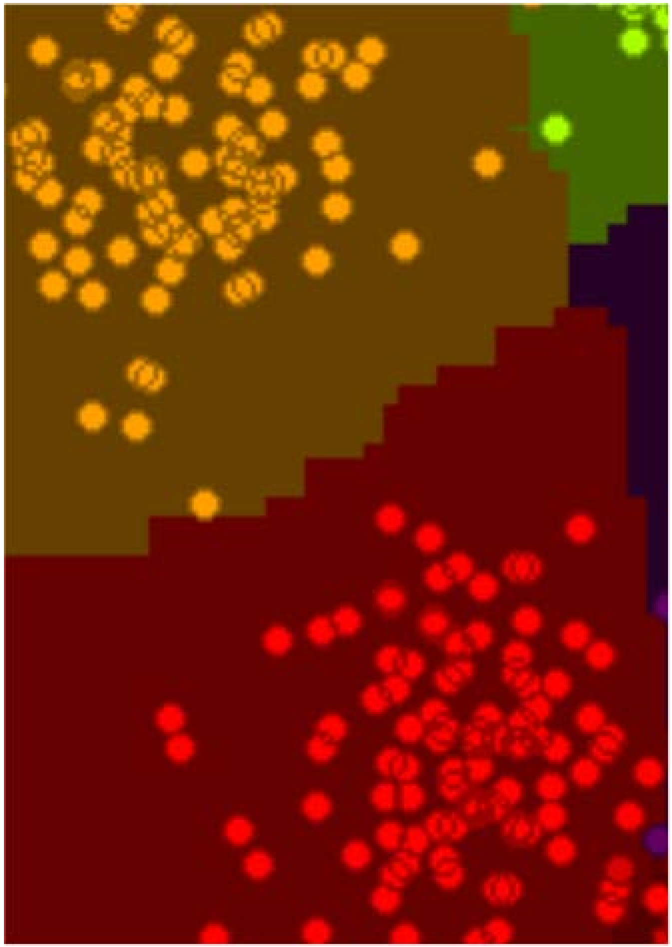
100 Decision Trees



500 Decision Trees

Limitations

- Oblique/curved frontiers
  - Staircase effect
  - Many pieces of hyperplanes
- Fundamentally discrete
  - Functional data? (Example: curves)



Christian-Albrechts-Universität zu Kiel

Kernel-Induced Random Forest (KIRF)

- Random Forest
  - Sample S is a vector
  - Features of S = components of S
- Kernel-induced features
  - Learning set $L = \{S_i, i \in [1, \dots, N]\}$
  - Kernel $K(x, y)$
    - Features of sample $S = \{K_i(S) = K(S_i, S), i \in [0, \dots, N]\}$
    - Samples S and S_i can be vectors or functional data

Kernel trick

- Maps samples into an inner product space
- Usually of higher dimension (possibly infinite)
- In which classification (or regression) is easier
  → Typically linear

Kernel K(x,y)

- Symmetric
- Positive semi-definite (Mercer's condition):

$$\int\int f(x)K(x,y)f(y)dx\,dy \geq 0$$

- $K(x,y) = \langle\varphi(x),\varphi(y)\rangle$
  → Note: mapping needs not to be known (might not even have an explicit representation; e.g., Gaussian kernel)

Examples of Kernels

- Polynomial (homogeneous):

$$K(x, y) = (x \cdot y)^d$$

- Polynomial (inhomogeneous):

$$K(x, y) = (x \cdot y + 1)^d$$

- Hyperbolic tangent:

$$K(x, y) = \tanh(\alpha x \cdot y + \beta)$$

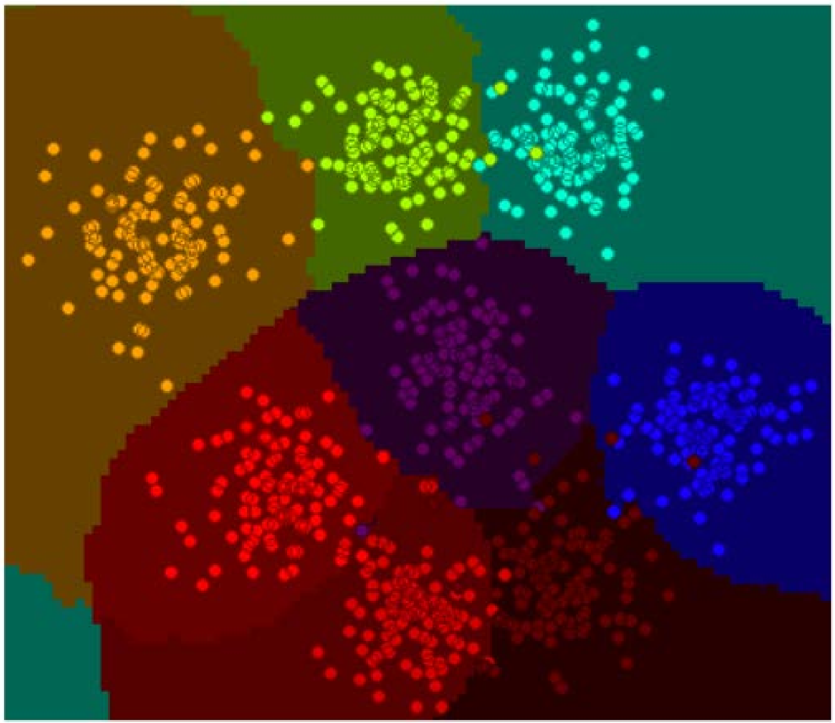
- Gaussian:

$$K(x, y) = \exp(-\gamma \, |x - y|^2)$$

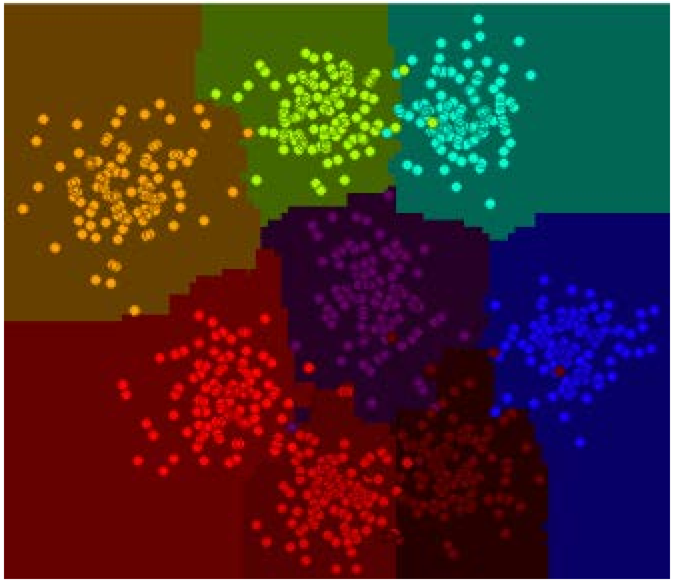
  – Function of the distance between samples
  – Straightforward application to functional data of a metric space
  → e.g. curves

Gaussian kernel

- Some similarity with vantage-point tree



KIRF with 100 random decision trees



Reference: Random Forest with 100 random decision trees

Limitations

- Which kernel?
  - Which kernel parameters?
- No “orange and apple” handling anymore
  - $(x \cdot y)$ or $(x \cdot y)^2$
- Computational load (kernel evaluations)
  - Especially during learning
- Needs to store samples
  - Instead of feature indices in Random Forest

Remarks

- To grow one random decision tree
  - Bootstrap sample set from learning set L
  - Remaining samples
    - Called out-of-bag samples
    - Can be used for testing
- Two points of view
  - For one random decision tree, out-of-bag samples = $\frac{L}{Bootstrap\ samples}$
    - Used for variable importance
  - For one sample S of L, set of random decision trees for which S was out-of-bag
  - Used for out-of-bag error

Out-of-bag error

- For each sample S of the learning set
  - Look for all the random decision trees for which S was out-of-bag
  - Build the corresponding sub-forest
  - Predict the class of S with it
  - Error = is prediction correct?
- Out-of-bag error = average over all samples of S
  - Note: predictions not made using the whole forest
  - But with some aggregation
- Provides an estimation of the generalisation error
  - Can be used to decide when to stop adding trees to the forest

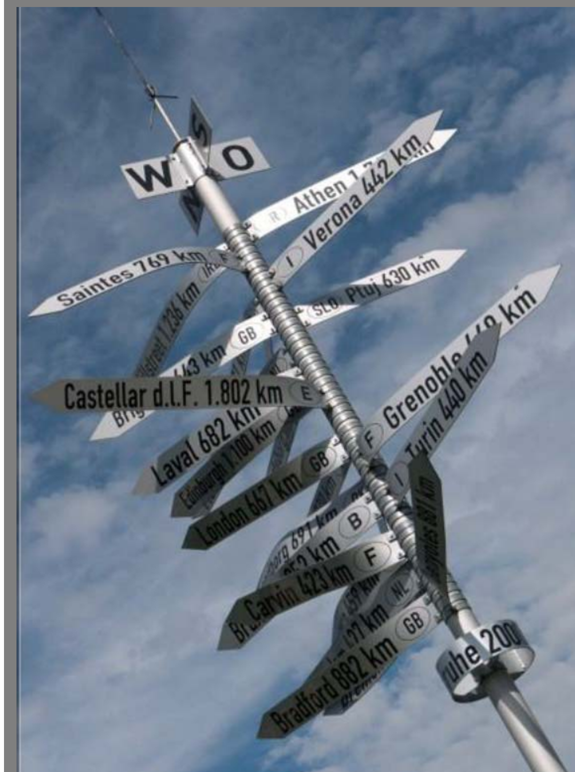
Variable importance

- For each random decision tree
  - Compute out-of-bag error $OOB_{original}$
    - Fraction of misclassified out-of-bag samples
  - Consider the i-th feature/variable of the samples
  - Randomly permute its values among the out-of-bag samples
  - Re-compute out-of-bag error $OOB_{permutation}$
  - Importance of random decision tree i is Imp_i
    $Imp_i = OOB_{permutation}$-$OOB_{original}$

- Variable $importance(i)$ = average overall random decision trees
- Note: random decision tree-based errors (no aggregation)
  → Avoid attenuation of individual errors

Christian-Albrechts-Universität zu Kiel

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Conclusion and further readings



Naïve Bayes Classification

- Takes all features into account (in contrast to 1-R)
- Probabilistic classifier:

$$P(C|x_1, \ldots, x_D)$$

- Based on Bayes‘ theorem by Thomas Bayer (1702-1761)
- Assumption: All features are equally important
- Originally for nominal feature, but can be modified to fit ordinal and other features

Source: https://simple.wikipedia.org/wiki/File:William_of_Ockham.png

Reminder: Probability theory

- Single random variable A and the corresponding probability $P(A)$
- Here more interesting: multiple random variables $A, B, C, \ldots$

- Compound probability: $P(A \cap B)$
  → Alternative notation: $P(A, B)$

- Conditional probability: $P(A|B)$
  - The probability for the occurrence of event A under the condition that event B was previously observed.
  - If one assumes event B, the probability of observing A is $P(A|B)$
  - Hence: It is not a logical condition for A

Reminder: Probability theory

- For arbitrary events A and B in combination with $P(B) > 0$ holds:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- By transforming the formula, we derive the multiplication axiom:

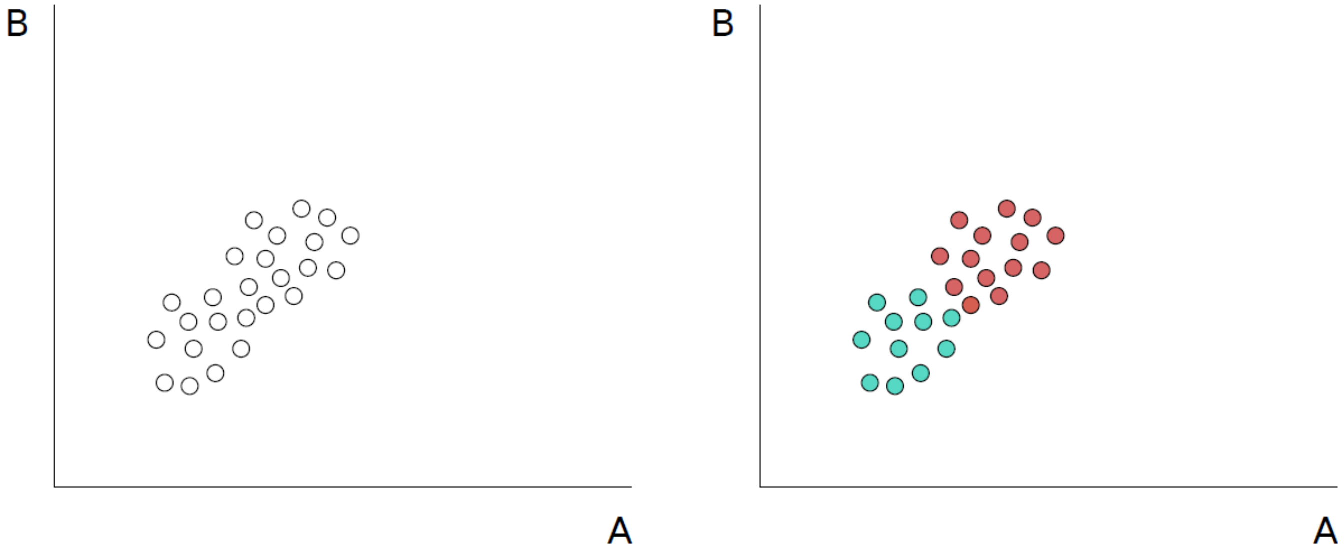
$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

- Independence: If A and B are independent of each other, then:

$$P(A|B) = P(A)$$
$$P(A \cap B) = P(A)$$

Naïve Bayes Classifier (4)

Christian-Albrechts-Universität zu Kiel



Conditional independent:

- Given C, A and B are conditionally independent if holds:
  $P(A, B|C) = P(A|C)\, P(B|)$
- Note: Conditional independence does not imply independence!

Naïve Bayes Classifier (5)

Bayes‘ theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A)$: A-priori probability of event A
- $P(B|A)$: Probability of event B given the occurrence of A
  → Also known as ‘likelihood‘
- $P(A|B)$: A-posteriori probability of event A
- $P(B)$: Evidence
- Usage: Intuition suggests that Bayes‘ theorem allows the inversion of conclusions:
  - Determination of $P(Event|Cause)$ is often easy
  - But usually required: $P(Cause|Event)$
  - Hence: ‘Exchange‘ of arguments

Bayes‘ theorem

- For countable many events $A_i (i = 1, \dots, N)$, the Bayes‘ theorem can be extended to:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_{i=1,\dots,N} P(B|A_i)P(A_i)}$$

- Whereas the relation

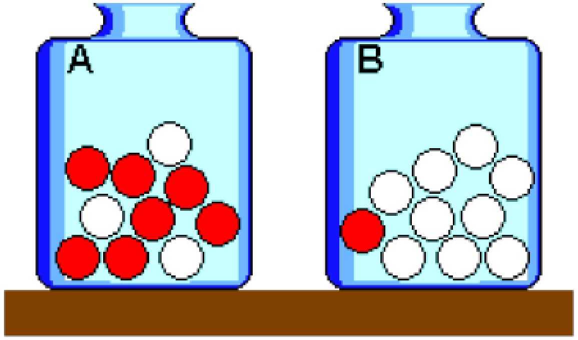
$$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + \cdots$$
$$= P(B|A_1)P(A_i) + P(B|A_2)P(A_2) + \cdots$$

is denoted as the law of the total probability.

Christian-Albrechts-Universität zu Kiel

Example:

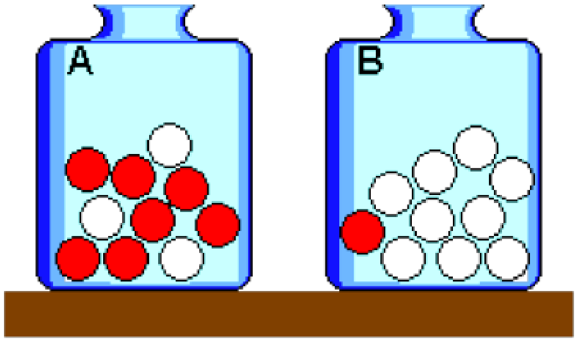
- A ball is randomly drawn from an urn (i.e., a priori uniformly distributed, either A or B)
- Urn contains red (R) and white (W) balls
- One may ask oneself what the probability is for having drawn a red ball (R) from urn A: $P(A|R)$



Source: de.wikipedia.org

Example

- $P(A) = P(B) = \frac{1}{2}$
- $P(R|A) = \frac{7}{10}$ (There are 7 red balls in urn A and 3 white ones)
- $P(R|B) = \frac{1}{10}$ (There is just one white ball in urn B)
- $P(R) = P(R|A)P(A) + P(R|B)P(B)$
$$= \frac{7}{10} \cdot \frac{1}{2} + \frac{1}{10} \cdot \frac{1}{2} = \frac{2}{5}$$
- This is the total probability



Source: de.wikipedia.org

Naïve Bayes Classifier (9)

Application of Bayes‘ theorem to classification

$$P(C|x_1, \dots, x_D) = \frac{P(x_1, \dots, x_D|C)P(C)}{P(x_1, \dots, x_D)}$$

- Likelihood $P(x_1, \dots, x_D|C)$ and class-a-priori probability C
  - In general: Determinable from training data (counting, calculate ratios)
  - Corresponds to maximum likelihood estimators of the parameters
- Evidence of the occurrence of the sample $(x_1, \dots, x_D)$ as a normalisation factor
  - Actually: Approximately derivable from the training data
  - But: Not relevant for the classification decision
  - Reason: Independent from C – hence constant for all classes
  - Absolute value of $P(C|x_1, \dots, x_D)$ not important as well – focus on the inter-class difference!
  - Assignment of sample to the class with maximum value.

Naïve Bayes‘ Classification

- So far: no naive assumption/restriction introduced
  - Fundamental mathematical/statistical foundation
  - Why then the name?
- Calculation of $P(x_1, \dots, x_D | C)$
  - Number of free parameters $O(K^D \cdot C)$
  - Where K is the average number of distinct feature values of a feature
  - In typical realistic applications: combinatorial explosion
- Hence: Naive assumption of conditional independence of the features of a class:

$$P(x_1, \dots, x_D | C) = \prod_{i=1}^{D} P(x_i | C)$$

- Only $O(K^D \cdot C)$ parameters left to determine.

Example:

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Calculation of the probabilities:

- A priori possibility
  - $P(Play = Yes) =?$
  - $P(Play = No) =?$
- For samples:
  - $P(Outlook = Sunny \mid Play = Yes)$
  - $P(Outlook = Rainy \mid Play = Yes)$
  - $P(Outlook = Sunny \mid Play = No)$
  - $P(Outlook = Rainy \mid Play = No)$

Naïve Bayes Classifier (12)

Example

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

A new day starts with an “Event“…

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Example

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

A new day starts with an “Event“…

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

Example

- Event E (fix values for 4 features):

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$P(yes|E) = \frac{P(E|yes)P(yes)}{P(E)}$$

$$= P(Outlook = sunny|yes)\,P(temperature = cool|yes)$$

$$P(humidity = high|yes)\,P(windy = true|yes)\frac{P(yes)}{P(E)}$$

$$= \frac{\frac{2}{9}\cdot\frac{3}{9}\cdot\frac{3}{9}\cdot\frac{3}{9}\cdot\frac{9}{14}}{P(E)}$$

- Remark: In comparison to $P(no|E)$, $P(E)$ does not necessarily have to be calculated.

Questions and answers:

- What shall we do if a feature value does not appear for every class (i.e., the probability would vanish)?
  - Addition of a constant value $\alpha > 0$ (cf. Laplace Smoothing)
  - In general: for a categorical dimension X with K possible distinct feature values $1, \dots, K$ and N observations holds:

$$P_{Lap}(X = i) = \frac{|X = i| + \alpha}{N + K \cdot \alpha}$$

  - with $k \in 1, \dots, K$
- How shall we treat missing values
  - Might be already solved due to pre-processing
  - Otherwise: Feature will not be considered for the calculation of the dependent probability

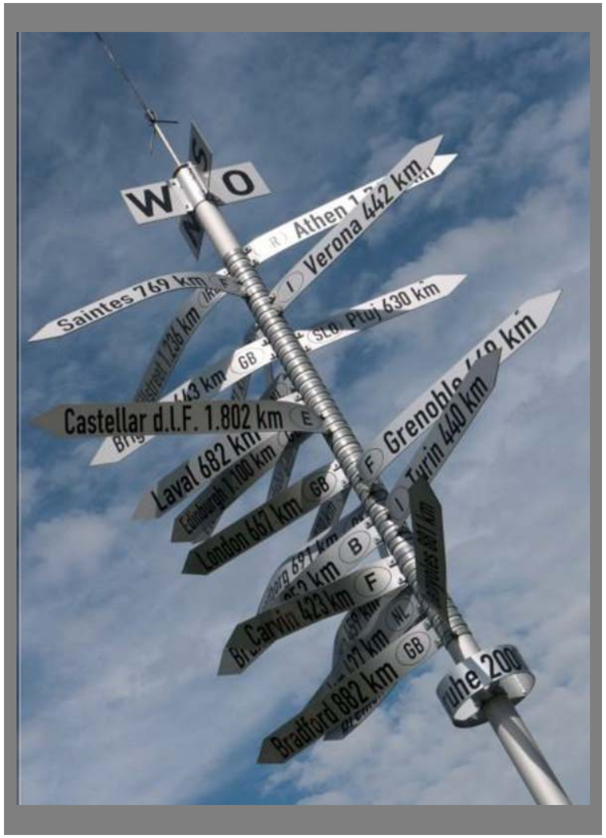
Questions and answers

- Why does the Naïve Bayes‘ Classification perform unexpectedly well even if the assumptions are not fulfilled?
  - The classification does not require a good estimator of the probabilities, because the event with maximum probability will be assigned to the correct class.
  - Real application: Spam filtering
- Hint for the implementation
  - Multiple multiplications with probability values (i.e. values below 1) results in a decrease of values below the available numerical precision
  - Solution: Logarithmic expressions → Product becomes a sum
- How to deal with numerical features?
  - Discretisation: partitioning into bins
  - Assumption of a normal distribution: For each class, calculate the mean μ_c and the variance σ_c^2

Agenda

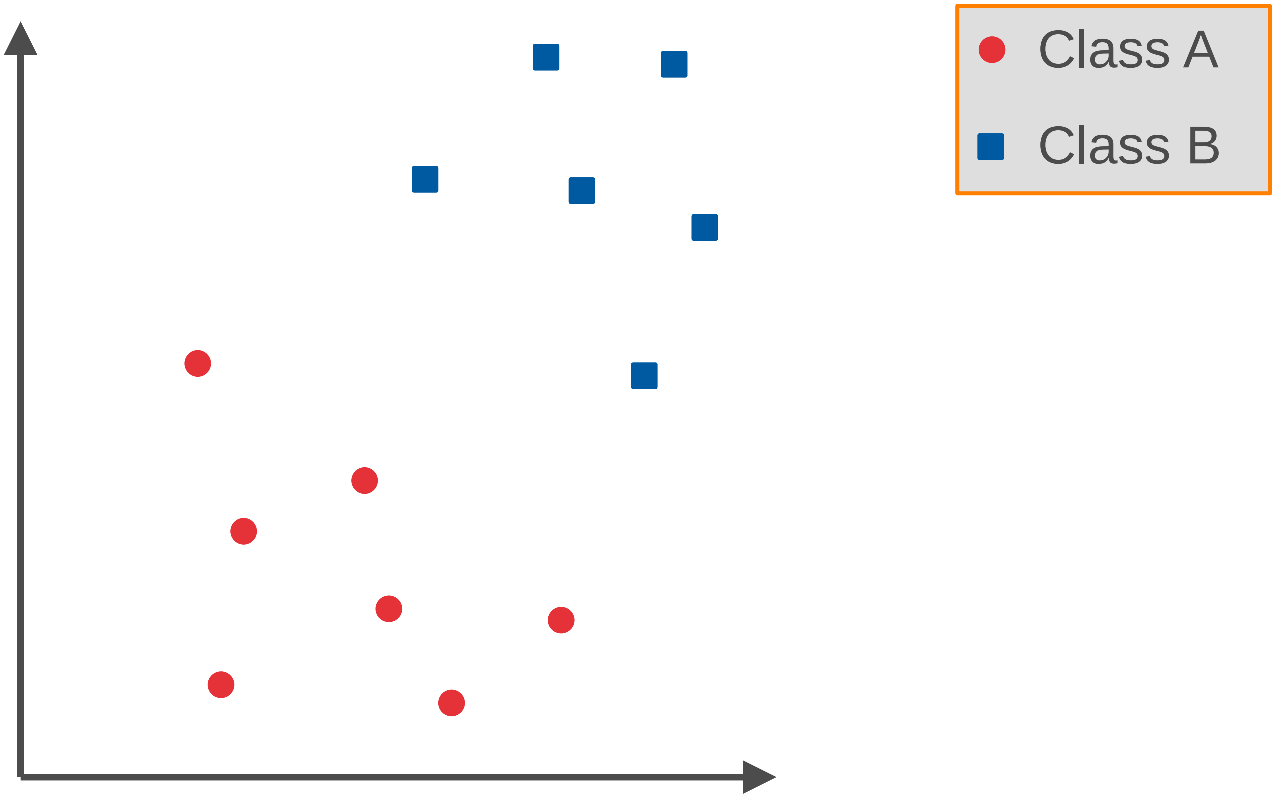
Christian-Albrechts-Universität zu Kiel

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- **Support Vector Machines**
- Conclusion and further readings



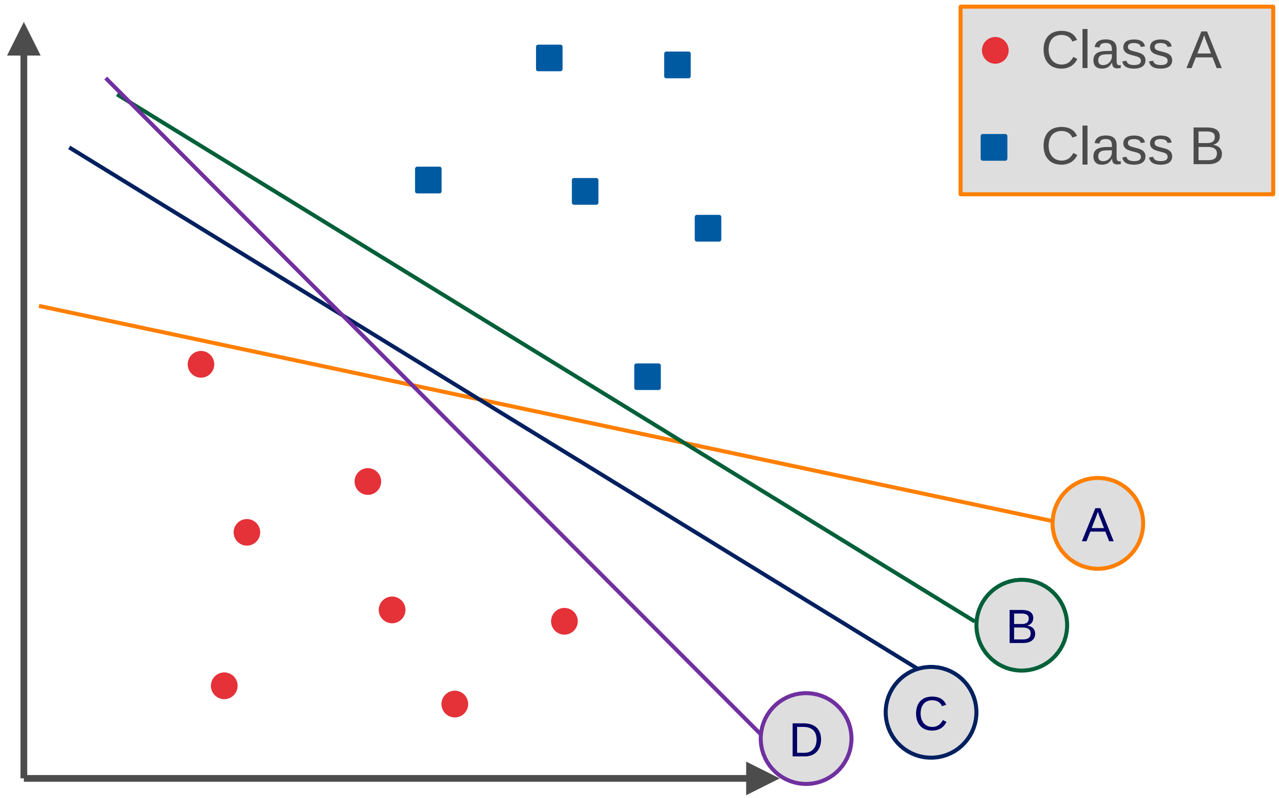
Decision boundary

- Focus: linear separable in a two-class problem



Christian-Albrechts-Universität zu Kiel

Decision boundary

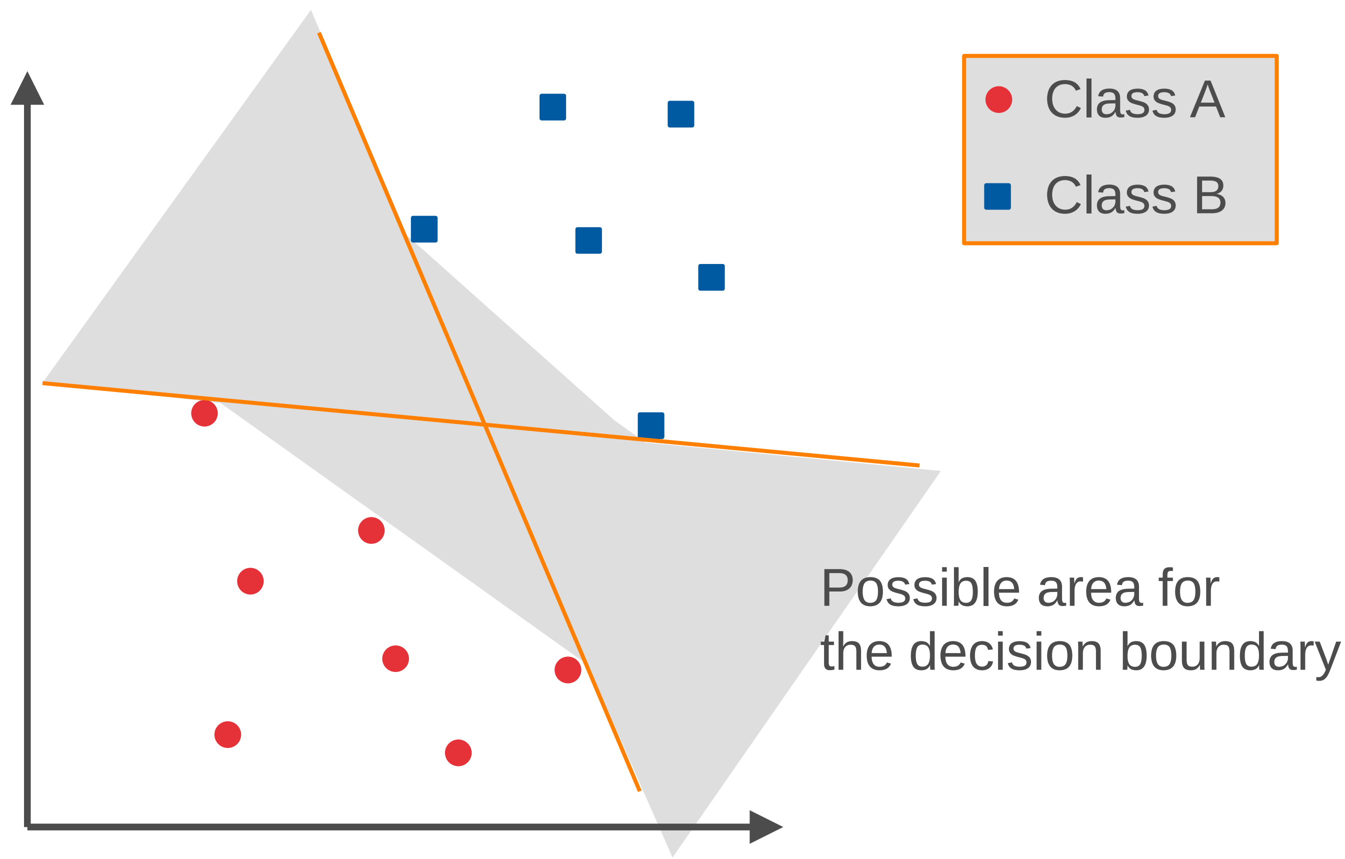


Question:

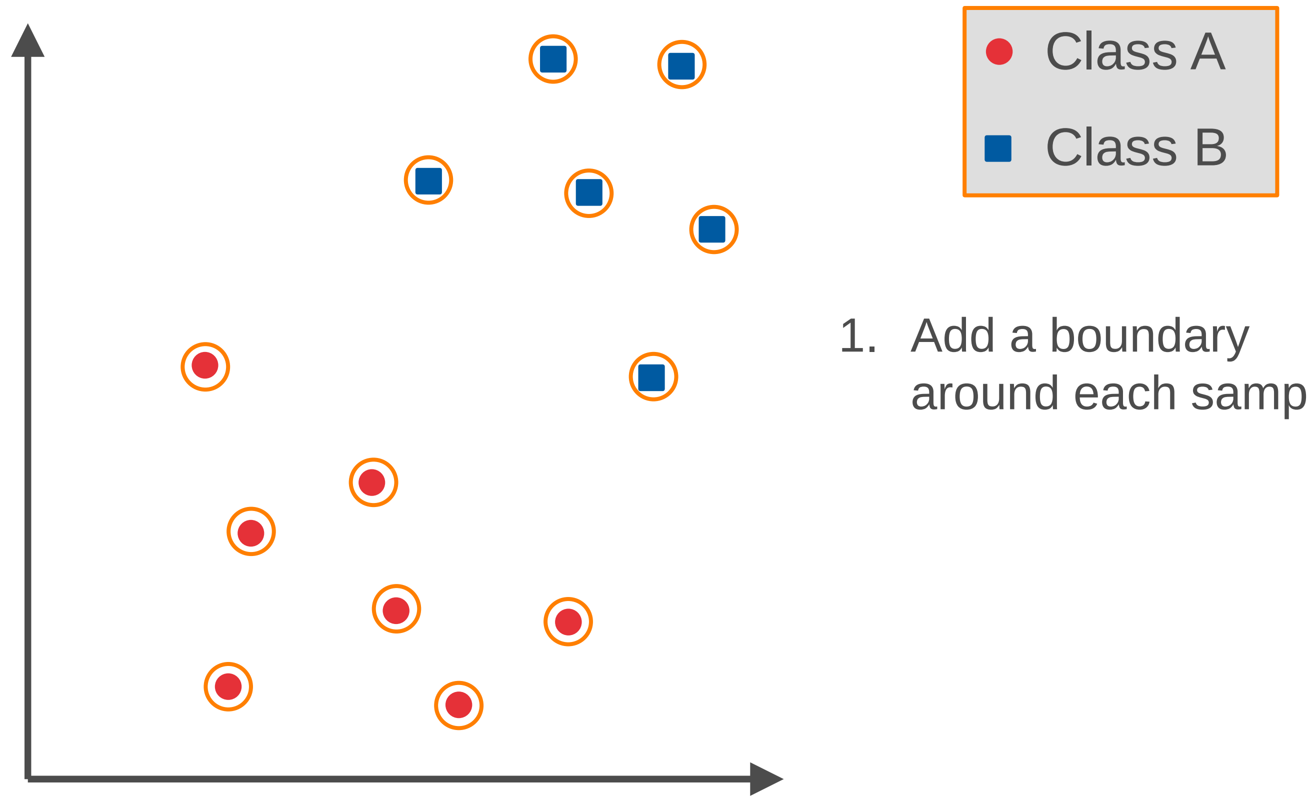
- Which of the four possible decision boundaries is 'better'?

Christian-Albrechts-Universität zu Kiel

Decision boundary

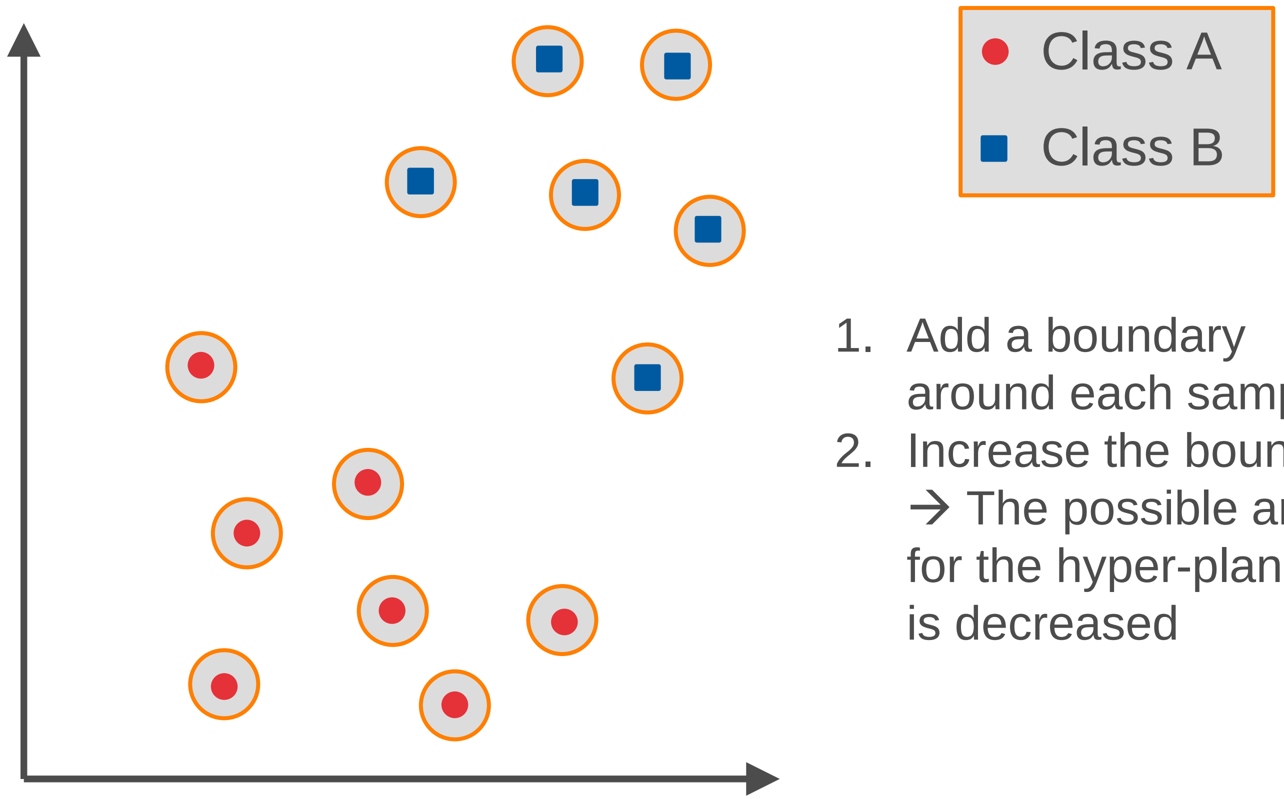


Process: Find the optimal decision boundary



1. Add a boundary around each sample

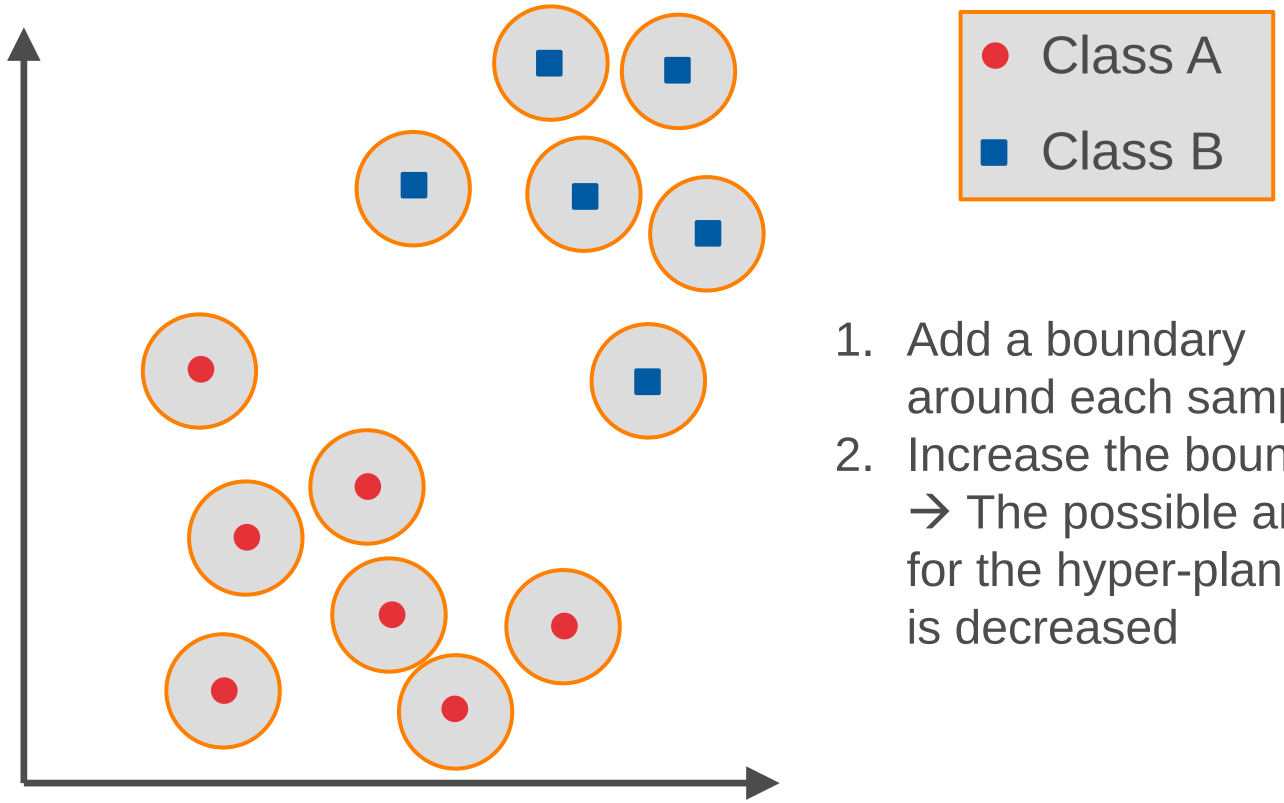
Process: Find the optimal decision boundary



1. Add a boundary around each sample
2. Increase the boundaries
   → The possible area for the hyper-plane is decreased

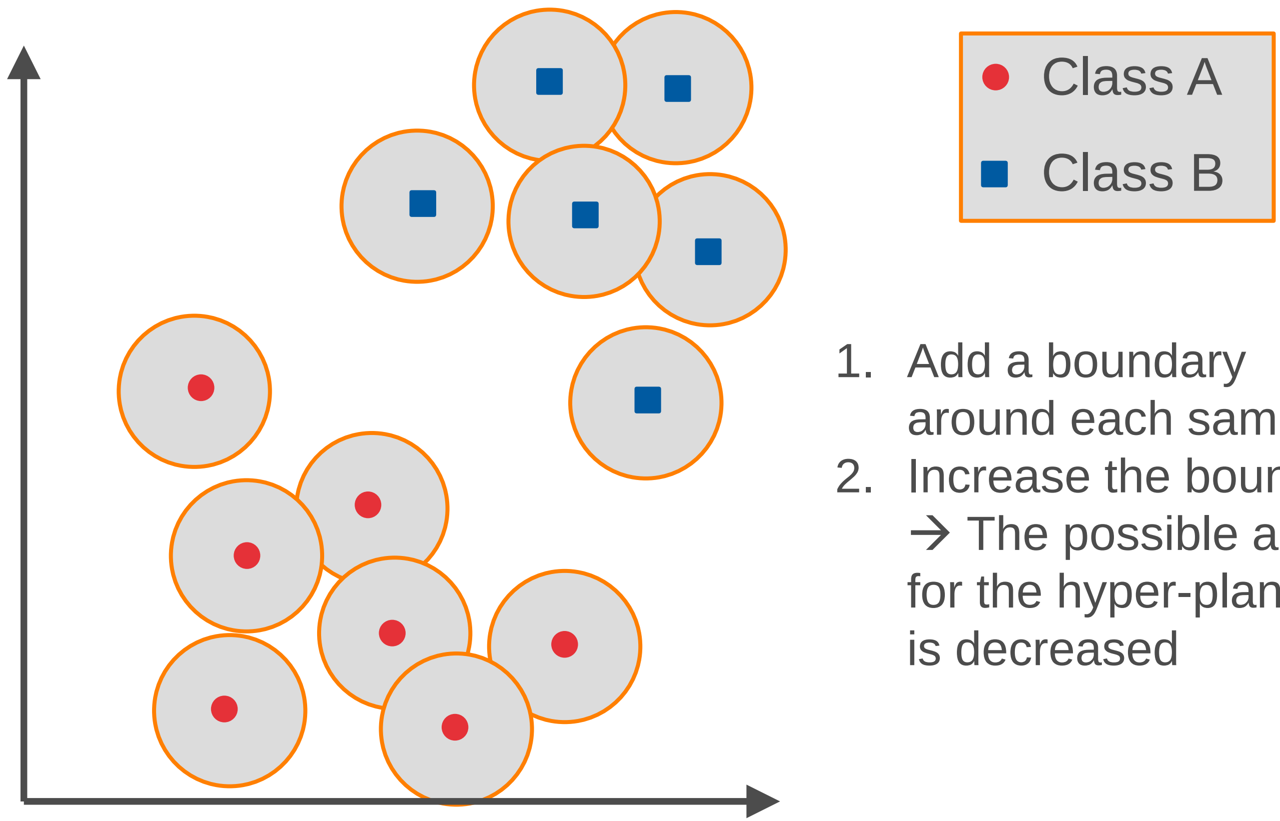
Christian-Albrechts-Universität zu Kiel

Process: Find the optimal decision boundary



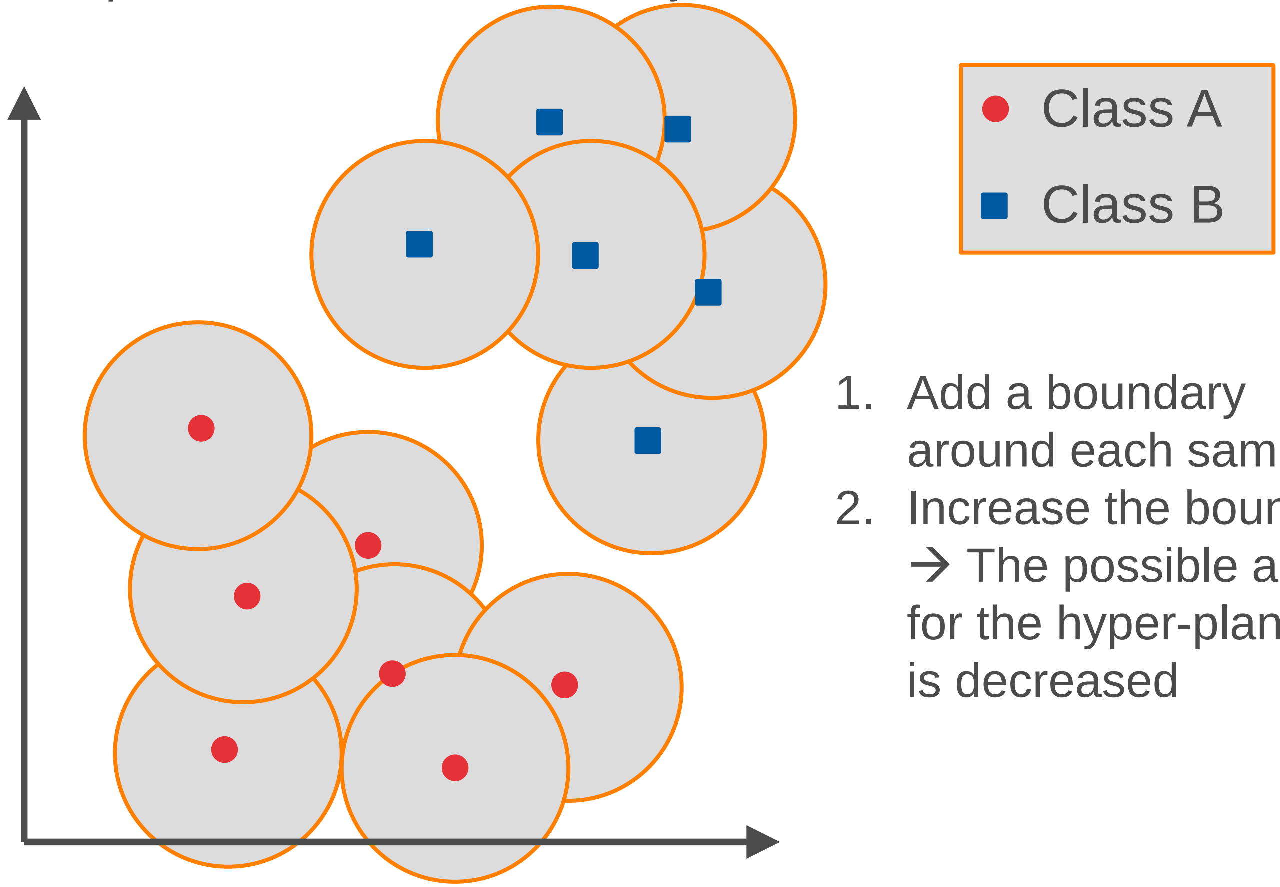
1. Add a boundary around each sample
2. Increase the boundaries
   → The possible area for the hyper-plane is decreased

Process: Find the optimal decision boundary

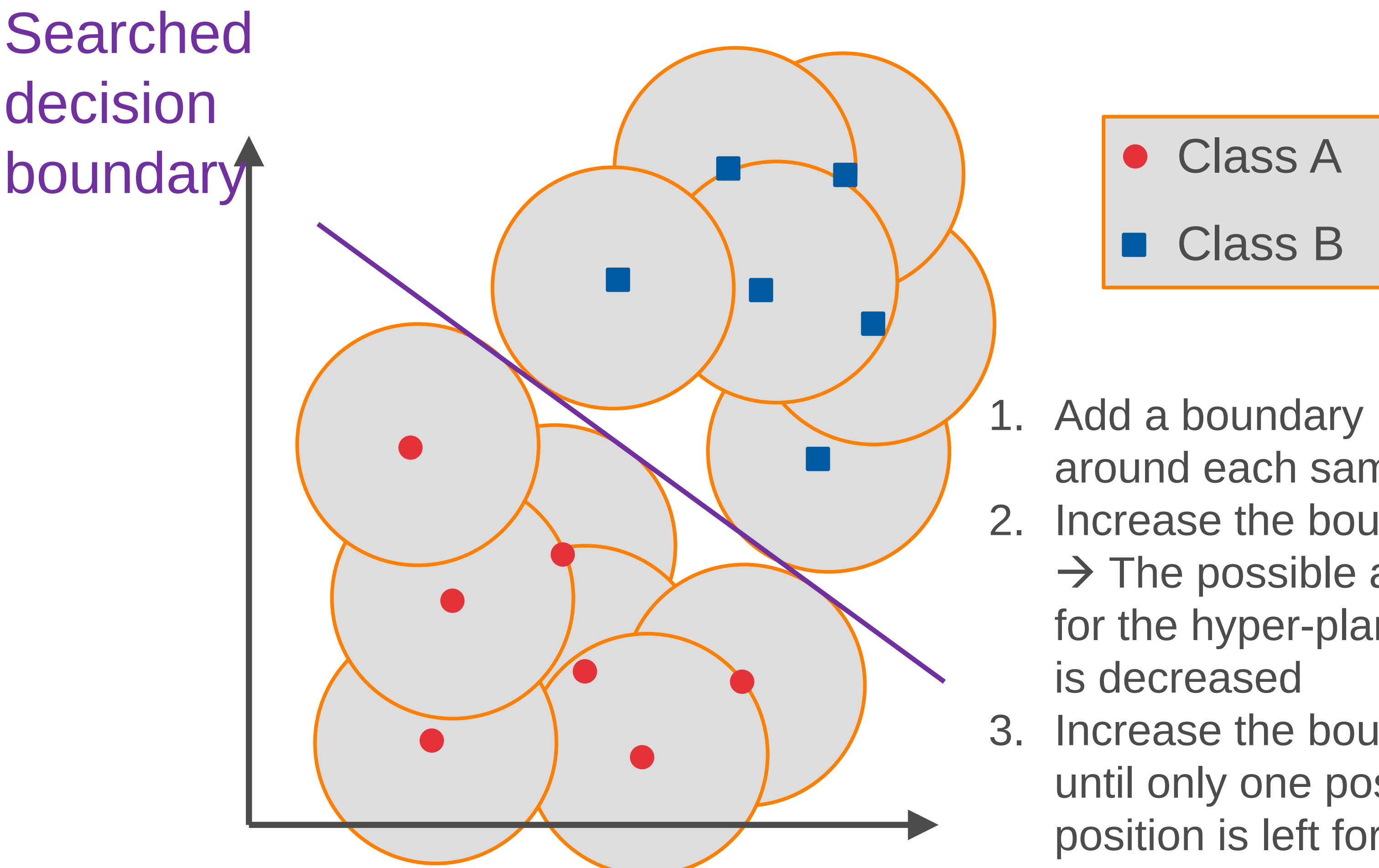


1. Add a boundary around each sample
2. Increase the boundaries
   → The possible area for the hyper-plane is decreased

Process: Find the optimal decision boundary



1. Add a boundary around each sample
2. Increase the boundaries
   → The possible area for the hyper-plane is decreased



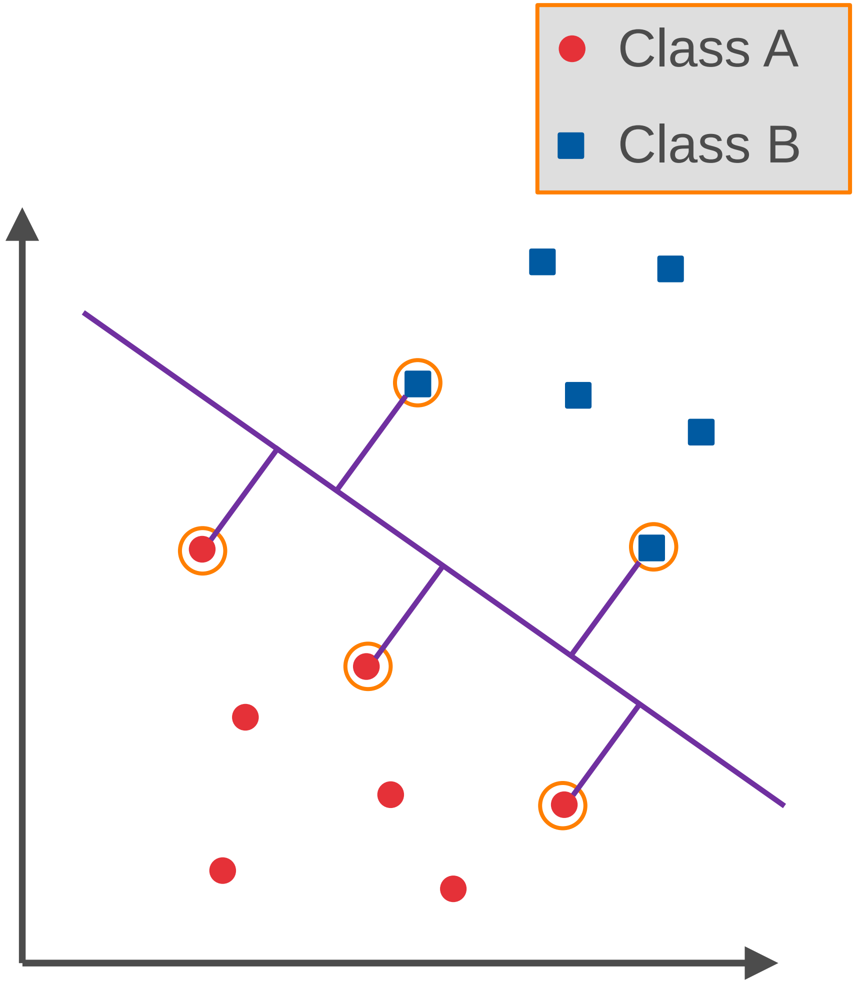
1. Add a boundary around each sample
2. Increase the boundaries
   → The possible area for the hyper-plane is decreased
3. Increase the boundaries until only one possible position is left for the hyper-plane!

Support Vector Machine (SVM)

- Given: Training data with class information
$$\{(\boldsymbol{x_i}, y_i) | i = 1, \dots, m; y_i \in \{-1,1\}\}$$
- Each sample is represented by a vector in the input or vector space.
- Task of the SVM:
  - Fit a hyper-plane into this space
  - Hyper-plane serves as a separation plane and partitions the search space into two classes.
  - Maximisation of the distance of those vectors that are closest to the hyper-plane.
- Goal: Generalisation
  - Maximally large, empty margin
  - Should allow for reliable classification of unknown samples later on.

Definition of the hyperplane

- The optimal (canonical) hyper-plane is the maximum distance to the nearest points of both classes
- Assumption first of all: The samples can be separated linearly.

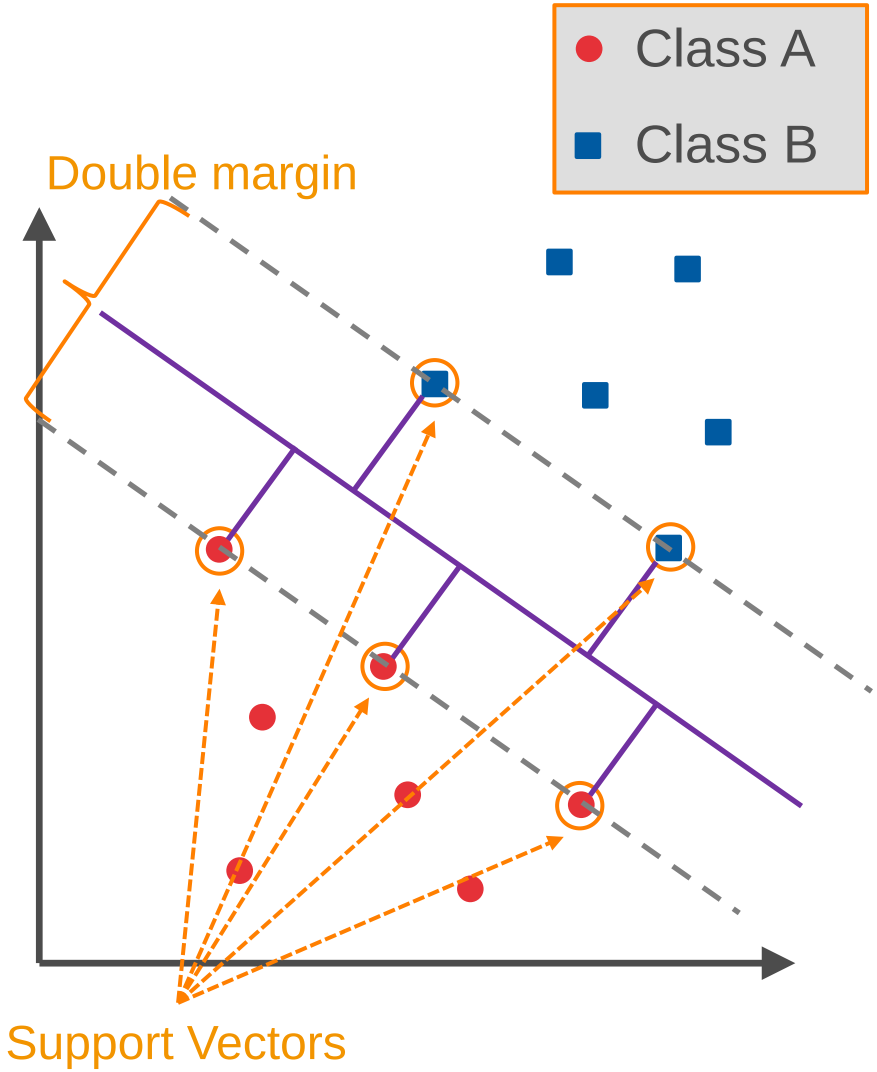


Support Vectors

- Those data points (samples) which have the smallest distance to the separating hyperplane.

Margin

- Double distance of the support vectors to the hyperplane



Christian-Albrechts-Universität zu Kiel

The formal structure of an SVM

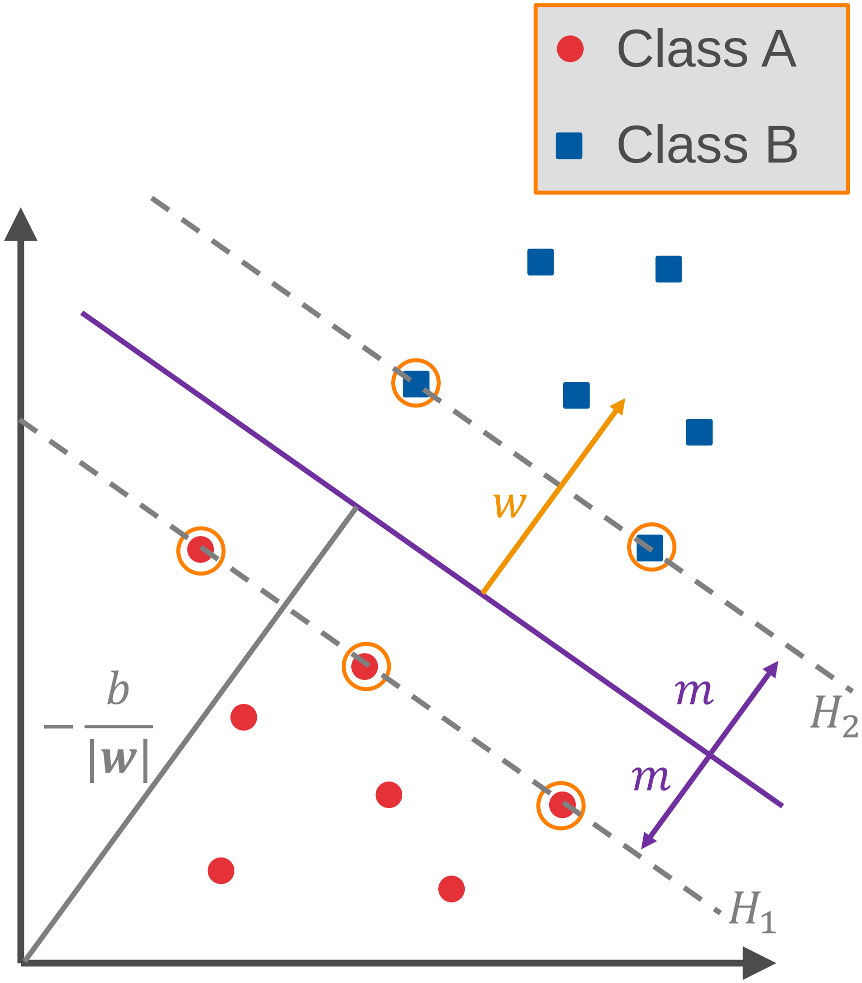
- Normal vector $\boldsymbol{w}$ describes a straight line through the coordinate origin.
- Hyper-planes run perpendicular to this straight line.
- Each hyper-plane intersects the line at a certain distance b from the origin (measured in the opposite direction to w).
- This distance is called bias.
- Normal vector and the bias clearly define a hyper-plane, for the samples belonging to it the following linear expression is 0:
  $$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$$
- Samples beyond the hyper-plane lead to positive (on the side to which w points) or negative (on the other side) values, i.e. they are not equal to 0.
- The aim is to use training data to calculate the parameters w and b of this "best"

Christian-Albrechts-Universität zu Kiel

Class assignment

- Class C_1
  $\boldsymbol{w}^T \cdot x_n + b \geq +1$
  if $y_n = +1$
- Class C_2
  $\boldsymbol{w}^T \cdot x_n + b \leq -1$
  if $y_n = -1$

Geometric distance



Class assignment and distances

- The algebraic distance from x to the straight line is defined by:

$$y = \boldsymbol{w}^T \cdot \boldsymbol{x} + b$$

- The geometric distance from x to the straight line is defined by:

$$\frac{|y|}{||w||}$$

Classification

- Determined by $\mathrm{sign}(y)$
- So: the sign of y.

The size of the margin

- For x_n, which lie on the hyper-planes H_1 and H_2 (these are the support vectors!), the algebraic distance (i.e., it is required):
  - $w^T \cdot x_n + b = +1$ if $y_n = +1$
    with geometric distance $\frac{1-b}{||w||}$ to the origin
  - $w^T \cdot x_n + b = -1$ if $y_n = -1$
    with geometric distance $\frac{-1-b}{||w||}$ to the origin

- Thus, for the size of the margin:

$$2m = \frac{1-b}{||w||} - \frac{-1-b}{||w||} = \frac{2}{||w||} \Leftrightarrow m = \frac{1}{||w||}$$

Christian-Albrechts-Universität zu Kiel

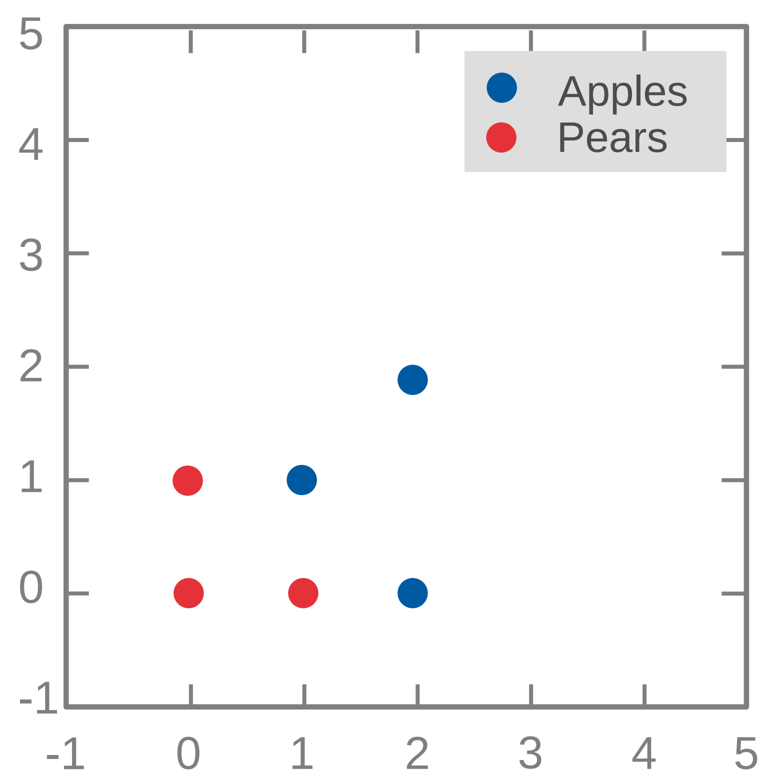
Example: Farm

- Fruit meadow with apples and pears
- Where is the optimal line for a fence?
- Last year's harvest:

 : [(1,1), (2,2), (2,0)]

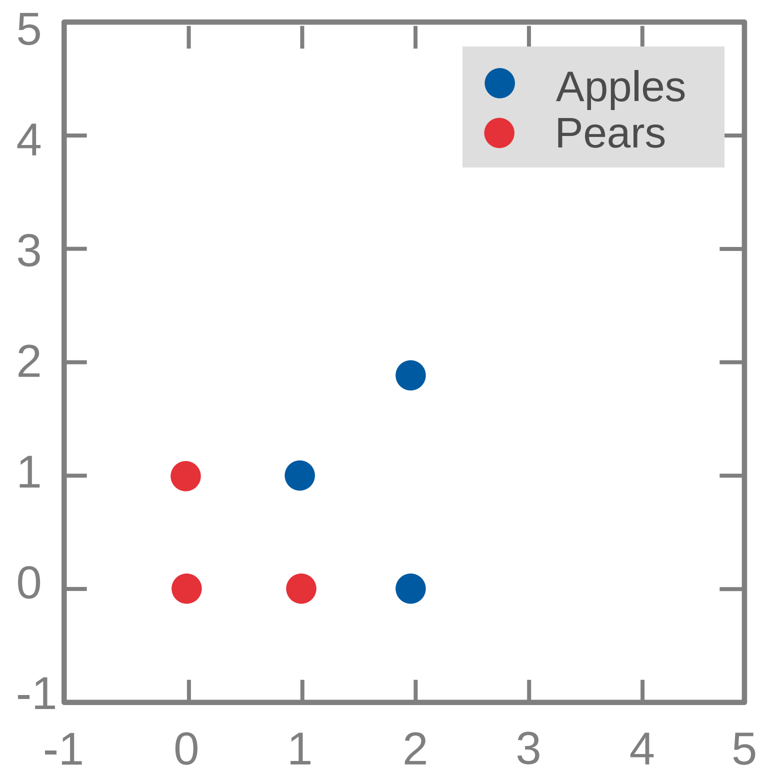
 : [(0,0), (1,0), (0,1)]

Visualisation



Question:

- Are the data points (apples and pears) linearly separable according to their positions?



Answer:

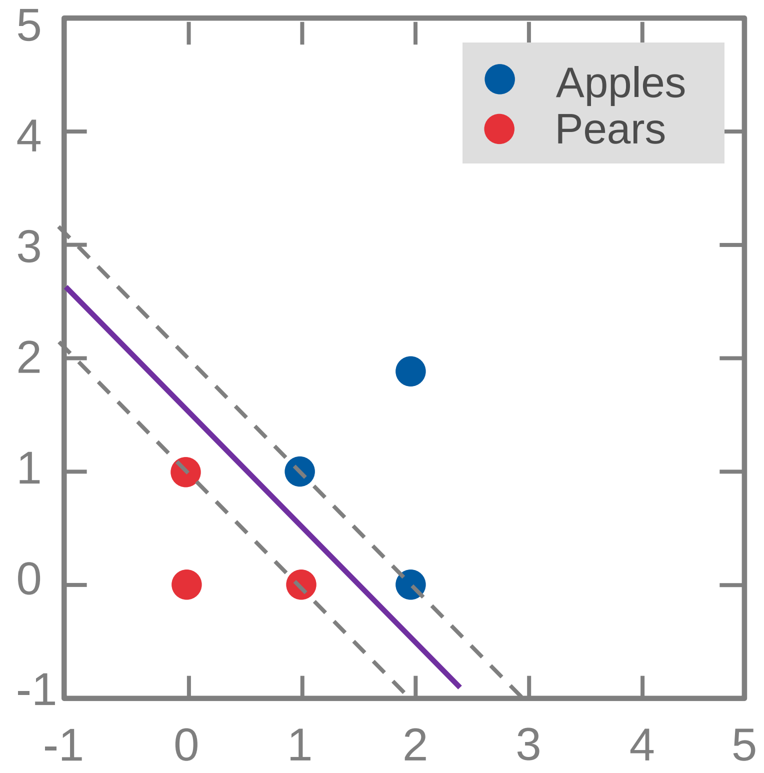
Solution

Support Vector Machines (19)

Christian-Albrechts-Universität zu Kiel

Assumption:

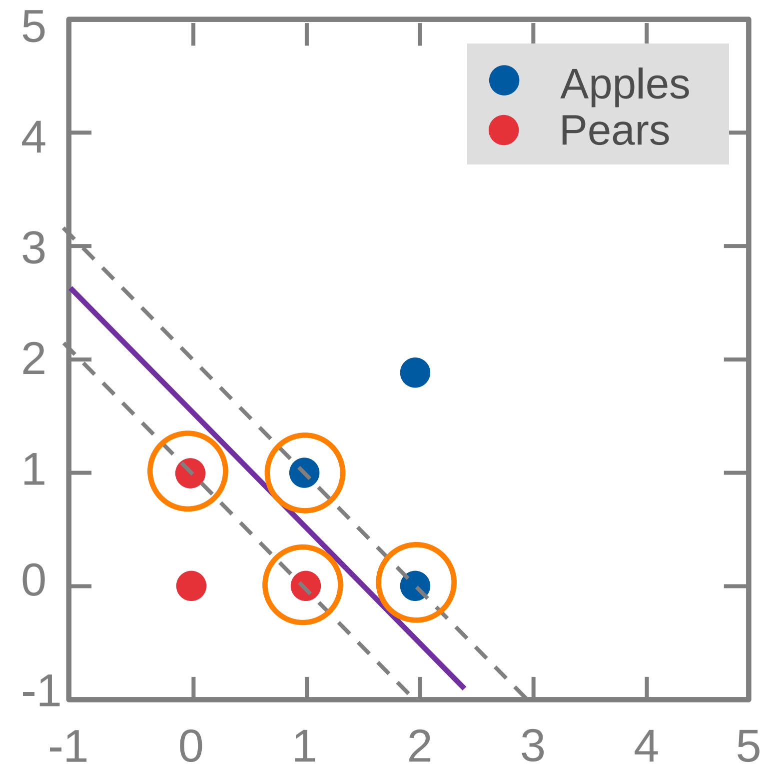
- We know the support vectors.
- For example: Drawing solution



Determine the support vectors:

- Apples
  $supportVectors_{Apple} = [(1{,}1), (2{,}0)]$
- Pears
  $supportVectors_{Pears} = [(1{,}0), (0{,}1)]$
- The equation of a polynomial of degree one in an SVM is based on the following requirements:
  $w_1 x + w_2 y + w_3 = 0$
- Define the equation system:

$$
\begin{aligned}
apple_1&: 1w_1 + 1w_2 + w_3 = +1\\
apple_2&: 0w_1 + 2w_2 + w_3 = +1\\
pear_1&: 1w_1 + 0w_2 + w_3 = -1\\
pear_2&: 0w_1 + 1w_2 + w_3 = -1
\end{aligned}
$$



Christian-Albrechts-Universität zu Kiel

Equation system

$$
\begin{aligned}
apple_1&: 1w_1 + 1w_2 + w_3 = +1 \\
apple_2&: 0w_1 + 2w_2 + w_3 = +1 \\
pear_1&: 1w_1 + 0w_2 + w_3 = -1 \\
pear_2&: 0w_1 + 1w_2 + w_3 = -1
\end{aligned}
$$

Solving the equation system:

- $apple_1 - pear_1 \Rightarrow 1w_1 - 1w_1 + 1w_2 - 0w_2 + w_3 - w_3 = 1 - (-1)$
- Results in: $w_2 = 2$
- Insert w_2 in $pear_2$: $0w_1 + 1 * 2 + w_3 = -1$
- Result: $w_3 = -3$
- Insert in $apple_1$: $1w_1 + 1 * 2 + (-3) = 1$
- Result: $w_1 = 2$

Linear equation

- With $w_1 = 2$, $w_2 = 2$ and $w_3 = -3$ follows:

$$2\mathrm{x} + 2\mathrm{y} - 3 = 0$$

- Hence, we can define the vector w of the SVM:

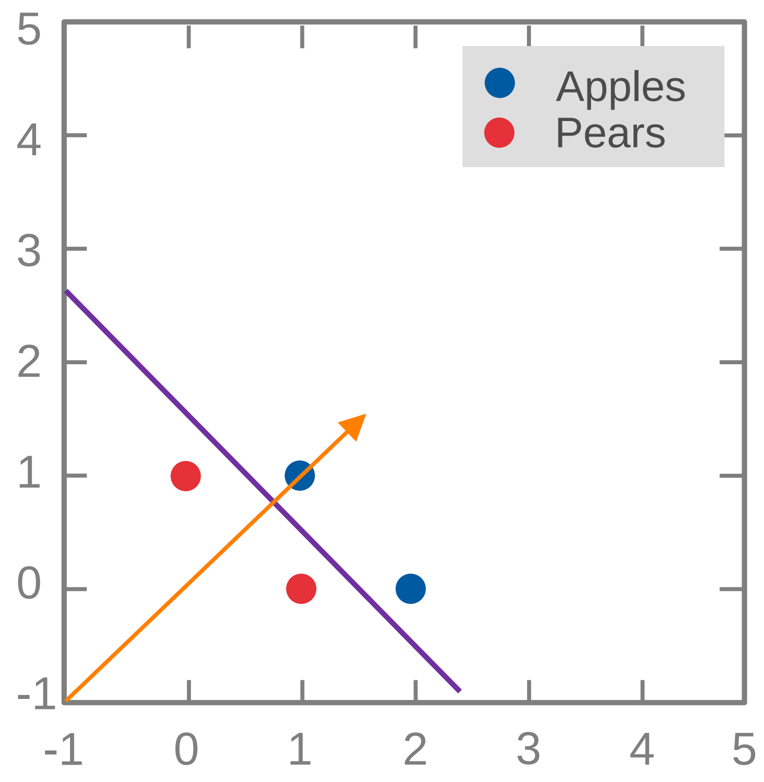
$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Christian-Albrechts-Universität zu Kiel

Illustration of results

- Support vectors
  - Apples
    $supportVectors_{Apples} = [(1,1),(2,0)]$
  - Pears
    $supportVectors_{Pears} = [(1,0),(0,1)]$

- Normal vector
  $\boldsymbol{w} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$



Christian-Albrechts-Universität zu Kiel

How does the SVM solve complex problems?

- Quadratic optimisation problem with linear inequalities as constraints
- Approach: Lagrange method

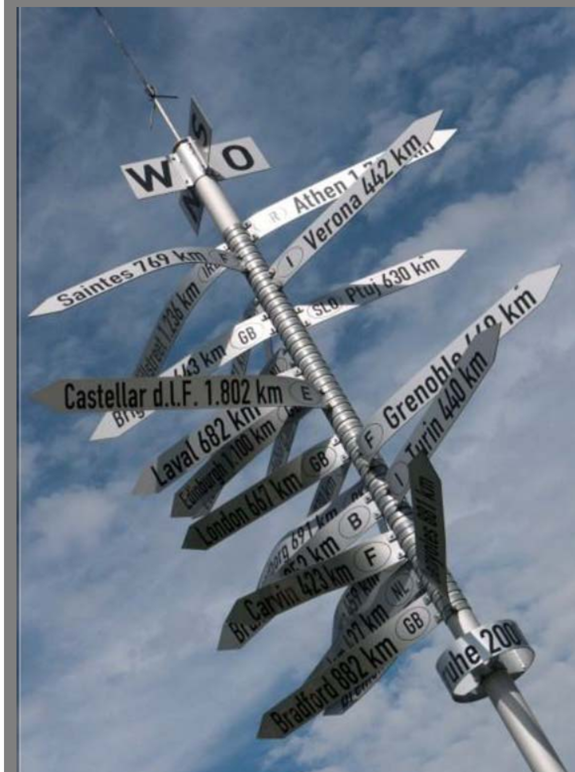
How does the SVM solve non-linear problems?

- Kernel trick
- Transformation of the data by means of a non-linear function into a (mostly) higher-dimensional space

How does the SVM solve multi-class problems?

- Mostly: Reduction of several two-tier problems

- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- **Conclusion and further readings**



Conclusion

Christian-Albrechts-Universität zu Kiel

Summary of the chapter

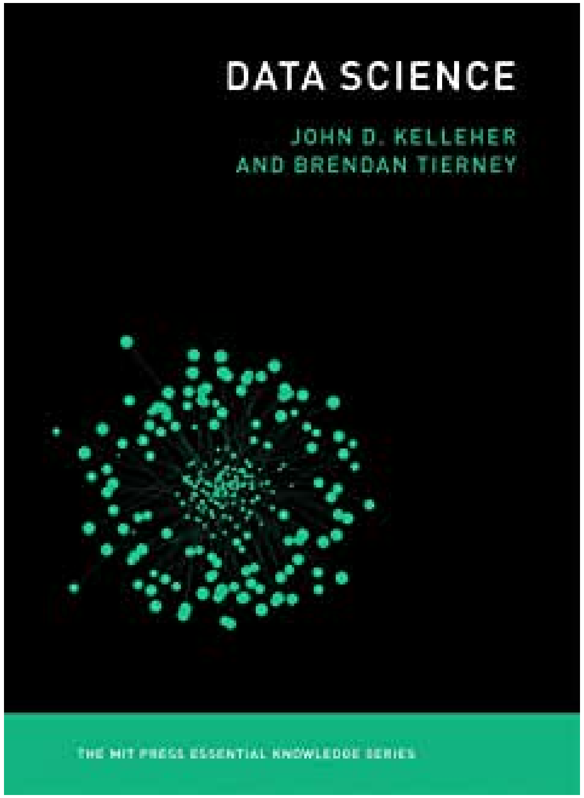
- Introduction to classification
- 1-R Classifier
- k-Nearest Neighbour
- Decision Trees
- Random Forest
- Naïve Bayes Classification
- Support Vector Machines
- Linear severability, Kernel trick

Further readings

Christian-Albrechts-Universität zu Kiel

Data Science

- John D. Kelleher, Brendan Tierney
- MIT Press
- Series “Essential Knowledge“
- ISBN-13 : 978-0262535434
- Edition from 2018



- Any questions…?