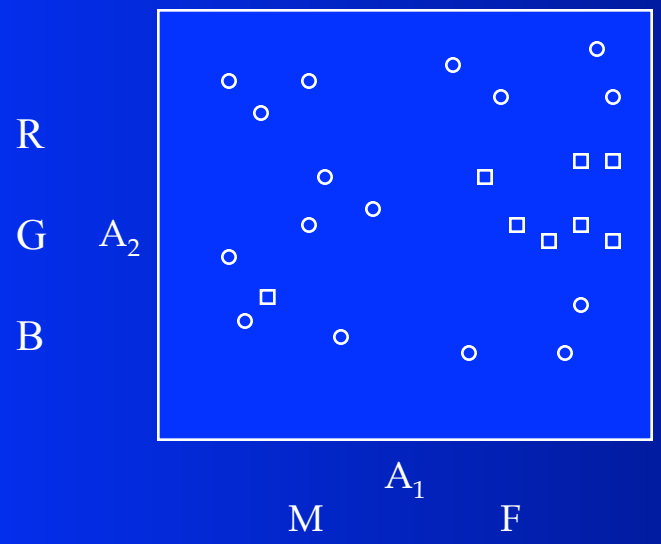
Decision Trees

- Highly used and successful
- Iteratively split the Data Set into subsets one attribute at a time, using most informative attributes first
  - Thus, constructively chooses which attributes to use and ignore
- Continue until you can label each leaf node with a class
- Attribute Features – discrete/nominal (can extend to continuous features)
- Smaller/shallower trees (i.e. using just the most informative attributes) generalizes the best
  - Searching for smallest tree takes exponential time
- Typically use a greedy iterative approach to create the tree by selecting the currently most informative attribute to use

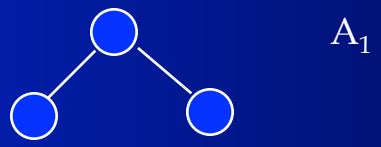
Decision Tree Learning

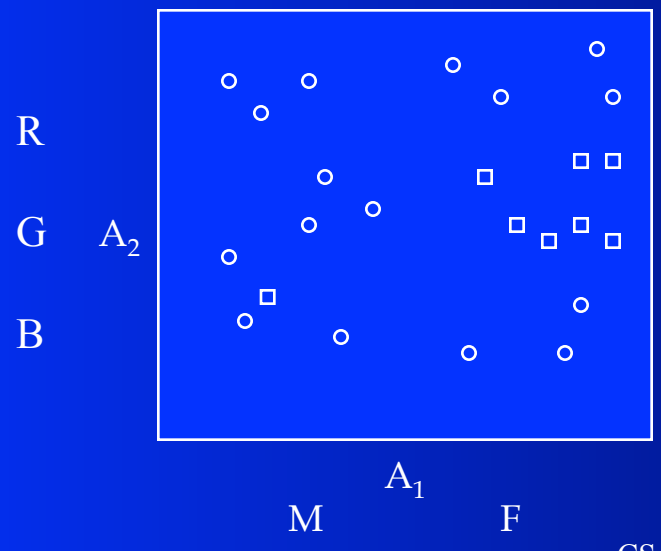
- Assume A_1 is nominal binary feature (Gender: M/F)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- A goal is to get "pure" leaf nodes. What would you do?

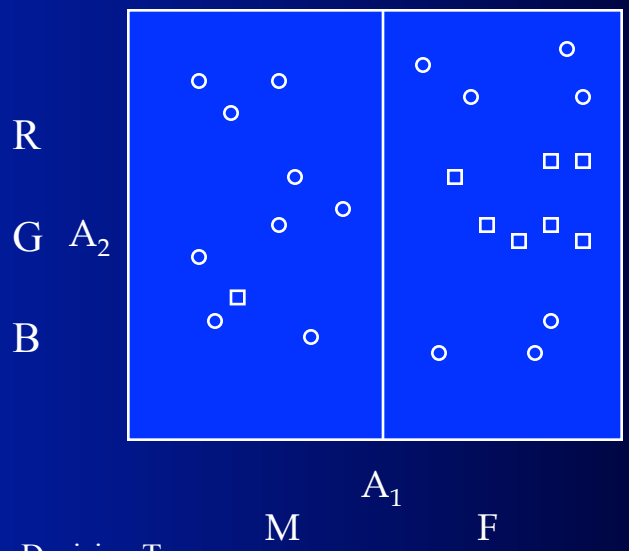


Decision Tree Learning

- Assume A_1 is nominal binary feature (Gender: M/F)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Next step for left and right children?

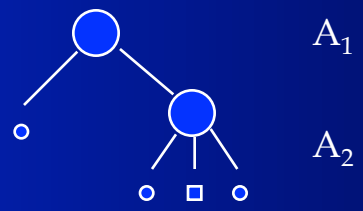


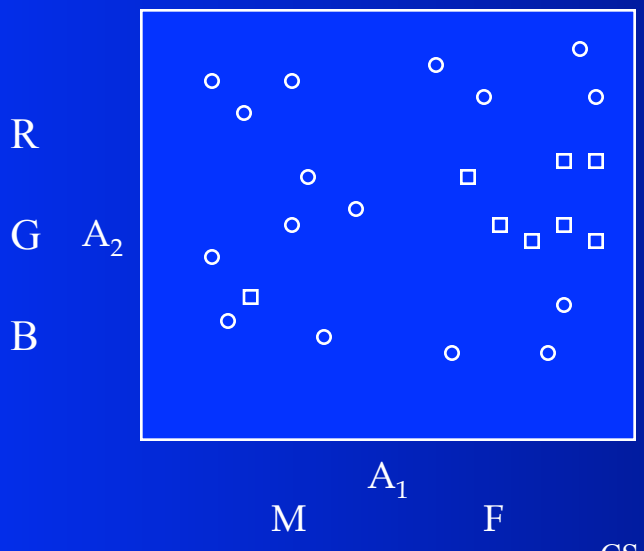


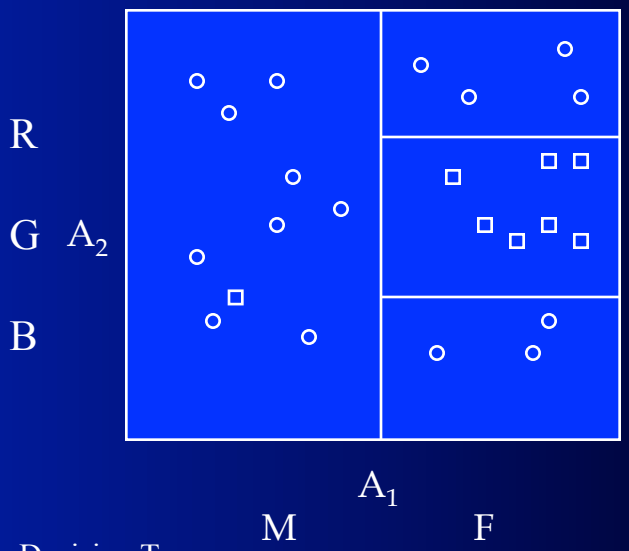


Decision Tree Learning

- Assume A_1 is nominal binary feature (Gender: M/F)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Decision surfaces are axis aligned Hyper-Rectangles

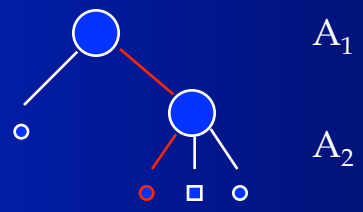


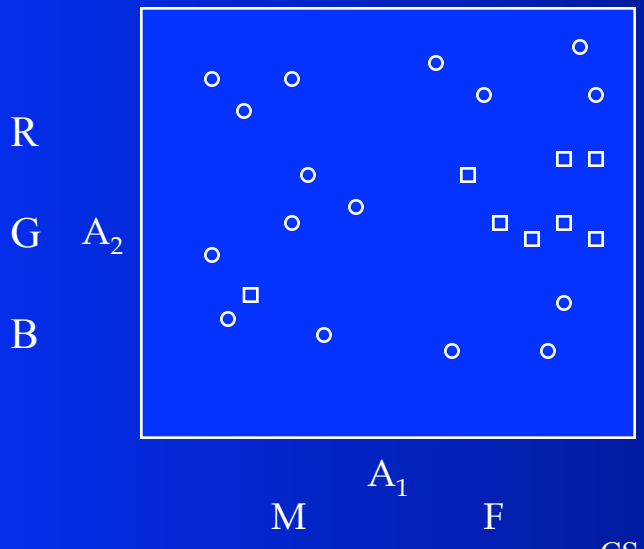


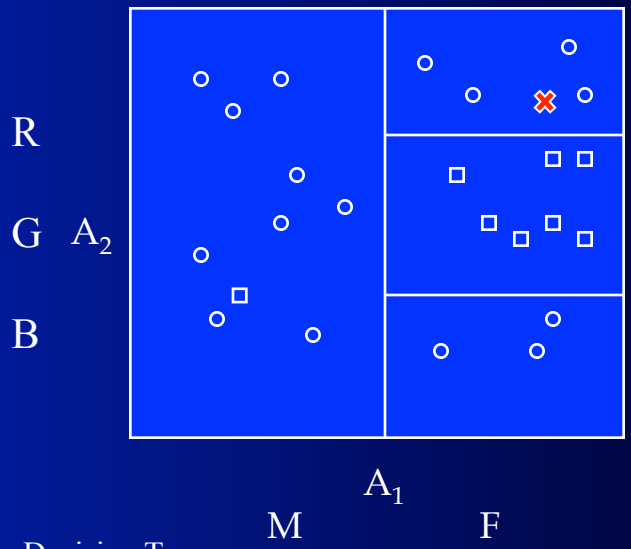


Decision Tree Learning

- Assume A_1 is nominal binary feature (Gender: M/F)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Decision surfaces are axis aligned Hyper-Rectangles







ID3 Learning Approach

- C is a set of examples
- A test on attribute A partitions C into $\{C_i, C_2,\ldots,C_{|A|}\}$ where $|A|$ is the number of values A can take on
- Start with *TS* as C and first find a *good* A for root
- Continue recursively until subsets unambiguously classified, you run out of attributes, or some stopping criteria is reached

Which Attribute/Feature to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

$$\frac{n_{majority}}{n_{total}}$$

Purity

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

$$\frac{n_{majority}}{n_{total}}$$

  - Want both purity and statistical significance (e.g SS#)

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

$$\frac{n_{majority}}{n_{total}} \qquad \frac{n_{maj}+1}{n_{total}+|C|}$$

  - Want both purity and statistical significance
  - Laplacian

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

$$\frac{n_{majority}}{n_{total}} \qquad \frac{n_{maj}+1}{n_{total}+|C|}$$

  - This is just for one node
  - Best attribute will be good across many/most of its partitioned nodes

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of a node after attribute selection

$$\frac{n_{majority}}{n_{total}} \qquad \frac{n_{maj}+1}{n_{total}+|C|} \qquad \sum_{i=1}^{|A|} \frac{n_{total,i}}{n_{total}} \cdot \frac{n_{maj,i}+1}{n_{total,i}+|C|}$$

  - Now we just try each attribute to see which gives the highest score, and we split on that attribute and repeat at the next level

Which Attribute to split on

- Twenty Questions - what are good questions, ones which when asked decrease the information remaining
- Regularity required
- What would be good attribute tests for a DT
- Let's come up with our own approach for scoring the quality of each possible attribute – then pick highest

$$\frac{n_{majority}}{n_{total}} \qquad \frac{n_{maj}+1}{n_{total}+|C|} \qquad \sum_{i=1}^{|A|} \frac{n_{total,i}}{n_{total}} \cdot \frac{n_{maj,i}+1}{n_{total,i}+|C|}$$

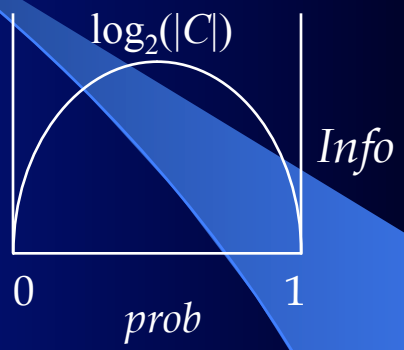
  - Sum of Laplacians – a reasonable and common approach
  - Another approach (used by ID3): Entropy
    - Just replace Laplacian part with information(node)

Information

- Information of a message in bits: $I(m) = -\log_2(p_m)$
- If there are 16 equiprobable messages, I for each message is $-\log_2(1/16) = 4$ bits
- If there is a set S of messages of only c types (i.e. there can be many of the same type [class] in the set), then information for one message is still: $I = -\log_2(p_m)$
- If the messages are not equiprobable then could we represent them with less bits?
  - Highest disorder (randomness) is worst case

Information Gain Metric

- Info(S) is the average amount of information needed to identify the class of an example in S



- Info(S) = Entropy(S) = $-\sum_{i=1}^{|C|} p_i \log_2(p_i)$
- $0 \le$ Info(S) $\le \log_2(|C|)$, $|C|$ is # of output classes
- Expected Information after partitioning using A:
- Info$_A(S)$ = $\sum_{i=1}^{|A|} \frac{|S_i|}{|S|} Info(S_i)$ where $|A|$ is # of values for attribute A
- Gain(A) = Info(S) - Info$_A(S)$ (i.e. minimize Info$_A(S)$)
- Gain does not deal directly with the statistical significance issue
  - more on that later

ID3 Learning Algorithm

1. S = Training Set
2. Calculate gain for each remaining attribute: Gain(A) = Info(S) - Info$_A$(S)
3. Select highest and create a new node for each partition
4. For each partition
   - if one class (or if stopping criteria met) then end
   - else if > 1 class then go to 2 with remaining attributes, or end if no remaining attributes and label with most common class of parent
   - else if empty, label with most common class of parent (or set as null)

Do Example

$$Info(S) = -\sum_{i=1}^{|C|} p_i \log_2(p_i)$$

$$Info_A(S) = \sum_{i=j}^{|A|} \frac{|S_j|}{|S|} Info(S_j) = -\sum_{j=1}^{|A|} \frac{|S_j|}{|S|} \cdot \sum_{i=1}^{|C|} p_i \log_2(p_i)$$

Meat N,Y	Crust D,S,T	Veg N,Y	Quality B,G,Gr
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

Example and Homework

$$Info(S) = -\sum_{i=1}^{|C|} p_i \log_2(p_i)$$

$$Info_A(S) = \sum_{i=j}^{|A|} \frac{|S_j|}{|S|} Info(S_j) = -\sum_{j=1}^{|A|} \frac{|S_j|}{|S|} \cdot \sum_{i=1}^{|C|} p_i \log_2(p_i)$$

- Info(S) = - 2/9·$\log_2$2/9 - 4/9·$\log_2$4/9 -3/9·$\log_2$3/9 = 1.53
  - Not necessary unless you want to calculate information gain
- Starting with all instances, calculate gain for each attribute
- Let's do Meat:
- $Info_{Meat}(S)$ = 4/9·(-2/4$\log_2$2/4 - 2/4·$\log_2$2/4 - 0·$\log_2$0/4) +
  5/9·(-0/5·$\log_2$0/5 - 2/5·$\log_2$2/5 - 3/5·$\log_2$3/5) = .98
  - Information Gain is 1.53 - .98 = .55
- Finish this level, find best attribute and split, and then find the best attribute for at least the left most node at the next level
  - Assume sub-nodes are sorted alphabetically left to right by attribute

ID3 Notes

- Attributes which best discriminate between classes are chosen
- If the same ratios are found in a partitioned set, then gain is 0
- Complexity:
  - At each tree node with a set of instances the work is
    - O(|*Instances*| * |*remaining attributes*|), which is Polynomial
  - Total complexity is empirically polynomial
    - O(|*TrainingSet*| * |*attributes*| * |*nodes in the tree*|)
    - where the number of nodes is bound by the number of attributes and can be kept smaller through stopping criteria, etc.

ID3 Overfit Avoidance

- Noise can cause inability to converge 100% or can lead to overly complex decision trees (overfitting). Thus, we usually allow leafs with multiple classes.
  - Can select the majority class as the output, or output a confidence vector
- Also, may not have sufficient attributes to perfectly divide data
- Even if no noise, statistical chance can lead to overfit, especially when the training set is not large. (e.g. some irrelevant attribute may happen to cause a perfect split in terms of info gain on the training set, but will generalize poorly)
- One approach - Use a validation set and only add a new node if improvement (or no decrease) in accuracy on the validation set – checked independently at each branch of the tree using data set from parent
  - shrinking data problem

ID3 Overfit Avoidance (cont.)

- If testing a truly irrelevant attribute, then the class proportion in the partitioned sets should be similar to the initial set, with a small info gain. Could only allow attributes with gains exceeding some threshold in order to sift noise. However, empirically tends to disallow some relevant attribute tests.
- Use Chi-square test to decide confidence in whether attribute is irrelevant. Approach used in original ID3. (Takes amount of data into account)
- If you decide to not test with any more attributes, label node with either most common, or with probability of most common (good for distribution vs function)
- C4.5 allows tree to be changed into a rule set. Rules can then be pruned in other ways.
- C4.5 handles noise by first filling out complete tree and then pruning any nodes where expected error could statistically decrease (# of instances at node becomes critical).

Reduced Error Pruning

- Validation stopping could stop too early (e.g. higher order combinations)
- Pruning a full tree (one where all possible nodes have been added)
  - Prune any nodes which would not hurt accuracy
  - Could allow some higher order combinations that would have been missed with validation set early stopping (though could do a VS window)
  - Can simultaneously consider all nodes for pruning rather than just the current frontier

1. Train tree out fully (empty or consistent partitions or no more attributes)
2. For EACH non-leaf node, test accuracy on a validation set for a modified tree where the sub-tree of the node is removed and the node is assigned the majority class based on the instances it represents from the training set
3. Keep pruned tree which does best on the validation set and does at least as well as the original tree on the validation set
4. Repeat until no pruned tree does as well as the current tree

Missing Values: C4.5 Approach

- Can use any of the methods we discussed previously – new attribute value quite natural with typical nominal data
- Another approach, particular to decision trees (used in C4.5):
  - When arriving at an attribute test for which the attribute is missing do the following:
  - Each branch has a probability of being taken based on what percentage of examples at that parent node have a particular value for the missing attribute
  - Take all branches, but carry a weight representing that probability. Weights could be further modified (multiplied) by other missing attributes in the current example as they continue down the tree.
  - Thus, a single instance gets broken up and appropriately distributed down the tree but its total weight throughout the tree will always sum to 1
- Results in multiple active leaf nodes. For execution, set output as leaf with highest weight, or sum weights for each output class, and output the class with the largest sum, (or output the class confidence).
- During learning, scale information gain by instance weights.
- This approach could also be used for labeled probabilistic inputs with subsequent probabilities tied to outputs

ID3/C4.5 Miscellaneous

- Continuous data handled by testing all *n*-1 possible binary thresholds to see which gives best information gain. The split with highest gain is used as the attribute at that level.
  - More efficient to just test thresholds where there is a change of classification.
  - Is binary split sufficient? Attribute may need to be split again lower in the tree, no longer have a strict depth bound
- Intelligibility of DT – When trees get large, intelligibility drops off
- C4.5 rules - transforms tree into prioritized rule list with default (most common output for examples not covered by rules). It does simplification of superfluous attributes by greedy elimination strategy (based on statistical error confidence as in error pruning). Prunes less productive rules within rule classes
  - How critical is intelligibility?

Information gain favors attributes with many attribute values

- If A has random values (SS#), but ends up with only 1 example in each partition, it would have maximum information gain, though a terrible choice.
- Occam's razor would suggest seeking trees with less overall nodes. Thus, attributes with less possible values might be given some kind of preference.
- Binary attributes (ASSISTANT) are one solution, but lead to deeper trees, and exponential growth in possible ways of splitting attribute sets
- Can use a penalty for attributes with many values such as Laplacian: $(n_c+1)/(n+|C|)$), though real issue is splits with little data
- Gain Ratio is the approach used in original ID3, though you do not have to do that in the project, but realize, you will be susceptible to the SS# variation of overfit, though it doesn't occur in your data sets

ID3 - Gain Ratio Criteria

- The main problem is splits with little data – What might we do?
  - Laplacian or variations common: $(n_c+1)/(n+|C|)$ where n_c is the majority class and $|C|$ is the number of output classes
- Gain Ratio: Split information of an attribute SI(A) = $-\sum_{i=1}^{|A|} \frac{S_i}{|S|} \log_2 \frac{S_i}{|S|}$
- What is the information content of "splitting on attribute A" - does not ask about output class
- SI(A) or "Split information" is larger for a) many valued attributes and b) when A evenly partitions data across values. SI(A) is $\log_2(|A|)$ when partitions are all of equal size.
- Want to minimize "waste" of this information. When SI(A) is high then Gain(A) should be high to take advantage. Maximize Gain Ratio: Gain(A)/SI(A)
- However, somewhat unintuitive since it also maximizes ratio for trivial partitions (e.g. $|S| \approx |S_i|$ for one of the partitions), so.... Gain must be at least average of different A before considering gain ratio, so that very small SI(A) does not inappropriately skew Gain ratio.

Decision Trees - Conclusions

- Good Empirical Results
- C4.5 uses the Laplacian and Pruning
- Comparable application robustness and accuracy with neural networks - faster learning (though NN are simpler with continuous - both input and output), while DT natural with nominal data
- One of the most used and well known of current symbolic systems - used widely to aid in creating rules for expert systems
- Higher order attribute tests - C4.5 can do greedy merging into *value sets*, based on whether that improves gain ratio. Executes the tests at each node expansion allowing different value sets at different parts of the tree. Exponential time based on order.

Decision Tree Assignment

- See http://axon.cs.byu.edu/~martinez/classes/478/Assignments.html
- Start Early!!