Decision Trees

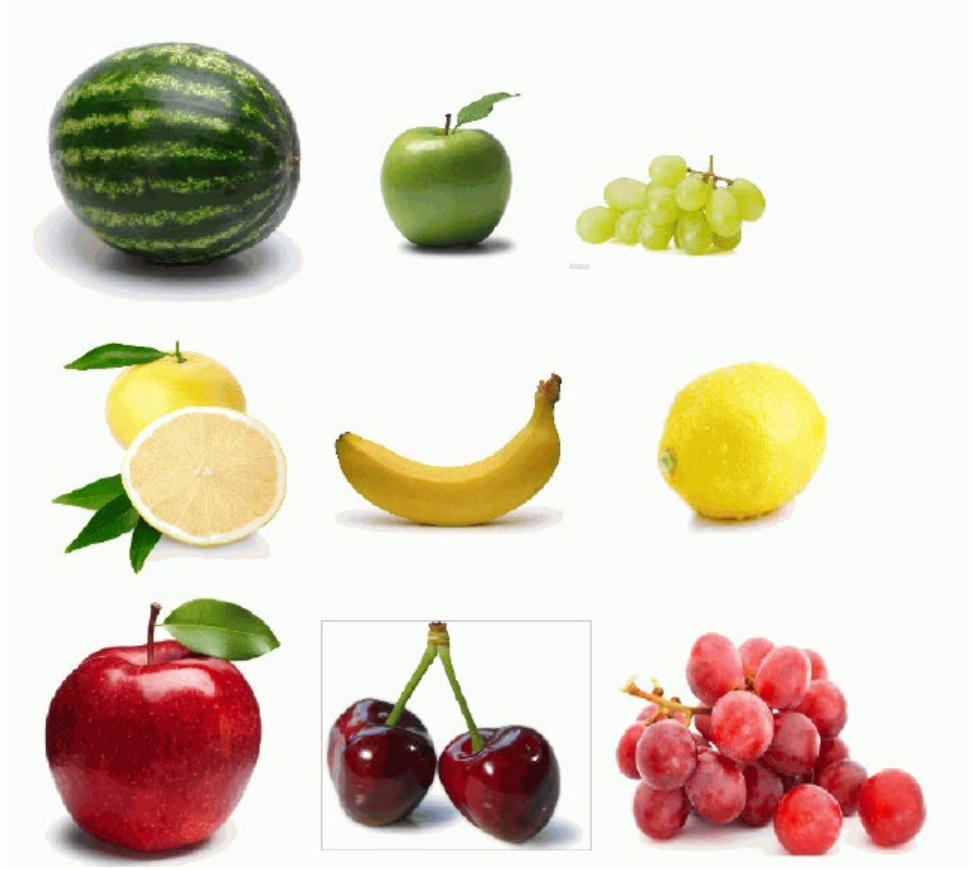
Reading: Chapter 8.7, 9.2, 10.9, and 15.

Decision Tree

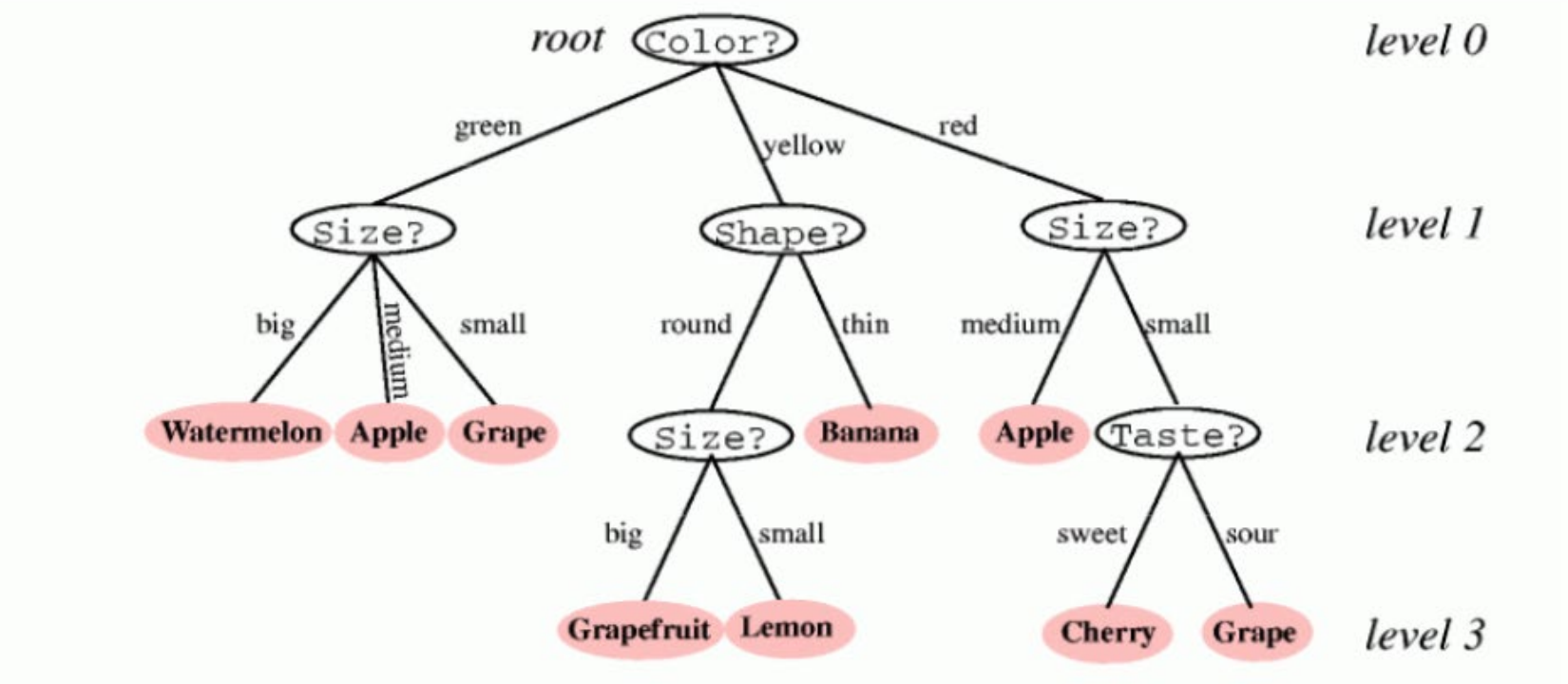
Task: Classifying some fruits

Attributes/features to extract

- Colour: green, yellow, red
- Size: big, medium, small
- Shape: round, thin
- Taste: sweet, sour



Decision Tree



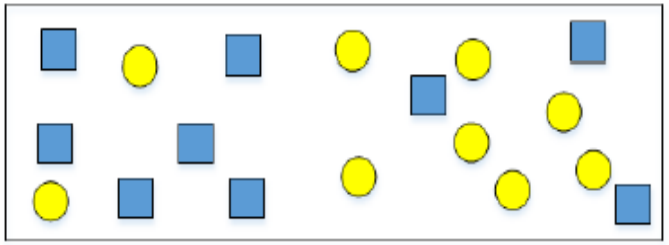
Decision Tree

- Description: flow chart-like tree structure
  - Each node: a test on an attribute
  - Each branch: an outcome of the test
  - Leaf nodes: classes or class distributions
- Which feature to split next?
  - General principle: Feature that maximizes the separation of classes
  - Common choices: Error, Gini index, entropy
- Where to split?
  - Depending on type of attributes: categorical vs numerical
  - Multi-way split or binary split
- When to stop splitting or how to control the tree size?
  - Size and purity thresholds
  - Pruning

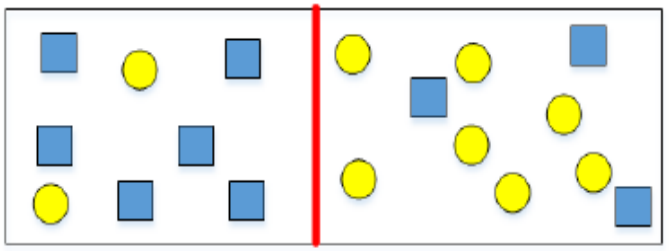
Split Criteria

- Goal: maximize the separation of the different classes among the children nodes
- Dependent on the type of attribute
  - Binary: only one choice
  - Categorical with r different values: r-way split, converting to binary
  - Numeric: several options
    - r-way split if containing a small number of r ordered values
    - Common: split using binary condition, e.g $x < a$
- How to precisely quantify the **best** split?
  - Error rate/purity
  - Gini index
  - Entropy

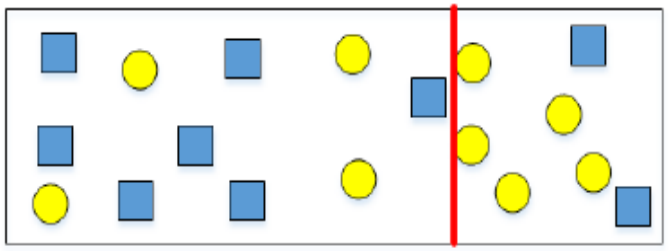
Which is the better split?



ORIGINAL DATA



SPLIT 1



SPLIT 2

Split Criteria

- Error rate/purity
  - Purity = fraction of samples having dominant class label
  - Error rate = 1 - Purity
  - Based on smallest weighted average of error rates
    - Given a set S
    - Given a r-way split of set S into $S_1, S_2, \ldots, S_r$
    - N: number of samples in S
    - N_r: number of samples in S_r
    - For each set compute the error rate $e_r = 1 - p_r$
    - Compute weighted average of error rates

$$\bar{e}_S = \frac{e_1 N_1 + e_2 N_2 + \ldots + e_r N_r}{N}$$

    - Repeat this for all possible r-way splits
    - Select the one with the lowest weighted average error rate

Split Criteria – Gini Index

- Gini index (used by CART algorithm)
  - Given a set S
  - Given a r-way split of set S into $S_1, S_2, \ldots, S_r$
  - N: number of samples in S
  - N_r: number of samples in S_r
  - For each subset S_r
    - $p_1, p_2, \ldots, p_k$ fraction of samples from k classes

$$G(S_r) = 1 - (p_1^2 + \ldots + p_k^2)$$

  - Compute weighted average of Gini indices

$$\bar{G}(S) = \frac{G(S_1)N_1 + G(S_2)N_2 + \ldots + G(S_r)N_r}{N}$$

  - Select the split with the lowest weighted average Gini index

Split Criteria: Entropy

- Entropy (used by ID3 and C4.5 algorithms (variation))
  - Entropy: measure of disorder or uncertainty
  - Given a set S
  - Given a r-way split of set S into $S_1, S_2, \ldots, S_r$
  - N: number of samples in S
  - N_r: number of samples in S_r
  - For each subset S_r
    - $p_1, p_2, \ldots, p_k$ fraction of samples from k classes

$$E(S_r) = -(p_1 \log_2(p_1) + \ldots + p_k \log_2(p_k))$$

    - Note: if $p_i = 0$: $p_i \log_2(p_i) = 0$
  - Compute weighted average of Gini indices

$$\bar{E}(S) = \frac{E(S_1)N_1 + E(S_2)N_2 + \ldots + E(S_r)N_r}{N}$$

  - Select the split with the lowest weighted entropy

Play Golf

ID	Outlook	Temperature	Humidity	Windy	Play Golf
1	Rainy	Hot	High	FALSE	No
2	Rainy	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Sunny	Mild	High	FALSE	Yes
5	Sunny	Cool	Normal	FALSE	Yes
6	Sunny	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Rainy	Mild	High	FALSE	No
9	Rainy	Cool	Normal	FALSE	Yes
10	Sunny	Mild	Normal	FALSE	Yes
11	Rainy	Mild	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
14	Sunny	Mild	High	TRUE	No
15	Rainy	Mild	Normal	FALSE	?
16	Rainy	Mild	High	TRUE	?
17	Overcast	Hot	High	TRUE	?
18	Sunny	Hot	High	TRUE	?

Play Golf

Sort and count

ID	Outlook	Temperature	Humidity	Windy	Play Golf
3	Overcast	Hot	High	FALSE	Yes
7	Overcast	Cool	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
9	Rainy	Cool	Normal	FALSE	Yes
11	Rainy	Mild	Normal	TRUE	Yes
1	Rainy	Hot	High	FALSE	No
2	Rainy	Hot	High	TRUE	No
8	Rainy	Mild	High	FALSE	No
4	Sunny	Mild	High	FALSE	Yes
5	Sunny	Cool	Normal	FALSE	Yes
10	Sunny	Mild	Normal	FALSE	Yes
6	Sunny	Cool	Normal	TRUE	No
14	Sunny	Mild	High	TRUE	No

Play Golf

Compute weighted errors

		PlayGolf				
		Yes	No	Total	Purity	Error
	Sunny	3	2	5	0.60	**0.40**
Outlook	Overcast	4	0	4	1.00	**0.00**
	Rainy	2	3	5	0.60	**0.40**
				14		
				Weighted Average		**0.29**

		PlayGolf				
		Yes	No	Total	Purity	Error
	High	3	4	7	0.57	**0.43**
Humidity	Low	6	1	7	0.86	**0.14**
				14		
				Weighted Average		**0.29**

		PlayGolf				
		Yes	No	Total	Purity	Error
	Hot	2	2	4	0.50	**0.50**
Temp	Mild	4	2	6	0.67	**0.33**
	Cool	3	1	4	0.75	**0.25**
				14		
				Weighted Average		**0.36**

		PlayGolf				
		Yes	No	Total	Purity	Error
	TRUE	6	2	8	0.75	**0.25**
Windy	FALSE	3	3	6	0.50	**0.50**
				14		
				Weighted Average		**0.36**

Decision Trees

Applications

Decision trees work very well when applied to problems that have the following characteristics:

1) instances are presented by attribute-value pairs

2) the target function has discrete output values

3) training data may contain errors

4) training data may contain missing attribute values

Learning Decision Trees

Basic Algorithm (Hunt's Divide and Conquer)

- *start with a node that represents the training data*

- *if all the training data belongs to the same class then the node becomes a leaf and is labeled with that classification*

- *if the training data does not belong to the same class, then the algorithm uses an entropy measure to select an attribute which will best separate the training data into individual classes. The attribute represents the test or the decision attribute at the node. This repeated until the data is classified correctly or until all attributes have been used*

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

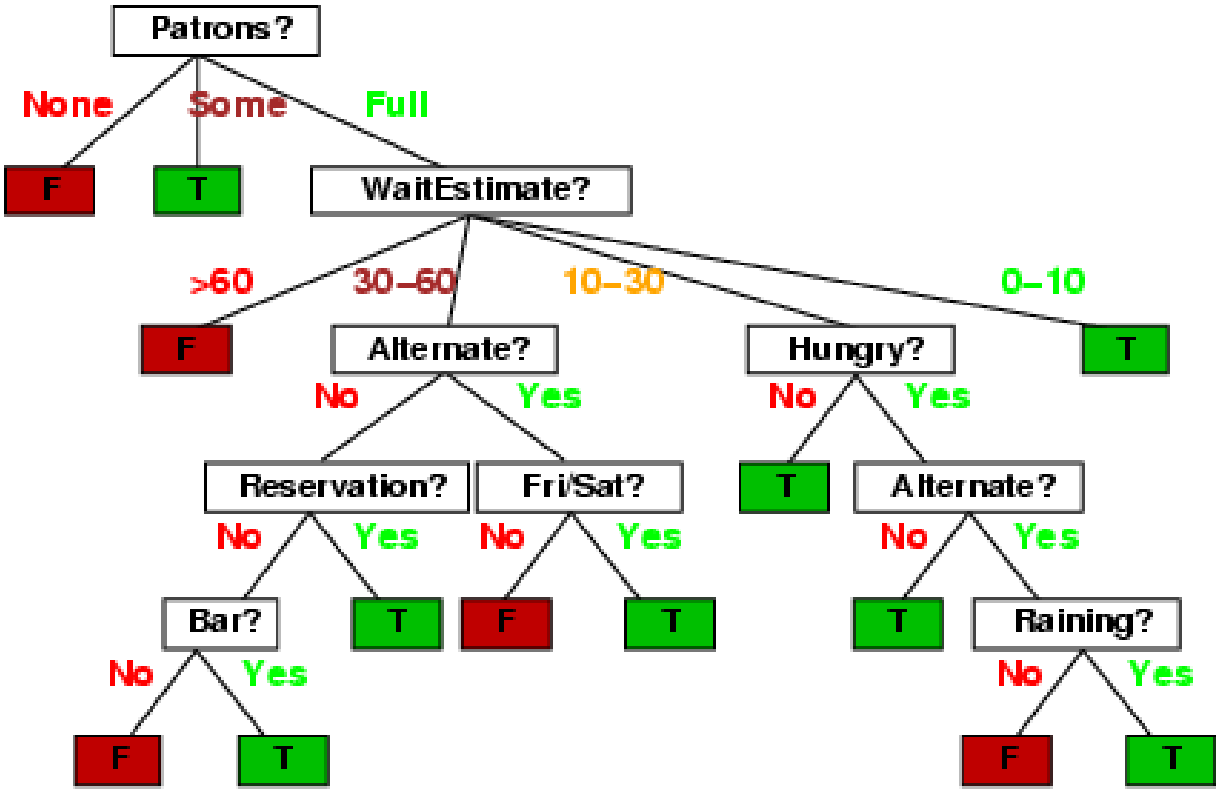
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g.

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)
-

Decision trees

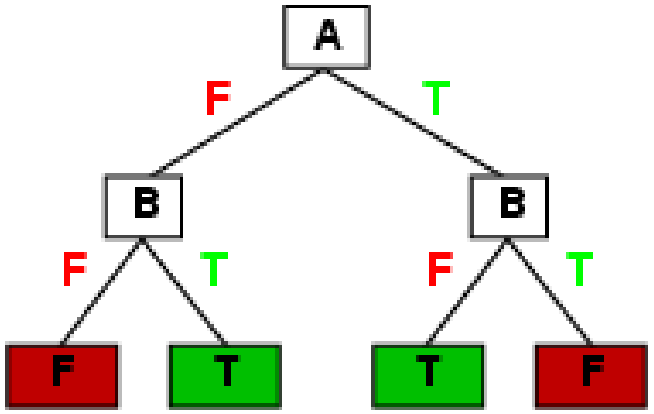
- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

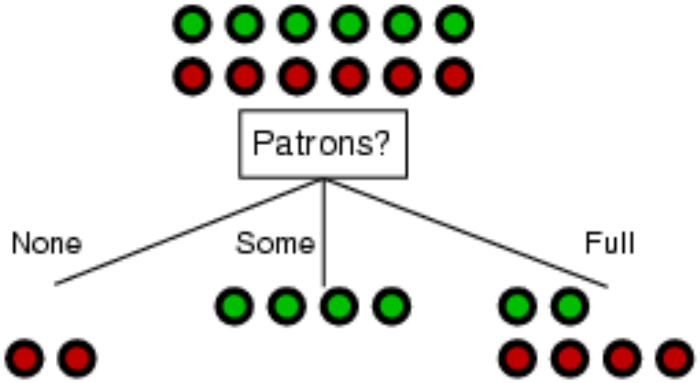
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

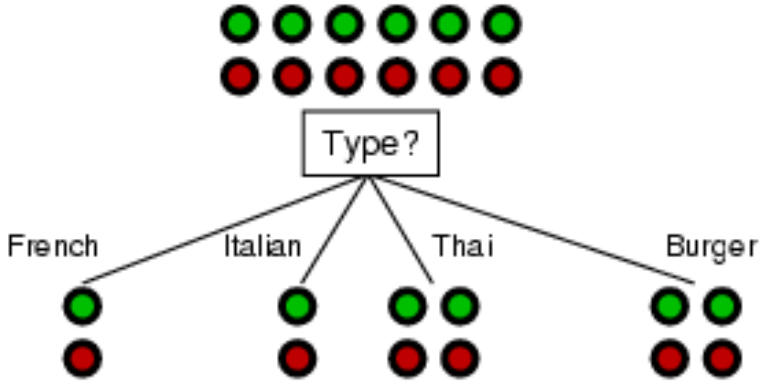


- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all

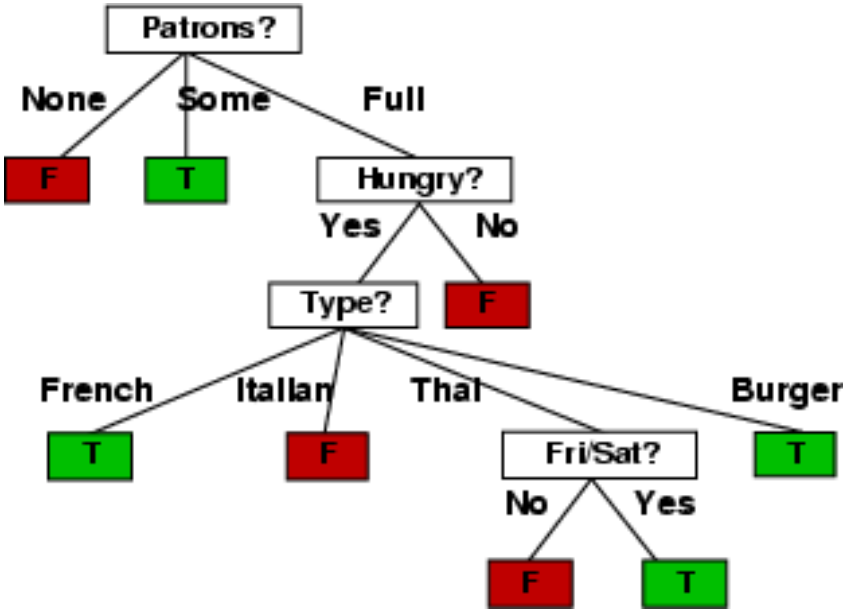




- *Patrons?* is a better choice

Example contd.

- Decision tree learned from the 12 examples:



- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Decision Trees

Bias and Decision Trees

- General approach - shorter trees preferred over larger trees.
- based on Occam's Razor - prefer the simplest hypothesis that fits the data
  - ***pro***
    - fewer short hypotheses than long ones
    - short hypothesis less likely to fit data due to coincidence
    - longer hypothesis may fit the data but often may fail to generalize correctly
  - ***against***
    - the size of the hypothesis generated depends on the internal representation of the learner
    - different learners generate different hypotheses for the same dataset (multiple different trees can be generated for the same dataset - which is best?)
    - some applications require complete explanations: medical applications

Decision Trees: issues

Problems with Decision Trees

1) Overfitting the data

- when noise or coincidental regularities are found in the data.

-major problem for decision trees - a hypothesis is said to overfit the training examples if there exists another hypothesis that classifies the training examples with less accuracy but actually produces better results over the entire set of examples.

- several methods are available to handle the problem of overfitting the data:

a) Using a test data set - use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree (training + validation approach).

b) Cross-validation method - split training data into k-subsets; train on k-1 subsets and test on remaining subset (do this k times so that all subsets are used once as test sets); average results to determine the effectiveness of the classification structure.

Decision Trees: issues

Problems with Decision Trees

1) Overfitting the data

c)Statistical test - use all the available data for training, but apply a statistical test to determine whether expanding or removing a given node is likely to produce an improvement beyond the training set.

d)Complexity analysis - one can also use an explicit measure of the complexity for encoding the training examples and the decision trees, halting the growth of the tree when this encoding size is minimized

Cross-validation

Cross-validation Example

Training Set – 90 examples
Number of folds (subsets) – 3

Split training set into 3 subsets (equal size if possible)
Subset 1 – Examples 1-30
Subset 2 – Examples 31-60
Subset 3 – Examples 61-90

Train Run 1 – Use Subset 1 & 2 – Test on Subset 3 -> Decision Tree 1
Train Run 2 – Use Subset 1 & 3 – Test on Subset 2 -> Decision Tree 2
Train Run 3 – Use Subset 2 & 3 – Test on Subset 1 -> Decision Tree 3

Final Step – Derive average of DT1, DT2 and DT3

Decision Trees: issues

2) Continuous Values

- need to find a threshold/s to produce the greatest information gain

- simple two step approach:

a) sort the examples according to the continuous attribute and identify the adjacent examples that differ in their target classification

b) build a set of candidate thresholds midway between the values of the corresponding values of the continuous attribute and then compute the information gain for each threshold - the best threshold is the one with the greatest information gain

Decision Trees: issues

3) Missing Attribute Values

- training data may be incomplete

- two approaches to deal with missing attribute values

a) assign the missing attribute the value that is most common among training examples at a given node D or the most common value among training instances at a node D that is labelled with class A.

b) compute the probability of each of the values of attribute and then use the probability when building the decision tree.

Pros and Cons of Trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than other classification approaches.
- Trees can be displayed graphically and are easily interpreted even by a non-expert.
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- However, by aggregating many decision trees, the predictive performance of trees can be substantially improved.

Bagging

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method
- It is particularly useful and frequently used in the context of decision trees.
- Procedure
  - Bootstrap: taking repeated samples from the (single) training data set.
  - Generate multiple different bootstrapped training data sets.
  - Train the tree on every bootstrapped training set
  - Majorite Voting for final decision.

Random Forests

- An extension over bagging.
- One extra step: in addition to taking the random subset of data, it also takes the random selection of features rather than using all features to grow trees.
- When you have many random trees. It's called Random Forest.

Pros and Cons of Random Forest

Advantages:

- Handles higher dimensionality data very well.
- Handles missing values and maintains accuracy for missing data.

Disadvantages:

- Since final prediction is based on the mean predictions or majority vote from subset trees, it is more difficult to interprete.

Boosting

- A general approach for many statistical learning methods.
- Boosing *is another ensemble technique to create a collection of predictors (e.g. decision trees).*
  - *In this technique, learners are learned sequentially with early learners fitting simple models to the data*
  - *then analyses data for errors and fits consecutive trees aiming to solve for net error from the prior tree.*

Summary

- Decision trees are simple and interpretable models
- However, they are often not competitive with other methods in terms of prediction accuracy
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods, random forests and boosting, are among the state-of-the-art methods for some supervised learning tasks. However, their results can be difficult to interpret.