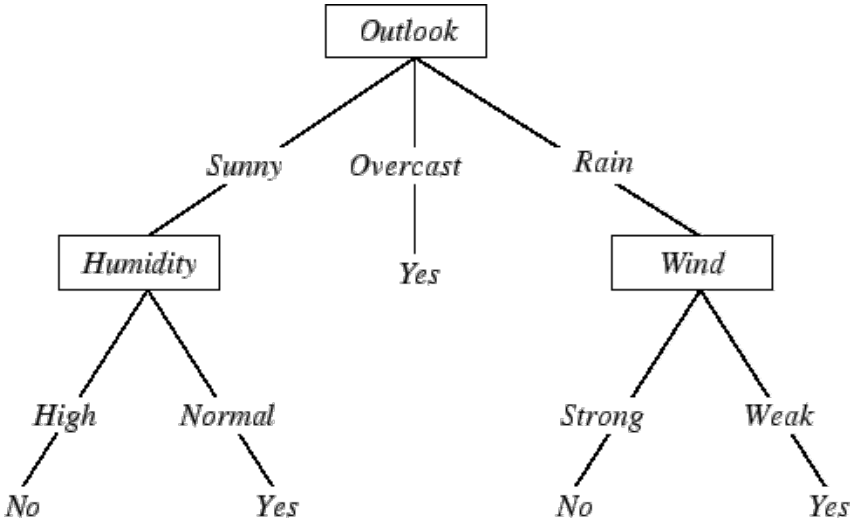
Introduction to Machine Learning

Decision Trees

PlayTennis problem

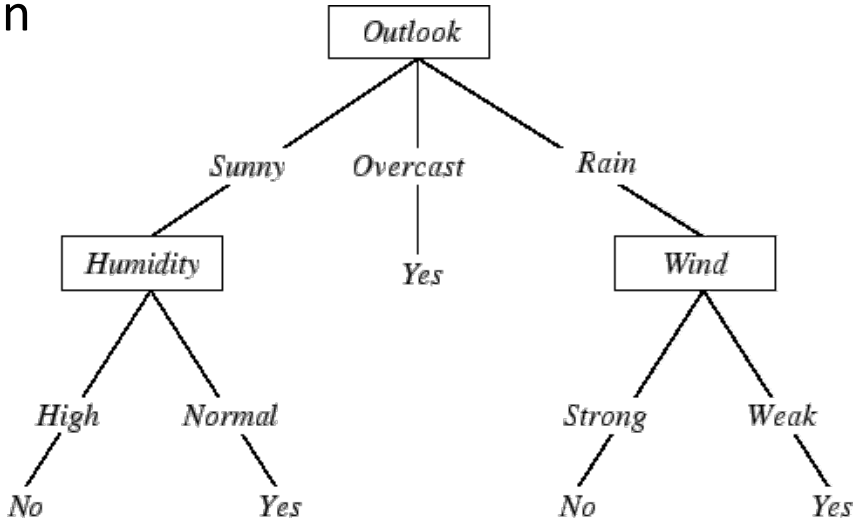
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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



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

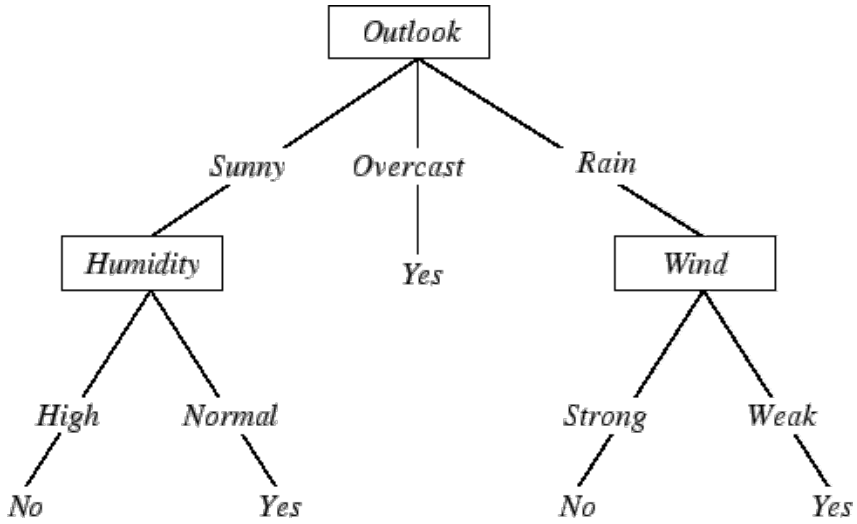
Decision tree representation

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



Decision tree characteristics

- Discrete class values
- Disjunctive hypothesis
- Can handle noisy training data



Decision tree for conjunction

Decision tree for disjunction

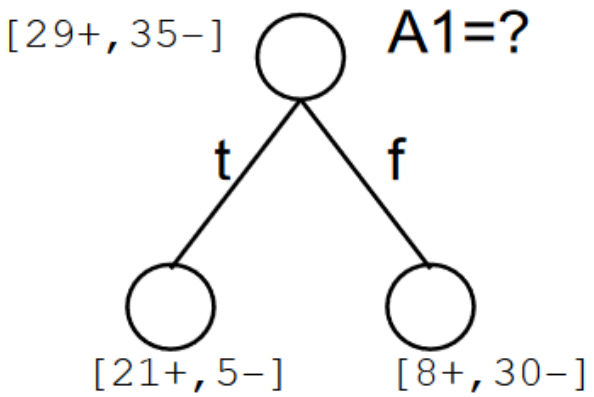
Decision tree for XOR

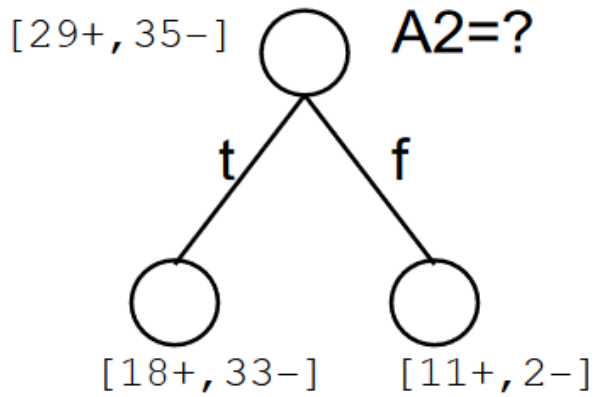
Top-down creation

Main loop:

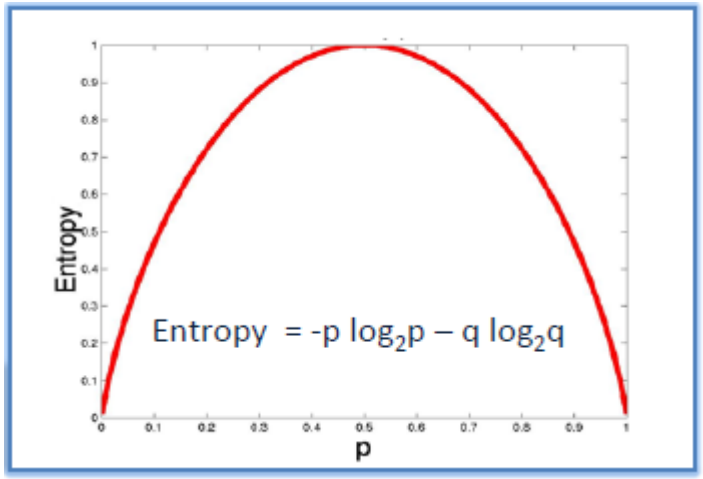
1. $A \leftarrow$ the "best" decision feature for next *node*
2. Assign A as decision feature for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which feature is best?





Entropy



- S is a sample of training examples
- Here, p is the proportion of positive examples in S
- q is the proportion of negative examples in S
- Entropy measures the impurity of S

Entropy

- Entropy(S) = expected #bits needed to encode class (+ or -) of randomly drawn element of S (using shortest-length code)
- Why?
- Information theory: optimal length code assigns $\log_2 p$ bits to message having probability p

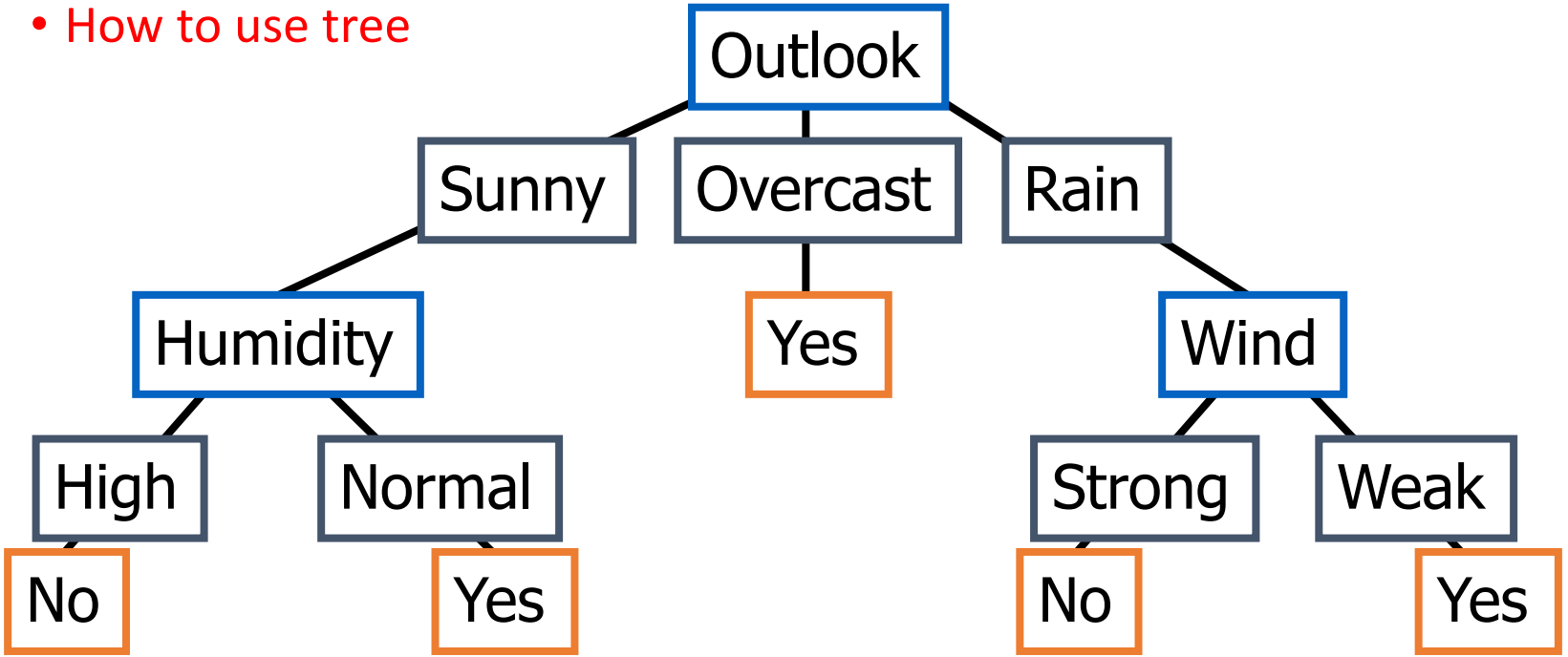
Entropy

- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- $Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$

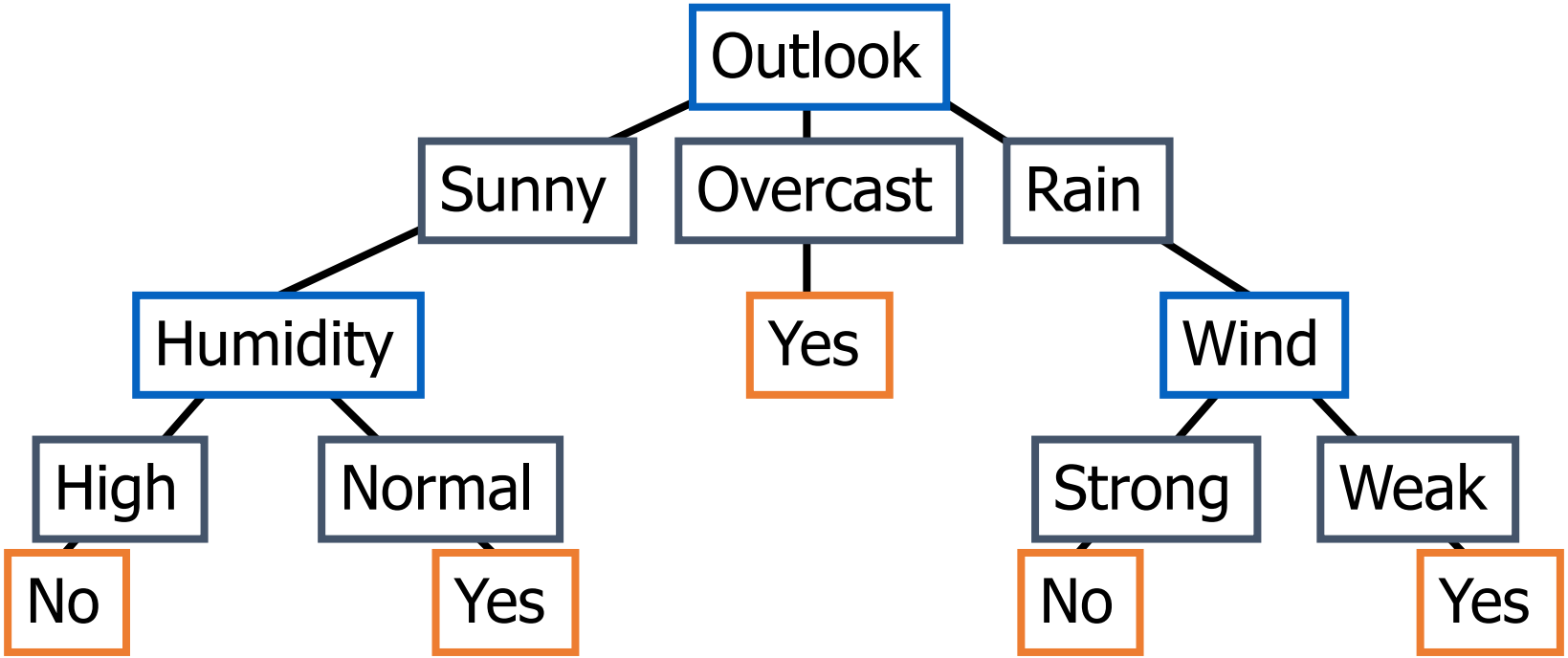
Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes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

Practical considerations

- How to use tree



Convert decision tree to rules



Practical considerations

- How to use tree
- Continuous feature values

Continuous valued attributes

Create a discrete attribute to test continuous attribute

- Temperature = 76^0F
- (Temperature > 70^0F) = {true, false}

Where to set the threshold?

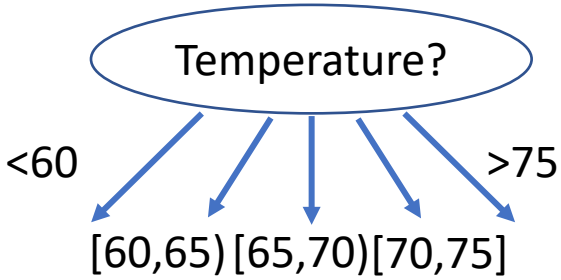
Temperature	59	64	66	71	75	90^0C
PlayTennis	No	No	Yes	Yes	Yes	No

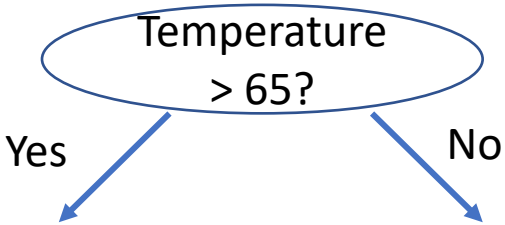
Continuous valued attributes

- Discretize
- Binary decision

Temperature	59	64	66	71	75	90^0C
PlayTennis	No	No	Yes	Yes	Yes	No

Continuous valued attributes



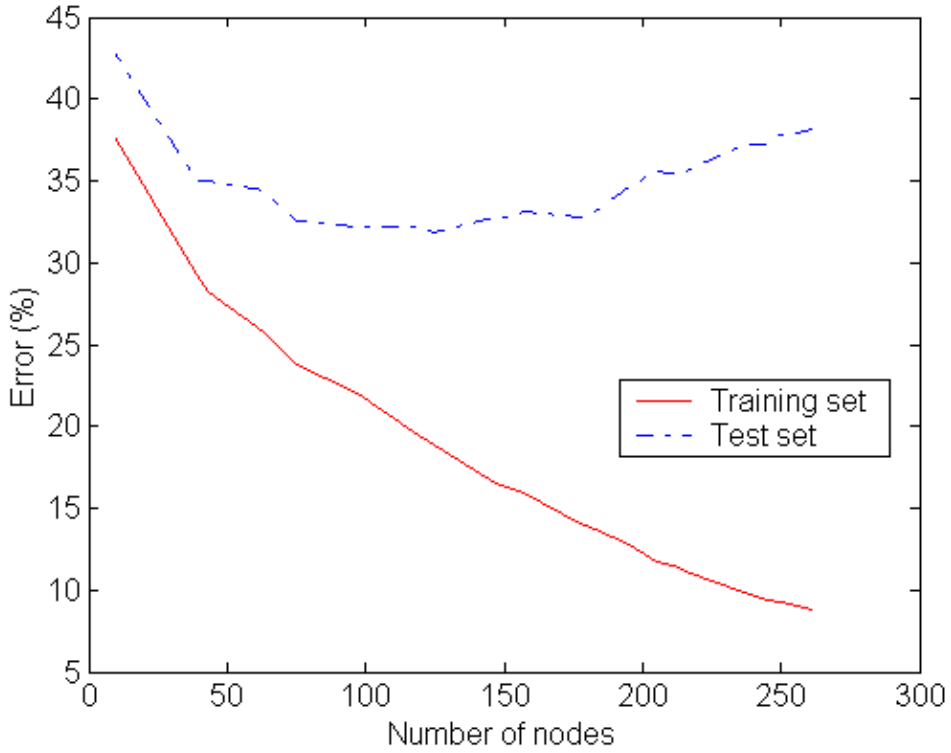


Temperature	59	64	66	71	75	90^0C
PlayTennis	No	No	Yes	Yes	Yes	No

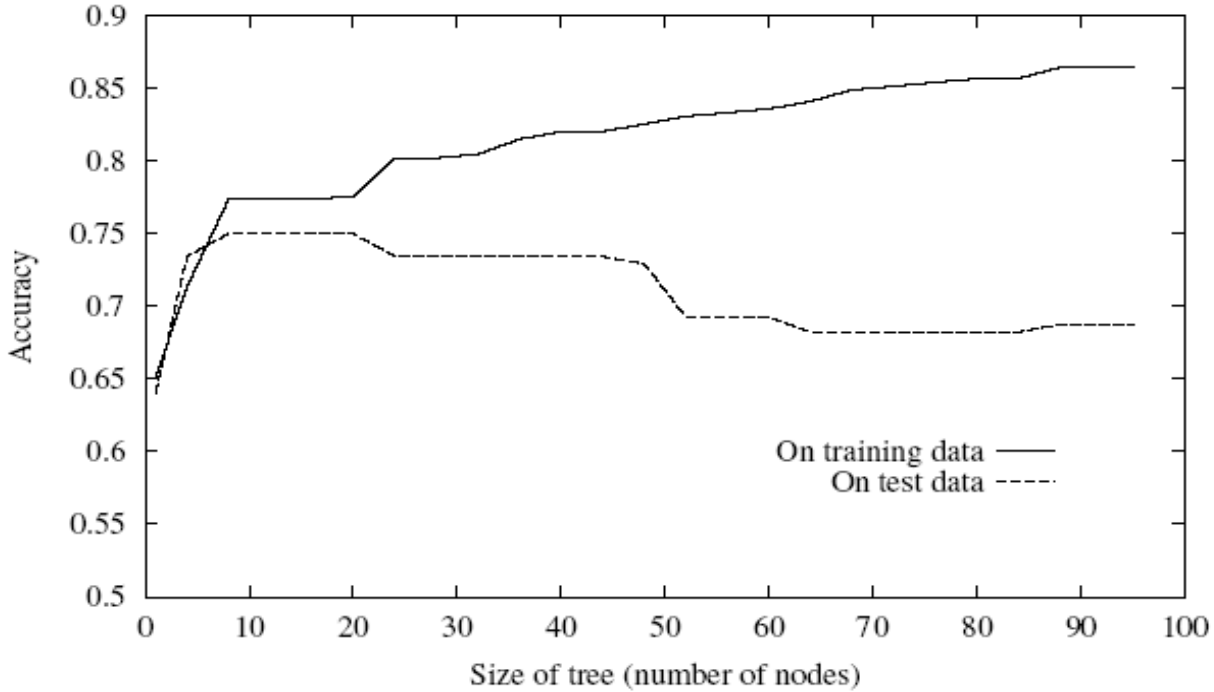
Practical considerations

- How to use tree
- Continuous feature values
- Depth limit / pruning

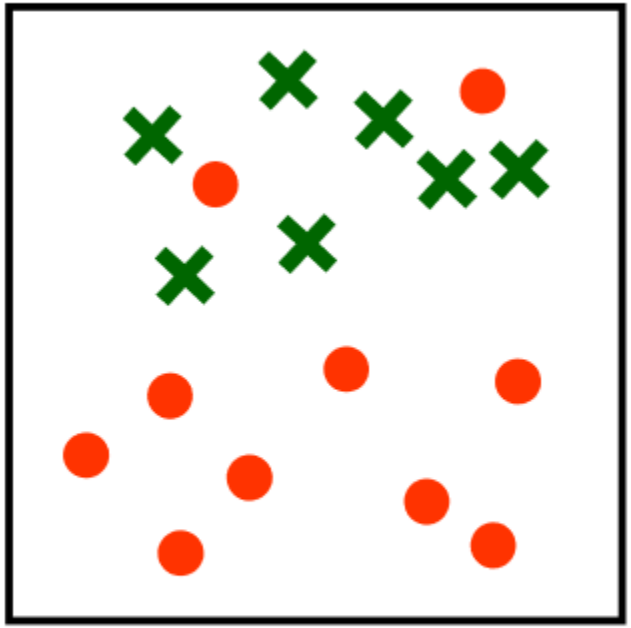
Underfitting

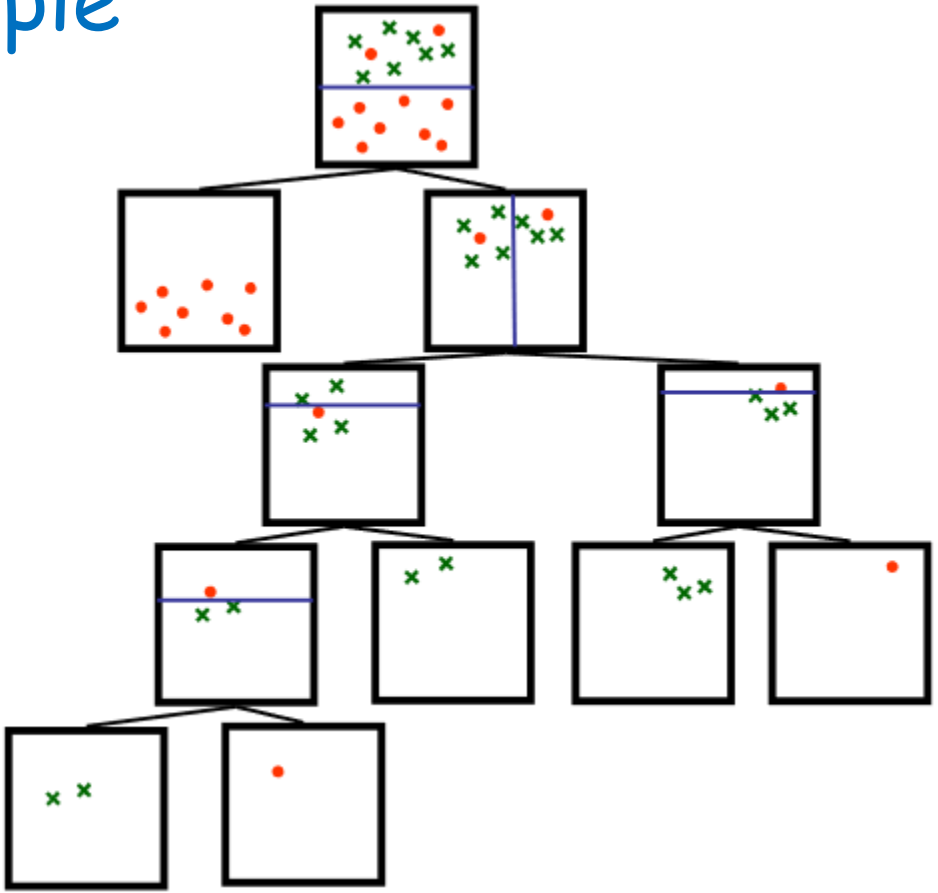


Overfitting



Overfitting example





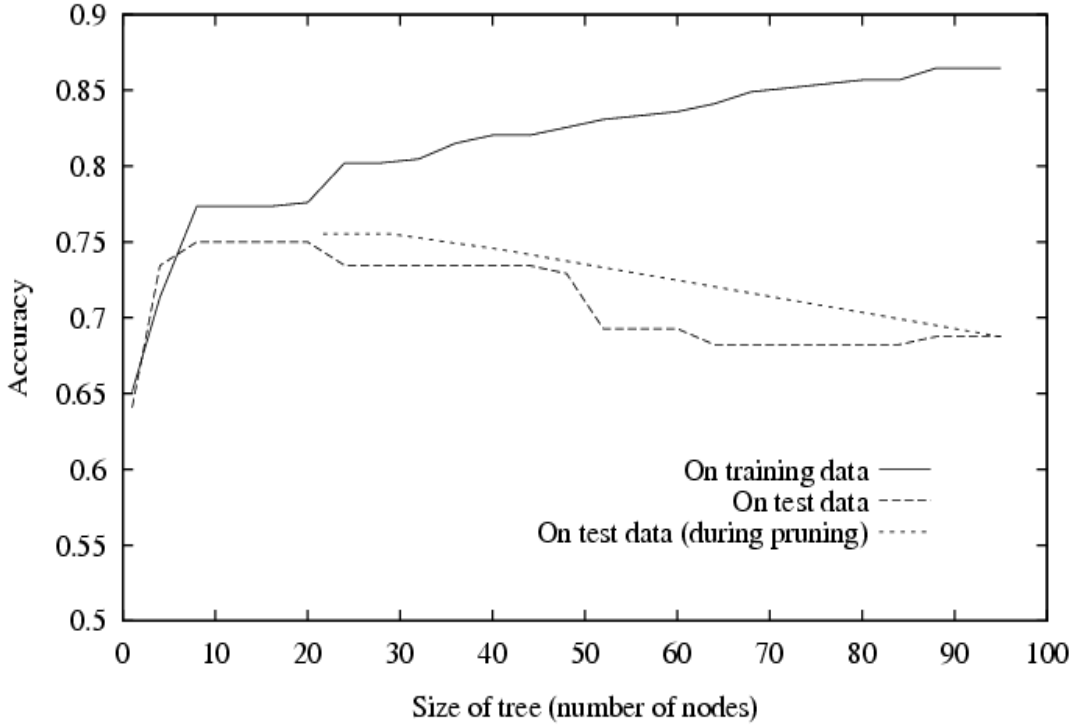
Pre-pruning

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is < threshold
  - Stop if expanding the current node does not improve information gain

Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming
  - Replace subtree by leaf
- Class label of leaf node
  - Determined from majority class of instances in the sub-tree

Effect of pruning



Practical considerations

- How to use tree
- Continuous feature values
- Depth limit / pruning
- Inconclusive leaves

Practical considerations

- How to use tree
- Continuous feature values
- Depth limit / pruning
- Inconclusive leaves
- Missing values

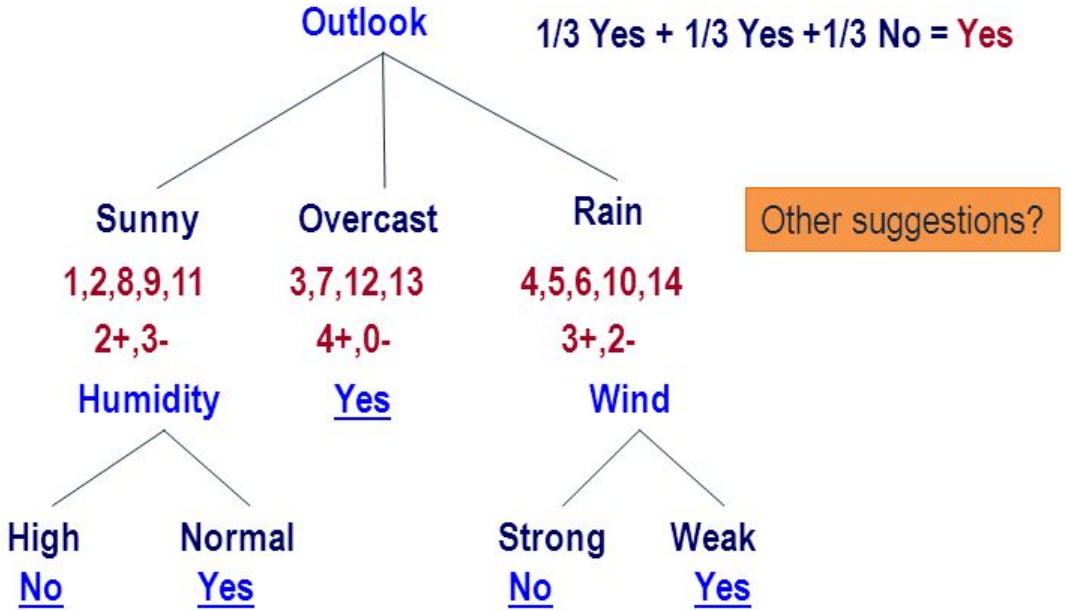
Missing feature values

- What if some examples have missing values of A?
- Use training example anyway, sort through tree
  - If node *n* tests A, assign most common value among examples in *n*.
  - Assign most common value of A among other examples with same class label
  - Assign probability p_i to each possible value v_i of A
    - Assign fraction p_i of example to each descendant in tree
- Classify new examples in the same fashion

Missing values

Outlook = Sunny, Temp = Hot, Humidity = ???, Wind = Strong, label = ?? Normal/High

Outlook = ???, Temp = Hot, Humidity = Normal, Wind = Strong, label = ??



Use decision trees?