# EID403: MACHINE LEARNING

Abhishikta Ummadi

GITAM Hyderabad

July 17, 2019

# Outline

1. Decision Tree Learning
   - Decision Tree Representation
   - Problems For Decision Tree Learning
   - The basic Decision Tree learning algorithm
   - Hypothesis space search in decision tree learning

# Decision Trees

## Decision Trees

- Consider Decision making process: For example, What to do this weekend?
  - If my parents are visiting
    - We'll go for movie
  - If not
    - Then, if its sunny, I'll play tennis.
    - But if it's windy and I'm rich, I'll go shopping.
    - But if it's windy and I'm poor, I'll go for a movie.
    - if it's rainy, I'll stay home.
- Decision Tree is a tree like graph which classifies examples by sorting them down the tree from root to some leaf node.

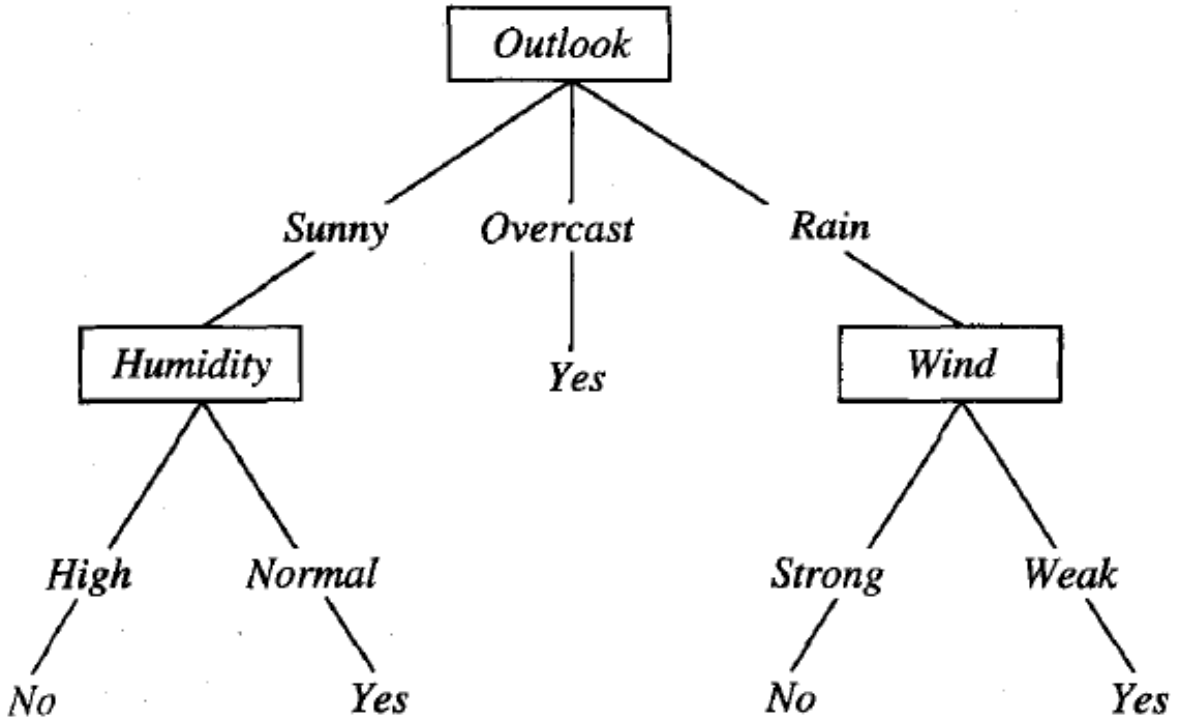
# Decision Tree Learning

- Decision tree learning : a method for approximating target functions, in which learned function is represented by a Decision Tree.
- Learned tress can also be re-represented as sets of if then rules to improve human readability.
- These learning algorithms have successfully applied to a broad range of tasks from training to diagnose medical cases to learning to assess credit risk of loan applicants.

# Decision Tree Representation

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.
- Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.
- An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example.
- This process is then repeated for the sub-tree rooted at the new node.

Example: A Decision Tree for concept *PlayTennis*. This tree classifies Sturday mornings according to whether or not they are suitable for playing tennis.



- The Decision Tree classifies Saturday Mornings according to whether they are suitable for playing tennis. For eg., instance (leftmost branch of given decision tree)

  $\langle Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = stron$

  This is classified as negative. i.e. PlayTennis= No.
- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.
  For example,

  $$(Outlook = sunny \wedge Humidity = normal) \vee (Outlook = overcast) \vee (Outlook = rain \wedge Wind = weak)$$

- There are many specific decision-tree algorithms. Notable ones include:
- ID3: (Iterative Dichotomiser 3)
- C4.5: (successor of ID3)
- CART: (Classification And Regression Tree)
- CHAID: (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.
- MARS: extends decision trees to handle numerical data better.

# Few Advantages Of Decision Tree

- Simple to understand and interpret
- Able to handle both numerical and categorical data
- Requires little data preparation
- Possible to validate a model using statistical tests
- Performs well with large datasets
- Mirrors human decision making more closely than other approaches
  Limitations:
- Trees do not tend to be as accurate as other approaches
- A small change in the training data can result in a big change in the tree

# Appropriate Problems For Decision Tree Learning

- Decision tree learning is generally best suited to problems with the following characteristics:
  - *Instances are represented by attribute-value pairs*: Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot).
  - *The target function has discrete output values*: For example: Output values for taken example are Yes or No. Decision tree methods easily extend to learning functions with more than two possible output values.
  - *Disjunctive descriptions may be required*: Decision trees naturally represent disjunctive expressions.
  - *The training data may contain missing attribute values*: Decision tree methods can be used even when some training examples have unknown values
  - *The training data may contain errors*: Decision tree learning methods are robust to errors.

# THE BASIC DECISION TREE LEARNING ALGORITHM

- Basic algorithm, ID3, learns decision trees by constructing them topdown, beginning with the question "which attribute should be tested at the root of the tree?".
- To answer this question, each instance 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 at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node.
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.
- This forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.

# ID3 Algorithm

---

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*

---

# Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree. We would like to select the attribute that is most useful for classifying examples.
- A statistical property, **Information gain**, that measures how well a given attribute separates the training examples according to their target classification.
- In order to define information gain precisely we calculate **entropy**- that characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this Boolean classification is

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

where $p_{\oplus}$, is the proportion of positive examples in S and $p_{\ominus}$, is the proportion of negative examples in S. In all calculations involving entropy we define 0 log 0 to be 0.

# An Example

- 9 +ve , 5 –ve examples.
- Training examples for the target concept PlayTennis.

| 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 |

To illustrate, suppose S is a collection of 14 examples of some Boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-1 to summarize such a sample of data). Then the entropy of S relative to this Boolean classification is

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

Notice that the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_{\oplus} = 1$), then $p_{\ominus}$ is 0, and Entropy(S) $= -1.log_2(1) - 0.log_20 = -1.0 - 0.log20 = 0$. Note the entropy is 1 when the collection contains an equal number of positive and negative examples. If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and 1.

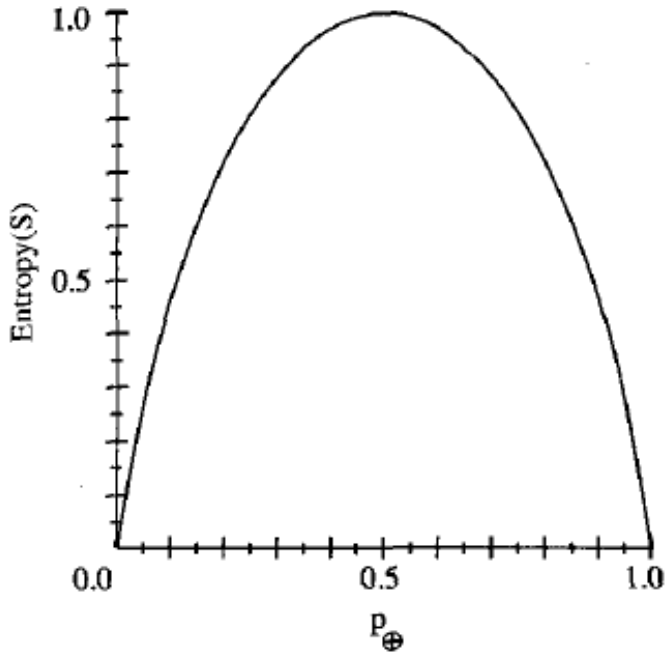



**FIGURE 3.2**
The entropy function relative to a boolean classification, as the proportion, $p_{\oplus}$, of positive examples varies between 0 and 1.

Figure 3.2 shows the form of the entropy function relative to a Boolean classification, as p, varies between 0 and 1.

- Thus far we have discussed entropy in the special case where the target classification is Boolean. More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

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

where $p_i$ is the proportion of S belonging to class i.

# Information Gain Measures The Expected Reduction In Entropy

- Entropy: Impurity in a collection of training examples.
- Information gain: A measure of the effectiveness of an attribute in classifying the training data. Information gain is simply expected reduction in entropy caused by partitioning the examples according this attribute.
- The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

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

where Values(A) is the set of all possible values for attribute A, and $S_v$, is the subset of S for which attribute A has value v (i.e., $S_v$, $= \{s \epsilon S \mid A(s) = v\}$).

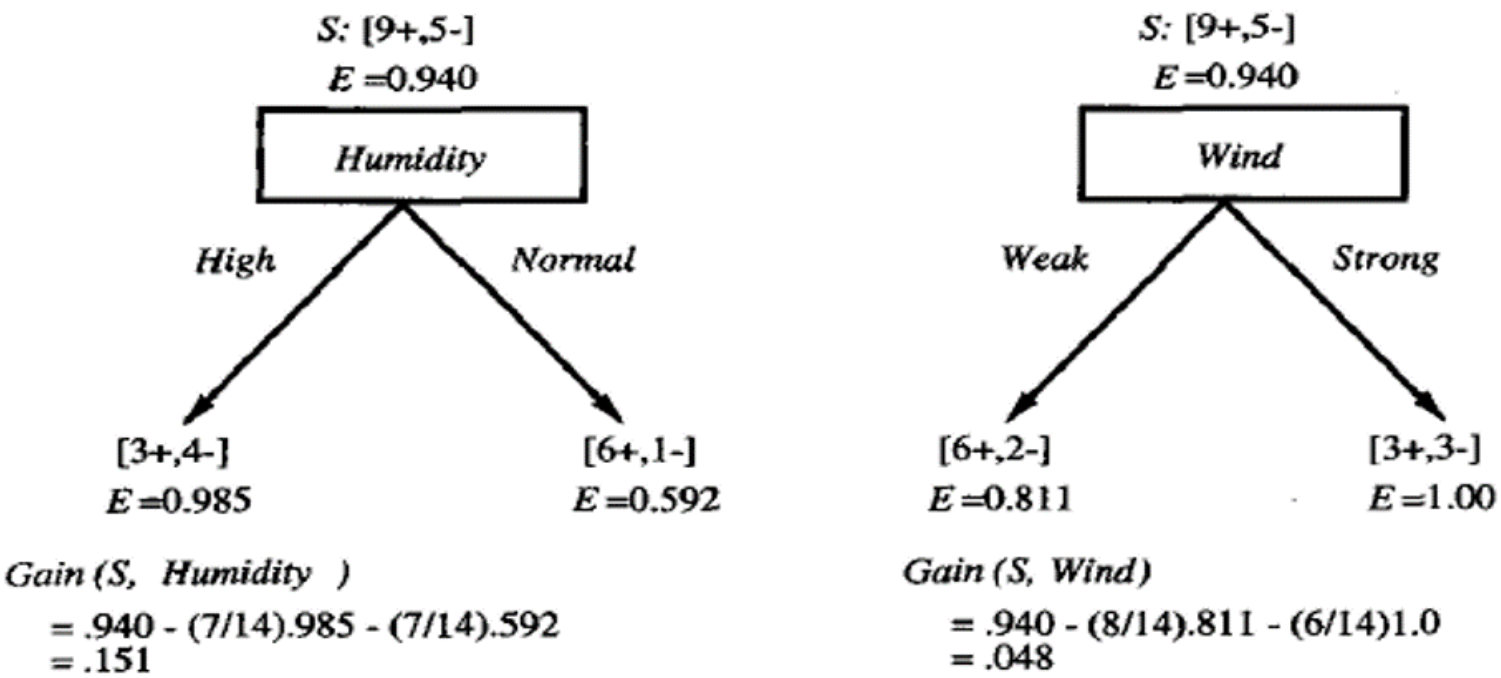
- The first term in Equation is just the entropy of the original collection S.
- The second term is the expected value of the entropy after S is partitioned using attribute A.
- The expected entropy described by this second term is simply the sum of the entropies of each subset $\frac{|S_v|}{|S|}$, weighted by the fraction of examples that belong to $S_v$.
- Gain(S, A) is therefore the expected reduction in entropy caused by knowing the value of attribute A.

For example, suppose S is a collection of training-example days described by attributes including Wind, which can have the values Weak or Strong. Assume S contains 14 examples, [9+, 5-]. Out of these 14 examples, suppose 6 of the positive and 2 of the negative examples have Wind = Weak, and the remainder have Wind = Strong. The information gain due to sorting the original 14 examples by the attribute Wind may then be calculated as

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

- The use of information gain to evaluate the relevance of attributes is summarized in Figure.
- In this figure, the information gain of two different attributes, Humidity and Wind, is computed in order to determine which is the better attribute for classifying the training examples.

**Which attribute is the best classifier?**



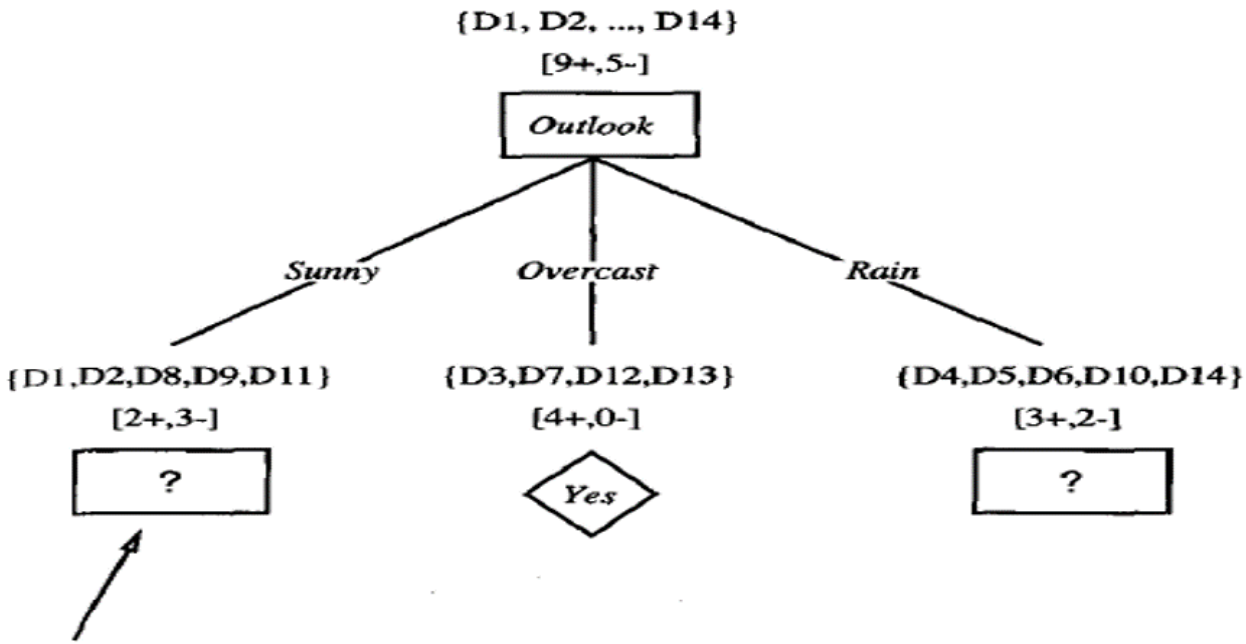
- ID3 determines the information gain for each candidate attribute (i.e., Outlook, Temperature, Humidity, and Wind), then selects the one with highest information gain as root node for the tree.
- The information gain values for all four attributes are:

$$Gain(S, Outlook) = 0.246$$
$$Gain(S, Humidity) = 0.151$$
$$Gain(S, Wind) = 0.048$$
$$Gain(S, Temperature) = 0.029$$

Outlook attribute provides the best prediction of the target attribute, PlayTennis, over the training examples. Therefore, Outlook is selected as the decision attribute for the root node.

- The process of selecting a new attribute and partitioning the training examples is now repeated for each non-terminal descendant node, this time using only the training examples associated with that node.
- Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.
- This process continues for each new leaf node until either of two conditions is met:
  - (1) every attribute has already been included along this path through the tree, or
  - (2) the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

The resulting partial decision tree is shown in Figure:




$S_{sunny}$ = {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

# HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- ID3 searches the space of possible decision trees: doing hill-climbing on information gain.
- It searches the complete space of all finite discrete-valued functions. All functions have at-least one tree that represents them.
- It maintains only one hypothesis (unlike Candidat-Elimination). It cannot tell us how many other viable ones are there.
- It does not do back tracking. Can get stuck in local optima.
- Uses all training examples at each step. Results are less sensitive to errors.

# HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

Hypothesis space search by ID3.