Karlsruher Institut für Technologie

Machine Learning 1 – Fundamentals

Decision Trees

Prof. Dr. J. M. Zöllner, M.Sc. Nikolai Polley, M.Sc. Marcus Fechner

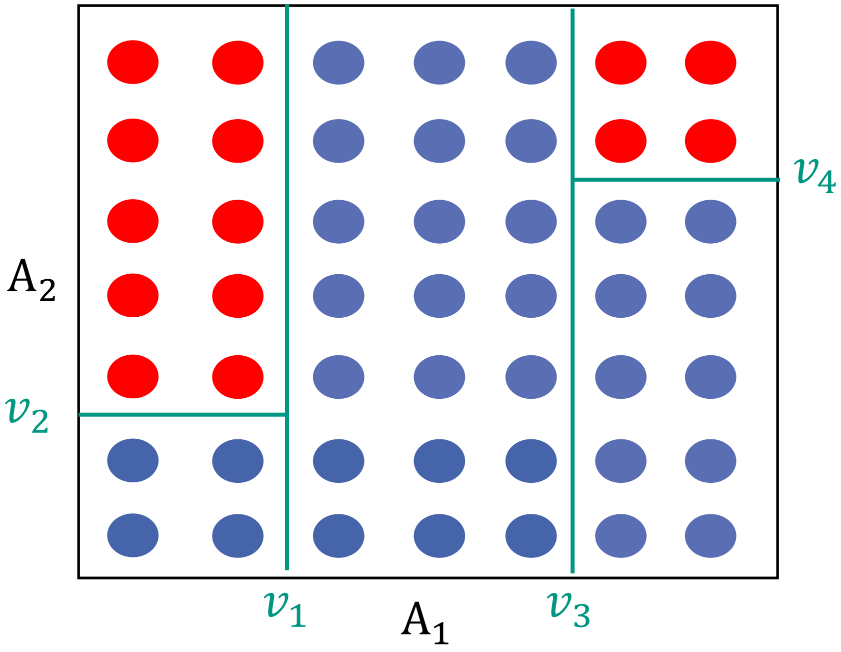
Overview

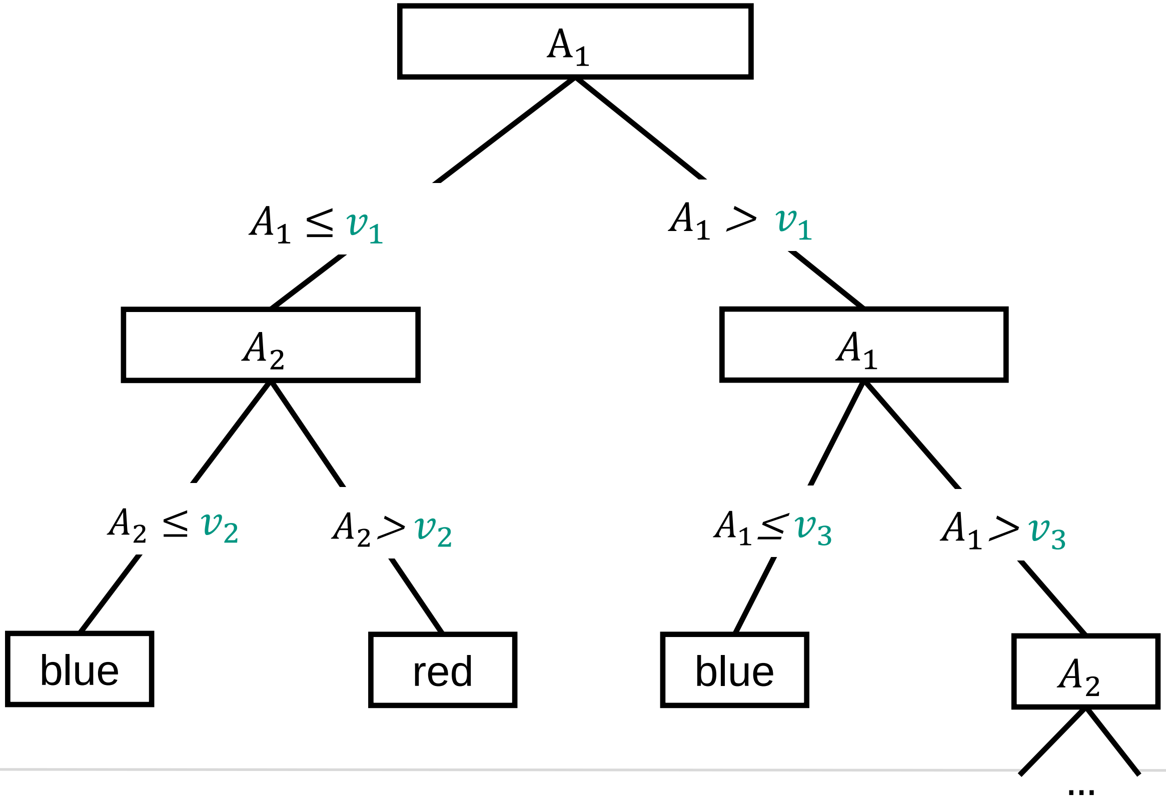
Karlsruher Institut für Technologie

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Motivation

- **Classification**: Partition the space so that all instances in a region have the same class
- Extensible to regression trees

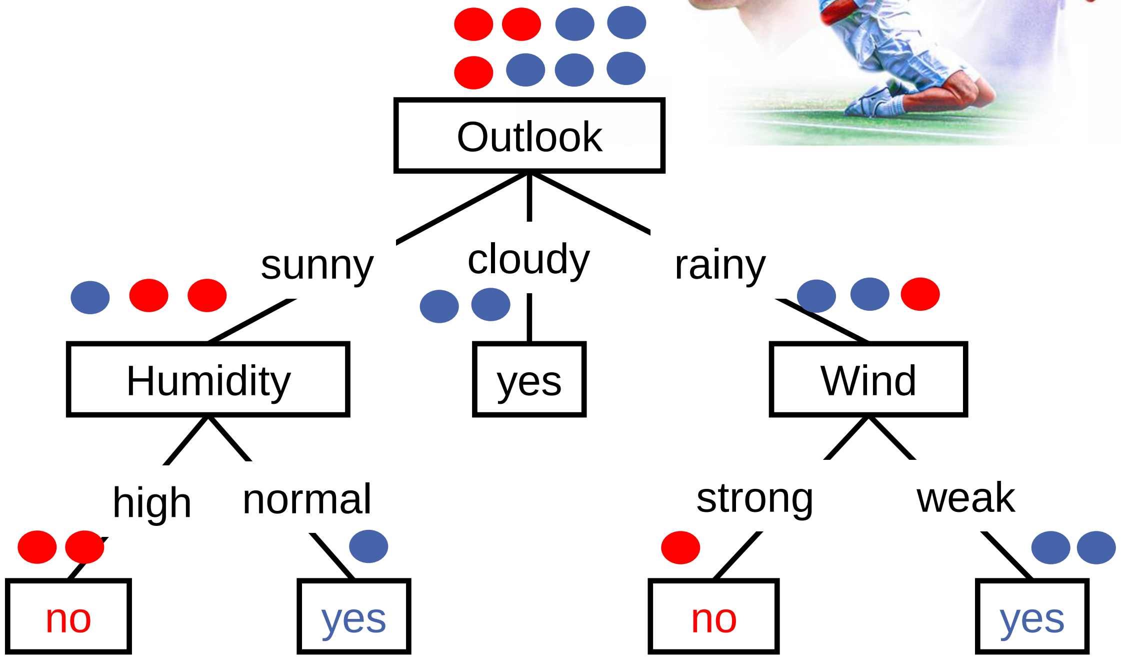




Motivation: Play Tennis

- **Question**: Which days are suitable for Roger Federer to play tennis?

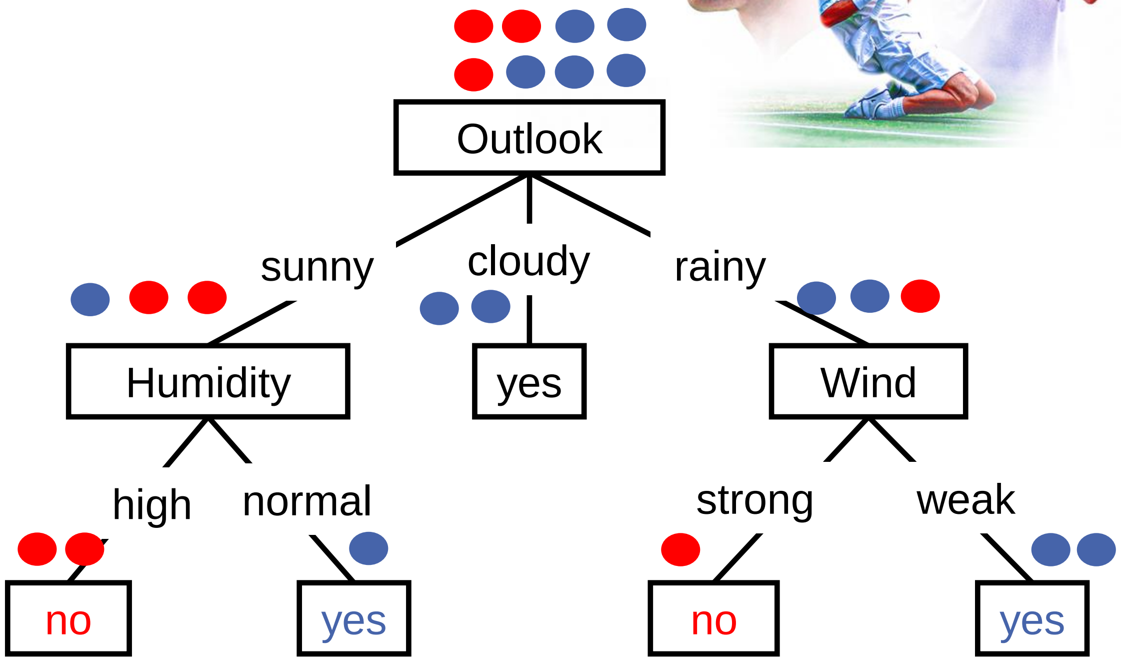
No	Outlook	Temperature	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Warm	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Cold	Normal	Weak	Yes
9	Sunny	Warm	High	Weak	???
10	Rainy	Warm	Normal	Weak	???



Motivation: Play Tennis

- **Question**: Which days are suitable for Roger Federer to play tennis?

No	Outlook	Temperature	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Warm	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Cold	Normal	Weak	Yes
9	Sunny	Warm	High	Weak	**No**
10	Rainy	Warm	Normal	Weak	**Yes**



When are Decision Trees suitable?

Basic Decision Trees:

- Instances are represented as attribute-value pairs (categorical)
- Target function returns discrete output values (e.g. classes)
- Non-parametric: No assumptions about underlying distribution
- Interpretability is possible
- Generally low computing resource requirements for inference
- Simple to learn non-linear problems
- Data doesn‘t need to be normalized and can be in diverse formats

Decision Tree with extensions:

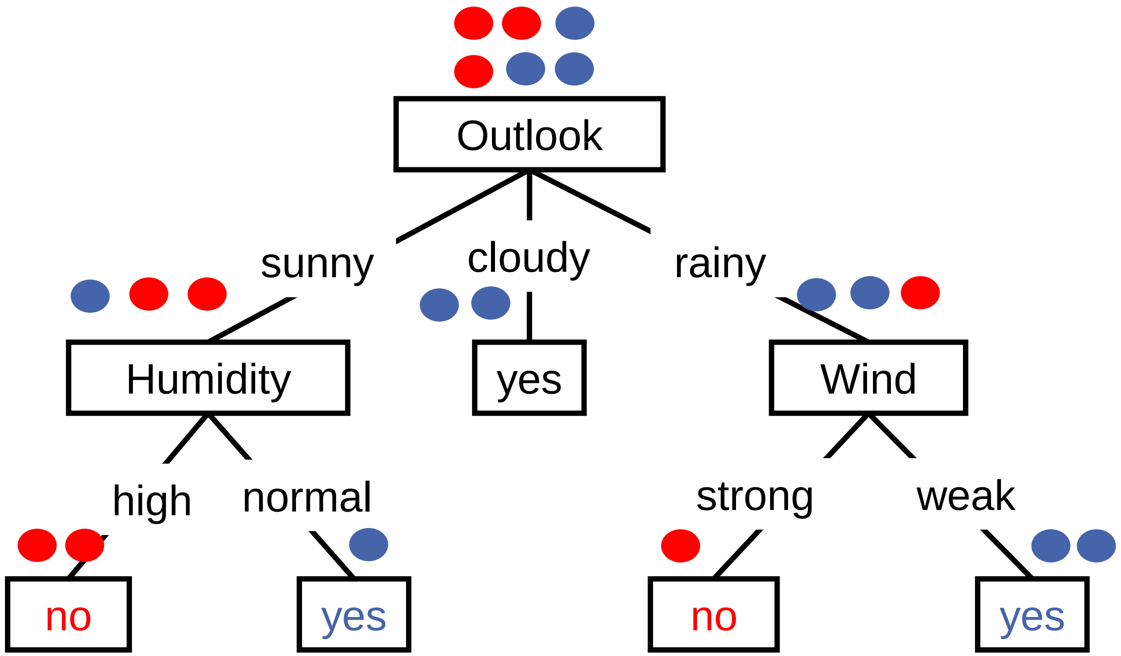
- Applicable for regression (Regression Tree)
- Noisy input data
- Continuous attribute values (features)
- Missing input data

Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Formalization

- **Nodes**: test feature/attribute A for value
  - E.g. Outlook, Humidity, Wind …
- **Root Node:** Top/first node in decision tree
- **Child Node**: Successor of a node
- **Parent Node**: Predecessor of a node
- **Leaf**: Final node containing classification result Y
  - E.g.: PlayTennis?=yes, PlayTennis?=no
- **Branch**: Attribute value v of testing A
  - E.g., for the attribute Outlook: sunny, cloudy, rainy

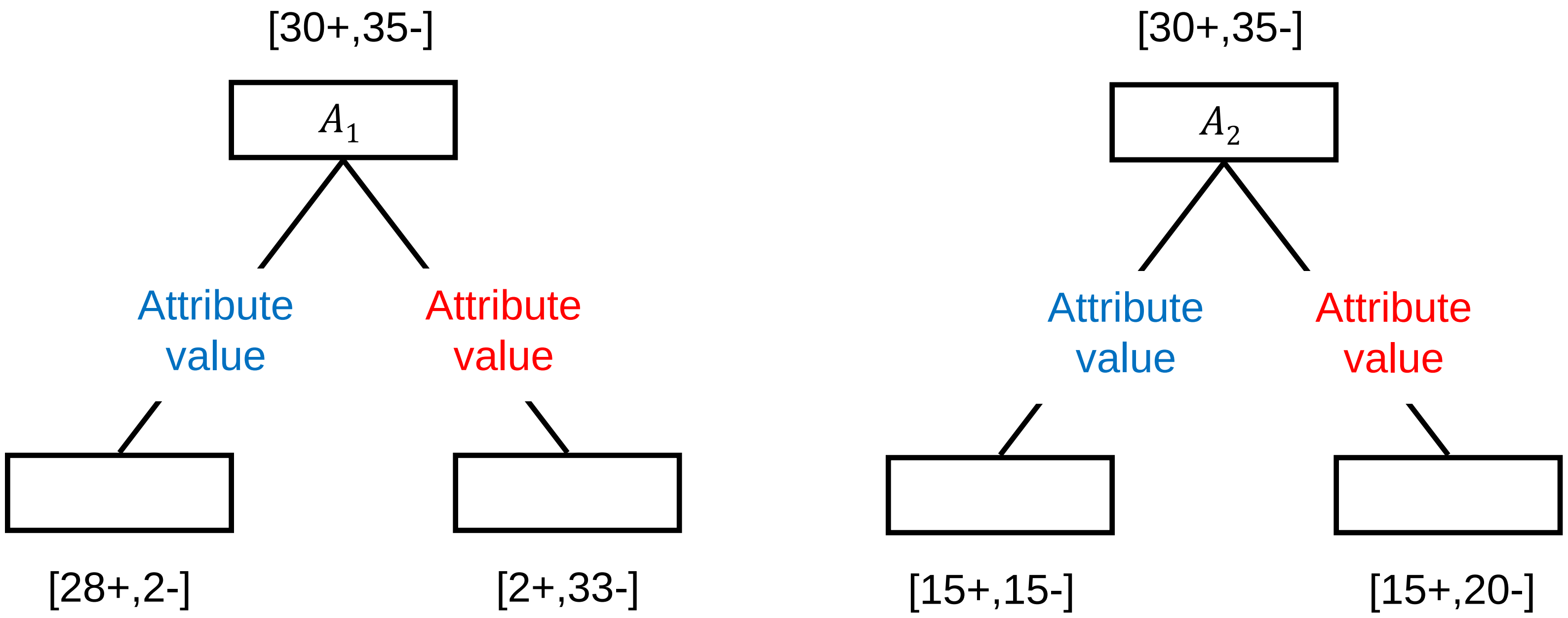


Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Attribute Selection

- **Problem**: How can we measure how well an attribute/feature splits the data?
- **Example**: In the following, is attribut A_1 or A_2 preferrable?
- **Solution**: Metrics to measure impurity of data sets
  - Entropy
  - Gini-Index
  - …



Notation: [number positive examples (+); number negative examples (-)]

Entropy

- **Definition**: *In information theory, the entropy of a random variable Y is the average level of "information", "suprise", or "uncertainty" inherent to the variable‘s possible outcomes y_i*
  随机变量Y的熵是指变量可能结果y_i所固有的“信息”、“惊奇”或“不确定性”的平均水平

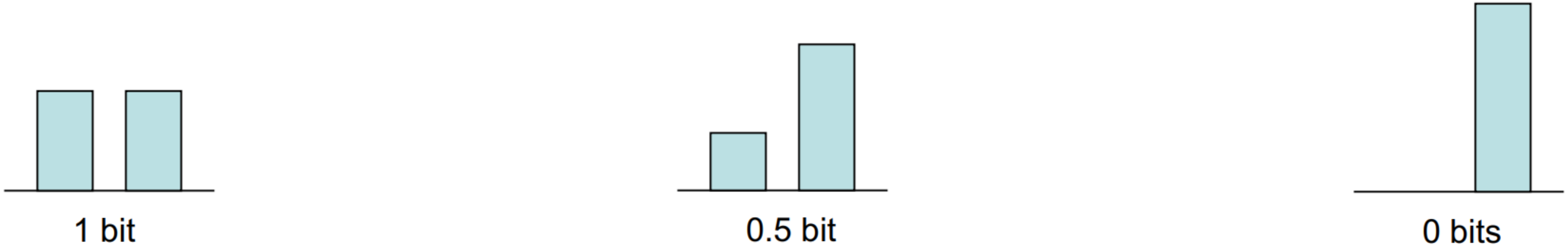
- **For decision trees**: Entropy is a measure of the homogeneity (in terms of class membership) of the current data S

- **Here**: Discrete random variable $Y \sim p(y)$, whereas $p(y) = \frac{|S_y|}{|S|}$ defined by data S and subset S_y of data with class y
  - Entropy for ***K*** **Classes**: $H(S) = -\sum_{i=1}^{K} p(y_i) \log_2 p(y_i)$
  - Entropy for **2 Classes**(⊕, ⊖): $H(S) = -\mathrm{p}_{\oplus} \log_2 \mathrm{p}_{\oplus} - \mathrm{p}_{\ominus} \log_2 \mathrm{p}_{\ominus}$

- **Info**: The entropy, measured with a logarithm of base 2 uses **Bit** as unit

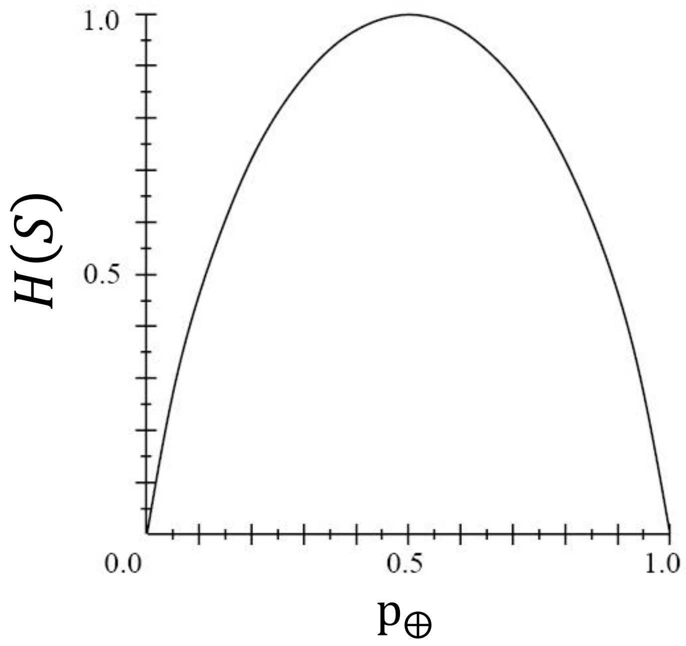
Entropy – Decision Tree

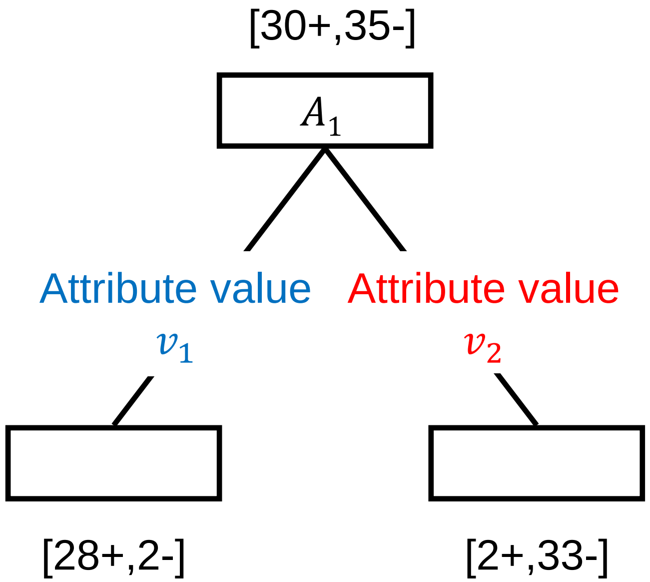
- **High entropy**
  - Y is derived from an almost uniformly distributed probability density
  - Data sampled from this density are difficult to predict
- **Low entropy**
  - Y is derived from a probability density with a high probability for one class, relative to the others
  - Data sampled from this density are highly predictable



Entropy – Decision Tree

- **Objective**: To quickly sort data into their respective classes through the selection of appropriate attributes
  - i.e. successively reduce the entropy as quickly as possible



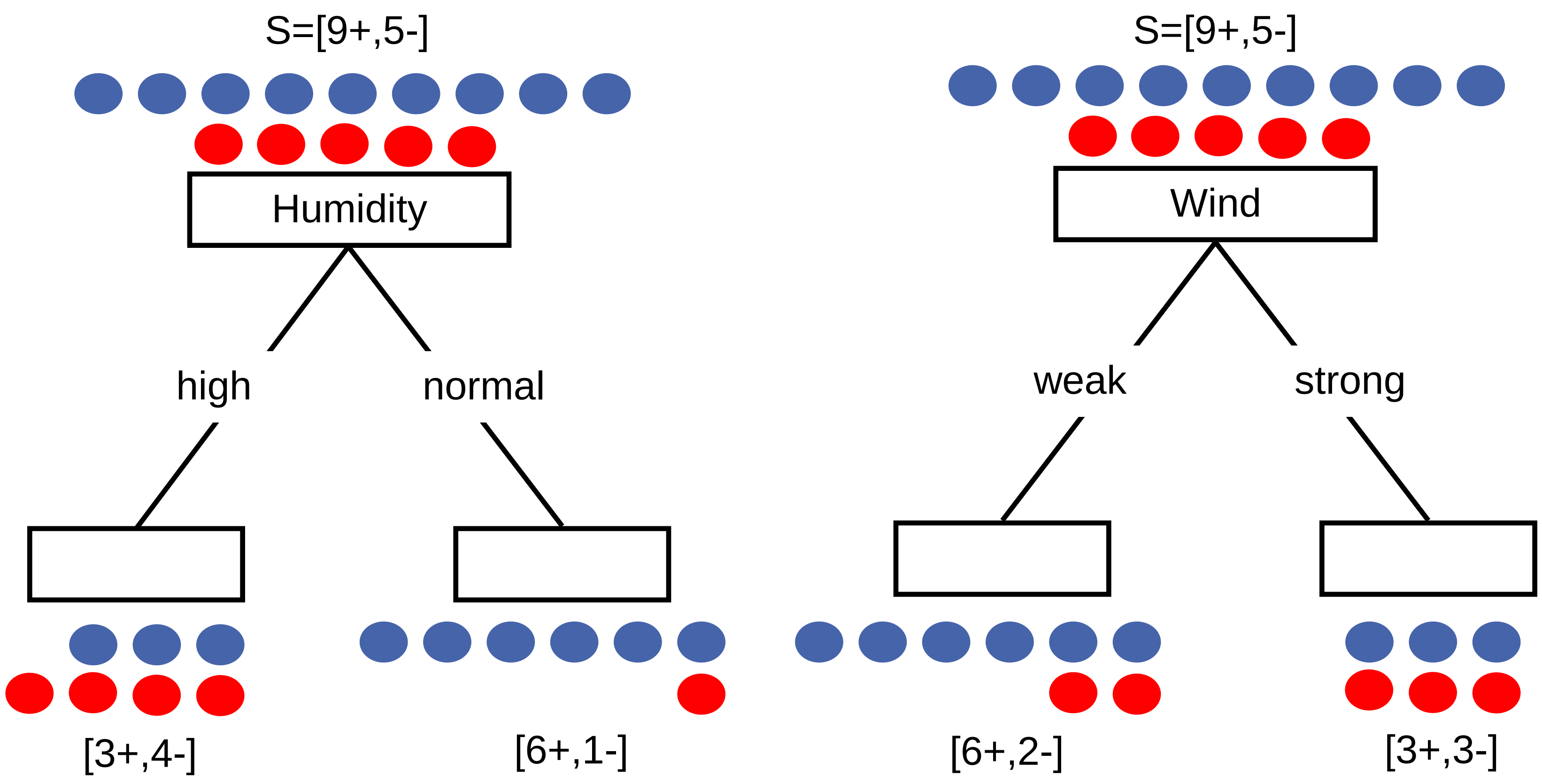


Information Gain

- **Information Gain** $\mathrm{IG}(\mathrm{S}, \mathrm{A})$: Expected entropy reduction of S by splitting on attribute A
- $\mathrm{IG}(\mathrm{S}, \mathrm{A}) = \mathrm{H}(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} \mathrm{H}(S_v)$
  - $V(A)$: Set of all possible attribute values v of attribute A
  - S_v: Subset of S for which A is value v
  - Calculate the difference between the entropy of the parent node and the entropy of the child nodes
- **Objective of learning with decision trees**
  - Select attributes that minimize entropy and maximize information gain.
    Classify learning examples with as few steps as possible
    → Tree with little depth

Information Gain – Attribute Selection

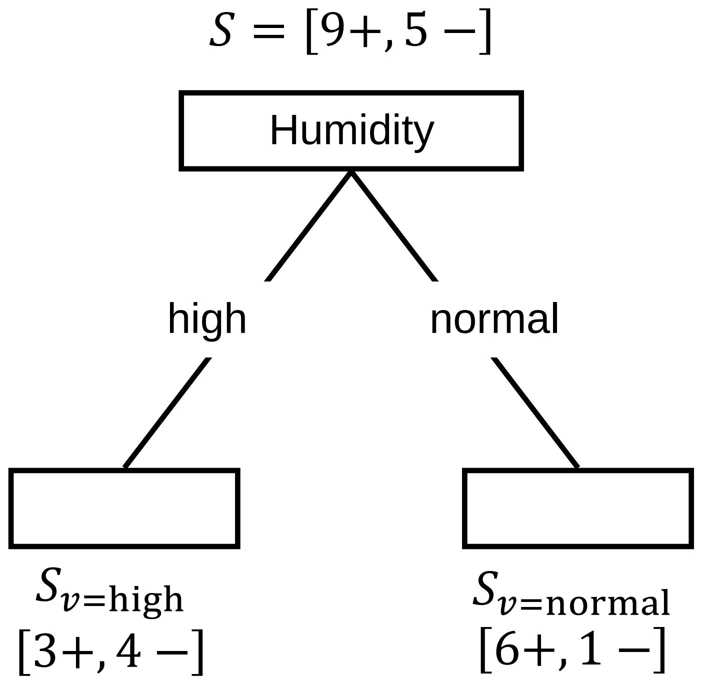
- **Idea**: A suitable attribute splits the instances into subsets, where (ideally) all instances in a subset are positive or negative



Information Gain – Attribute Selection

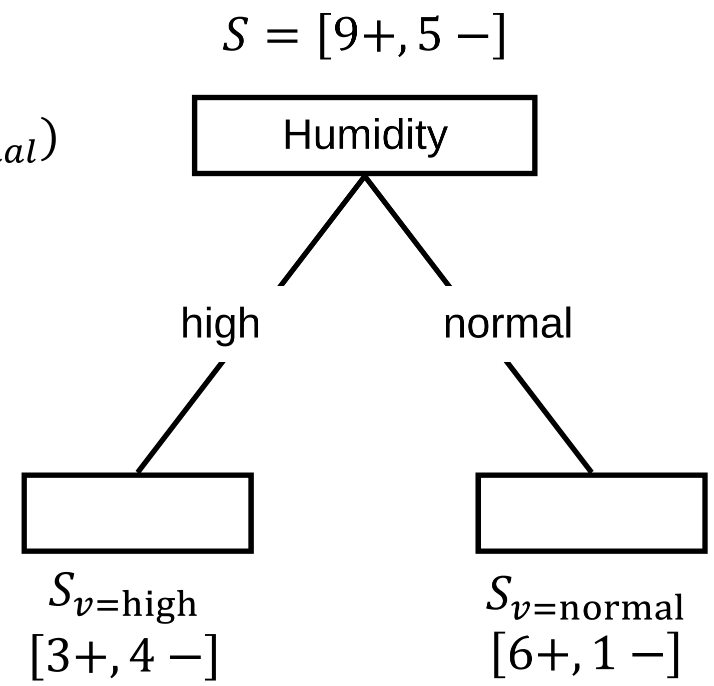
- **Step 1: Calculate entropy of parent node**
  - Entropy of S:
    - $H(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$
    - $H(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$

- **Step 2: Calculate entropy of child nodes**
  - Entropy of $S_{v=\text{high}}$:
    - $H\left(S_{v=\text{high}}\right) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$
  - Entropy of $S_{v=\text{normal}}$:
    - $H\left(S_{v=\text{high}}\right) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.592$



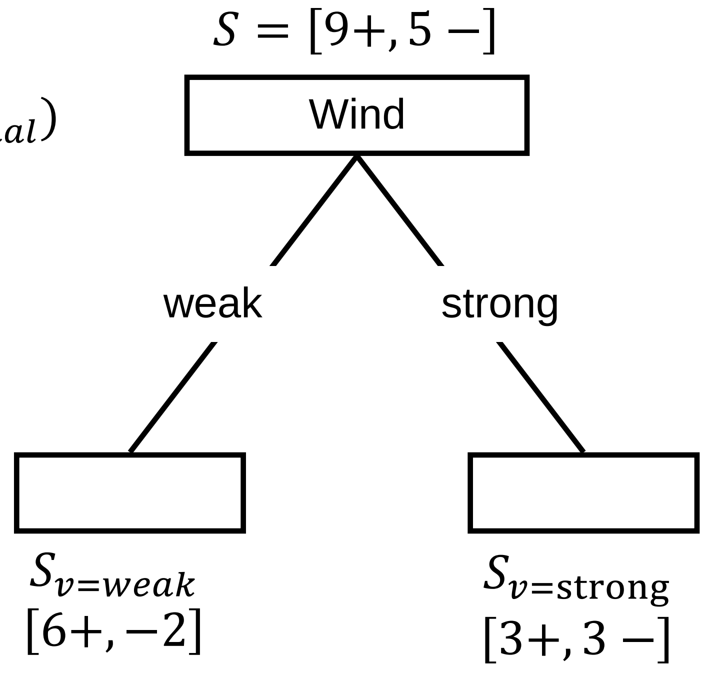
Information Gain – Attribute Selection

- **Step 3: Calculate Information gain for all attributes**
  - Information gain for **humidity:**
    - $IG(S, Humidity) = H(S) - \frac{|S_{v=high}|}{|S|} H\left(S_{v=high}\right) - \frac{|S_{v=normal}|}{|S|} H\left(S_{v=normal}\right)$
    - $IG(S, Humidity) = 0.940 - \frac{7}{14} 0.985 - \frac{7}{14} 0.592 = 0.151$



Information Gain – Attribute Selection

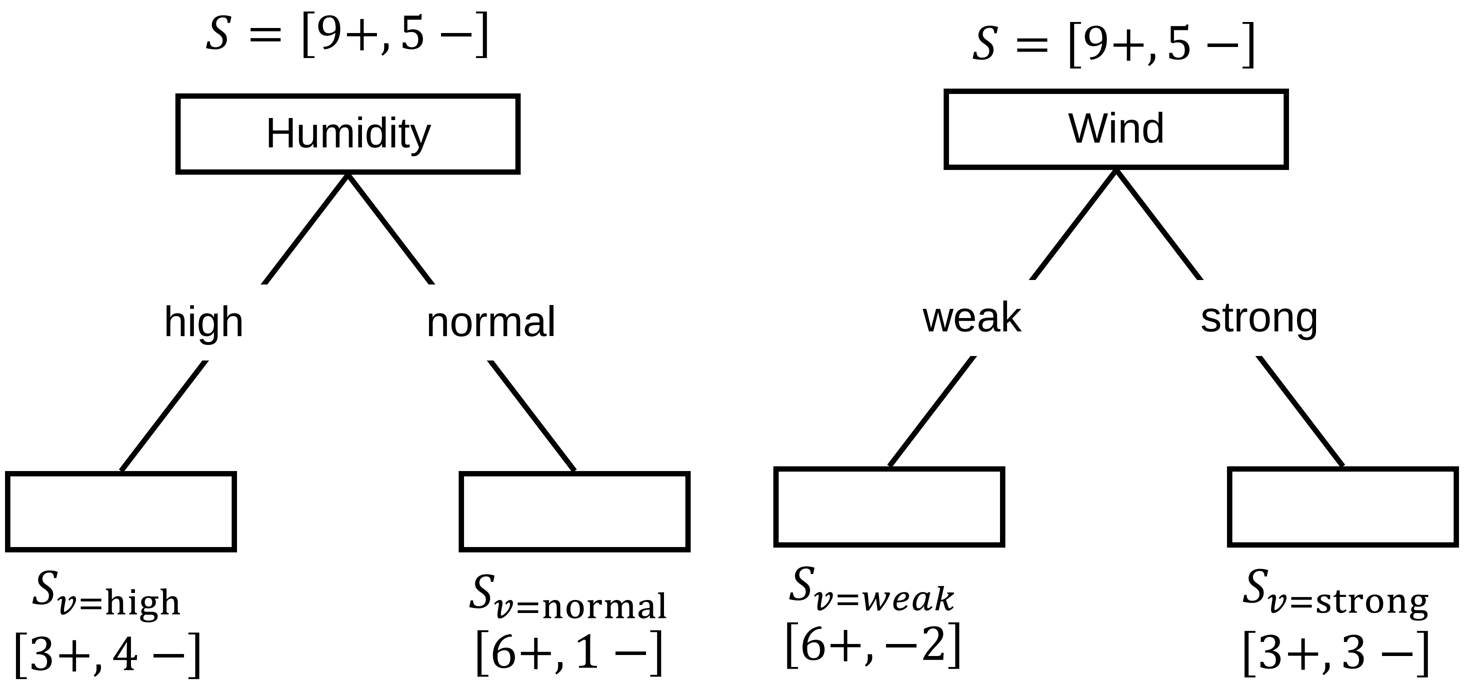
- **Step 3: Calculate Information gain for all attributes**
  - Information gain for **humidity:**
    - $IG(S, Humidity) = H(S) - \frac{|S_{v=high}|}{|S|} H(S_{v=high}) - \frac{|S_{v=normal}|}{|S|} H(S_{v=normal})$
    - $IG(S, Humidity) = 0.940 - \frac{7}{14} 0.985 - \frac{7}{14} 0.592 = 0.151$
  - Information gain for **wind:**
    - $H(S) = 0.940$ (entropy of parent stays the same)
    - $H(S_{v=\text{weak}}) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.811$
    - $H(S_{v=\text{strong}}) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$
    - $IG(S, Wind) = H(S) - \frac{|S_{v=weak}|}{|S|} H(S_{v=weak}) - \frac{|S_{v=strong}|}{|S|} H(S_{v=strong})$
    - $IG(S, Wind) = 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1 = 0.048$



Information Gain – Attribute Selection

- **Step 3: Calculate information gain for all attributes**
  - Information gain for **humidity:**
    - $IG(S, Humidity) = 0.151$
  - Information gain for **wind:**
    - $IG(S, Wind) = 0.048$

- **Step 4: Choose Attribute with highest information gain**
  - $IG(S, Humidity) > IG(S, Wind)$
  - Humidity attribute minimizes entropy, and is therefore used to split the tree

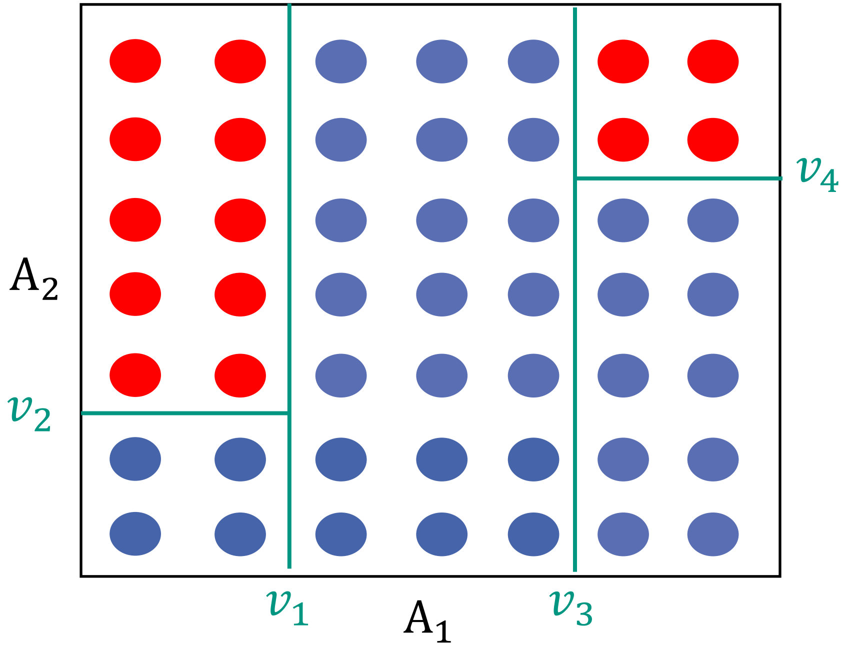


Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Base Method: ID3

- **ID3**: Iterative Dichotomizer 3 ≈ iterative division of instances
- **Top-Down**: Build decision tree **recursively** from root node
- **Greedy**: Each time an attribute is selected, the attribute that maximizes the information gain is used to split the data



Base Method: ID3 – Algorithm

ID3(Examples, Target Attribute, Attributes)
  Create node for tree
  IF (all examples positive), **Return** (node with label=$\oplus$)
  IF (all examples negative), **Return** (node with label=$\ominus$)
  IF (*Attribute*=∅), **Return** (node with label= most common target attribute of examples in node)

  Calculate A = Attribute with largest information gain for examples
  Assign node the attribute = A
  FOR ALL attribute values v_i of A:
    Create new branch with v_i
    Examples(v_i) = Subset of examples containing attribute value v_i
    IF (Examples(v_i)=∅):
      Add leaf node to branch with label = most common target attribute of examples in node
    THEN:
      Add subtree **ID3**(Examples(v_i), Target Attribute, Attributes \ $\{A\}$)
  Return node

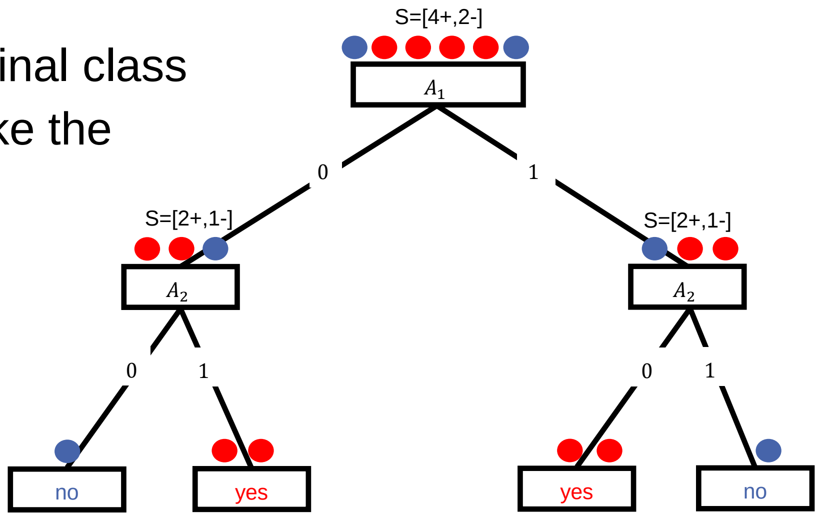
Break recursion

Create subtree and sort examples into nodes

Recursively build decision tree

Base Method: ID3 – Greedy Algorithm

- Class is A_1 XOR A_2
  - Attribute A_3 does not correlate with final class
  - A perfect tree tests A_1 then A_2 to make the classification with depth of tree = 2
- **BUT:** $\mathrm{IG}(\mathrm{S}, \mathrm{A}_1)$ and $\mathrm{IG}(\mathrm{S}, \mathrm{A}_2) = 0$
  - $H(S) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.92$
  - $H(S_0) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.92$
  - $H(S_1) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.92$
  - $IG(S, A_1) = 0.92 - \frac{3}{6}0.92 - \frac{3}{6}0.92 = 0$



A_1	A_2	A_3	Class
1	1	0	No
1	0	1	Yes
0	0	0	No
0	1	1	Yes
1	0	0	Yes
0	1	0	Yes

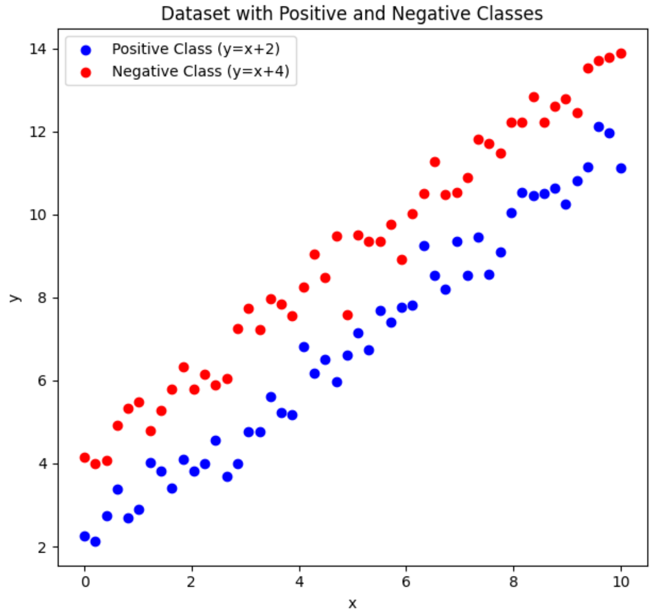
- Information gain for split with A_3 is 0.25 and therefore preferred.
  - → ID3 choses A_3 and creates tree with depth = 3
  - Greedy algorithms don‘t find the optimum. Finding an optimum tree is a NP-complete problem and is therefore rarely/never done. [Hyafil]

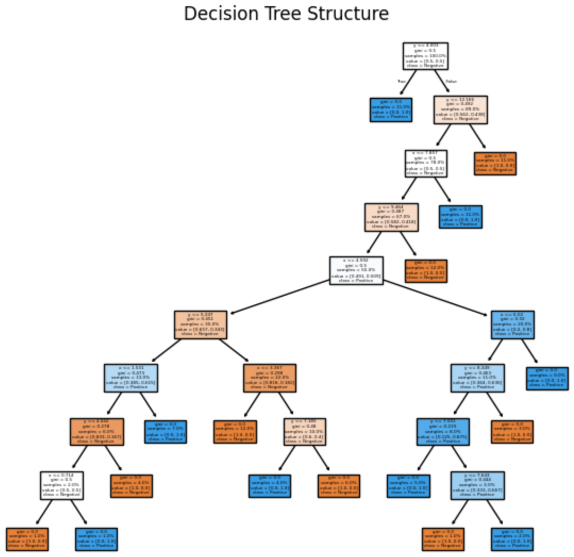
Base Method: ID3 – Properties

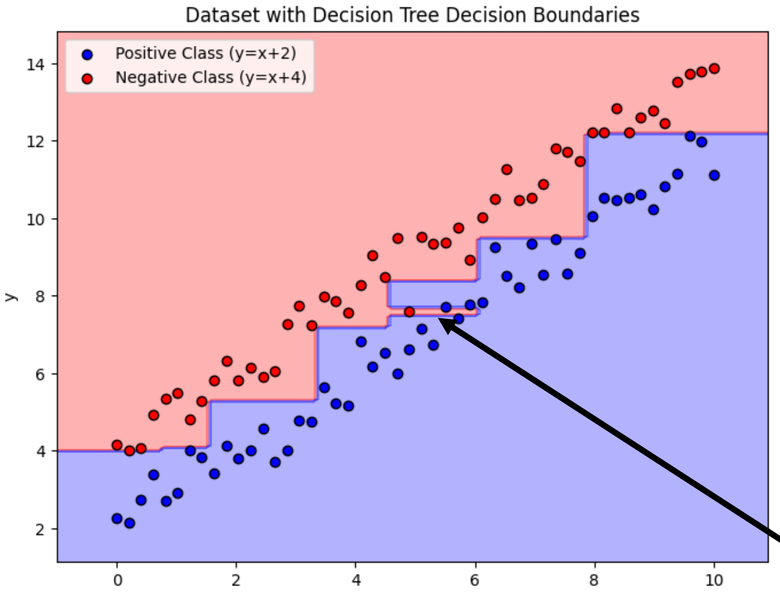
- The hypothesis space H of ID3 is the complete space of finite discrete-valued functions, relative to the available attributes ➔ contains target function
- ID3 does not guarantee an optimal solution
  - Greedy algorithms do not backtrack and do not consider future steps
- Inductive bias: smaller trees are preferred to larger trees
- ID3 is prone to overfitting
  - Stops only when all instances have been classified perfectly, even if the data is noisy/wrong
- Nowadays modern libraries mostly use CART which is a slightly improved ID4.5 algorithm which is a slightly improved ID3 algorithm
  - The basics remain the same

Trees do not have an additive structure

- Decision Trees are unable to handle linearly correlated features.
- Example: Data from $y = x + 2$ and $y = x + 4$ with little bit of noise







Decision Tree is unable to learn decision boundary $y = x + 3$

What is happening here?

Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

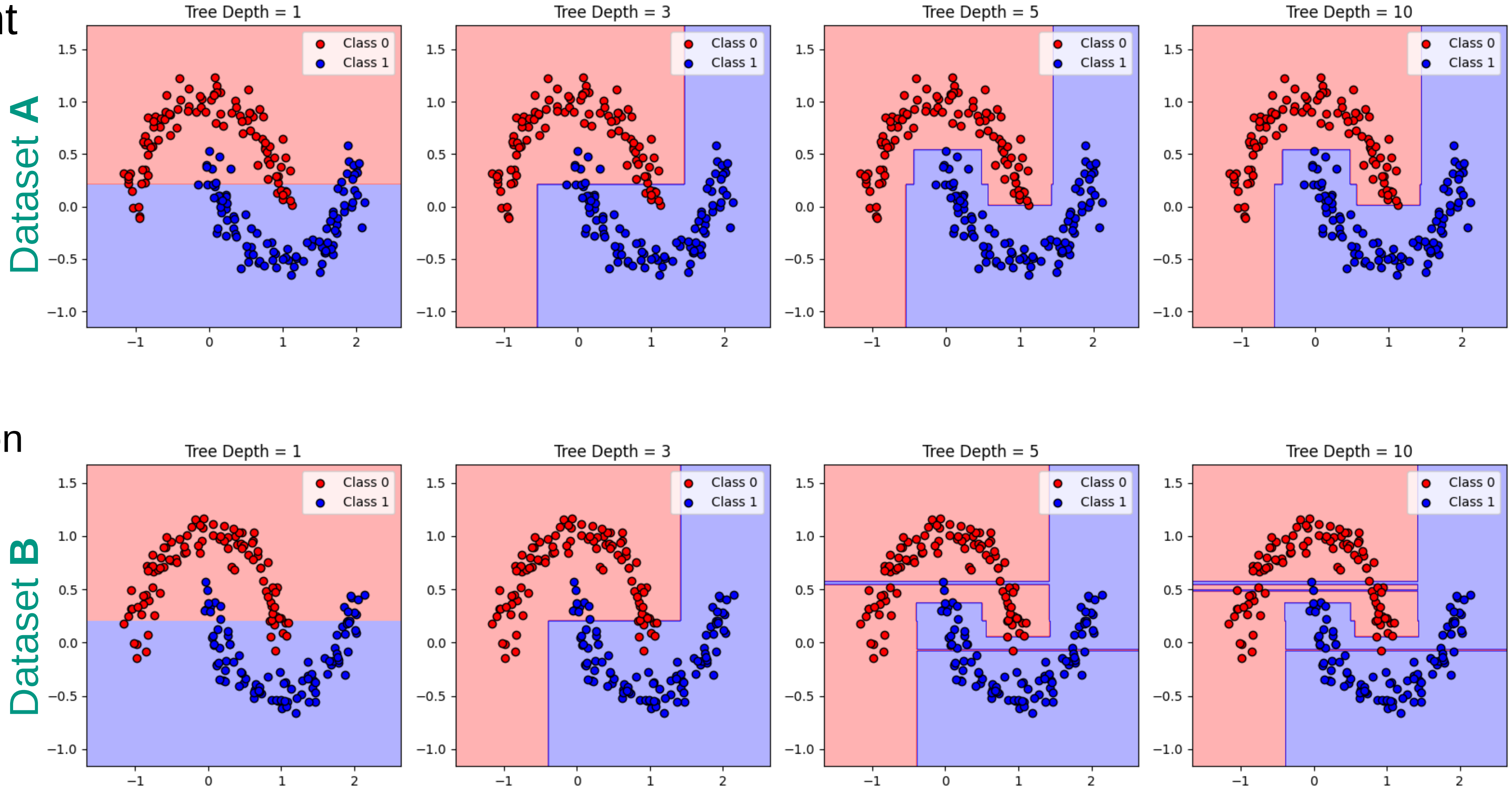
Base Method ID3 – Overfitting

- **Building trees with ID3**
  - Add nodes until all training examples are perfectly classified
  - Based on statistical approximation of information gain (e.g. entropy)

➔ **Often leads to overfitting because**
  - some examples in dataset are noisy or mislabeled
  - examples are not representative (e.g. too little data)

Overfitting - Example

- **A, B**: Slightly different samples from same distribution
- Empirical error of train dataset is reduced to zero with deeper trees
- **BUT:**
  - Decision boundary on **A** seems to be suitable
  - The deeper decision boundaries on **B** include outliers and noisy samples. It overfits the data distribution



Occam‘s Razor

- Why should simpler hypotheses be preferred?
  - In this case, smaller trees?
- There are fewer simple hypotheses than complex ones.
  - A short hypothesis that correctly explains the training examples is most likely not a coincidence
  - A long hypothesis, that correctly explains the training examples may be a coincidence.
- Short trees are more efficient

Overfitting formal

Reminder

- **Definition:** A hypothesis overfits the training examples, if some other hypothesis, that fits the training examples less well, actually performs better over the entire distribution of instances
  - Learning system memorizes training data rather than learning the underlying structure

- Formal:

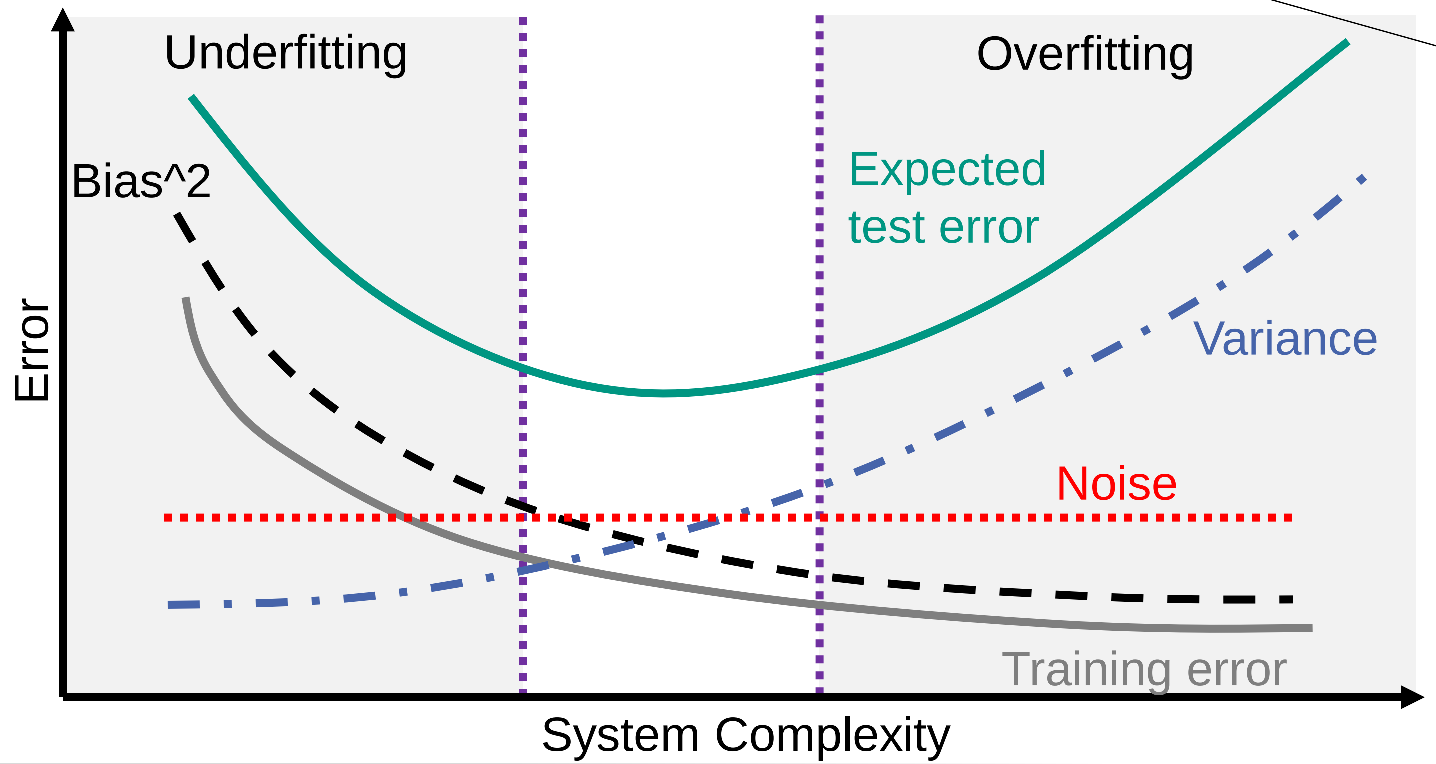
$$h \in H \text{ overfitting} \Leftrightarrow \exists h' \in H \text{ such that given } D_{Tr} \text{ and } D_V$$
$$\hat{\mathcal{L}}_{D_{Tr}}(h) < \hat{\mathcal{L}}_{D_{Tr}}(h') \wedge \hat{\mathcal{L}}_{D_V}(h) > \hat{\mathcal{L}}_{D_V}(h')$$

  - Whereas:
    - D_{Tr} – Training Data
    - D_V – Validation Data

Bias-Variance Tradeoff

- $\underbrace{\mathbb{E}_{x,y,D}[(h_D(x) - y)^2]}_{\text{Expected error}} = \underbrace{\mathbb{E}_{x,D}\left[\left(h_D(x) - \bar{h}(x)\right)^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x}\left[\left(\bar{h}(x) - \bar{y}(x)\right)^2\right]}_{\text{Bias^2}} + \underbrace{\mathbb{E}_{x,y}[(\bar{y}(x) - y)^2]}_{\text{Noise}}$

Common mistake in past exams: Forgetting the „squared“

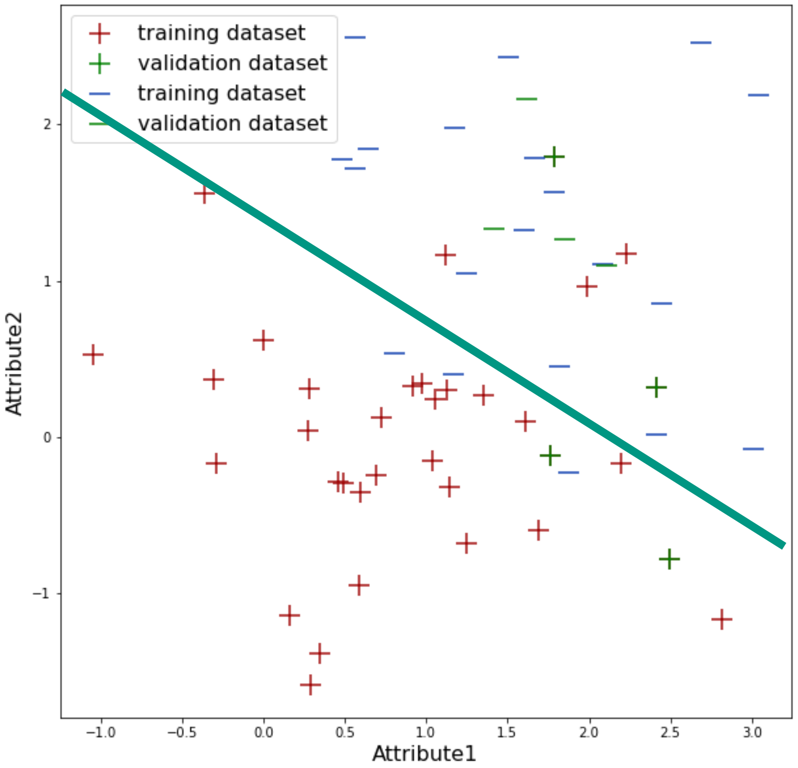


Bias-Variance Tradeoff

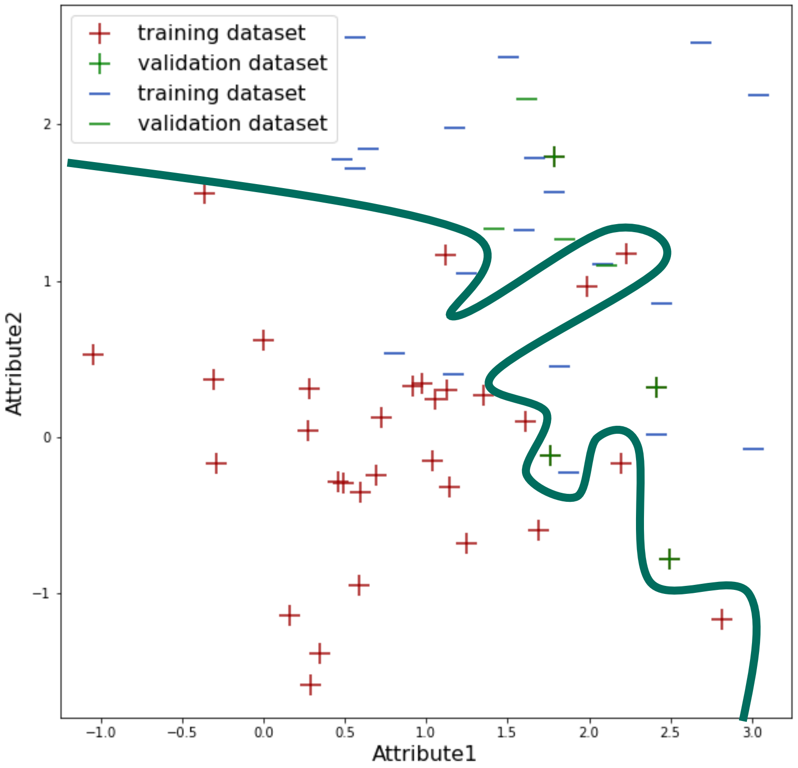
- **Variance**: Measures how much the classifier changes when trained on different splits of training data.
  - I.e. how much do we “overspecialize” for this particular training dataset
- **Bias**: What is the inherent error you get from your classifier even with infinite training data?
  - Can be related to the hypothesis space, e.g. a linear model can't predict non-linear data
- **Noise:** How large is the intrinsic noise in the data?
  - Measures ambiguity due to data distribution and feature representation.
  - Is an inherent aspect of the data and cannot be removed

Bias vs Variance

High bias, low variance
(usually underfitting)



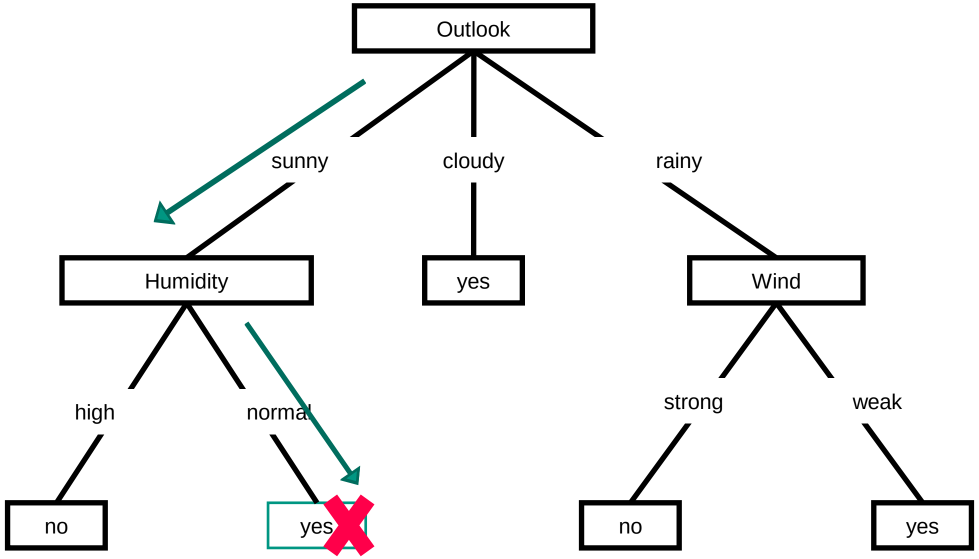
Low bias, high variance
(usually overfitting)



Example – Noisy/Wrong Data

- What happens if a noisy example is added to training data?

Nr.	Outlook	Temperature	Humidity	Wind	Tennis?
1	Sunny	Hot	Normal	Strong	No

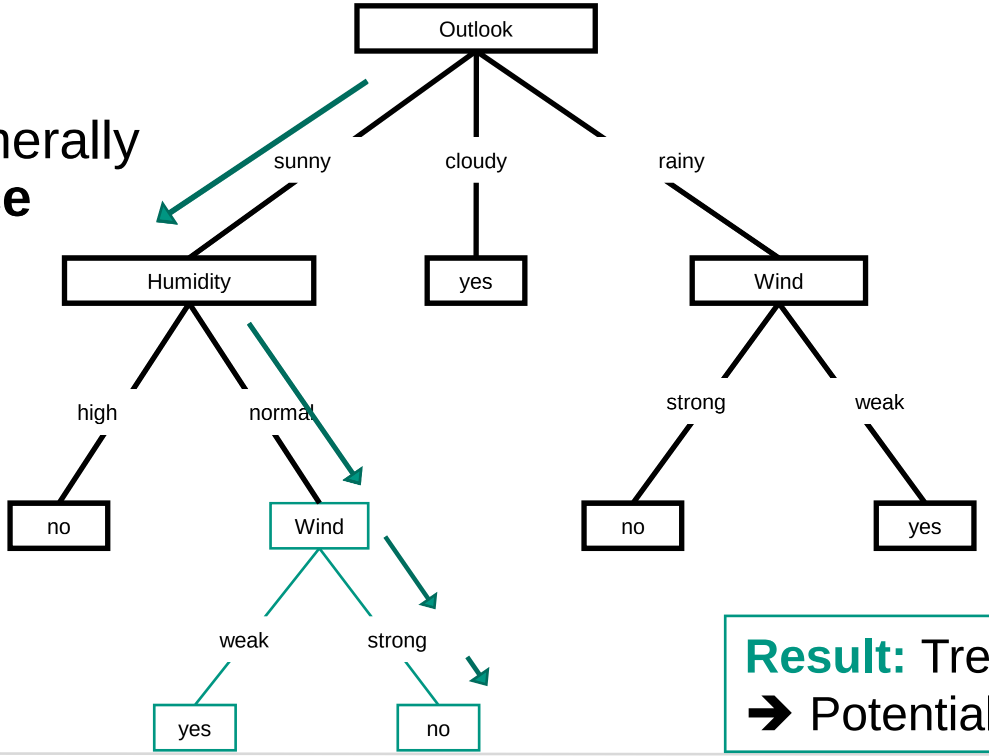


Example – Noisy/Wrong Data

- What happens if a noisy example is added to training data?

Nr.	Outlook	Temperature	Humidity	Wind	Tennis?
1	Sunny	Hot	Normal	Strong	No

- Decision Trees are generally **low bias high variance** models and usually **overfit** the data



Result: Tree complexity increases
➔ Potentially more errors on unseen data

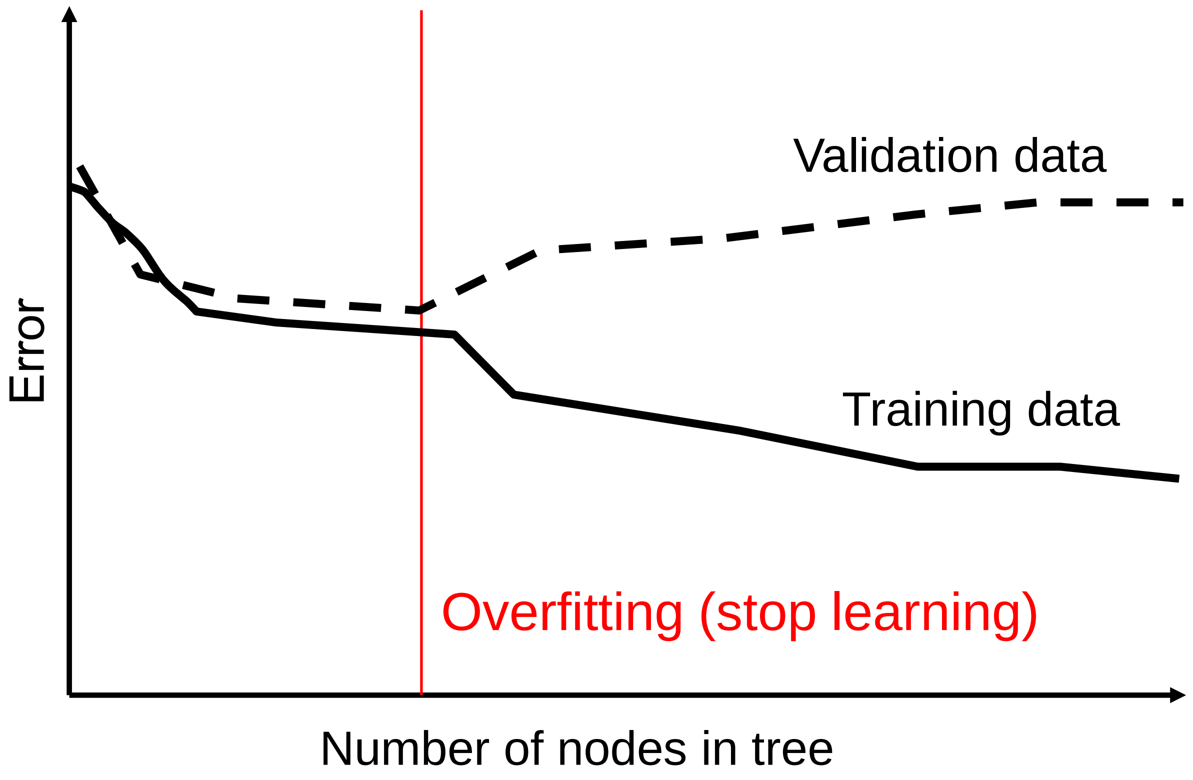
Reduce Overfitting for ID3

- Combat the high overfitting properties of ID3 with **regularization**
  - **Maximum depth**: Growing the tree stops after a certain number of steps, even if not all data is perfectly classified.
  - **Minimum samples:** A node cannot be split if it contains less than a specified number of training examples
  - **Early stopping:** Stop growing the tree if the validation error increases
  - **Pruning**: Remove non-critical parts of the tree to reduce complexity

- Reduce overfitting with multiple trees:
  - **Bagging**
  - **Random Forests**

Early Stopping – Classification Error

- **Intuition**: Stop tree growth before overfitting occurs
- **Idea**: Validation error must decrease by ϵ per iteration, otherwise we stop
- **Pros**:
  - Easy to implement
- **Cons**:
  - Too short-sighted: Validation error could increase in the current step, but decrease again in the next step
    - Reminder the XOR problem with the greedy ID3 algorithm. Early stopping might stop before we even get the correct solution.

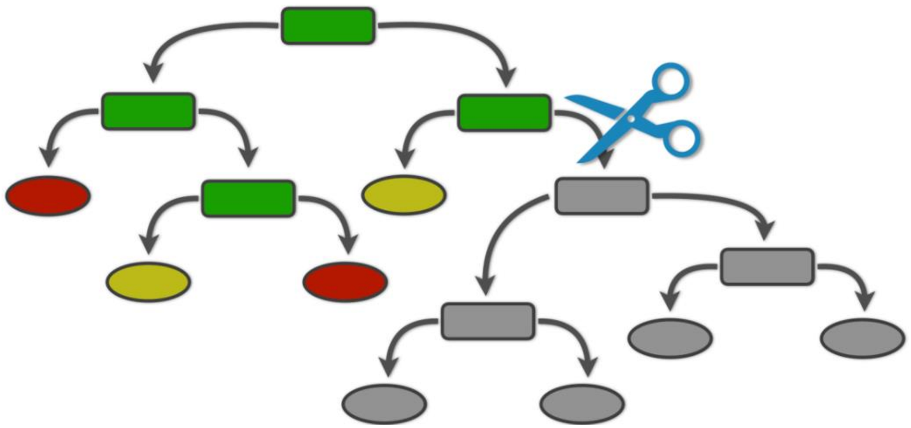


Pruning

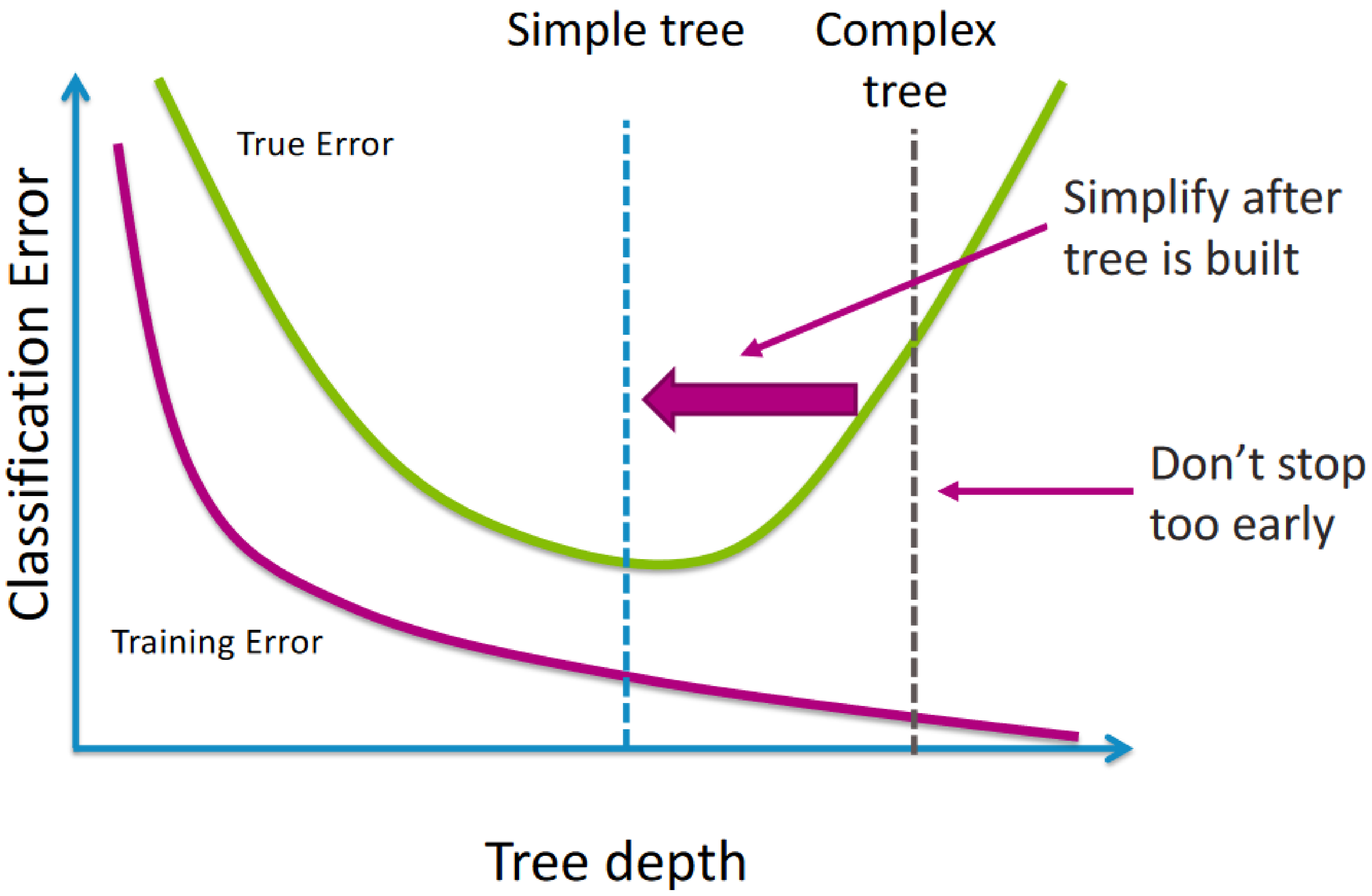
- **Intuition**: Simplify the tree after learning is completed.
  - Stopping “too early” no longer a problem

- **Pruning**: Remove a node (subtree) and replace it with a leaf node of the most common class

- **Bottom up:** Starting from the leaves, replace each node with a leaf node until the validation error starts to increase



Pruning



[Guestrin]

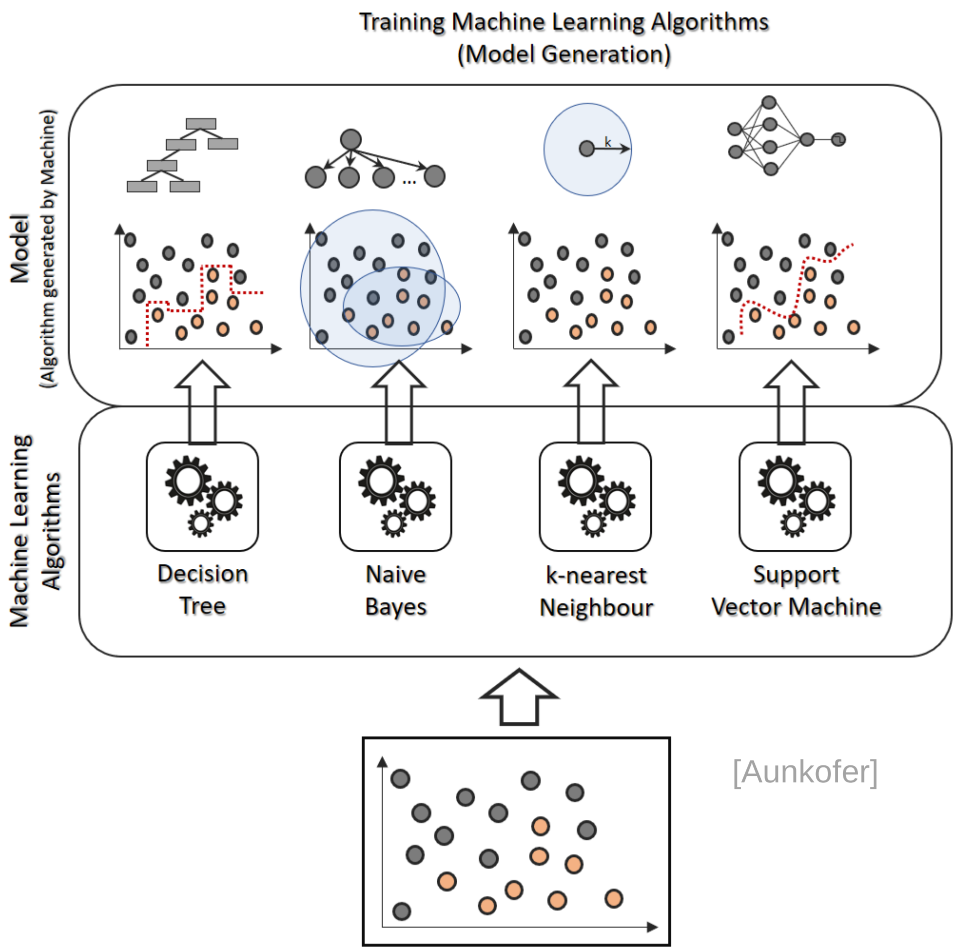
Overview

Karlsruher Institut für Technologie

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Ensemble Learning

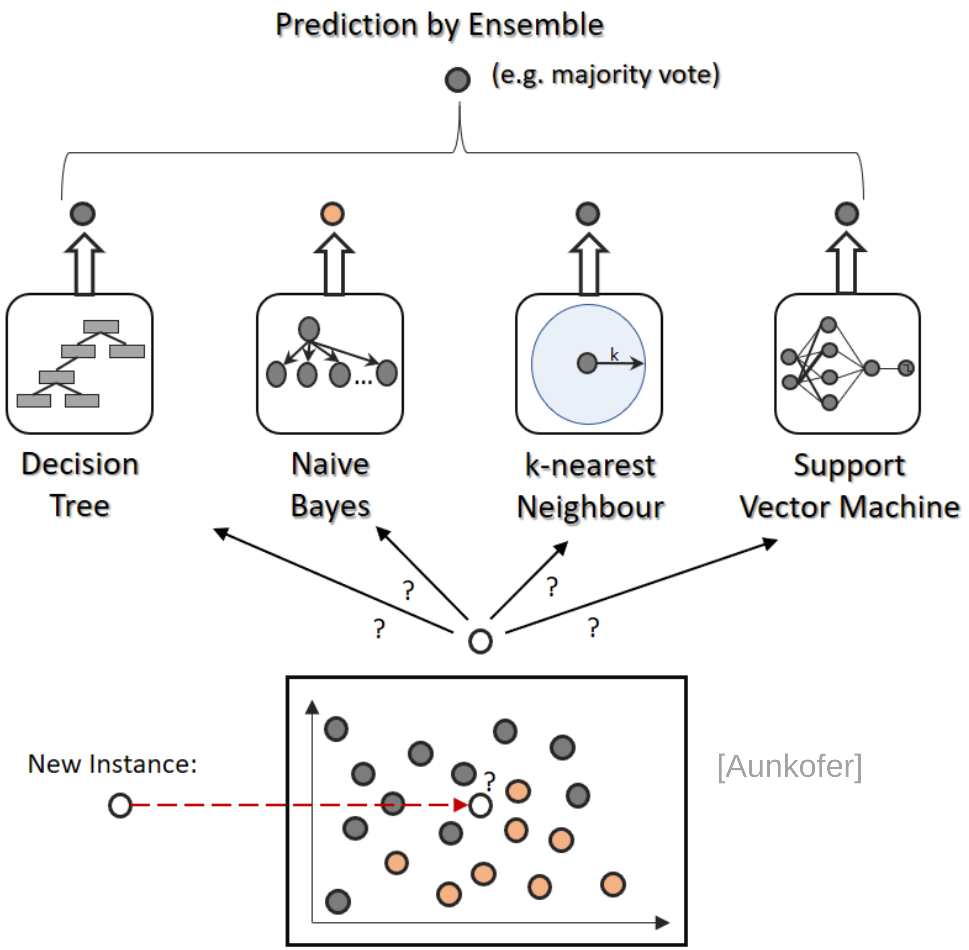
- **Question**: Which learning system is the best for my specific task?
  - Decision Trees
  - Support-Vector-Machines
  - Naive Bayes
  - ...?
- **Ensembles**: Learn several models and combine them into a stronger model



[Aunkofer]

Ensemble Learning

- **Advantages**
  - Better prediction quality
  - Better robustness (overfitting)
- **Disadvantages**
  - Time consuming and computationally expensive
  - Loses interpretability
- Predictions from all machines can be treated the same or weighted differently
  - E.g.: use majority vote for classification or average for regression



[Aunkofer]

Ensemble Learning - Motivation

- **Reminder**: $\text{Generalization Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$
- **Idea**: When several complex hypothesis with small bias and large variance are aggregated, their variance decreases but the bias remains small.
  → Simple way to decrease the generalization error

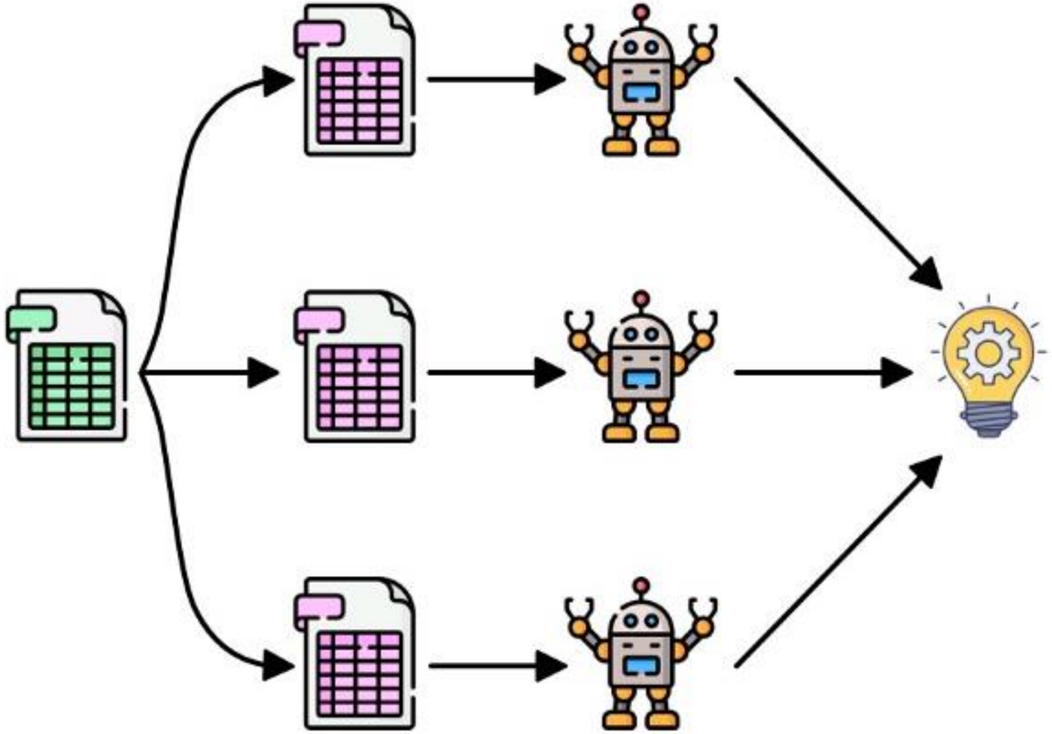
Ensemble Learning - Idealistic

Karlsruher Institut für Technologie

- **Reminder**: Generalization Error = $\mathbb{E}_{x,D}\left[\left(h_D(x) - \bar{h}(x)\right)^2\right]$ + Bias² + Noise
- Instead of one dataset D, use several datasets D_1 to D_k
  - With the same number of learning examples and from the same distribution as D
- Learn k hypotheses with k different models
- Final hypothesis is e.g. the mean of the k hypotheses
- $\hat{h} = \frac{1}{k}\sum_{i=1}^{k} h_{D_i} \rightarrow \bar{h}$, if $k \rightarrow \infty$
  - Variance approaches 0!
- **Problem**: We don‘t have k datasets and if we split D into k datasets, the variance of the hypotheses increases even more due to the lack of training examples
- **Solution**: Bagging

Bagging

- **Process**
  - **Bootstrap**: Create k datasets D_1 to D_k from D via layback sampling
  - **Training**: Train a model h_{D_1} to h_{D_k} for each dataset
  - **Aggregation**: Combine models by majority vote or average their predictions
    - Regression: $h(x) = \frac{1}{k}\sum_{i=1}^{k} h_{D_i}(x)$
- **Advantage**: Reduce variance without increasing bias



Bagging – Pros & Cons

Pros

- **Reduces variance** and can therefore be used with high variance models
- **Uncertainty**: Bagging not only gives you the expected value, but also makes the variance of the prediction easily visible
- **Out-of-bag error**: Learning examples from D not sampled in dataset D_i can be used to validate h_i. No separate validation dataset is required.
- **Parallelizable**

Cons

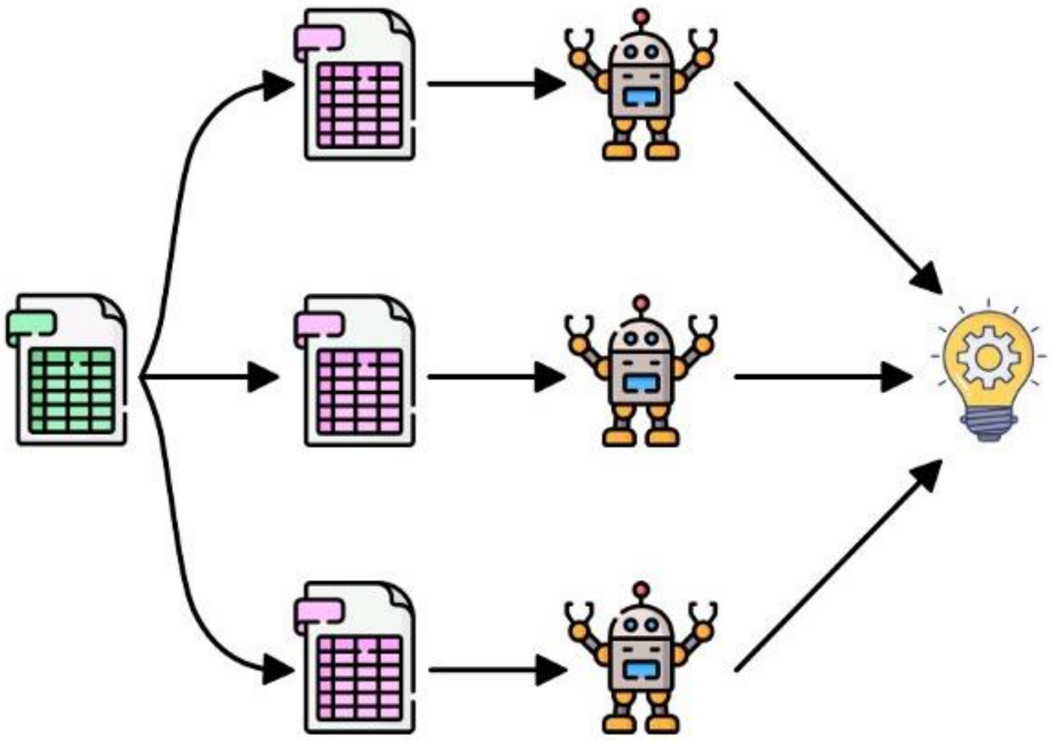
- **Computational cost** increases by at least k
- Variance cannot be fully reduced because bootstrapping samples is not **I.I.D**
  - **I.I.D:** Independent and identically distributed random samples
- **Loss of interpretability**

Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

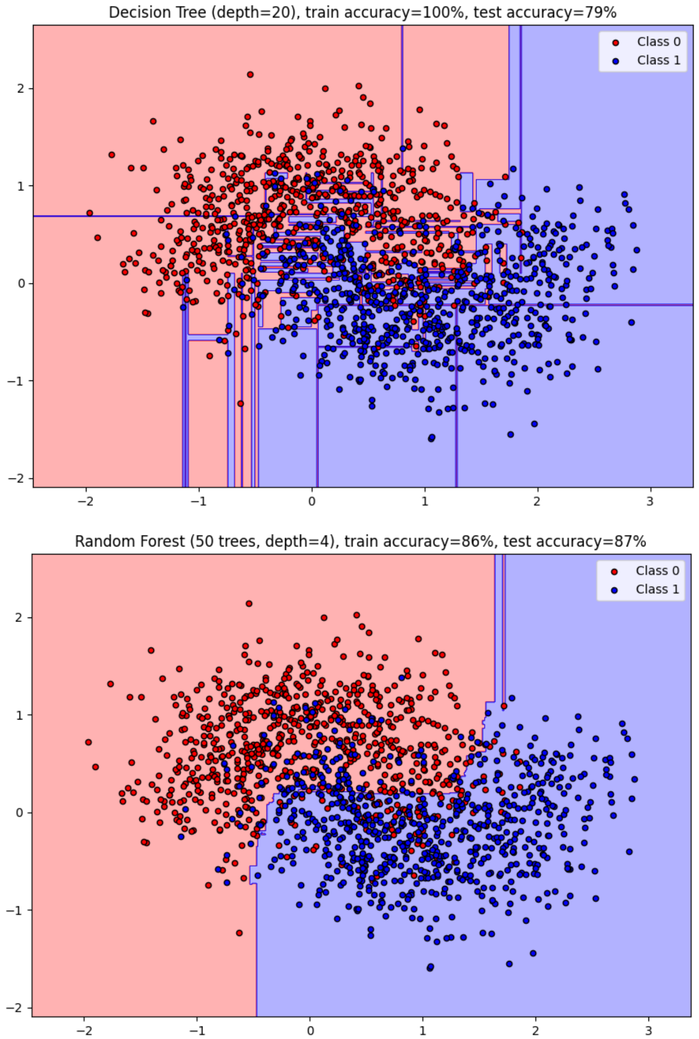
Random Forests

- **Problem of bagging**: Models are highly correlated
- **Previous**: Bagging with decision trees. Splits can use all d attributes of drawn samples
- **Now**: Choose random subset $s < d$ of attributes for each split. Create nodes using the best attribute for maximizing information gain in s.
- **Why**: To ensure, that the k learned trees are less correlated. Each tree uses a different subset of attributes.
  - Using s instead of d attributes will increase the bias. However, the reduction in variance from decorrelating the trees outweighs this.



Random Forests – Modification of Bagging

- **Process**
  - **Bootstrap**: Create k datasets D_1 to D_k from D via layback sampling.
  - **Training**: Train a model h_{D_1} to h_{D_k} for each dataset with one small modification
    - **Before each split, sample a subset of $s < d$ attributes as possible candidates for splitting**
  - **Aggregation**: Combine models by majority vote or average their predictions
- **Advantage**: Generally, much better results than single decision tree



Overview

- Motivation
- Formalization
- Attribute selection
- Build the Tree - ID3-algorithm
- Overfitting
- Bagging
- Random Forest
- Extensions

Continuous Attribute Values I

- **Given**
  - Attribute A with continuous values

- **Process**
  - Dynamic definition of a new discrete attribute: $A_{\mathrm{c}} = \mathrm{true}$ if $A > c$

- **Problem**:
  - Choice of threshold c?
    → Use information gain:
    - Sort examples by their attribute values
    - Optimal threshold lies in the middle between two adjacent examples with different class affiliations

Continuous Attribute Values II

- **Example**: Continuous temperature

Temperature	4°	9°	16°	22°	27°	32°
Tennis	No	No	Yes	Yes	Yes	No

c_1 c_2

- Potential thresholds and respective information gain:
  - $c_1 = (9° + 16°) / 2 = 12.5° \rightarrow IG = 1 - \frac{2}{6}0 - \frac{4}{6}0.81 = 0.46$
  - $c_2 = (27° + 32°) / 2 = 29.5° \rightarrow IG = 1 - \frac{5}{6}0.97 - \frac{1}{6}0 = 0.19$
- Information gain is higher $c_1 = 12.5°$
  categorical attributes are < 12.5 and > 12.5

Summary

- Decision trees are a **non-parametric** supervised learning method
- Generally, very **fast training and inference**
- **Interpretable** as an „if-else“ flow chart
- Decision Trees usually **overfit the data** and show **low accuracies** on test data
  - It’s rare in practice to use a single decision tree
  - But ensemble methods like **random forests** and **boosting** drastically improve performance as they decrease variance
    - But we lose interpretability
    - Even today, they achieve better results on tabular data than neural networks [2022: Why do tree-based models still outperform deep learning on tabular data]

Literature

- Tom Mitchell: *Machine Learning*, Chapter 3. 1997
  - Homepage: http://www-2.cs.cmu.edu/~tom/
  - Only individual Decision Trees, no ensembles

- Murphy: *Probabilistic Machine Learning*, Chapter 18. 2022
  - PDF
  - Contains ensemble methods like random forest and gradient boosting
  - Also includes XGBoost, which is currently one of the best tree-based methods.