.

Machine Learning

Decision Trees

Prof. Dr. Martin Riedmiller
AG Maschinelles Lernen und Natürlichsprachliche Systeme
Institut für Informatik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Martin.Riedmiller@uos.de

Acknowledgment Slides were adapted from slides provided by
Tom Mitchell, Carnegie-Mellon-University
and Peter Geibel, University of Osnabrück

Outline

- Decision tree representation
- ID3 learning algorithm
- Which attribute is best?
- C4.5: real valued attributes
- Which hypothesis is best?
- Noise
- From Trees to Rules
- Miscellaneous

Decision Tree Representation

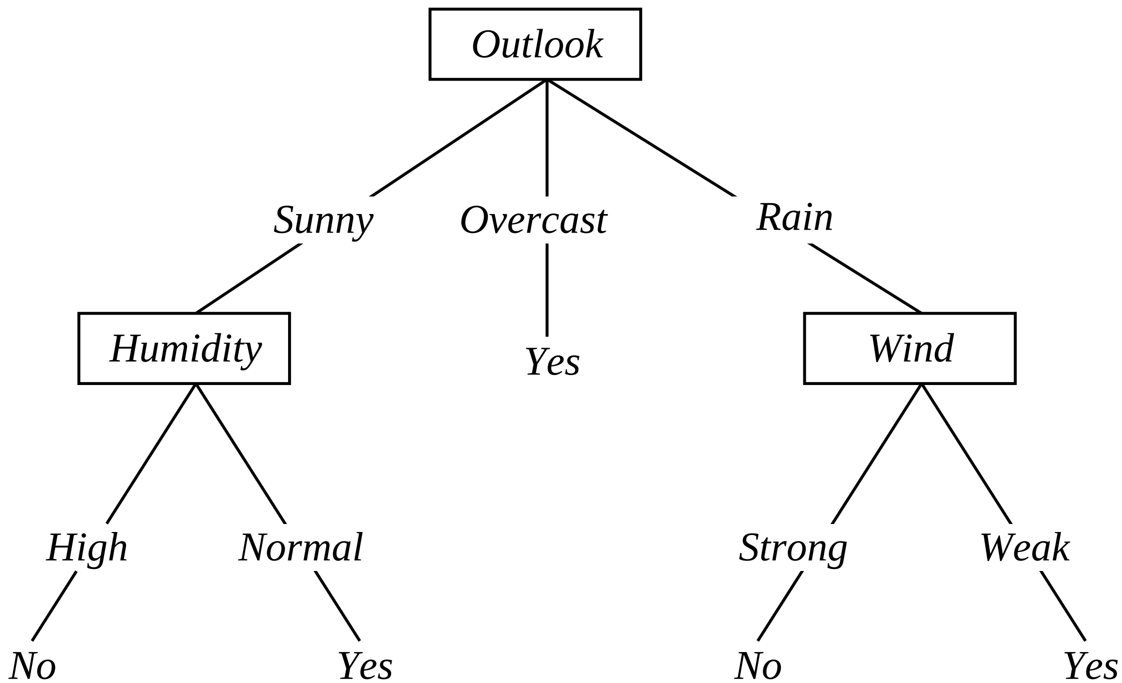
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Outlook, Temperature, etc.: attributes

PlayTennis: class

Shall I play tennis today?

Decision Tree for $PlayTennis$



Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee$, XOR

When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Interpretable result of learning is required

Examples:

- Medical diagnosis
- Text classification
- Credit risk analysis

Top-Down Induction of Decision Trees, ID3 (R. Quinlan, 1986)

ID3 operates on whole training set S

Top-Down Induction of Decision Trees, ID3 (R. Quinlan, 1986)

ID3 operates on whole training set S
Algorithm:

1. create a new $node$

Top-Down Induction of Decision Trees, ID3 (R. Quinlan, 1986)

ID3 operates on whole training set S
Algorithm:

1. create a new $node$

2. If current training set is sufficiently pure:

   - Label $node$ with respective class
   - We're done

Top-Down Induction of Decision Trees, ID3 (R. Quinlan, 1986)

ID3 operates on whole training set S
Algorithm:

1. create a new $node$

2. If current training set is sufficiently pure:
   - Label $node$ with respective class
   - We're done

3. Else:
   - $x \leftarrow$ the "best" decision attribute for current training set
   - Assign x as decision attribute for $node$
   - For each value of x, create new descendant of $node$
   - Sort training examples to leaf nodes
   - Iterate over new leaf nodes and apply algorithm recursively

Example ID3

- Look at current training set S

$$S=\{1,\ldots,14\}$$

Example ID3

- Look at current training set S

  $S=\{1,...,14\}$

- Determine best attribute

  Outlook

Example ID3

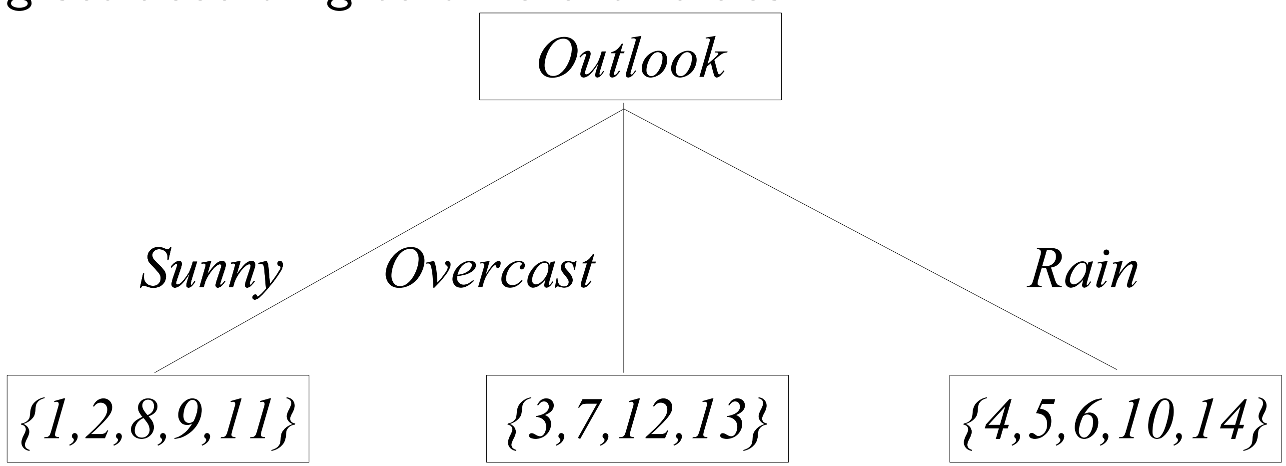
- Look at current training set S

$S=\{1,\ldots,14\}$

- Determine best attribute

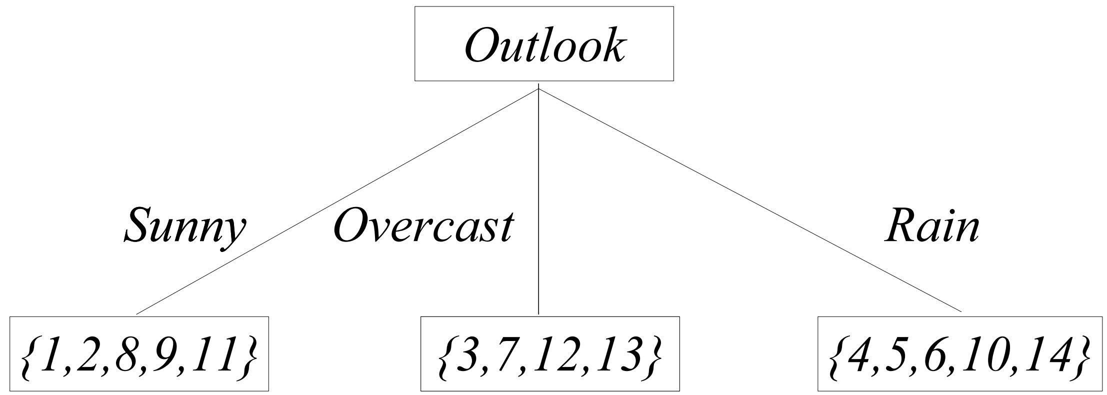
Outlook

- Split training set according to different values



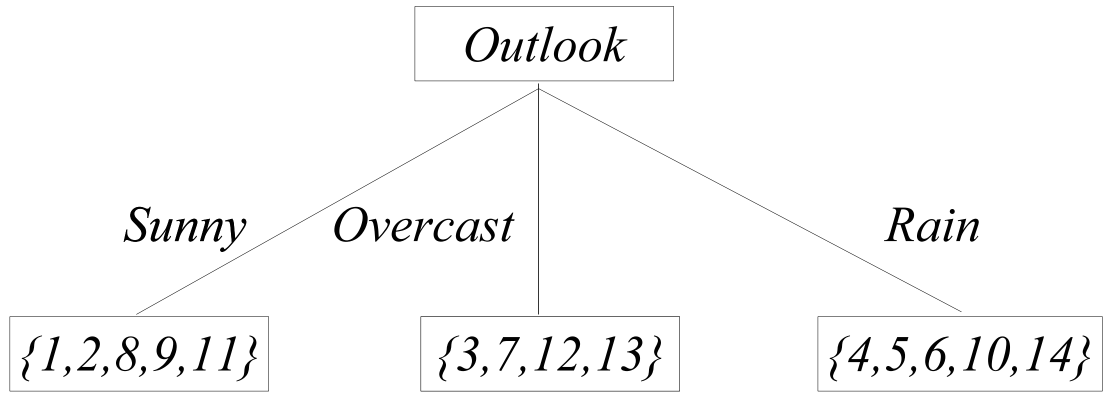
Example ID3

- Tree

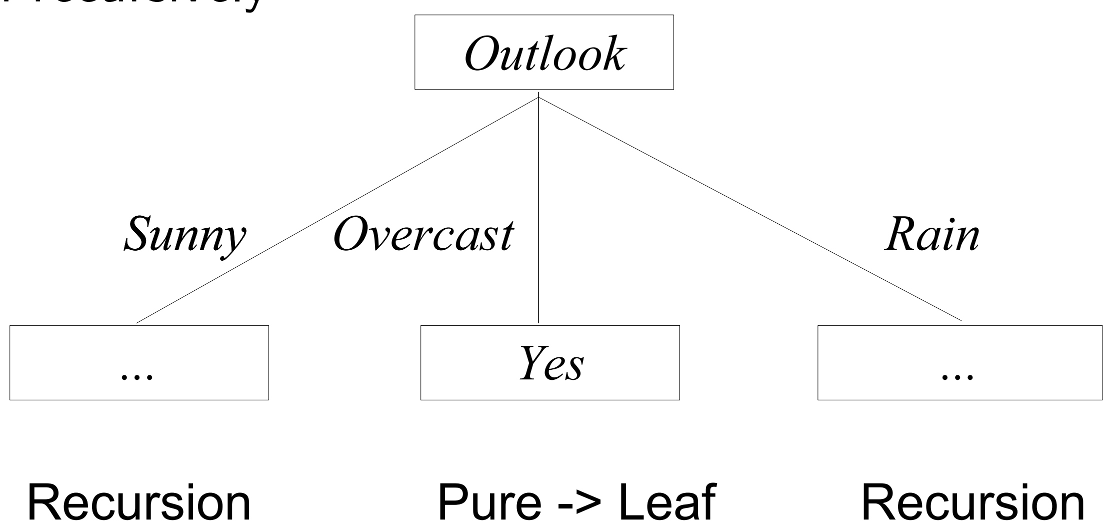


Example ID3

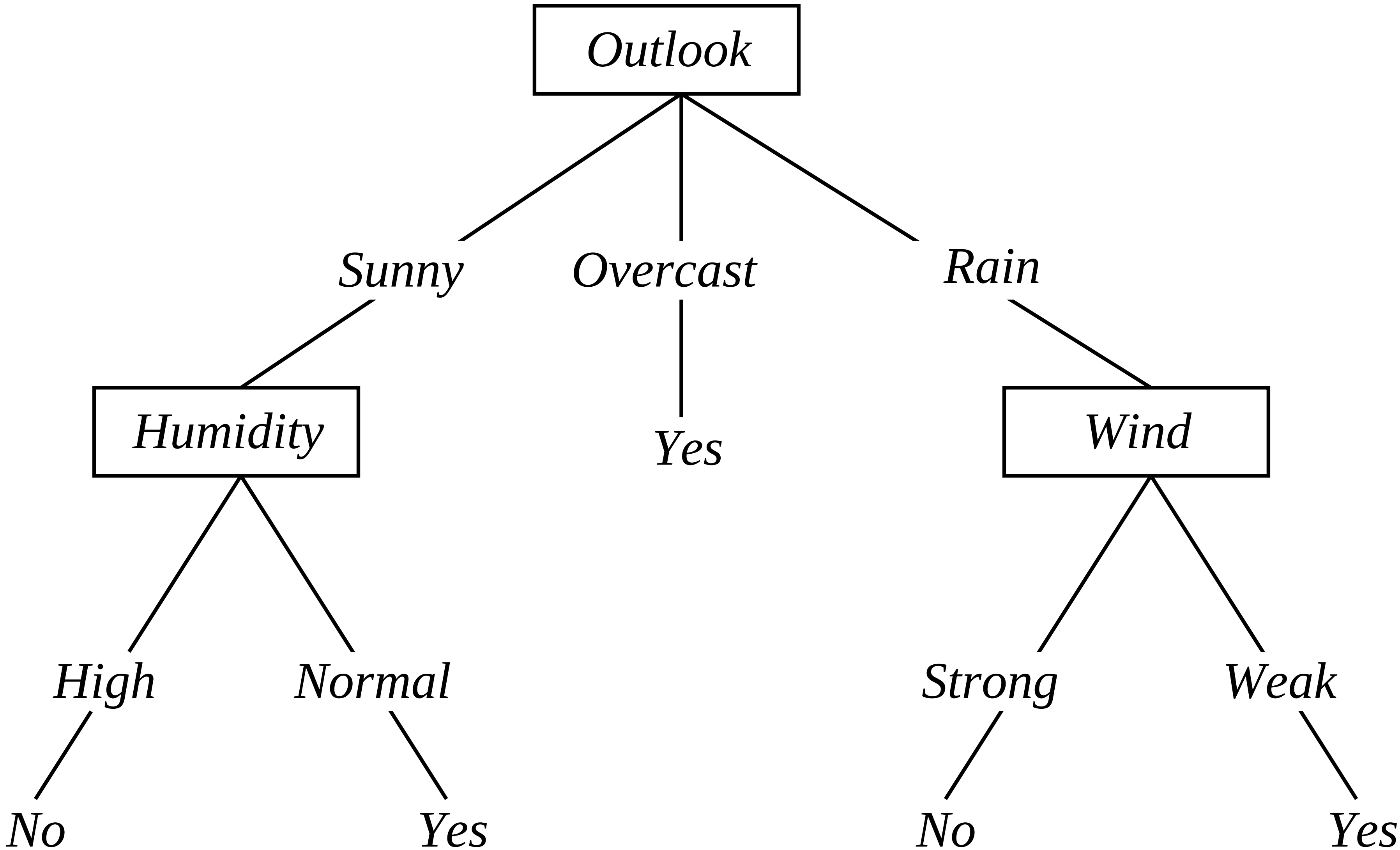
- Tree



- Apply algorithm recursively



Example – Resulting Tree



ID3 – Intermediate Summary

- Recursive splitting of the training set
- Stop, if current training set is sufficiently pure

ID3 – Intermediate Summary

- Recursive splitting of the training set
- Stop, if current training set is sufficiently pure
- ... What means pure? Can we allow for errors?
- What is the best attribute?
- How can we tell that the tree is really good?
- How shall we deal with continuous values?

Which attribute is best?

- Assume a training set $\{+, +, -, -, +, -, +, +, -, -\}$ (only classes)
- Assume binary attributes x_1, x_2, and x_3

Which attribute is best?

- Assume a training set $\{+,+,-,-,+,-,+,+,-,-\}$ (only classes)
- Assume binary attributes x_1, x_2, and x_3
- Produced splits:

	Value 1	Value 2
x_1	$\{+,+,-,-,+\}$	$\{-,+,+,-,-\}$
x_2	$\{+\}$	$\{+,-,-,+,-,+,+,-,-\}$
x_3	$\{+,+,+,+,-\}$	$\{-,-,-,-,+\}$

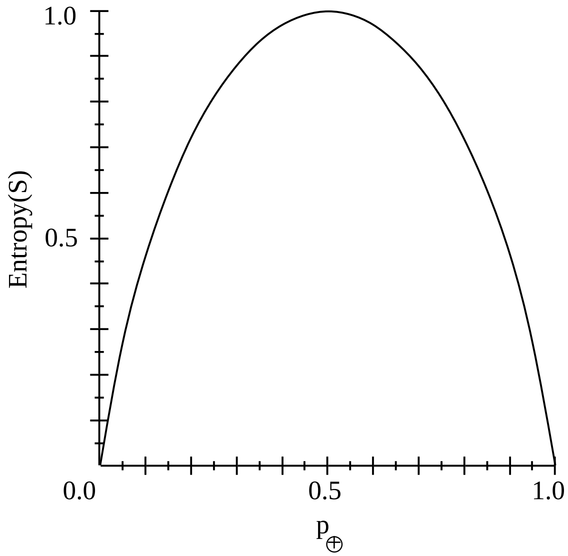
Which attribute is best?

- Assume a training set $\{+,+,-,-,+,-,+,+,-,-\}$ (only classes)
- Assume binary attributes x_1, x_2, and x_3
- Produced splits:

	Value 1	Value 2
x_1	$\{+,+,-,-,+\}$	$\{-,+,+,-,-\}$
x_2	$\{+\}$	$\{+,-,-,+,-,+,+,-,-\}$
x_3	$\{+,+,+,+,-\}$	$\{-,-,-,-,+\}$

- No attribute is perfect
- Which one to choose?

Entropy



- $p_\oplus$ is the proportion of positive examples
- $p_\ominus$ is the proportion of negative examples
- Entropy measures the 'impurity' of S
- $Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$

- task from information theory: find a code, that minimizes the information to be transmitted. Idea: the more likely the information, the shorter the code. The optimal code uses $-log_2(p_i)$ bits, where p_i is the probability of message i (Shannon, Weaver, 1949).
- entropy can then be intepreted as the expected length for transmitting one message, $\sum_i -p_i \, log_2(p_i)$

Information Gain

- Assume an attribute x with two possible values. Measuring x creates subsets S_1 and S_2 with different entropies
- Taking the mean of $Entropy(S_1)$ and $Entropy(S_2)$ gives conditional entropy $Entropy(S|x)$, i.e. in general: $Entropy(S|x) = \sum_{v \in Values(x)} \frac{|S_v|}{|S|} Entropy(S_v)$
- $\rightarrow$ Choose that attribute that maximizes the 'information gain'

$$Gain(S, x) := Entropy(S) - Entropy(S|x)$$

- $Gain(S, x) =$ expected reduction in entropy due to partitioning on x

$$Gain(S, x) \equiv Entropy(S) \; - \sum_{v \in Values(x)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Selecting the Next Attribute

- For whole training set:
  $Gain(S, Outlook) = 0.246$
  $Gain(S, Humidity) = 0.151$
  $Gain(S, Wind) = 0.048$
  $Gain(S, Temperature) = 0.029$
- $\rightarrow$ $Outlook$ should be used to split training set!

Selecting the Next Attribute

- For whole training set:
  $Gain(S, Outlook) = 0.246$
  $Gain(S, Humidity) = 0.151$
  $Gain(S, Wind) = 0.048$
  $Gain(S, Temperature) = 0.029$
- $\rightarrow Outlook$ should be used to split training set!
- Further down in the tree, $Entropy(S)$ is computed locally
- Usually, the tree does not have to be minimized
- Reason of good performance of ID3!

Real-Valued Attributes

- $Temperature = 82.5$

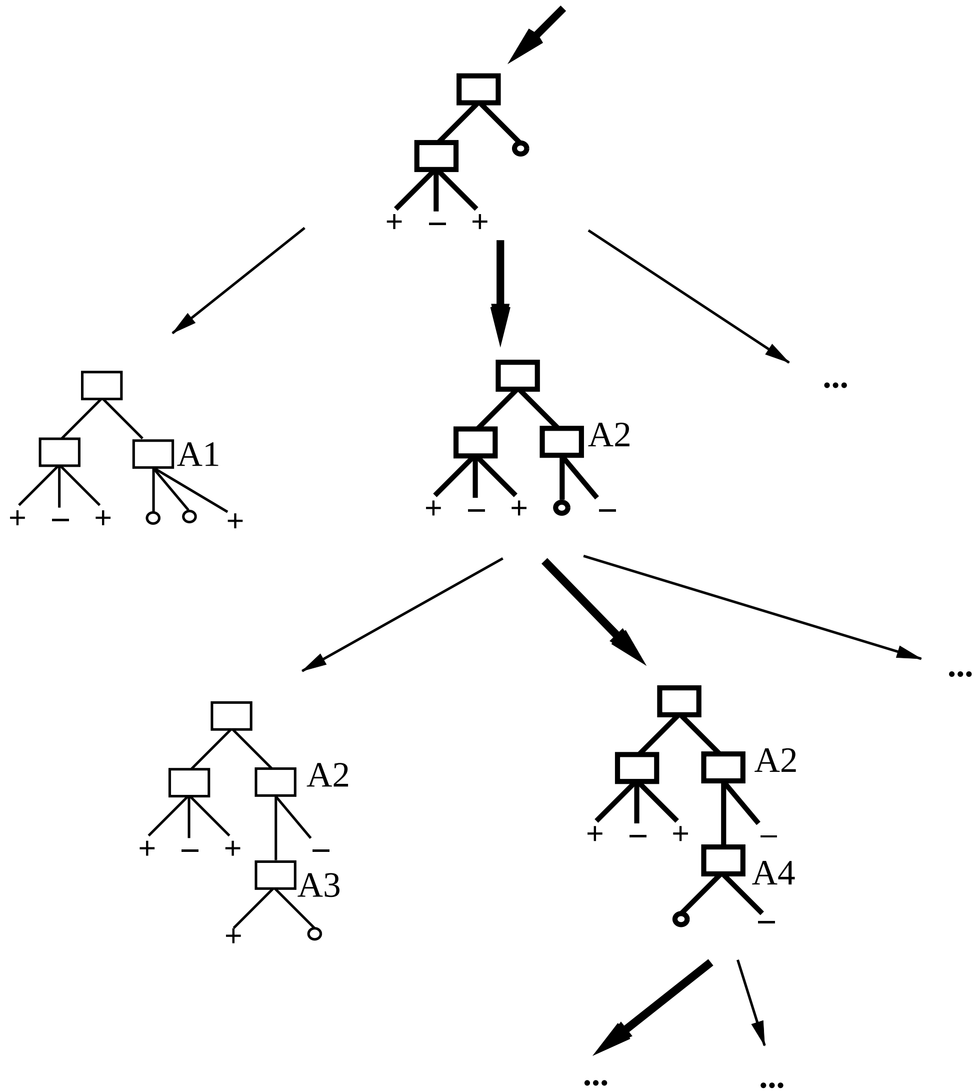
Real-Valued Attributes

- $Temperature = 82.5$
- Create discrete attributes to test continuous:
  - $(Temperature > 54) = true$ or $= false$
  - Sort attribute values that occur in training set:

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

  - Determine points where the class changes
  - Candidates are $(48 + 60)/2$ and $(80 + 90)/2$
- Select best one using info gain
- Implemented in the system C4.5 (successor of ID3)

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis H space is complete:
  - This means that every function on the feature space can be represented
  - Target function surely in there for a given training set
- The training set is only a subset of the instance space
- Generally, several hypotheses have minimal error on training set
- Best is one that minimizes error on instance space
  - ... cannot be determined because only finite training set is available
  - Feature selection is shortsighted
  - ... and there is no back-tracking $\rightarrow$ local minima...
- ID3 outputs a single hypothesis

Inductive Bias in ID3

- Inductive Bias corresponds to explicit or implicit prior assumptions on the hypothesis
  - E.g. hypothesis space H (language for classifiers)
  - Search bias: how to explore H
  - Bias here is a preference for some hypotheses, rather than a restriction of hypothesis space H

Inductive Bias in ID3

- Inductive Bias corresponds to explicit or implicit prior assumptions on the hypothesis
  - E.g. hypothesis space H (language for classifiers)
  - Search bias: how to explore H
  - Bias here is a preference for some hypotheses, rather than a restriction of hypothesis space H
- Bias of ID3:
  - Preference for short trees,
  - and for those with high information gain attributes near the root
- Occam's razor: prefer the shortest hypothesis that fits the data
- How to justify Occam's razor?

Occam's Razor

- Why prefer short hypotheses?
- Argument in favor:
  - Fewer short hyps. than long hyps.

  $\rightarrow$ A short hyp that fits data unlikely to be coincidence

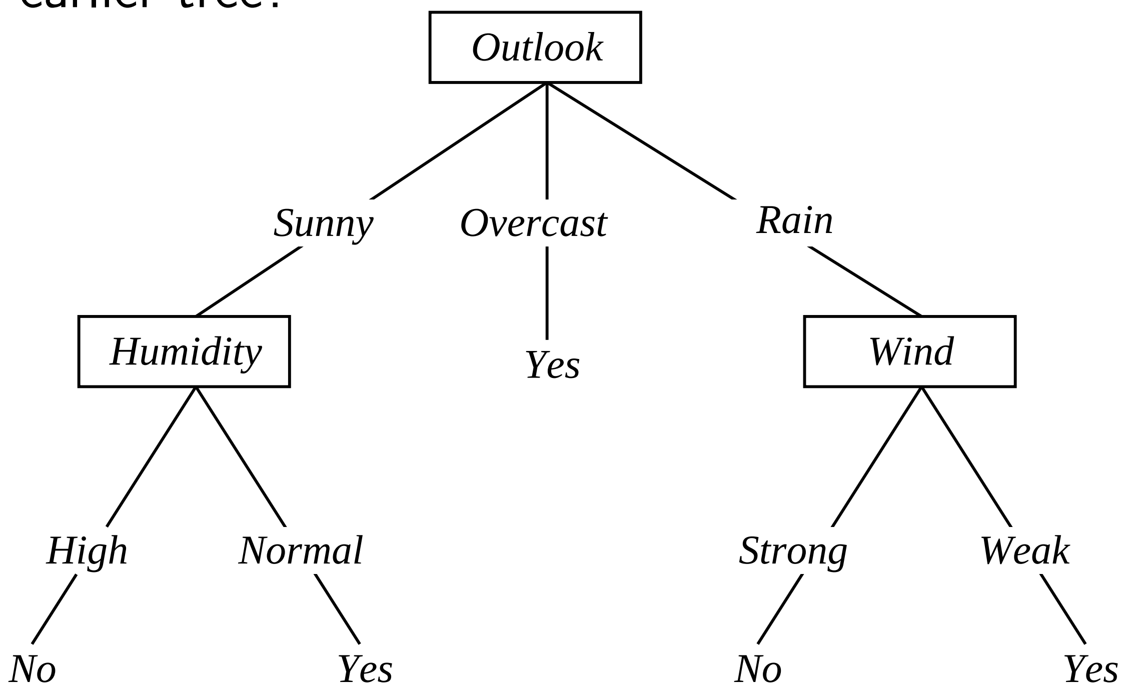
  $\rightarrow$ A long hyp that fits data might be coincidence
- refinement: compromise between short hypothesis and low training error (e.g. Bayesian approach)

Noise

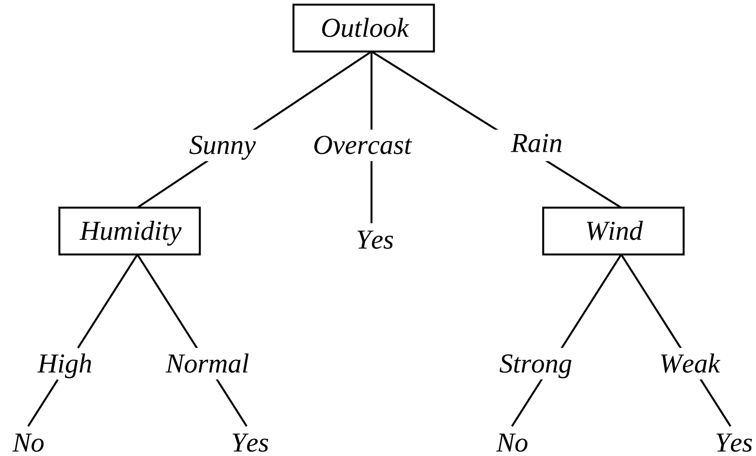
Consider adding noisy training example #15:

$$Sunny,\ Mild,\ Normal,\ Weak,\ PlayTennis = No$$

What effect on earlier tree?



Overfitting in Decision Trees



- Algorithm will introduce new test
- Unnecessary, because new example was erroneous due to the presence of Noise
- → Overfitting corresponds to learning coincidental regularities
- Unfortunately, we generally don't know which examples are noisy
- ... and also not the amount, e.g. percentage, of noisy examples

Overfitting

Consider error of hypothesis h over

- training data $(\mathbf{x}_1, k_1), \ldots, (\mathbf{x}_d, k_d)$: training error

$$error_{train}(h) = \frac{1}{d} \sum_{i=1}^{d} L(h(\mathbf{x}_i), k_i)$$

  with loss function $L(c, k) = 0$ if $c = k$ and $L(c, k) = 1$ otherwise

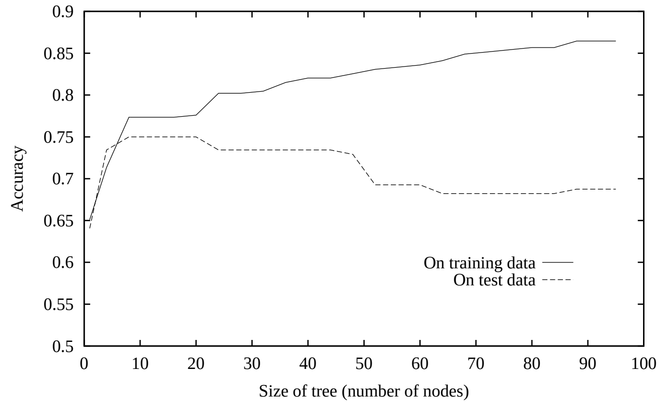
- entire distribution $\mathcal{D}$ of data $(\mathbf{x}, k)$: true error

$$error_{\mathcal{D}}(h) = P(h(\mathbf{x}) \neq k)$$

Definition Hypothesis $h \in H$ overfits training data if there is an alternative $h' \in H$ such that

$$error_{train}(h) < error_{train}(h') \qquad \text{and} \qquad error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



- The accuracy is estimated on a separate test set
- Learning produces more and more complex trees (horizontal axis)

Avoiding Overfitting

1. How can we avoid overfitting?

   - Stop growing when data split not statistically significant (pre-pruning)
     - e.g. in C4.5: Split only, if there are at least two descendant that have at least n examples, where n is a parameter
   - Grow full tree, then post-prune (post-prune)

2. How to select "best" tree:

   - Measure performance over training data
   - Measure performance over separate validation data set
   - Minimum Description Length (MDL): minimize $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

1. An example for post-pruning
2. Split data into $training$ and $validation$ set
3. Do until further pruning is harmful:
   (a) Evaluate impact on $validation$ set of pruning each possible node (plus those below it)
   (b) respective node is labeled with most frequent class
   (c) Greedily remove the one that most improves $validation$ set accuracy
4. Produces smallest version of most accurate subtree
5. But, if data is limited, splitting in training and validation set will further reduce accuracy.

Rule Post-Pruning

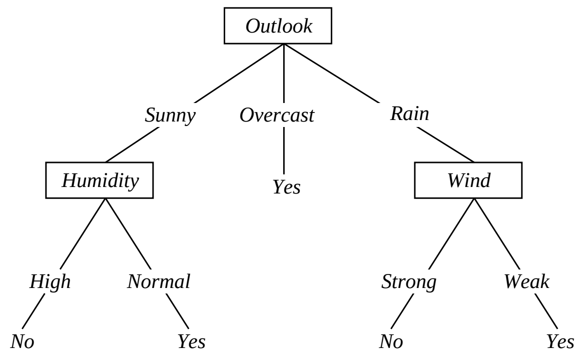
1. Grow tree from given training set that fits data best, and allow overfitting

2. Convert tree to equivalent set of rules by creating one rule for each path from root node to leaf node

3. Prune each rule by removing any preconditions that results in improving accuracy on a validation set.

- Perhaps most frequently used method (e.g., $\mathrm{C4.5}$)
- allows more fine grained pruning
- converting to rules increases understandability

Converting A Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$
IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$
. . .

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- For example, imagine using Date as attribute (very many values!) (e.g. $Date = Day1, ...$)
- Sorting by date, the training data can be perfectly classified $\rightarrow$ high information gain
- this is a general phenomen with attributes with many values, since they split the training data in small sets.
- but: generalisation suffers!

One approach: use $GainRatio$ instead

Gain Ratio

Idea: Measure how broadly and uniformly A splits the data:

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i and c is the number of different values.
Example:

- Attribute 'Date': n examples are completely seperated. Therefore: $SplitInformation(S,' Date') = \log_2 n$
- other extreme: binary attribute splits data set in two even parts: $SplitInformation(S,' Date') = 1$

By considering as a splitting criterion the

$$GainRatio(S,A) \ = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

one relates the Information gain to the way, the examples are split

Attributes with Costs

- Consider
  - medical diagnosis, $BloodTest$ has cost $150
  - robotics, $Width\_of\_obstacle$ has cost 23 sec.
- How to learn a consistent tree with low expected cost?
- One approach: replace gain by
  - Tan and Schlimmer (1990): $\frac{Gain^2(S,A)}{Cost(A)}$
  - Nunez (1988): $\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$, where w is a hyperparameter to choose.
- The algorithm tries to order the attribute with respect to the costs for testing them

Unknown Attribute Values

- What if an example x has a missing value for attribute A?
- To compute gain (S, A) two possible strategies are:
  - Assign most common value of A among other examples with same target value $c(x)$
  - Assign a probability p_i to each possible value v_i of A
- Classify new examples in same fashion

Summary

- Decision trees are a symbolic representation of knowledge
- $\rightarrow$ Understandable for humans
- Learning:
  - Incremental, e.g., CAL2
  - Batch, e.g., ID3
- Issues:
  - Assessment of Attributes (Information Gain)
  - Several extensions: continuous attributes, noisy patterns, pruning