Decision Tree Learning

- Among most popular of inductive inference algorithms
- A method for approximating discrete-valued target function
- Learned function is represented by a decision tree
- The decision tree can also be interpreted as a set of 'if-then' rules
- Ex:
  - Diagnosis medical cases, assess credit risk of loan applications

The Basic Decision Tree Learning Algorithm

- ID3 (Quinlan 1986)
  - Successor C4.5, C5.0
  - Many extensions
- ID3 learn decision tree by following a top-down approach
- At each step, it finds out
  - Q:"What attribute should be selected for the node?"
  - Ans: Compute the goodness of an attribute to classify the training examples

Which Attribute is the Best Classifier?

- How do you measure that?
- A measure called 'Information Gain'
  - Measures how well a given attribute separates the training examples according to their target classification
- How to measure 'Information Gain'?
  - Here we use a concept call 'Entropy' from information theory

Building a Decision Tree (ID3 Algorithm)

- Assume attributes are discrete
  - Discretize continuous attributes
- Choose the attribute with the highest Information Gain
- Create branches for each value of attribute
- Examples partitioned based on selected attributes
- Repeat with remaining attributes
- Stopping conditions
  - All examples assigned the same label
  - No examples left

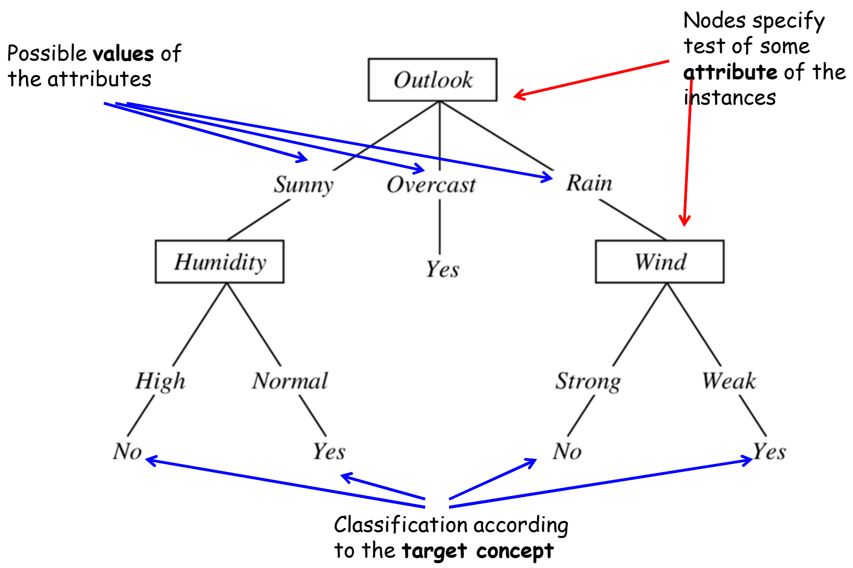
The ID3 Algorithm

ID3 (Examples, Target_Attribute, Attributes)

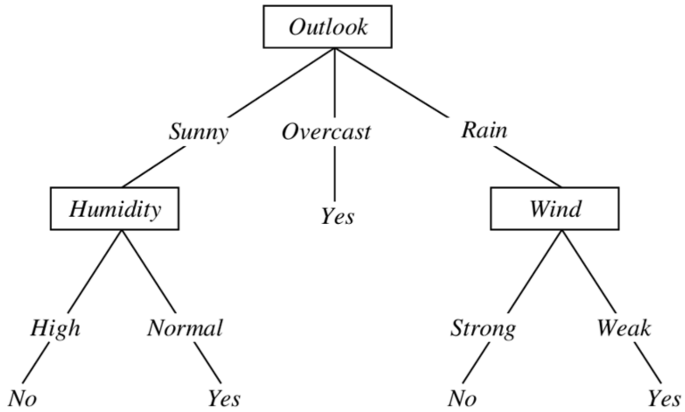
- Create a root node for the tree
- If all examples are positive, Return the single-node tree Root, with label = +.
- If all examples are negative, Return the single-node tree Root, with label = -.
- Otherwise Begin
  - A ← The Attribute that best classifies examples.
  - Decision Tree attribute for Root = A.
  - For each possible value, V_i, of A,
    - Add a new tree branch below Root, corresponding to the test A = V_i
    - Let Examples(V_i) be the subset of examples that have the value V_i for A
    - If Examples(V_i) is consists of only one class of example
      - Then below this new branch add a leaf node with label = most common target value in the examples
    - Else below this new branch add the subtree ID3 (Examples(V_i) Target_Attribute, Attributes – {A})
- End

Return Root

Decision Tree for Play Tennis



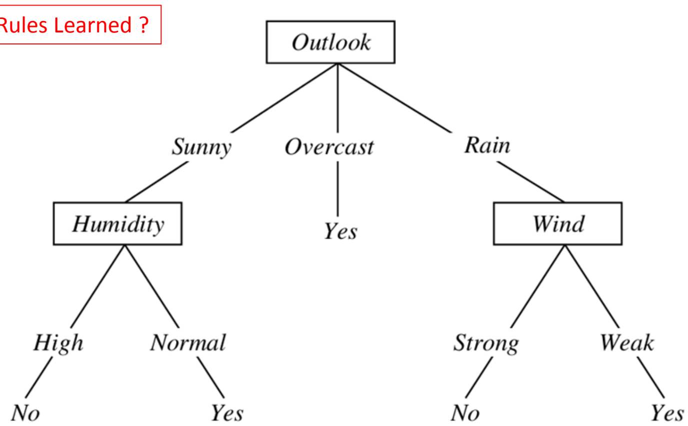
Decision Tree for Play Tennis



<Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong, Play Tennis = ?>

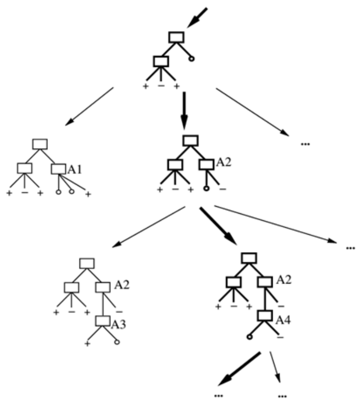
Decision Tree for Play Tennis

The Rules Learned ?



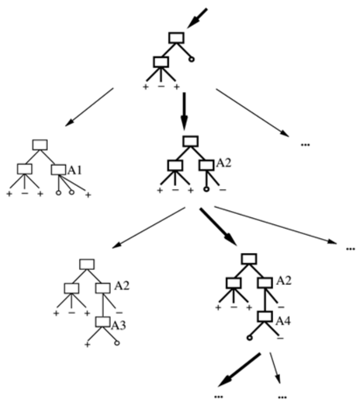
Hypothesis Space Search in DTL

- Hypothesis Space – Set of all decision trees
- Search is heuristic based
- No backtracking
- Evaluation Function
  - Information Gain
- Inductive bias?



Hypothesis Space Search in DTL

- Inductive bias?
  - "Shorter trees are preferred over the others!"
- Why prefer shorter hypotheses?
- Occam's razor
  - "**Prefer the simplest hypothesis that fits the data**"



Issues in Decision Tree Learning

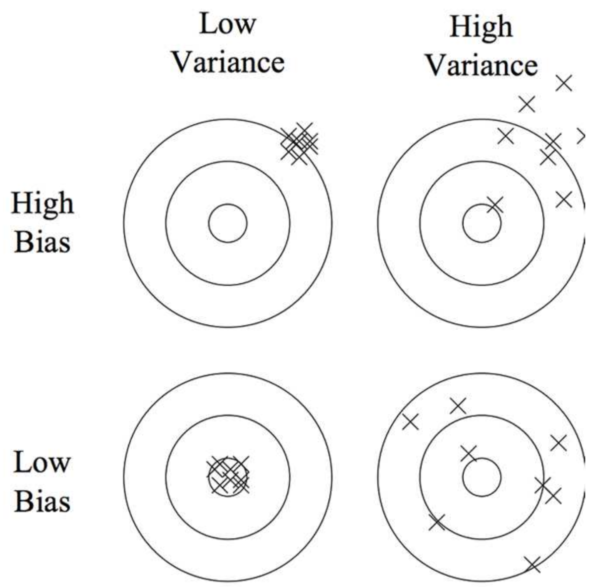
1. Over-fitting of training data
2. Handling continuous attribute
3. Choosing an appropriate attribute selection measure
4. Handling training data with missing attribute values
5. Expensive to Train

1. Over-fitting of Training Data

- ID3 provides a decision tree that perfectly classify training examples
- Difficulties leading to over-fitting
  - Noise in data
  - No of training examples are too small

Underfitting:
    High Bias
    Low Variance

Overfitting:
    Low Bias
    High Variance



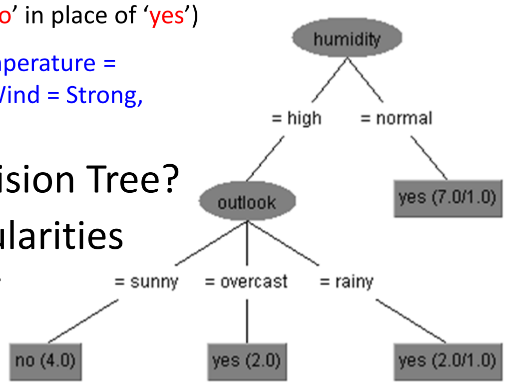
src: Domingo 2012

Over-fitting of Training Data

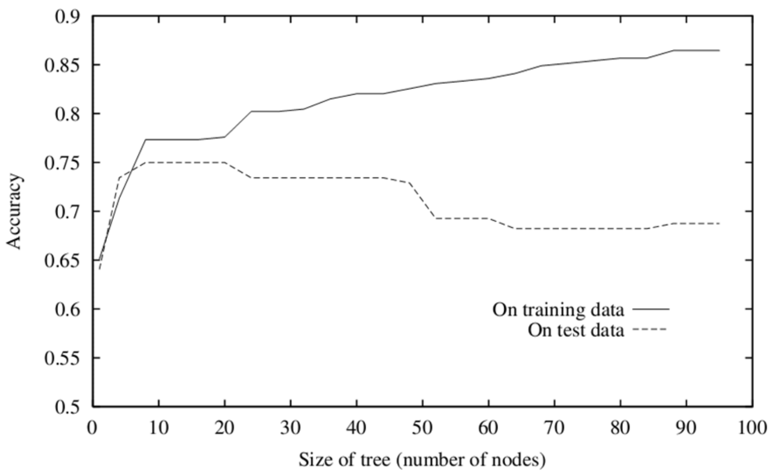
- An Example: Play Tennis
- Consider the following instance in the training example (labeled as 'No' in place of 'yes')

  D15 <Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = **No**>

- The resulting Decision Tree?
- Coincidental regularities can also cause OF



Over-fitting of Training Data



Learning Task: which medical patients have a form of diabetes?

Avoiding Over-fitting

1. Approaches that stop growing the tree earlier
2. Approaches that allow the tree to over-fit the data, and then post-prune the tree

- How to decide the final size of the tree?
  – Use separate set of examples for both training and testing of the learned tree, i.e., a training and validation set approach

Is the model able to *generalize*? Can it deal with unseen data, or does it overfit the data? Test on hold-out data:

- split data to be modeled in training and test set
- train the model on training set
- evaluate the model on the training set
- evaluate the model on the test set
- difference between the fit on training data and test data measures the model's ability to *generalize*

Evaluation

Division into training and test sets

- Fixed
  - Leave out random N% of the data
- k-fold Cross-Validation
  - Select K folds without replace
- Leave-One-Out Cross Validation
  - Special case
- Related: Bootstrap
  - Generate new training sets by sampling with replacement

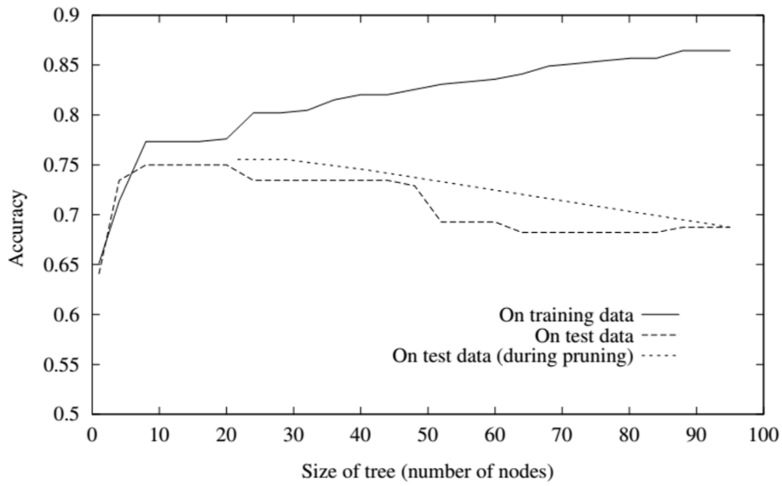
The Bootstrap

- Given a dataset of size N
- Draw N samples with replacement to create a new dataset
- Repeat ~1000 times
- You now have ~1000 sample datasets
  - All drawn from the same population
  - You can compute ~1000 sample statistics
  - You can interpret these as repeated experiments, which is exactly what the frequentist perspective calls for
- Very elegant use of computational resources

Avoiding Over-fitting

- Tree Pruning
  1. Reduced Error Pruning
  2. Rule Post Pruning

The Impact of Pruning



Reduced Error Pruning

Rule Post Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

2. Handling Continuous Attributes

C4.5 Extension

outlook	temperature	humidity	windy	play
overcast	cool	60	TRUE	yes
overcast	hot	80	FALSE	yes
overcast	hot	63	FALSE	yes
overcast	mild	81	TRUE	yes
rainy	cool	58	TRUE	no
rainy	mild	90	TRUE	no
rainy	cool	54	FALSE	yes
rainy	mild	92	FALSE	yes
rainy	mild	59	FALSE	yes
sunny	hot	90	FALSE	no
sunny	hot	89	TRUE	no
sunny	mild	90	FALSE	no
sunny	cool	60	FALSE	yes
sunny	mild	62	TRUE	yes

outlook	temperature	humidity	windy	play	
rainy	mild	54	FALSE	yes	E(6/6) = 0.0
overcast	hot	58	FALSE	yes	
overcast	cool	59	TRUE	yes	
rainy	cool	60	FALSE	yes	
overcast	mild	60	TRUE	yes	
overcast	hot	62	FALSE	yes	
rainy	mild	63	TRUE	no	E(3/8) + E(5/8) = 0.95
sunny	cool	80	FALSE	yes	
rainy	mild	81	FALSE	yes	
sunny	mild	89	TRUE	yes	
sunny	hot	90	FALSE	no	
rainy	cool	90	TRUE	no	
sunny	hot	90	TRUE	no	
sunny	mild	92	FALSE	no	

Expect = 8/14*0.95 + 6/14*0
= 0.54

Consider every possible binary partition; choose the partition with the highest gain

outlook	temperature	humidity	windy	play
rainy	mild	54	FALSE	yes
overcast	hot	58	FALSE	yes
overcast	cool	59	TRUE	yes
rainy	cool	60	FALSE	yes
overcast	mild	60	TRUE	yes
overcast	hot	62	FALSE	yes
rainy	mild	63	TRUE	no
sunny	cool	80	FALSE	yes
rainy	mild	81	FALSE	yes
sunny	mild	89	TRUE	yes
sunny	hot	90	FALSE	no
rainy	cool	90	TRUE	no
sunny	hot	90	TRUE	no
sunny	mild	92	FALSE	no

E(6/6) = 0.0

E(3/8) + E(5/8) = 0.95

E(9/10)+ E(1/10) = 0.47

E(4/4) = 0.0

Expect = 8/14*0.95 + 6/14*0
= 0.54

Expect = 10/14*0.47 + 4/14*0
= 0.33

3. Alternatives for Selecting Attributes

One approach: use $GainRatio$ instead

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

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

where S_i is subset of S for which A has value v_i

4. Dealing with Missing Attribute Values

- Missing attribute values are simply not used in gain and entropy calculations in C4.5

C4.5/J48 Decision Tree Algorithms

- Differences from ID3
  - Handling both continuous and discrete attributes
  - Handling training data with missing attribute values
  - Pruning trees after creation
  - A more robust, powerful Information Gain measure
- J48 is Open Source Java implementation of C4.5 in Weka tool
- Industrial applications

What we have learned so far …

1. We have to **choose a bias** toward selecting a potential approximate target function
2. Whatever functions we would like to learn from training examples are likely to have **some error** with future examples
3. If we try to learn too much from training example, may lead to **over fitting** …
4. So the all important question is - **How good are the hypotheses learned?**

Decision tree: Summary

- An efficient method of learning discrete value target function
- Inductive bias in ID3 have preference for smaller trees
- Over-fitting on training data is an important issue for which Tree-pruning can help to some extent
- Other issues
  - Handling continuous-valued attributes
  - Missing attribute values
  - Selection measures like information gain