CS-F415: Data Mining

13 Decision Tree Random Forest

Dr. Kamlesh Tiwari
Associate Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

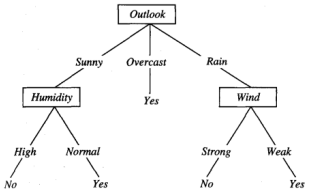
Sept 09, 2024 M/T/W - 9/8/9 am 6108 @ BITS-Pilani [Jan-May 2024]

http://ktiwari.in/dm

Recap: Decision Tree

Decision Tree

A method to approximate discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.



- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value
- Decision trees represent a disjunction of conjunctions

$$(Outlook = Sunny \wedge Humidity = Normal) \vee (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$$

Recap: Iterative-Dichotomiser-3 (ID3) Algorithm

Algorithm 1: ID3(Examples,Target_attribute,Attributes)

1 *Examples* are the training data, *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. Algorithm returns a decision tree that correctly classify the given example.
2 Create a single-node tree *Root*
3 **IF** all *Examples* are +ve **THEN return** *Root* with label +ve
4 **IF** all *Examples* are -ve **THEN return** *Root* with label -ve
5 **IF** *Attributes* $= \phi$ **THEN return** *Root* with most common *Target_attribute*
6 A $\leftarrow$ attribute from *Attributes* that best classifies *Examples*
7 Decision attribute for *Root* $\leftarrow$ A
8 **foreach** *value* v_i *of A* **do**
9 Add a new tree branch below Root, to test A=v_i
10 Examples$_{v_i}$ $\leftarrow$ subset of *Examples* having value v_i for A
11 **IF** Examples$_{v_i} = \phi$ **THEN** below this branch add a leaf with label = most common value of *Target_attribute* in *Examples*
12 **ELSE** below this branch add subtree ID3(Examples$_{v_i}$,*Target_attribute*,*Attributes*-{*A*})
13 **return** *Root*

Recap: Information Gain

Information Gain of an attribute, A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

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

here S_V is a subset of S where value of the attribute A is v

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

For example

S_{Sunny} = {D1, D2, D8, D9, D11}
$S_{Overcast}$ = {D3, D7, D12, D13}
S_{Rainy} = {D4, D5, D6, D10, D14}
S_{Cool} = {D5, D6, D7, D9}
S_{Hot} = {D1, D2, D3, D13}
S_{Normal} = {D5, D6, D7, D9, D10, D11, D13}
S_{High} = {D1, D2, D3, D4, D8, D12, D14}

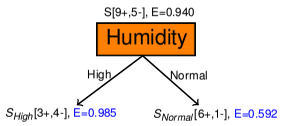
And so on....

[1] Outlook, Temperature, Humidity, Wind

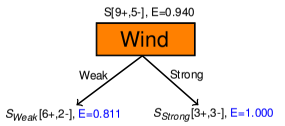
Recap: Information Gain

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

S_{High} = {D1, D2, D3, D4, D8, D12, D14}
S_{Normal} = {D5, D6, D7, D9, D10, D11, D13}



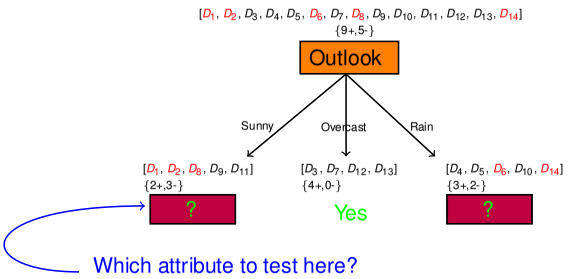
S_{Weak} = {D1, D3, D4, D5, D8, D9, D10, D13}
S_{Strong} = {D2, D6, D7, D11, D12, D14}



$$Gain(S,Humidity) = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$
$$Gain(S,Wind) = 0.940 - (8/14)0.811 - (6/14)1.000 = 0.048$$

$Gain(S, Outlook) = 0.246$, $Gain(S, Temperature) = 0.029$

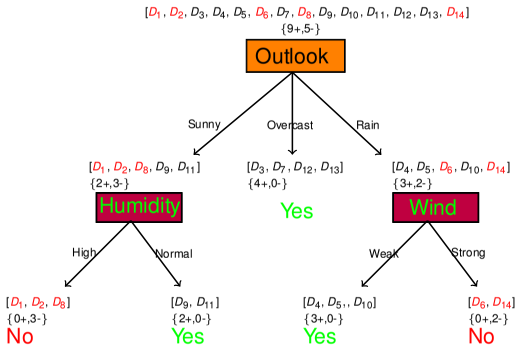
Recap: Recursively apply the same



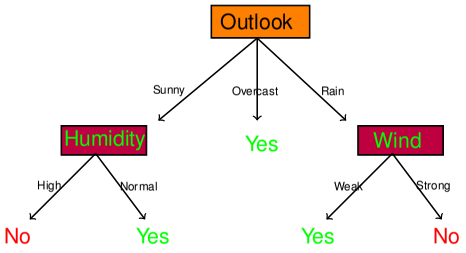
S_{sunny}=[D1, D2, D8, D9, D11]

Gain(S_{sunny},Humidity) = 0.970 − (3/5)0.0 − (2/5)0.0 = 0.970
Gain(S_{sunny},Temperature)= 0.970 − (2/5)0.0 − (2/5)1.0 − (1/5)0.0 = 0.57
Gain(S_{sunny},Wind)= 0.970 − (2/5)1.0 − (3/5)1.0 = 0.019

Recap: Recursively apply the same



Recap: Example



Classification for (*Outlook* = *Rain*, *Humidity* = *High*, *Wind* = *Weak*) is

YES

Issues with Decision Tree

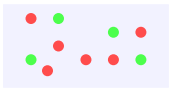
Given a collection of training examples, there could be many decision trees could be consistent with the examples

- ID3 search strategy
  - Selects in favor of shorter trees over longer ones, and
  - Selects trees that place the attributes with highest information gain closest to the root
- Issues in decision trees include
  1. How deeply to grow?
  2. Handling continuous attributes
  3. Choosing an appropriate attribute selection measure
  4. Missing attribute values
  5. Attributes with differing costs, and
  6. Improving computational efficiency

Attribute Selection with **Gini Index**

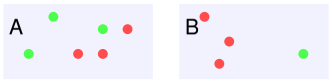
$$Gini = \sum p_c^2 \qquad \text{Gini Impurity} = 1 - \sum p_c^2$$

Gini Impurity is a value between 0 and 0.5

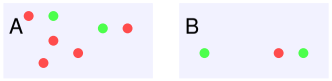


$$\begin{aligned} Gini &= (0.4)^2 + (0.6)^2 \\ &= 0.52 \end{aligned}$$

Consider following two splits of the data



Gini(A)=$(3/6)^2 + (3/6)^2 = 0.5$
Gini(B)=$(3/4)^2 + (1/4)^2 = 0.625$
Gini(split)=$(6/10) * 0.5 + (4/10) * 0.625 = 0.55$



Gini(A)=$(5/7)^2 + (2/7)^2 = 0.591$
Gini(B)=$(2/3)^2 + (1/3)^2 = 0.555$
Gini(split)=$(7/10) * 0.591 + (3/10) * 0.555 = 0.5802$
The better split

Issues with Decision Tree

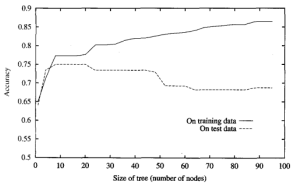
Given a collection of training examples, there could be many decision trees could be consistent with the examples

- ID3 search strategy
  - Selects in favor of shorter trees over longer ones, and
  - Selects trees that place the attributes with highest information gain closest to the root
- Issues in decision trees include
  1. How deeply to grow?
  2. Handling continuous attributes
  3. Choosing an appropriate attribute selection measure
  4. Missing attribute values
  5. Attributes with differing costs, and
  6. Improving computational efficiency

Issues in Decision Tree

Overfitting

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.



	Train	Test
h	70%	55%
h'	65%	59%

- This can occur when training examples contain random errors or noise.

Approaches to avoid overfitting

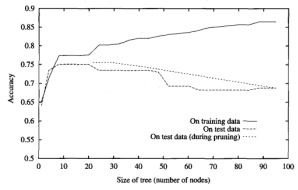
1. Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
2. Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test (such as chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.
- Use an explicit measure of the complexity for encoding the training examples (such as Minimum Description Length) and the decision tree, halting growth of the tree when this encoding size is minimized.

Reduced-error pruning

- Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training affiliated with that node.
- Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set
- Nodes are pruned iteratively, always choosing the node whose removal must increases the decision tree accuracy over the validation set

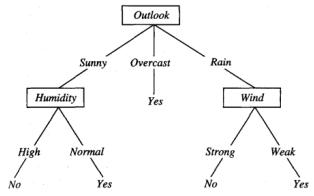


- Drawback: number of examples available for training is reduced.

Rule post-pruning

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node
- Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances

Rule post-pruning



- If $(Outlook = Sunny) \wedge (Humidity = High)$ Then $PlayTennis = No$

Each rule is pruned by removing any antecedent, or precondition, whose removal does not worsen its estimated accuracy

(on validation or test?)

C4.5 Algorithm

It is a statistical classifier that extends ID3 algorithm by dealing with both **continuous** and discrete attributes, **missing values** and **pruning** trees after construction.

- Determines pessimistic estimate by calculating the rule accuracy over the training example, then calculating the standard deviation in this estimated accuracy assuming a binomial distribution.
- For large data sets, the pessimistic estimate is very close to the observed accuracy (the standard deviation is very small), whereas it grows further from the observed accuracy as the size of the data set decreases.
- Although this heuristic method is not statistically valid, it has nevertheless been found useful in practice.

Incorporating Continuous-Valued Attributes

- Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete intervals
- Dynamically create a new boolean attribute A_c that is true if $A < c$ and false otherwise
- The question is, how to select the best value for the threshold c
- Pick a threshold, c, that produces the greatest information gain
- By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A.
- Continuous attribute can also be splits into multiple intervals

Example

$S =$

Temperature	40	48	60	72	80	90
playTennis	N	N	Y	Y	Y	N

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

- Entropy(S)=1
- Partitioned at 44 produces [0+,1-] and [3+,2-] having entropies 0 and $-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.29$
  $Gain(S, A_{44}) = 1 - \frac{1}{6}0 - \frac{5}{6}0.29 = 0.243$

Similarly

- Determine $Gain(S, A_{54})$?
- Determine $Gain(S, A_{66})$?
- Determine $Gain(S, A_{76})$?
- Determine $Gain(S, A_{85})$?

Alternate Measures for Attribute Selection

- There is a natural bias in the information gain measure that favors attributes with many values over those with few values
- Consider attribute *Date*, it would have the highest information gain of any of the attributes. Because *Date* alone perfectly predicts the target attribute over the training data.
- Thus, it would be selected as the decision attribute for the root node of the tree
- **Gain Ratio** is a measure that penalizes attributes such as *Date* by incorporating a term, called split information, that is sensitive to how broadly and uniformly the attribute splits the data:

$$splitInformation(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}$$

- **Gain Ratio:** measure is defined as

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

Alternative Measures for Selecting Attributes

- One practical issue arises using **GainRatio** when denominator is zero or very small ($|S_i| \ll |S|$)
- This makes **GainRatio** undefined or very large
- We can adopt some heuristic such as first calculating the Gain of each attribute, then applying the **GainRatio** test only considering those attributes with above average Gain
- An alternative to the **GainRatio**, is a distance-based measure introduced by Lopez de Mantaras (1991)[2]. Each attribute is evaluated based on the distance between the data partition it creates and the perfect partition (i.e., the partition that perfectly classifies the training data). The attribute whose partition is closest to the perfect partition is chosen.

[2] De Mántaras, R López, "A distance-based attribute selection measure for decision tree induction", Machine learning, 6(1), pp 81–92, Springer-1991

Missing Attribute Values

- In a medical domain, in which we wish to predict patient outcome based on various laboratory tests, it may be that the lab test Blood-Test-Result is available only for a subset of the patients.
- let for $< x, c(x) >$ we do not have value of $A(x)$
- One strategy for dealing with the missing attribute value is to assign it the value that is most common among training examples at node n
- A second, more complex procedure is to assign a probability to each of the possible values of A

Attributes with Differing Costs

Instance attributes may have associated costs

- Consider attributes Temperature, BiopsyResult, Pulse, BloodTestResult
- We would prefer decision trees that use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classifications.
- We might divide the Gain by the cost of the attribute, so that lower-cost attributes would be preferred (without guarantee it puts a bias)
- Tan and Schlimmer for robots perception, demonstrated more efficient recognition strategies using $Gain^2(S, A)/Cost(A)$
- Nunez for medical diagnosis rules proposed (for $w \in [0, 1]$)

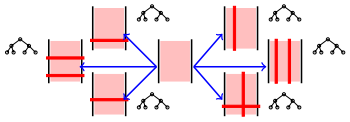
$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[3] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



- For every subset, built a decision tree ($2^{m+n} - 1$ possible)
- To classify new element: **apply voting**

[3] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Thank You!

Thank you very much for your attention!

Queries ?