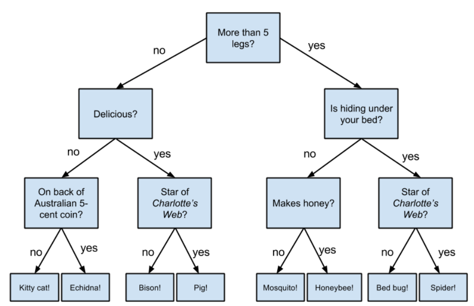
Machine Learning

Decision Trees

The “Animal” Game



Decision Tree Learning

- A **decision tree** is a supervised learning algorithm used for both classification and regression problems.
- It is a method for approximating discrete-valued functions that is robust to noisy data and is capable of learning disjunctive expressions.
- Disjunctive Expressions are of the form:

$$(A \wedge B \wedge C) \vee (D \wedge E \wedge F)$$

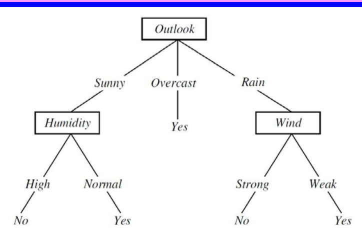
- The learned function is represented as a tree.

Decision Tree Learning Algorithms

There are many different decision tree learning algorithms:

- ID3
- C4.5
- CART
- CHAID
- MARS

Example Decision Tree



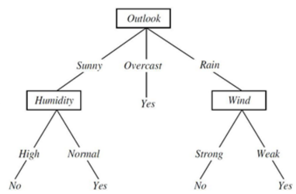
PlayTennis: This decision tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

Decision Tree Properties

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

Decision Tree Classification

An example is classified by sorting it through the tree from the root to a leaf node.



For instance:

(Outlook = Sunny, Humidity = High) → (PlayTennis = No)

Good Problems for Decision Trees

- Problems with instances describable by attribute-value pairs.
- Problems whose target function is discrete-valued.
- Problems where a disjunctive hypothesis may be required.
- Problems with noisy data, or where the training data may contain missing attribute values.
- Examples:
  - Medical diagnosis
  - Credit risk analysis

ID3 Learning Algorithm

- The algorithm operates through the top-down construction of a tree, beginning with the question “Which attribute should be tested at the root?”.
- Each attribute is evaluated using a statistical test to determine how well it alone classifies the training examples.
- The best attribute is selected and used as the test for the root node of the tree.

ID3 Learning Algorithm (2)

- A descendant of the root node is then created for each possible value of this attribute.
- The training examples are then sorted into the appropriate descendant node.
- The entire process is then repeated for each descendant node using the training examples associated with that descendant node.

ID3 Algorithm (3)

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- 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 = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - 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 value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* $- \{A\}$))
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

Top-Down Induction

1. Find A, the best decision attribute for the next node.
2. Assign A as the decision attribute.
3. For each value of A, create the new descendants of that node.
4. Sort the training examples into the descendant nodes.
5. If the training examples are all classified then STOP, else repeat this process over the new descendant nodes.

Entropy

- **Entropy** is the measure of disorder or uncertainty in the data.
- If the target attribute can take *c* different values, then entropy is measured as:

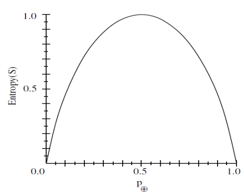
$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

- If the attribute is Boolean with only positive and negative values, this becomes

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy (2)

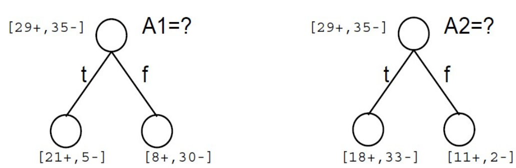
- Entropy varies between 0 and 1.
- Entropy is 0 if all members belong to the same class.
- Entropy is 1 for a binary attribute if there is an equal number of positive and negative examples.



Entropy Example:

Entropy([29+, 35-]) =

- (29/64) $\log_2$(29/64) - (35/64) $\log_2$(35/64) = 0.994



Information Gain

- **Information Gain** is a statistical property that measures how well a given attribute separates the training examples according to their target classification.
- This measure is used to select among the candidate attributes at each step while growing the tree.

Information Gain (2)

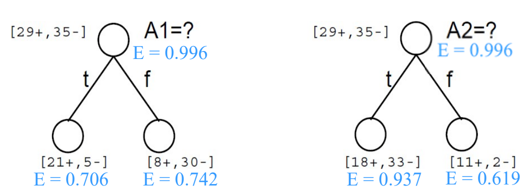
- Gain(S,A) is the expected reduction in entropy due to sorting on A.
- Information Gain is measured as:

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

- S_v is entropy of each attribute value.
- $|S_v/S|$ is the fraction of the total examples of S that belong to attribute value S_v.

Information Gain Example

Gain(S,A1) =
0.994 – (26/64)*.706 – (38/64)*.742 = **0.266**
Gain(S,A2) =
0.994 – (51/64)*.937 – (13/64)*.619 = **0.121**



ID3 Example

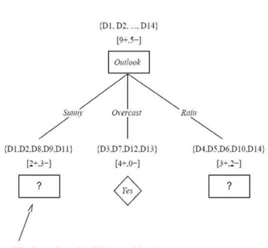
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool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

ID3 Example (2)

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

- Gain(*S*, *Outlook*) = 0.246
- Gain(*S*, *Humidity*) = 0.151
- Gain(*S*, *Wind*) = 0.048
- Gain(*S*, *Temperature*) = 0.029
- Since the *Outlook* attribute provides the best prediction of the target attribute ***PlayTennis,*** it is selected as the decision attribute for the root node, and branches are then created for the other possible attributes (*Sunny, Overcast,* and *Rain*).

ID3 Example (3)



- S_{sunny} = {D1,D2,D8,D9,D11}
- Gain (S_{sunny}, *Humidity*)
  = .970 - (3/5) 0.0 - (2/5) 0.0
  = .970
- Gain (S_{sunny}, *Temperature*)
  = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0
  = .570
- Gain (S_{sunny}, *Wind*)
  = .970 - (2/5) 1.0 - (3/5) .918
  = .019

Inductive Bias in ID3

- **Inductive bias** is the set of assumptions that along with the training data justify the classifications assigned by the learner to future instances.
- ID3 has a preference for short trees with high information gain attributes near the root.
- ID3 has a preference for certain hypothesis forms over others.

Occam's Razor

- **Occam's Razor** states that "entities should not be multiplied beyond necessity".
- Argument in favor:
  - A short hypothesis that fits the data is *unlikely* to be a coincidence.
  - A long hypothesis that fits the data *might* be a coincidence.
- Argument opposed:
  - There are many ways to define a small set of hypotheses.
  - Two different hypotheses from the same training set are possible.

Issues with Decision Tree Learning

- Overfitting
- Incorporating continuous-valued attributes
- Attributes with many values
- Handling examples with missing attribute values

Overfitting

- Causes:
  - the training data contains errors or noise
  - a small numbers of examples are associated with leaf nodes
- Avoiding Overfitting:
  - stop growing the tree when the data split is not statistically significant
  - grow a full tree, then “prune” it afterwards

Continuous-Valued Attributes

- Create a discrete-valued attribute to test the continuous one.
- As an example, if Temperature = 75, we can infer that PlayTennis = Yes.

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

Attributes with Many Values

- If an attribute has many values, Gain will select any value.
- A good example would be the use of a *date* attribute.
- One approach to avoid this is to use Gain Ratio:

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

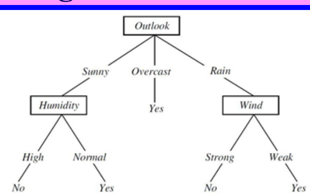
$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- S_i is the subset of S that has attribute value v_i

Examples with Missing Values

- If some examples are missing attribute values, use the training examples anyway and sort through the tree:
  - If node *n* tests *A*, assign the most common value to any missing values among the examples at node *n*.
  - Assign a probability p_i to each possible value of *A* and assign a fraction p_i of examples to each descendant in the tree.

Converting Trees to Rules

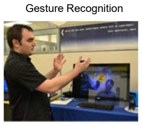


IF (Outlook = Sunny) ∧ (Humidity = High) THEN PlayTennis = No

IF (Outlook = Sunny) ∧ (Humidity = Normal) THEN PlayTennis = Yes

Latest Applications

Gesture Recognition



Motion Detection

Xbox 360 Kinect