Decision Trees

Jay Urbain, PhD

Credits:

Gareth James, Daniela Witten, Trevor Hastie , Robert Tibshirani. An Introduction to Statistical Learning: with Applications in R.

Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

Tom Mitchell, Machine Learning, CMU

Nazli Goharian, IIT/Georgetown

Jiawei Han, Micheline Kamber, and Jian Pei, Data Mining


CONSULTANTS SAY THREE QUINTILLION BYTES OF DATA ARE CREATED EVERY DAY.
DilbertCartoonist@gmail.com
IT COMES FROM EVERYWHERE. IT KNOWS ALL.
ACCORDING TO THE BOOK OF WIKIPEDIA, ITS NAME IS "BIG DATA."
BIG DATA
BIG DATA LIVES IN THE CLOUD. IT KNOWS WHAT WE DO.
IN THE PAST, OUR COMPANY DID MANY EVIL THINGS.
BUT IF WE ACCEPT BIG DATA IN OUR SERVERS, WE WILL BE SAVED FROM BANKRUPTCY.
LET US PAY.
www.dilbert.com
IS IT TOO LATE TO SIDE WITH EVIL?
SHHHH! IT HEARS YOU.

Top Machine Learning Methods on Kaggle

- Gradient Boosting
  - XGBoost using Decision Trees
  - Tabular data, feature engineering

- Neural Networks
  - Deep Learning for NLP and Computer Vision
  - Perceptual tasks, automated learning of features

https://www.kaggle.com/

Tree-based Methods

- Tree-based methods for regression and classification.
- Involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision-tree* methods.

Pros and Cons

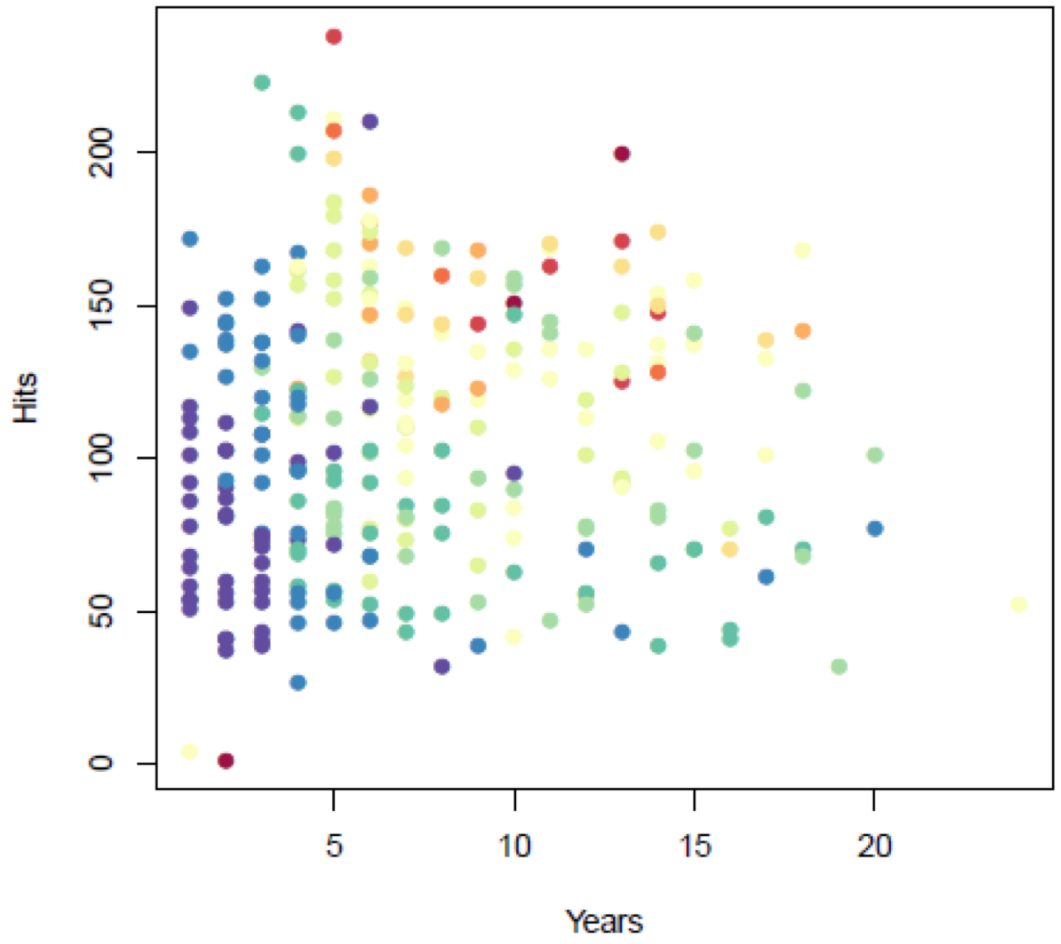
- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Need to augment trees with bagging, random forests (ensembles), and boosting methods.
  - These methods grow multiple trees which are then combined to yield a single consensus prediction.
  - Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss of interpretation.

Basics of Decision Trees

- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

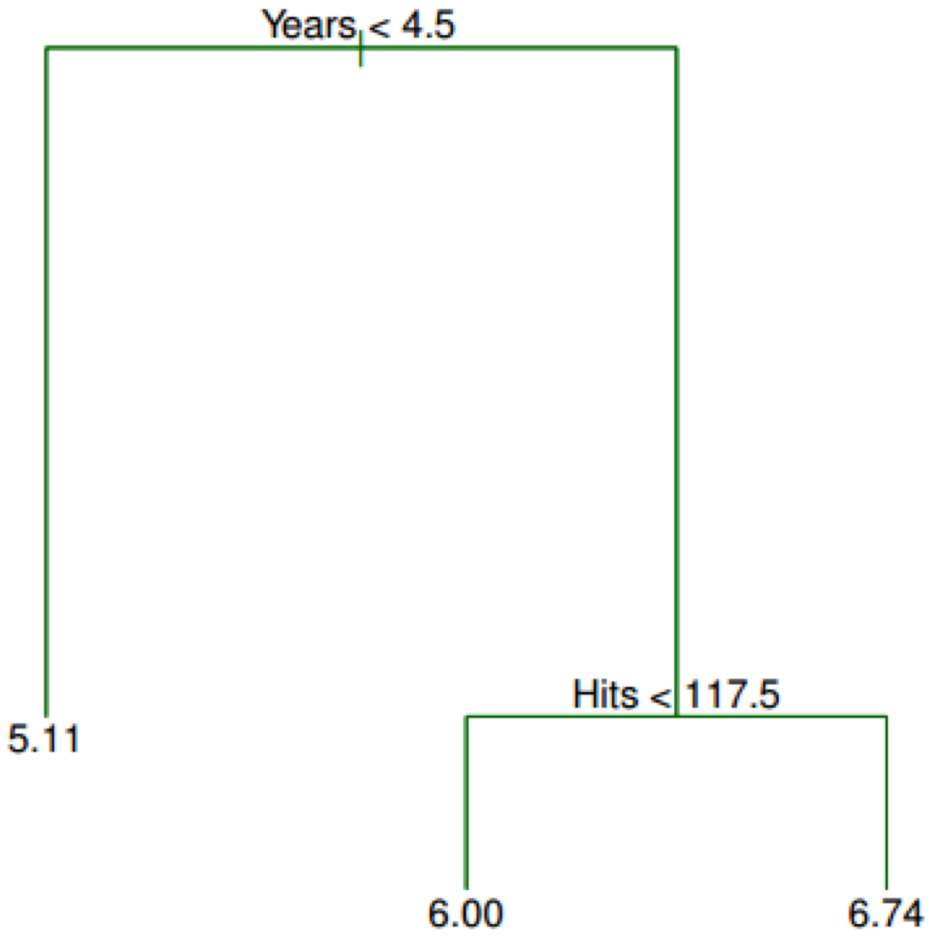
Baseball salary data

- How to stratify (segment) the data?
- Salary is color-coded from *low* (blue, green) to *high* (yellow, red)



DT for Baseball salary data

- Regression tree for predicting the *log salary* of a baseball player.
- Based on the number of years played and the number of hits made in the previous year.
- The label at each node ($X_j < t^k$) indicates the left-hand branch, $X_j \geq t_k$ is the right-hand branch.
- E.g., The left-hand branch corresponds to *Years<4.5*,right-hand branch *Years>=4.5*.
- The number in each leaf is the *mean of the response* for the observations that fall there.



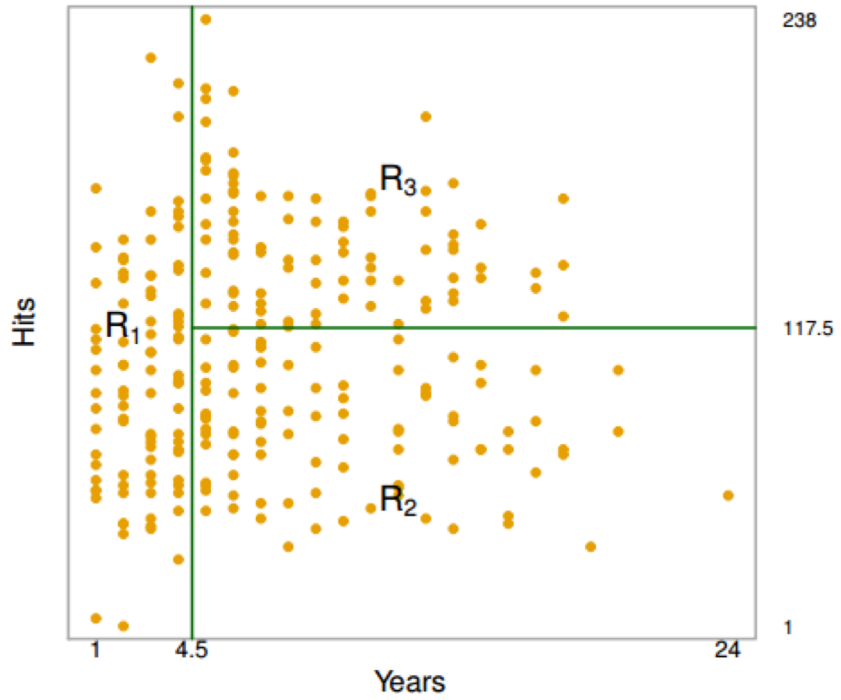
DT Results

- The tree below stratifies (or segments) the players into three regions of predictor space:

  R1 ={X | Years< 4.5},

  R2 ={X | Years>=4.5, Hits<117.5}, and

  R3 ={X | Years>=4.5, Hits>=117.5}.



DT Results

- The tree stratifies (or segments) the players into three regions of predictor space:

  R1 ={X | Years< 4.5},

  R2 ={X | Years>=4.5, Hits<117.5}, and

  R3 ={X | Years>=4.5

Examples predictions:

- Years=3, then predict 5.11 ($\$1000 \times e^{5.11} \approx \$166000$)
- Years=5 and Hits=100, then predict 6.00 ($\$1000 \times e^{6.00} \approx \$403000$)
- Years=8 and Hits=120, then predict 6.74 ($\$1000 \times e^{6.74} \approx \$846000$)

Note: salary scaled by *ln()*

DT Terminology

- The regions R_1, R_2, and R_3 are known as terminal nodes (leaves).
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as internal nodes.
- In the hitter's tree, the two internal nodes are indicated by the text Years<4.5 and Hits<117.5.

Interpretation of Results

- *Years* is the most important factor in determining *Salary*, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of *Hits* that he made in the previous year seems to play little role in his *Salary*.
- **But** among players who have been in the major leagues for five or more years, the number of *Hits* made in the previous year does affect *Salary*, and players who made more Hits last year tend to have higher salaries.
- An over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

Tree-building process

- Divide the predictor space, i.e., the set of possible values for attributes $X_1, X_2, \ldots, X_p$ into j distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- For every observation that falls into the region R_j, (leaf) we make the same prediction, which is simply the *mean of the response* values for the training observations in R_j.

Details of the tree-building process

- Theoretically, the regions could have any shape.
- However, with DTs, the predictor space is partitioned into high-dimensional rectangles, or boxes, for ease of interpretation.
- The goal is to find boxes $R_1,\ldots,R_J$ that minimize the RSS, given by:

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2,$$

- Where $\hat{y}_{Rj}$ is the mean response for the training observations (targets) within the j^{th} box.

More details of tree-building process

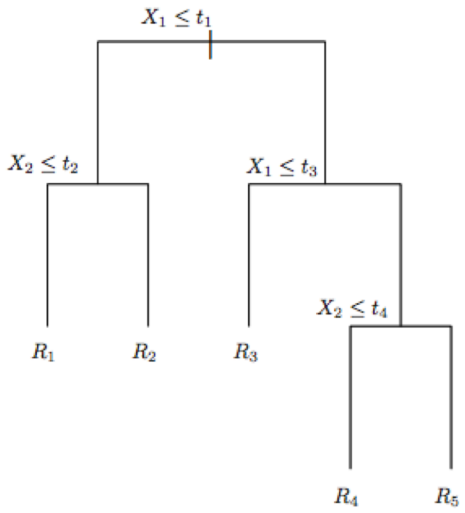
- Computationally infeasible to consider every possible partition of the feature space into *j* boxes.
- Take a top-down, ***greedy*** approach that is known as ***recursive binary splitting.***
- The approach is ***top-down*** because it begins at the top of the tree and then recursively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is ***greedy*** because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

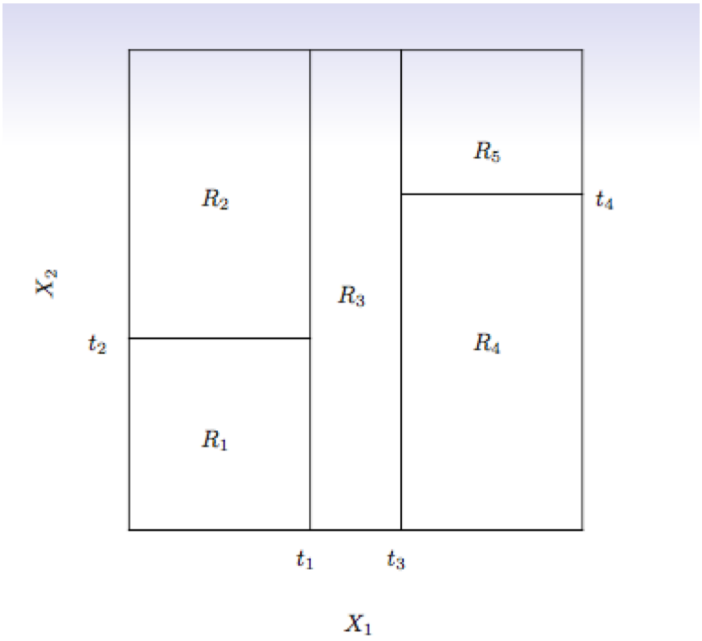
Details— Continued

- First select the predictor X_j and the cutpoint ***s*** such that splitting the predictor space into the regions $\{X|X_j < s\}$ *and* $\{X|X_j \geq s\}$ leads to the greatest possible reduction in ***RSS***.
- Next, repeat the process, looking for the next best predictor and best cutpoint in order to split the data further so as to minimize the ***RSS*** within each of the resulting regions.
  - This time, instead of splitting the entire predictor space, split one of the two previously identified regions. We now have three regions.
  - Again, split one of these three regions further, so as to minimize the RSS.
- The process continues until a stopping criterion is reached, e.g., we may continue until no region contains more than five observations, or we've exhausted the number of features.

Pruning a tree

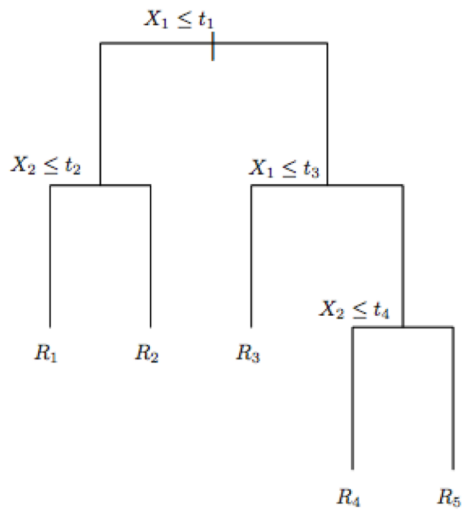
- The process described above may produce good predictions on the training set, but is likely to ***overfit*** the data, leading to poor test set performance. ***Why?***

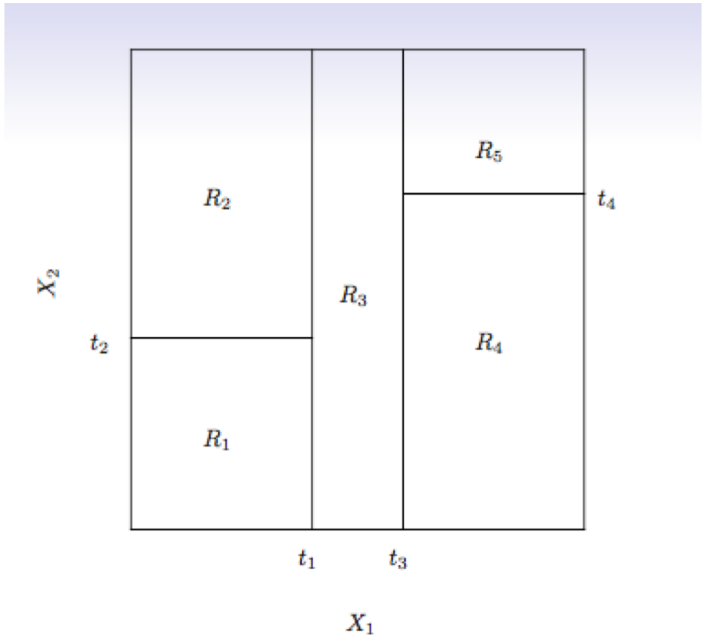




Predictions

- Predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- Right: The output of recursive binary splitting on a 2D example.
- Left: Tree corresponding to the partition in(right image)





Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to ***overfit*** the data, leading to poor test set performance. ***Why?***

- A smaller tree with fewer splits (i.e., fewer regions $R_1, \ldots, R_J$) might lead to ***lower variance*** and better interpretation at the cost of a ***little bias***.
- One alternative is to grow the tree only so long as the decrease in the ***RSS*** due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, however, a seemingly worthless split early on in the tree might be followed by a very good split that leads to a large reduction in RSS later on.

Pruning a tree

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree.
- *Cost complexity pruning* is used to do this.
- Consider a sequence of trees indexed by a non-negative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that:

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible.

- |T| indicates the number of terminal nodes of the tree T, R_m is the rectangle corresponding to the m^{th} terminal node, and $\hat{y}_{R_m}$ is the mean of the training observations in R_m.

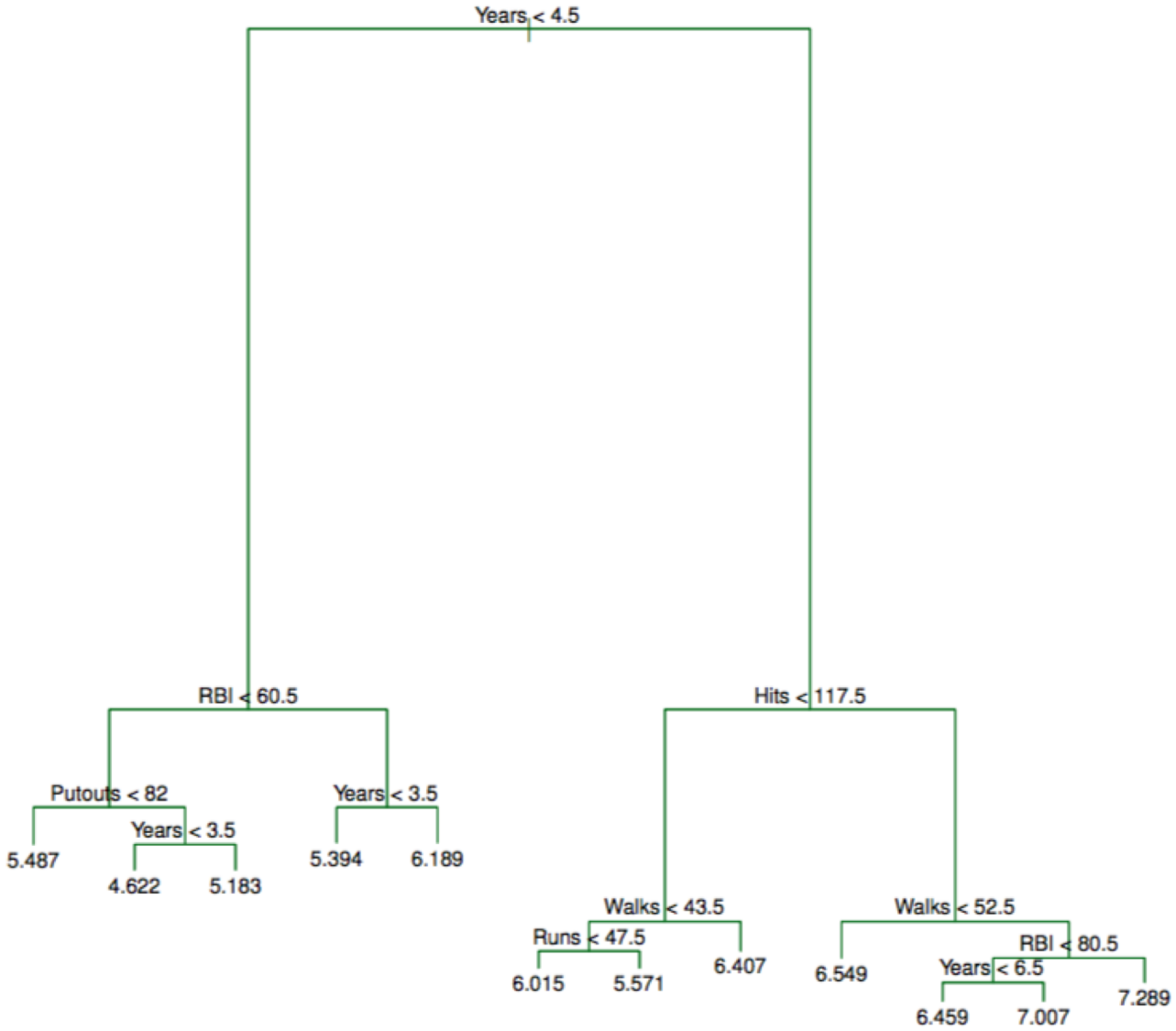
Choose the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- Select an optimal value $\hat{\alpha}$ (α "hat" for approximation) using cross-validation.
- Then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

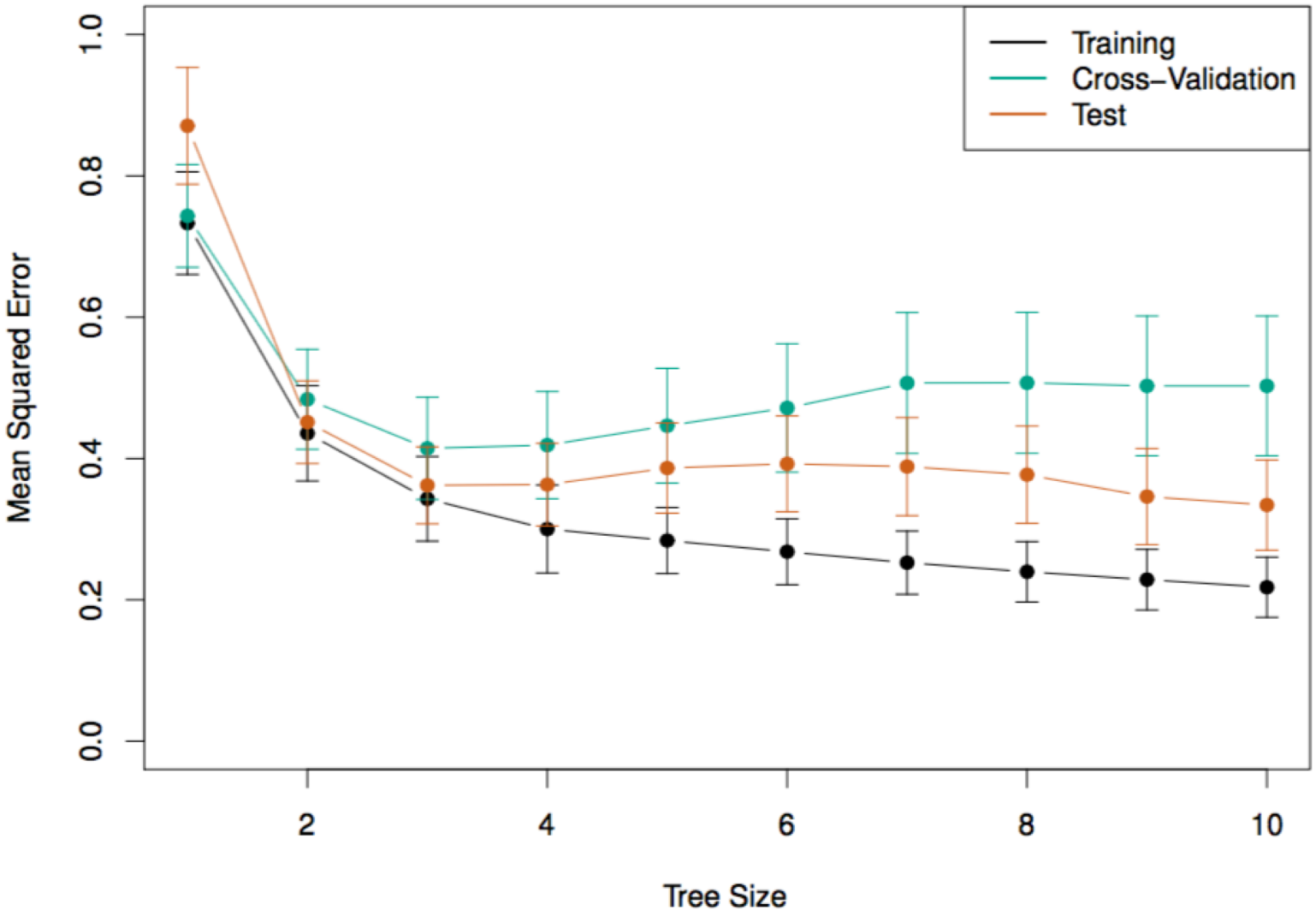
Summary: tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. For each k = 1,...,K:
   - Repeat Steps 1 and 2 on the (K−1)/Kth fraction of the training data, excluding the kth fold.
   - Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.
   - Average the results, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Baseball example



Baseball example



Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative (class) response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

Classification Trees

- Use recursive binary splitting to grow a classification tree.
- In the classification setting, RSS cannot be used as a criterion for making the binary splits
- A natural alternative to RSS is the *classification error rate*. This is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

- $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the k^{th} class.
- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

Gini index and Deviance

- The Gini index is defined by:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

- Is a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.
- For this reason the Gini index is referred to as a measure of node purity — a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is cross-entropy, given by K

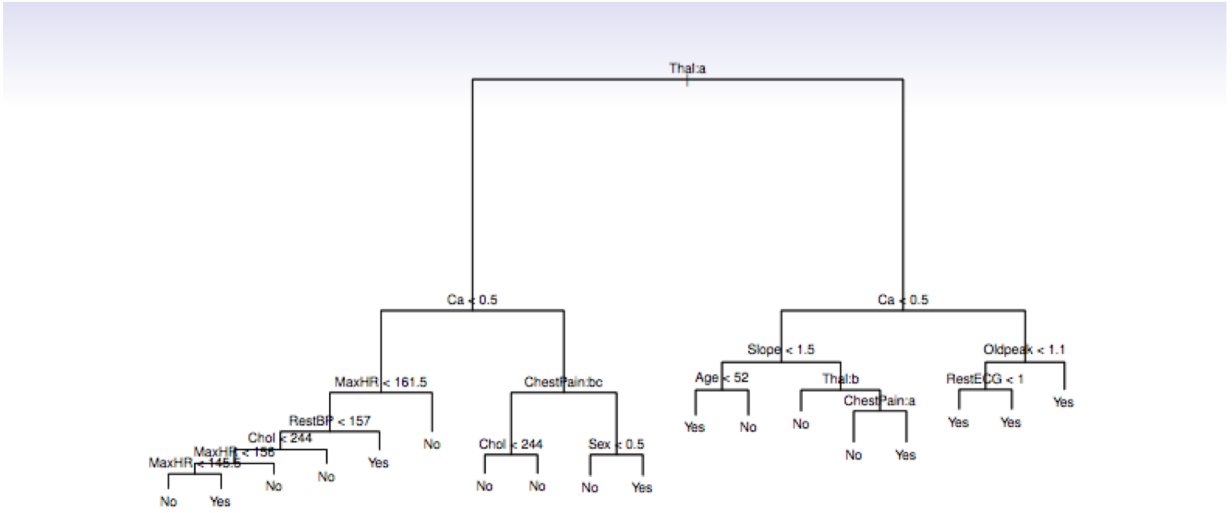
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

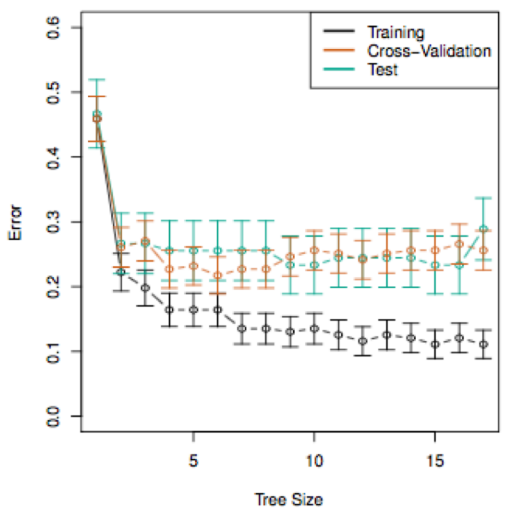
- It turns out that the Gini index and the cross-entropy are very similar numerically.

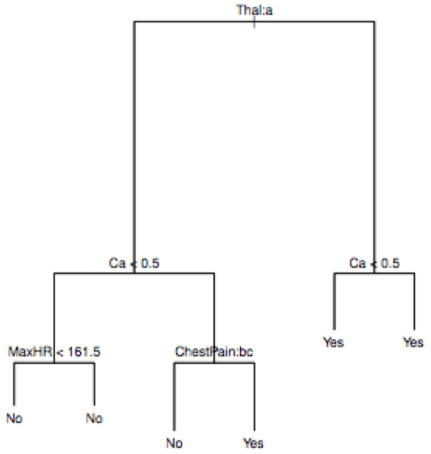
Example: heart data

- Data contain a binary outcome heart data for 303 patients who presented with chest pain.
- An outcome value of *Yes* indicates the presence of heart disease based on an angiographic test, while *No* means no heart disease.
- There are 13 predictors including Age, Sex, Chol (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with six terminal nodes.

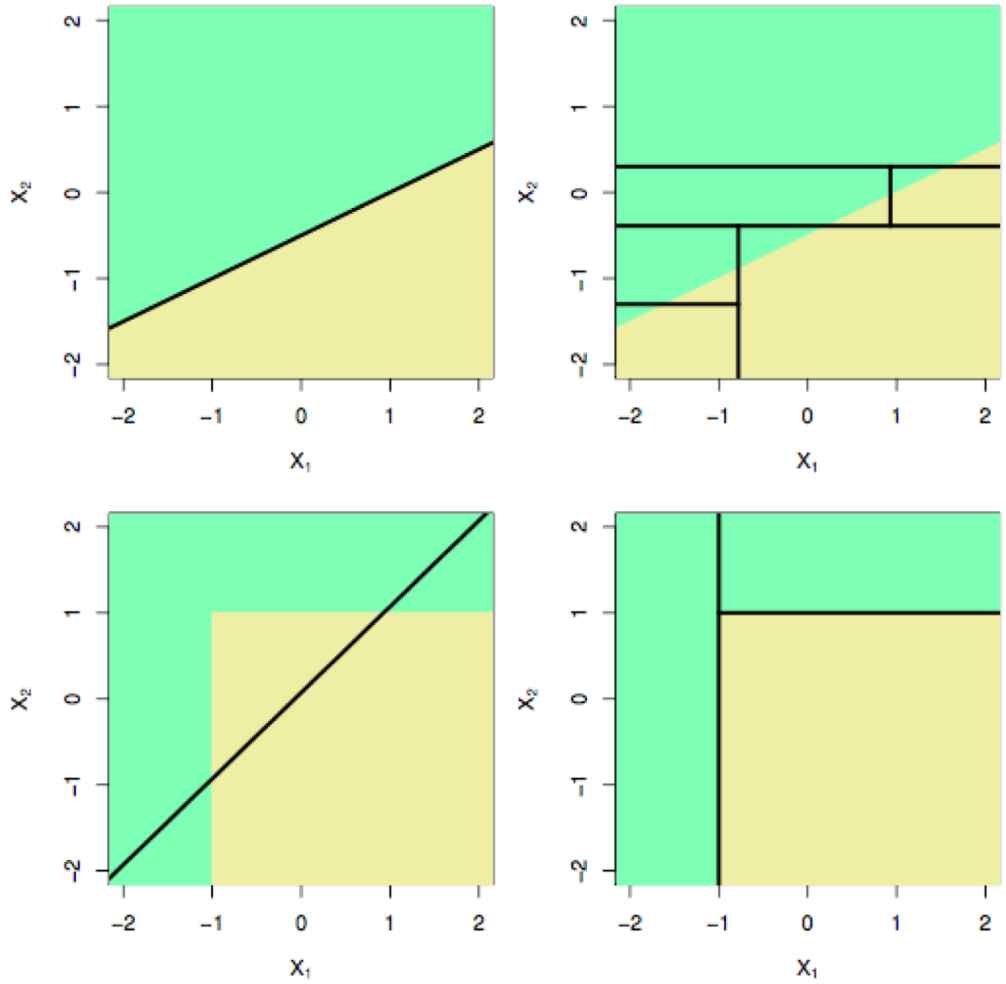
Example: heart data







Trees versus Linear Models



Top Row: True linear boundary; Bottom row: true non-linear boundary.
Left column: linear model; Right column: tree-based model

Advantages of Trees

- Trees are very easy to explain to people.
- Can generate conjunctive rules. Tree is a disjunction of conjunctive rules.
- Some believe that decision trees more closely mirror human decision-making than do the regression and classification approaches we talked about earlier.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Disadvantages of Trees

- 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. We introduce these concepts later.

STOP – supplemental material

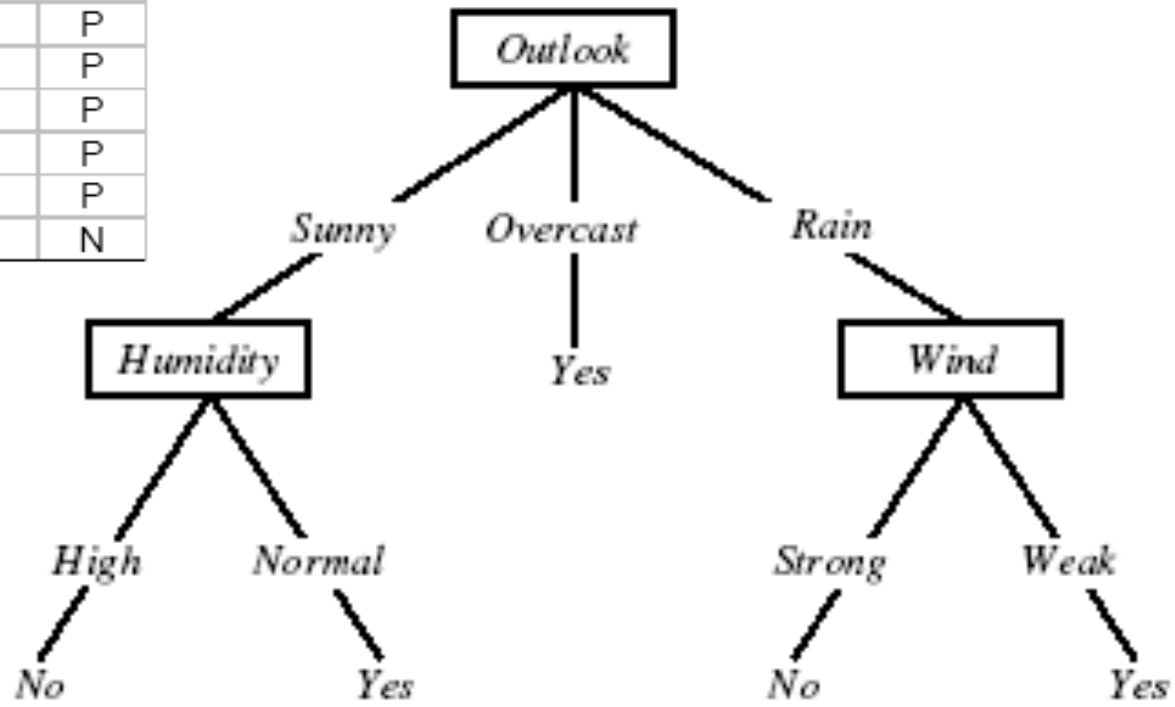
DT Classification

Classifying a sample using a decision tree:

1. Starting at the root node of the tree
2. Test the attribute specified by this node.
3. Move down the tree branch corresponding to the value of the attribute in the given example.
4. Repeat process for each subtree rooted at the new node.
5. Each leaf represents classification of instance.

DT for Play Tennis

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Play Tennis Example

- How to classify?:
  - {outlook=sunny, temperature=hot, humidity=high, wind=strong}

- Decision trees represent a *disjunction of conjunctions* of the attribute values of instances.

- For positive classification, the previous decision tree corresponds to:

  (outlook=sunny ^ humidity=high) v (outlook=overcast) v (outlook=rain ^ wind=weak)

Appropriate problems DT's

- Instances describable by *attribute-value* pairs
- Target function is discrete values
  - Note DT's can be modified to handle real valued parameters.
- Disjunctive hypothesis may be required
- Possibly noisy training data

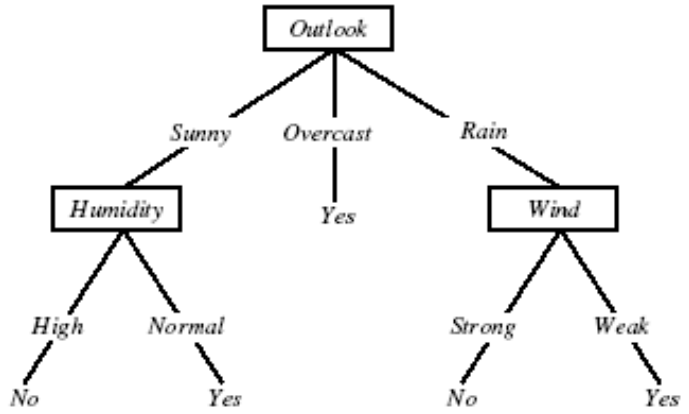
- Examples:
  - Gene expression
  - Equipment, or medical diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Skeletal tracking (random forests)

Top-down induction (creation) of DTs

Main loop:

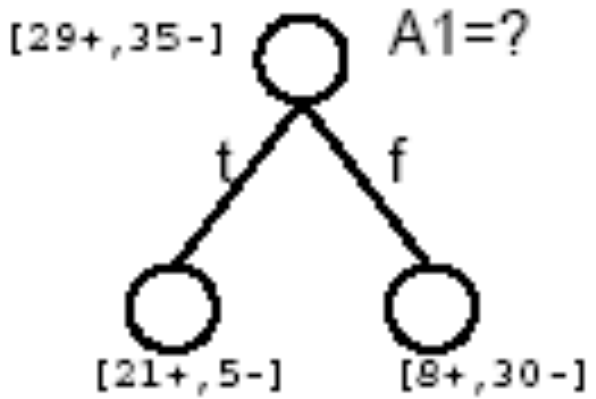
1. *A* <- select the "*best*" decision attribute for next *node*
2. Assign *A* as decision attribute for *node*
3. For each value of *A*, create new descendant of *node*
4. Sort training examples to current leaf *nodes*
5. If training examples perfectly classified, then STOP
6. Else iterate over new leaf nodes

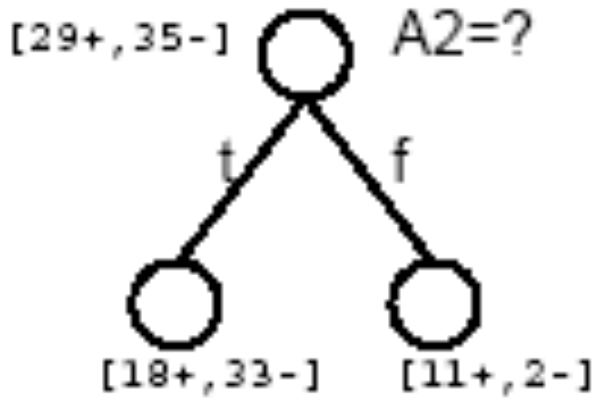
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Note: Show how samples are sorted

Which attribute is *best*?

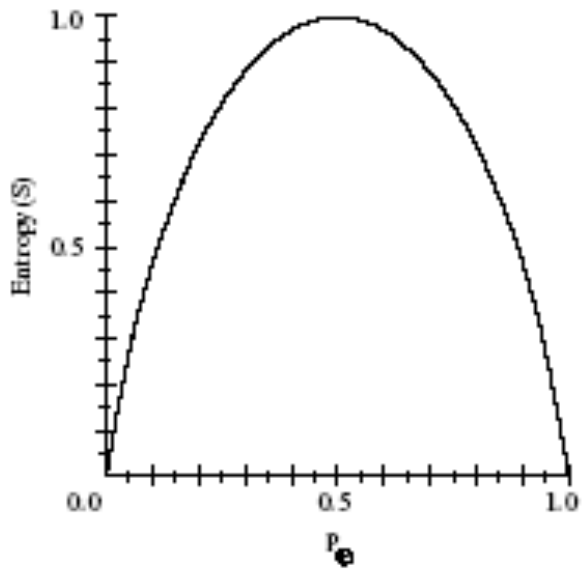




Entropy

Given *S*, a set of training examples:

- Entropy measures the impurity of S
- p+ is proportion of positive examples of S
- p- is proportion of negative examples of S



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

Entropy

Entropy(S) = expected number of bits needed to encode class(+/-) of randomly drawn member of S (under the optimal, shortest-length code)

- MDL – minimum description length

Why?

Information theory (Claude Shannon): optimal length code assigns

$-log_2 p$ bits to message having probability p.

So, expected number of bits to encode + or – examples of random member of S:

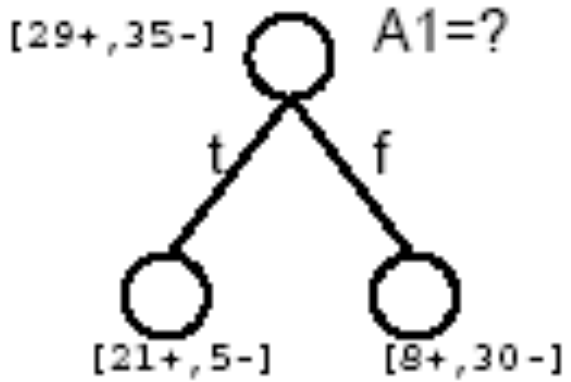
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

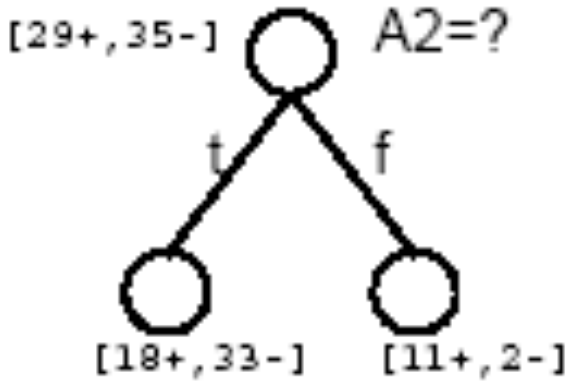
$$Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

Information Gain

Gain(S,A) = expected *reduction* in entropy due to sorting of A

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





Calculating Info Gain (Humidity)

entropy(s) 0.940286 // 9/14*(-LOG(9/14,2))+5/14*(-LOG(5/14,2))

entropy (s, humidity for high) 0.985228 // 3/7*(-LOG(3/7,2))+4/7*(-LOG(4/7,2))

entropy (s, humidity for normal) 0.591673 // 6/7*(-LOG(6/7,2))+1/7*(-LOG(1/7,2))

Gain(S, Humidity) = entropy(s) - 7/14*entropy (s, humidity for high) - 7/14*entropy (s, humidity for normal)

Gain(S, Humidity) = 0.151836 // D2- 7/14*D4 - 7/14*D6

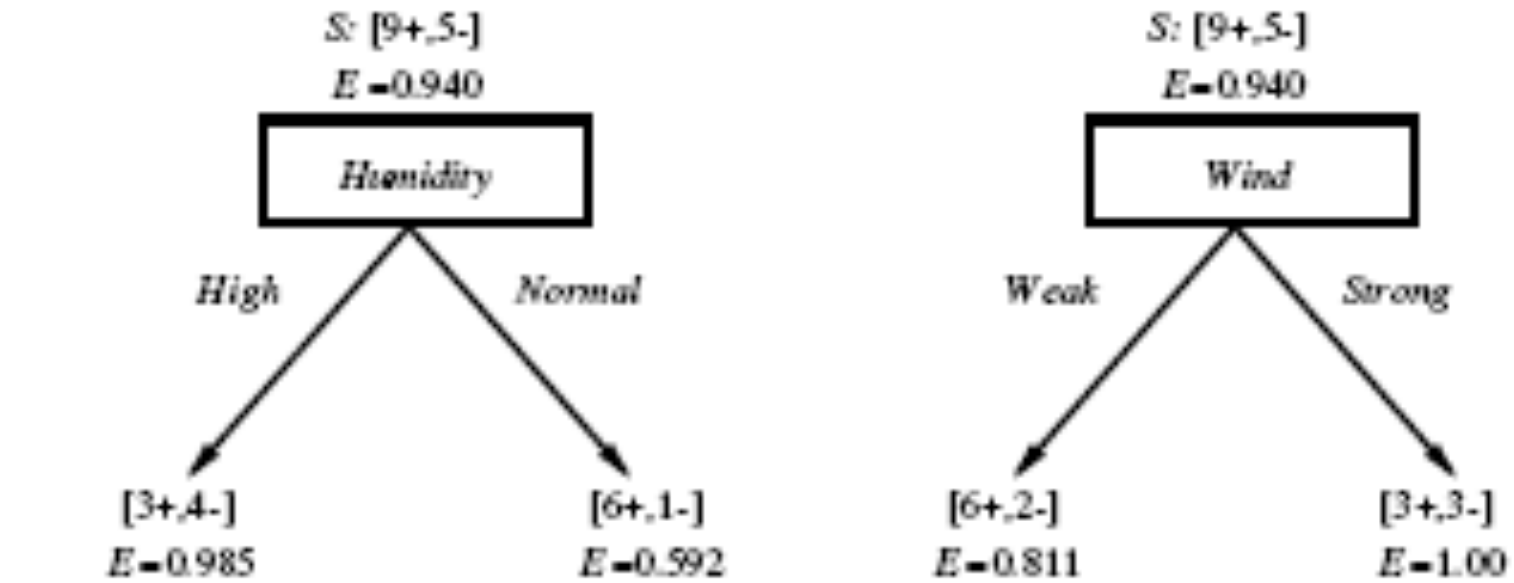
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

Training Examples

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

Selecting the Next Attribute

Which attribute is the best classifier?

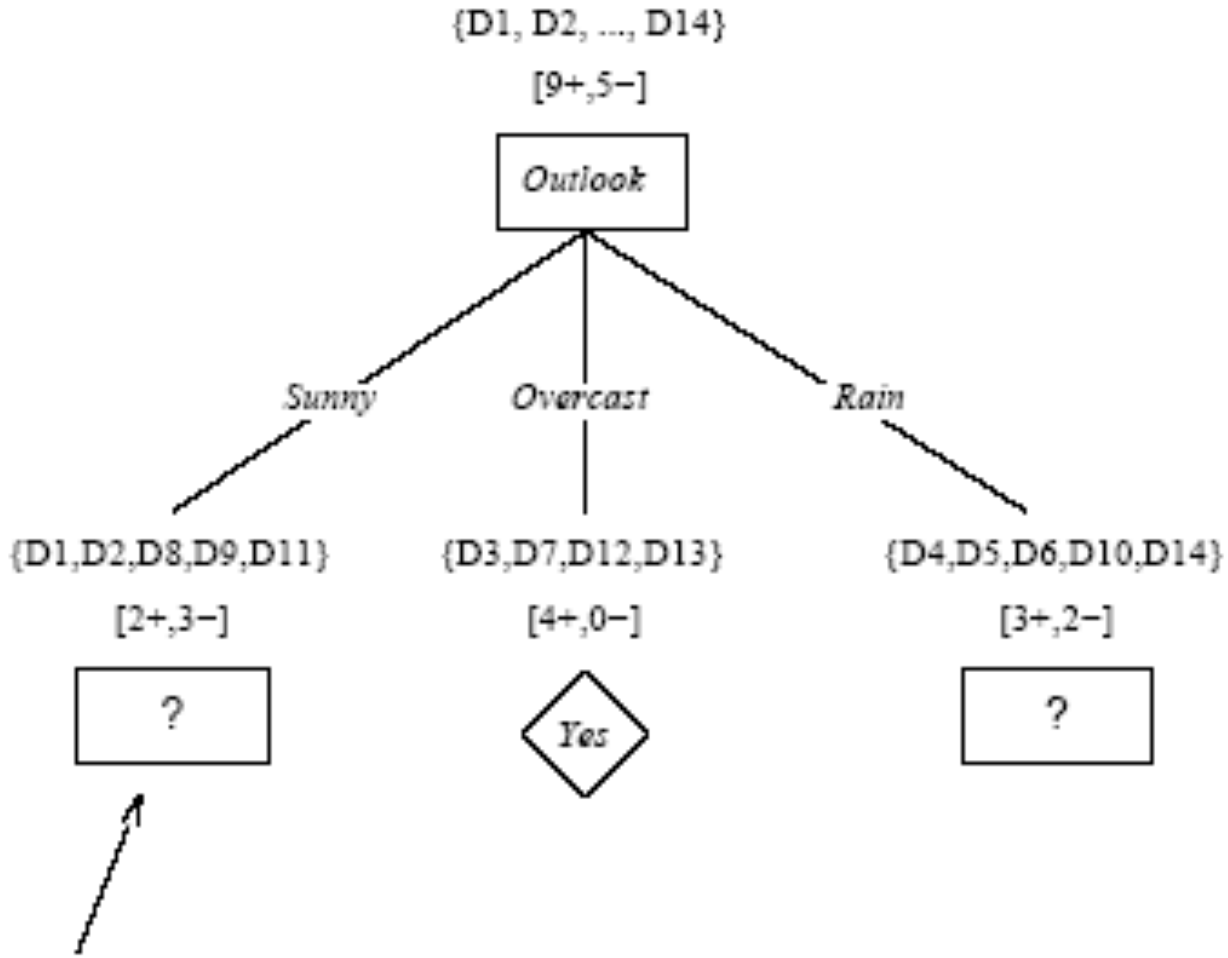


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



Which attribute should be tested here?

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

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function is surely in there…
- Outputs a single hypothesis
- No backtracking
  - Local minima
- Statically based search choices
  - Robust to noisy data
- Inductive bias: approximate "*prefer shortest tree*"

Inductive Bias of ID3

Note *H* is the power set of instances *X*

- Unbiased?

Not really…

- Prefer short trees, and for those with high info gain attributes near the root
- Bias is a preference for some *hypothesis*, rather than a restriction of hypothesis space *H*
- *Occam's razor: prefer the shortest hypothesis that fits the data.*

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypo's than long hypo's
- A short hypo that fits data unlikely to be coincidence
- A long hypo that fits data might be coincidence

Argument opposed:

- There are many ways to define small sets of hypo's
  - *e.g., all trees with a prime number of nodes use attributes with "Z"*
- *What's so special about small sets based on size of hypothesis?*

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Training data (observations, measurements, etc.) is accompanied *by labels indicating the class* of each observation
  - New data is classified based on a model built from the training set
- Supervised learning (regression)
  - Training data (observations, measurements, etc.) is accompanied *by numeric output value* for each observation
  - Numeric output values are predicted based on a model built from the training set
- Unsupervised learning (clustering, frequent item sets, subsequences)
  - The class *labels of training data is unknown*
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes, associations, or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

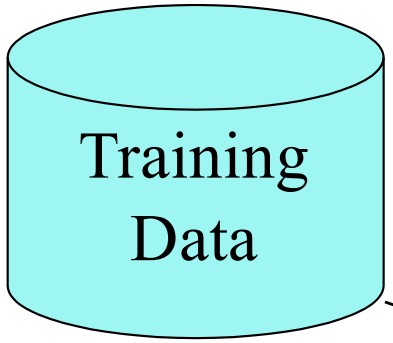
- Classification
  - *predicts categorical class* labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction (Regression)
  - *models continuous-valued functions*, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

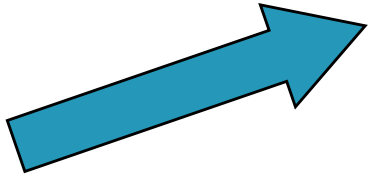
Classification—Two-Step Process

1. Model construction: describing a set of predetermined classes
   - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
   - The set of tuples used for model construction is training set
   - The model is represented as a statistical model, neural net, classification rules, decision trees, or mathematical formulae
2. Model usage: for classifying future or unknown objects
   - Estimate accuracy of the model
     - The known label of test sample is compared with the classified result from the model
     - Accuracy rate is the percentage of test set samples that are correctly classified by the model
     - Test set is independent of training set (otherwise *overfitting*)
   - If the accuracy is acceptable, use the model to classify new data

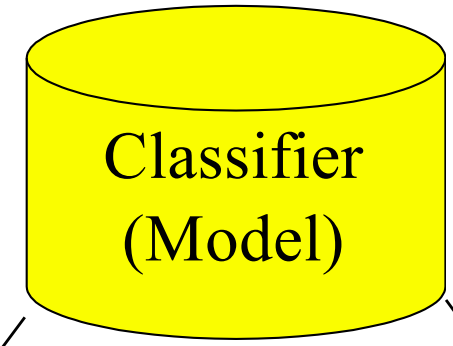
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction





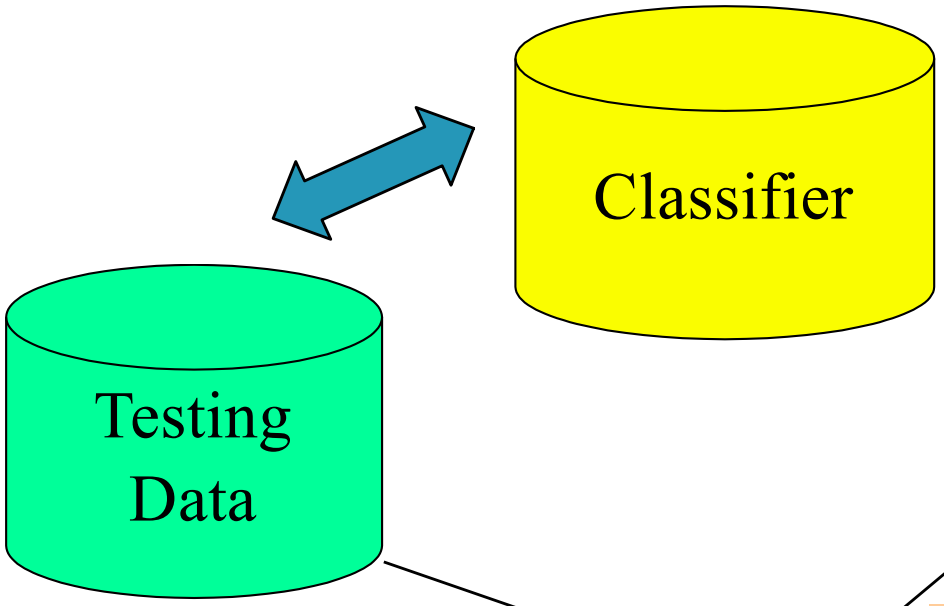
Classification
Algorithms



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

(Jeff, Professor, 4)

Tenured?

Yes

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction 
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Play Tennis?

(Tom Mitchell, Machine Learning)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision Trees (DT)

- Widely used, practical method for *inductive inference* (probable from evidence).
- Method for approximating discrete valued functions.
- Exhaustively search *hypothesis space*.
- Can be used for representing *if-then-else* rules.

DT Representation

- Classify instances by sorting them down a tree from the root to some leaf node.
- Each node in the tree specifies a test of some attribute (feature) of the instance.
- Each branch descending from that node represents one of the possible values for this attribute.
- Leaf node provides classification instance.

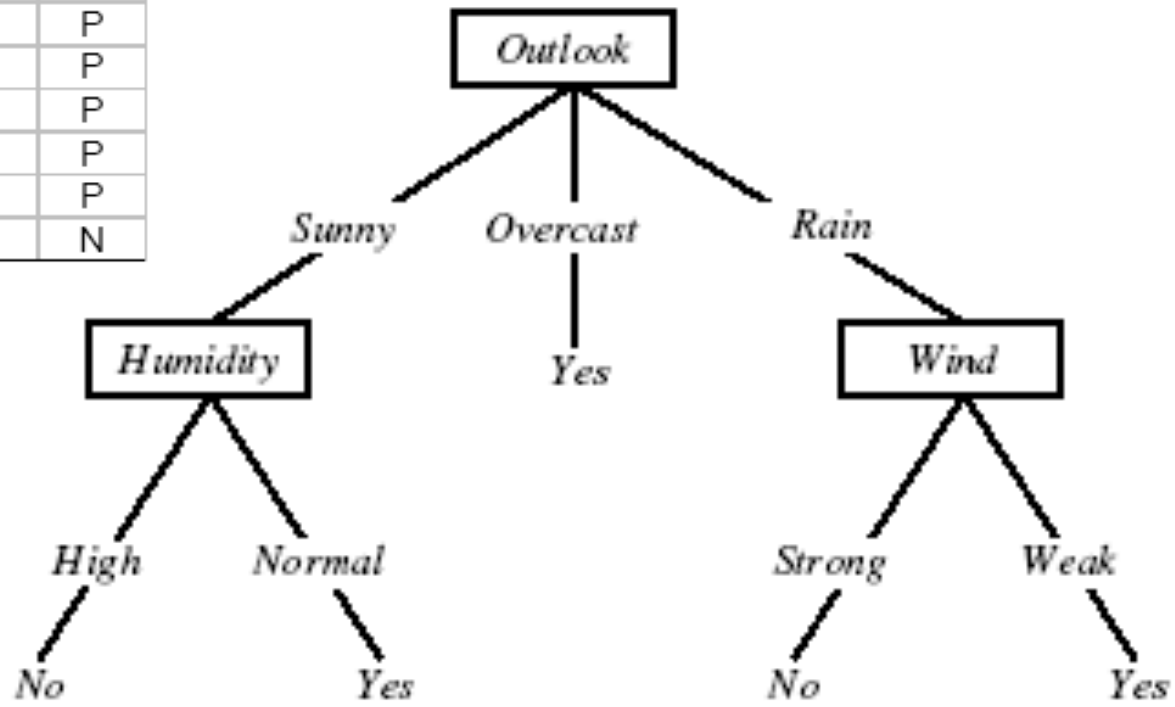
DT Classification

Classifying a sample using a decision tree:

1. Starting at the root node of the tree
2. Test the attribute specified by this node.
3. Move down the tree branch corresponding to the value of the attribute in the given example.
4. Repeat process for each subtree rooted at the new node.
5. Each leaf represents classification of instance.

DT for Play Tennis

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Play Tennis Example

- How to classify?:
  - {outlook=sunny, temperature=hot, humidity=high, wind=strong}

- Decision trees represent a *disjunction of conjunctions* of the attribute values of instances.

- For positive classification, the previous decision tree corresponds to:

  (outlook=sunny ^ humidity=high) v (outlook=overcast) v (outlook=rain ^ wind=weak)

Predict C-Section Risk Example

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Appropriate problems DT's

- Instances describable by *attribute-value* pairs
- Target function is discrete values
  - Note DT's can be modified to handle real valued parameters (regression)
- Disjunctive hypothesis may be required
- Possibly noisy training data

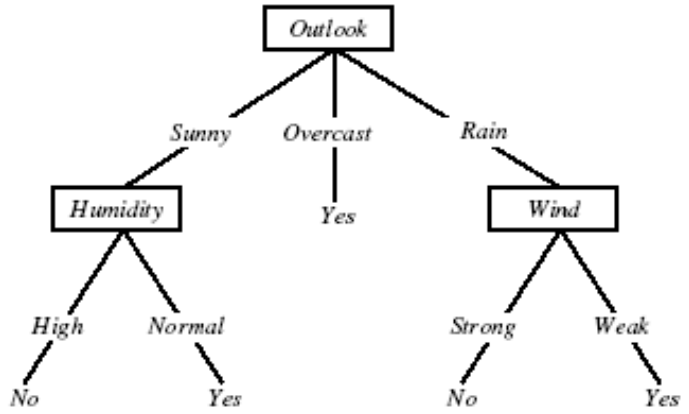
- Examples:
  - Gene expression
  - Equipment, or medical diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Skeletal tracking (random forests)

Top-down induction (creation) of DTs

Main loop:

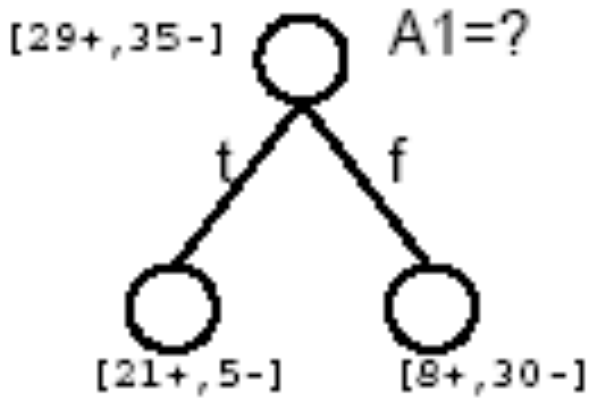
1. *A* <- select the "*best*" decision attribute for next *node*
2. Assign *A* as decision attribute for *node*
3. For each value of *A*, create new descendant of *node*
4. Sort training examples to current leaf *nodes*
5. If training examples perfectly classified, then STOP
6. Else iterate over new leaf nodes

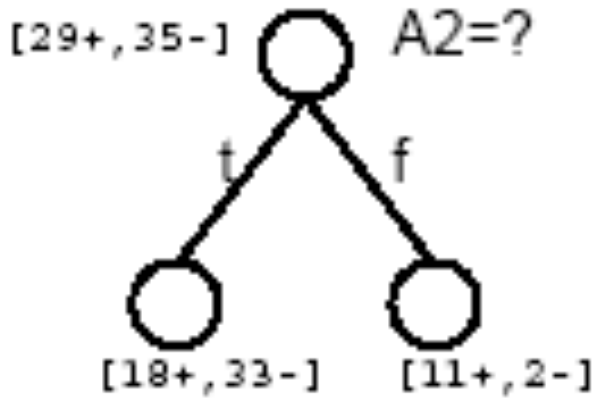
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Note: Show how samples are sorted

Which attribute is *best*?

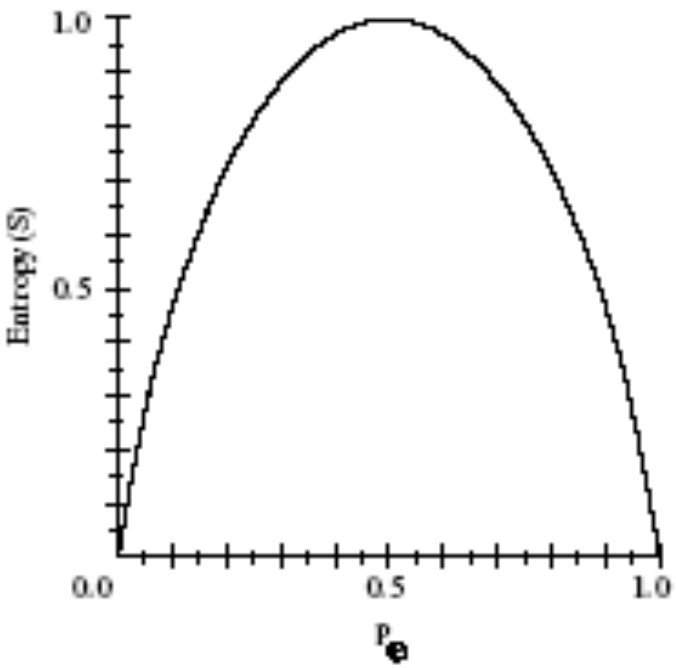




Entropy

Given *S*, a set of training examples:

- Entropy measures the impurity of S
- p+ is proportion of positive examples of S
- p- is proportion of negative examples of S



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

-0.5*math.log(0.5,2) -0.5*math.log(0.5,2)

Entropy

Entropy(S) = expected number of bits needed to encode class(+/-) of randomly drawn member of S (under the optimal, shortest-length code)

- MDL – minimum description length

Why?

Information theory (Claude Shannon): optimal length code assigns

$-log_2 p$ bits to message having probability p.

So, expected number of bits to encode + or – examples of random member of S:

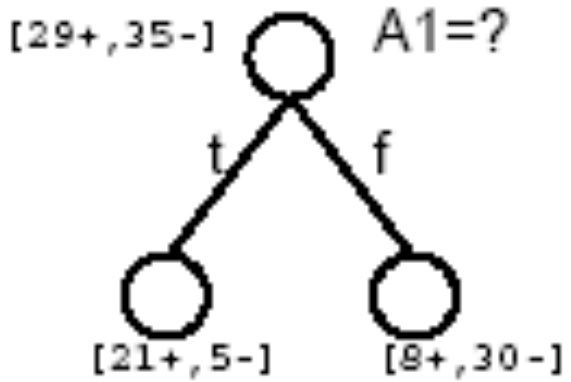
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

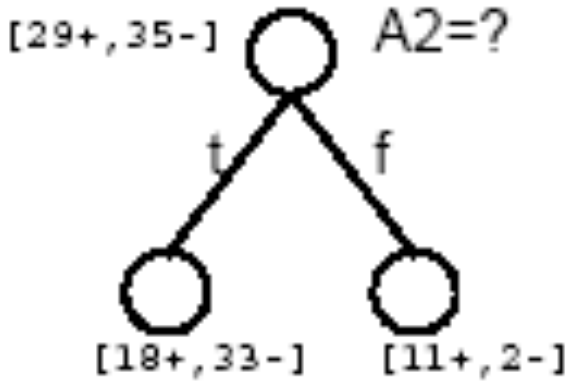
$$Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

Information Gain

Gain(S,A) = expected *reduction* in entropy due to sorting of A

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





Calculating Info Gain (Humidity)

entropy(s)) //D2 0.940286 // 9/14*(-LOG(9/14,2))+5/14*(-LOG(5/14,2))

entropy (s, humidity for high) //D4 0.985228 // 3/7*(-LOG(3/7,2))+4/7*(-LOG(4/7,2))

entropy (s, humidity for normal) //D6 0.591673 // 6/7*(-LOG(6/7,2))+1/7*(-LOG(1/7,2))

Gain(S, Humidity) = entropy(s) - 7/14*entropy (s, humidity for high) - 7/14*entropy (s, humidity for normal)

Gain(S, Humidity) = 0.151836 // D2- 7/14*D4 - 7/14*D6

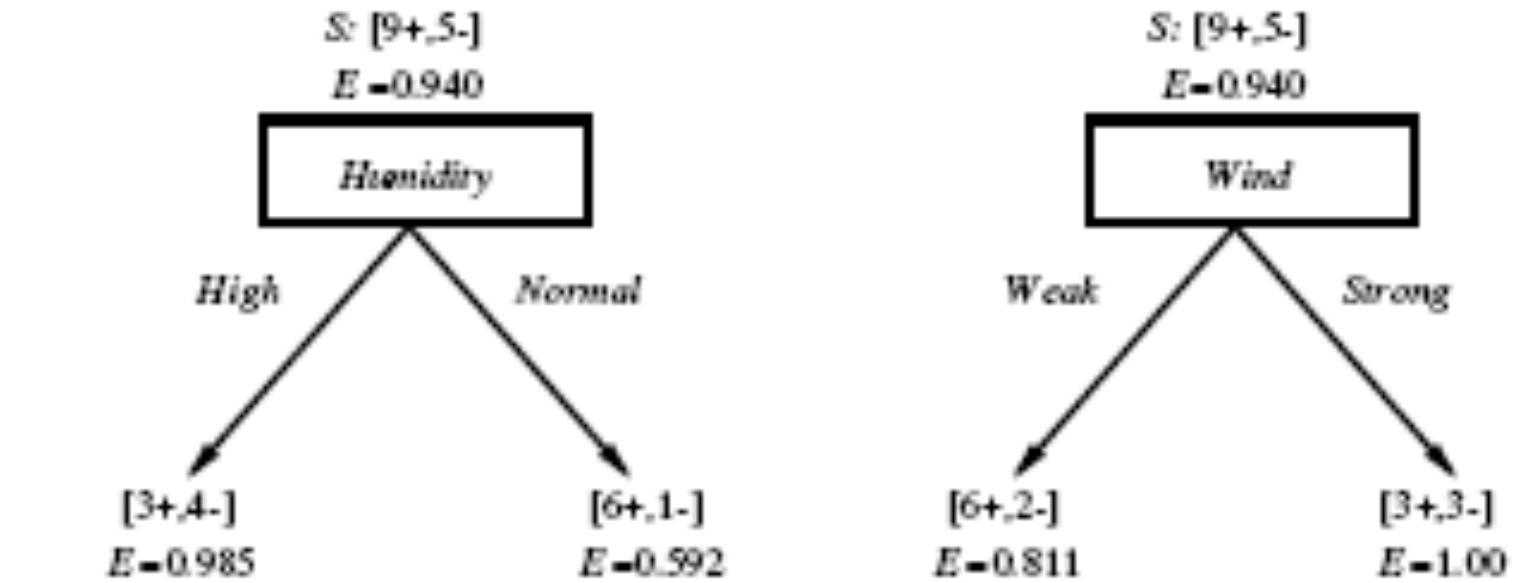
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

Training Examples

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

Selecting the Next Attribute

Which attribute is the best classifier?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

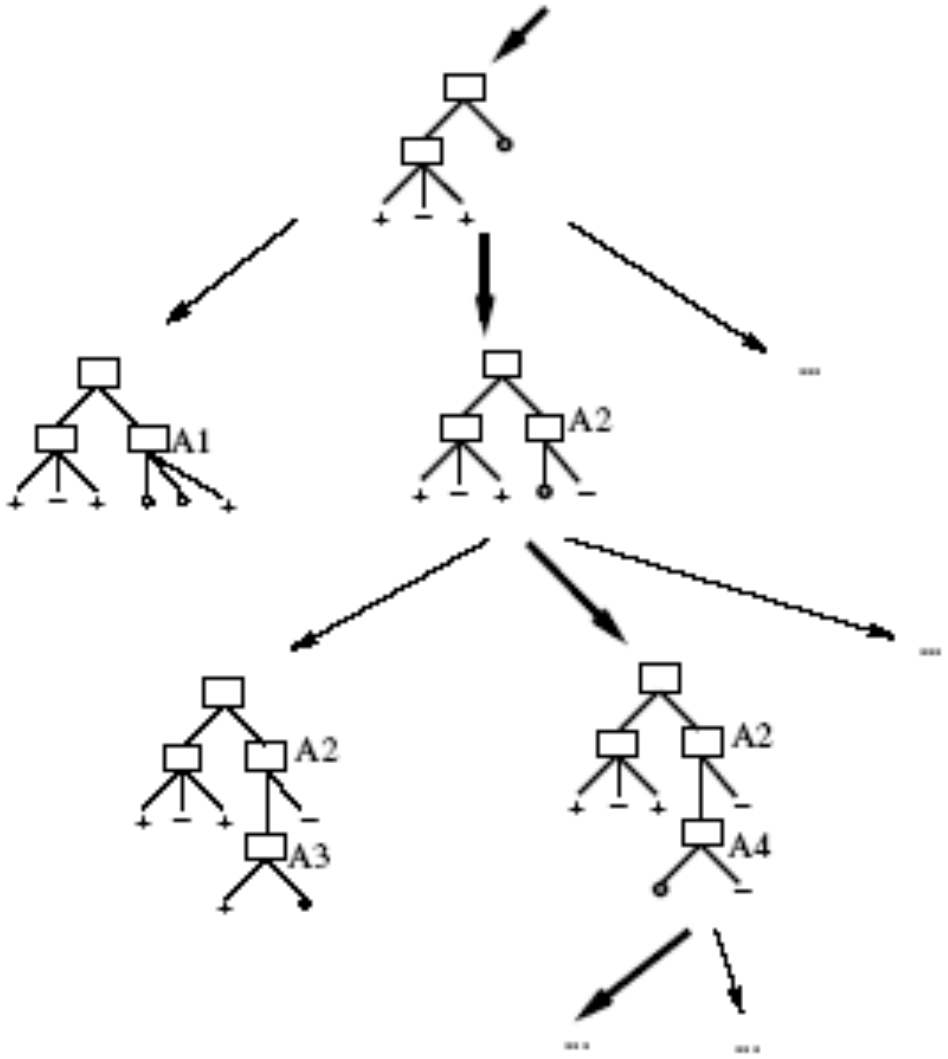
= .940 - (8/14).811 - (6/14)1.0
= .048

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function is surely in there…
- Outputs a single hypothesis
- No backtracking
  - Local minima
- Statically based search choices
  - Robust to noisy data
- Inductive bias: approximate "*prefer shortest tree*"

Inductive Bias of ID3

Note *H* is the power set of instances *X*

- Unbiased?

Not really…

- Prefer short trees, and for those with high info gain attributes near the root
- Bias is a preference for some *hypothesis*, rather than a restriction of hypothesis space *H*
- *Occam's razor: prefer the shortest hypothesis that fits the data.*

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypo's than long hypo's
- A short hypo that fits data unlikely to be coincidence
- A long hypo that fits data might be coincidence

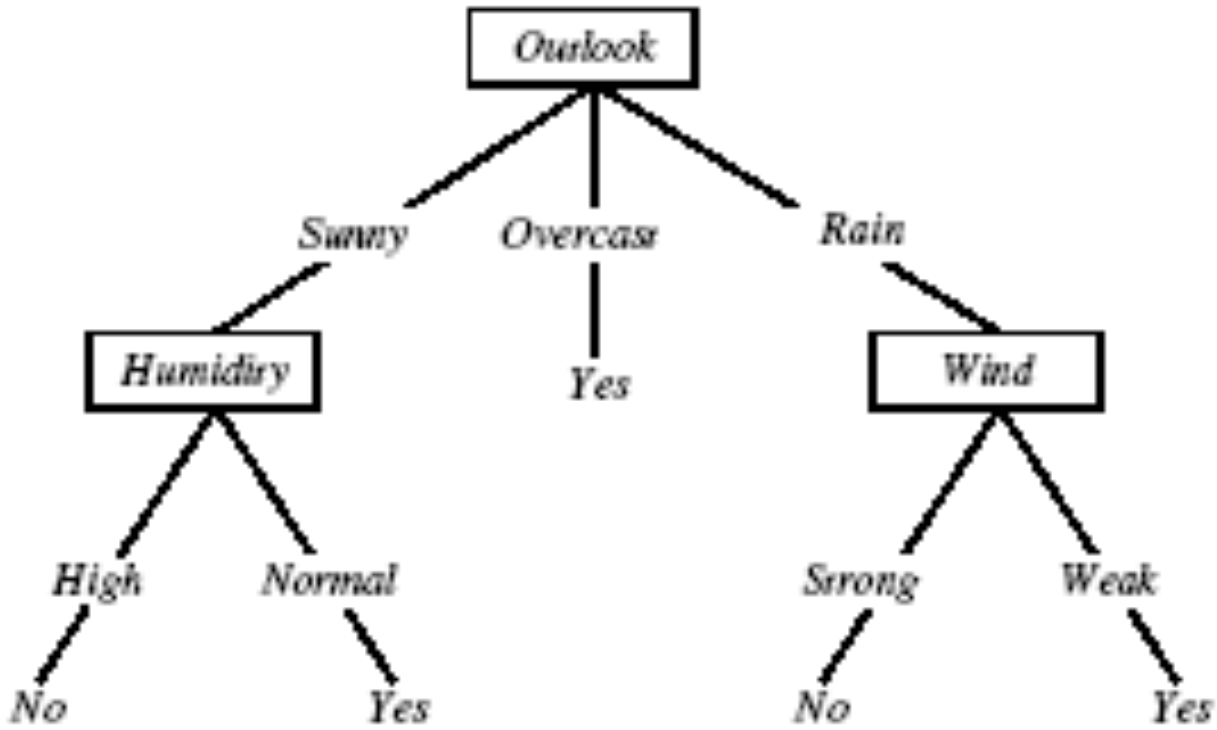
Argument opposed:

- There are many ways to define small sets of hypo's
  - *e.g., all trees with a prime number of nodes use attributes with "Z"*
- *What's so special about small sets based on size of hypothesis?*

Overfitting in Decision Trees

Consider adding noisy training example D15:

- *Sunny, Hot, Normal, Strong, Play tennis=No*



Overfitting

- Consider error of hypothesis *h* over:
  - Training data: $error_{train}(h)$
  - Entire distribution *D* of data: $error_D(h)$

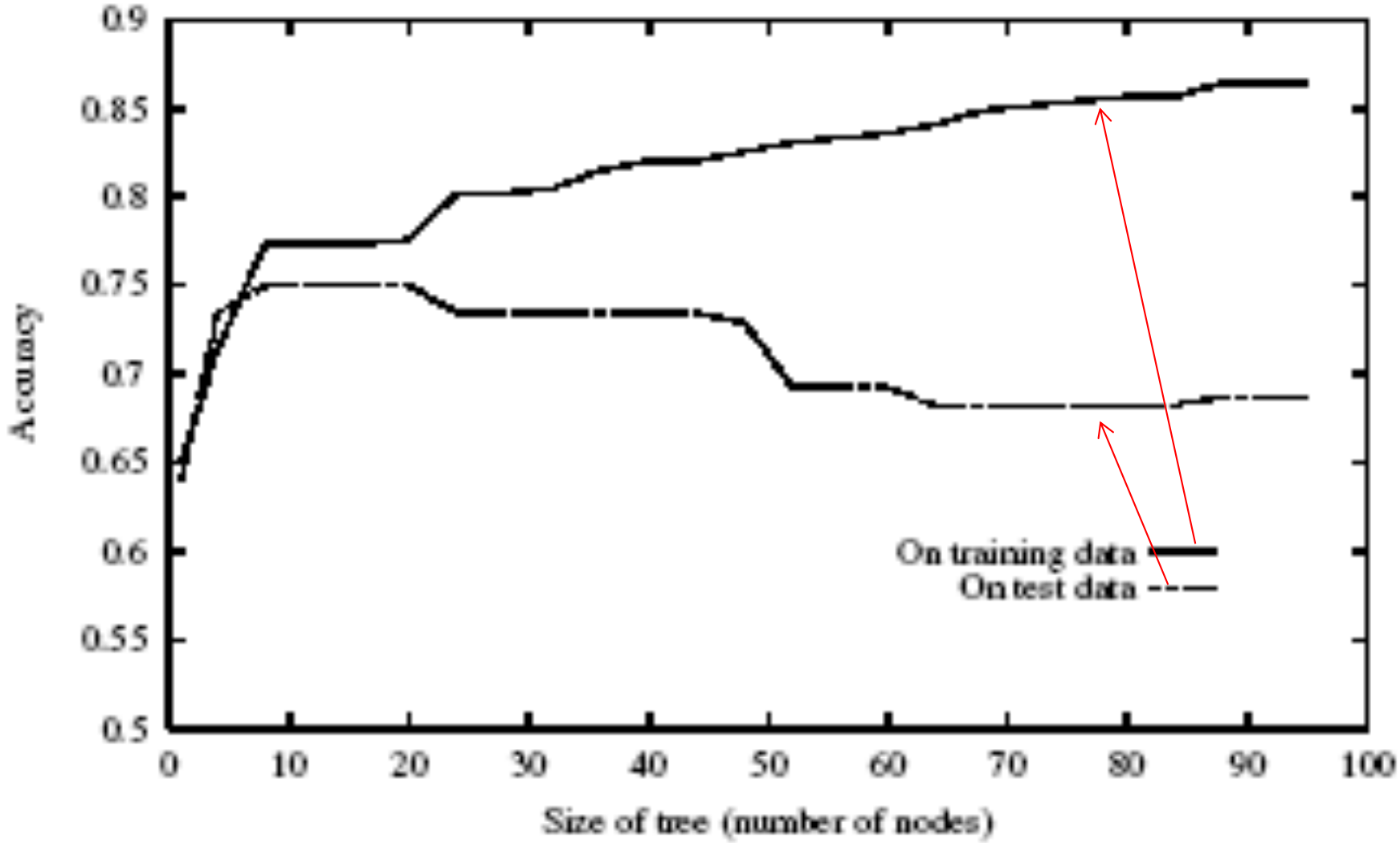
- Hypothesis $h \in H$ *overfits* training data if there is an alternative hypothesis $h' \in H$ such that:

$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

Overfitting



Avoiding Overfitting

Avoid *Overfitting:*

1. Stop growing when data split is not statistically significant
2. Grow full tree, then prune
3. Grow many trees w/ randomly selected reduced numbers of attributes, learn function to weight trees (Random Forests – state of the art technique)

How to select best tree:

- Measure performance over training data
- Measure performance over separate validation set

Minimum Description Length

- *MDL: minimize (size(tree)+size(misclassification))*

Reduced-Error Pruning

- Start with completed tree
- Split data into training and validation set
- Do until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node (plus those below it)
  2. *Greedily* remove the one that most improves validation set accuracy

- Produces smallest version of most accurate subtree

Rule Post-Pruning

1. Convert tree to equivalent rules
2. Prune each rule independently of others
3. Evaluate impact on validation set
4. Sort final rule into desired sequence of use.

- Most frequently used method, C4.5

Regression

- The goal is to find boxes $R_1,\ldots,R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2,$$

- where y^{Rj} is the mean response for the training observations within the j^{th} box.

Summary

- Learning needed for unknown environments
- For supervised learning, the aim is to find a simple hypothesis *approximately* consistent with training examples that performs well on unseen data
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set
- Overfitting hampers a machine learners ability to generalize to unseen examples

Pros and Cons

- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Hence we also discuss bagging, random forests, and boosting. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

STOP DT

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods 
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayes*, makes strong independence assumptions, but can perform well.
- Comparable performance with basic decision trees and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct (online learning) — prior knowledge can be combined with observed data
- Standard: Provides a theoretically sound standard of optimal

Naïve Bayes Classifier

- Bayes theorem
- Combines probability of each feature with respect to a class label.
- Makes strong independence assumption between features, i.e., independence between features
- Sample applications:
  - Classify email as spam based on sender, and text
  - Diagnose meningitis based on chest-xray, symptom
  - Classify fruit from shape and color
  - Determine life style from education and salary

Review conditional probability

- Can factor joint probability using the chain rule:

  $P(a \wedge b) = P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$

- And express the joint probability by conditioning on a or b:

  $P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$

- … and derive *Bayes* Theorem:

  $P(a \mid b) = P(b \mid a)\, P(a) / P(b)$

  $P(b \mid a) = P(a \mid b)\, P(b) / P(a)$

Naïve Bayes Classifier

- Lets say we have a hypothesis H, & we want to calculate the probability of the hypothesis being correct.
- Hypothesis H: given feature x_1, x_2 => object is a Peach
- Calculate probability that x_1, x_2 is a Peach
  - P(H: x_1, x_2 is a Peach)
  - P(H: x_1, x_2 is an Apricot)

1. Calculate each of these probabilities
2. Choose the highest probability

Naïve Bayes Classifier

- P(H|X) *Posterior* probability of hypothesis H (class)
  - X: $\{x_1, x_{2\ldots}, x_n\}$
  - Shows the confidence/probability of H given X
  - x_1: shape=round, x_2: color=orange
  - H: x_1, x_2 is a peach

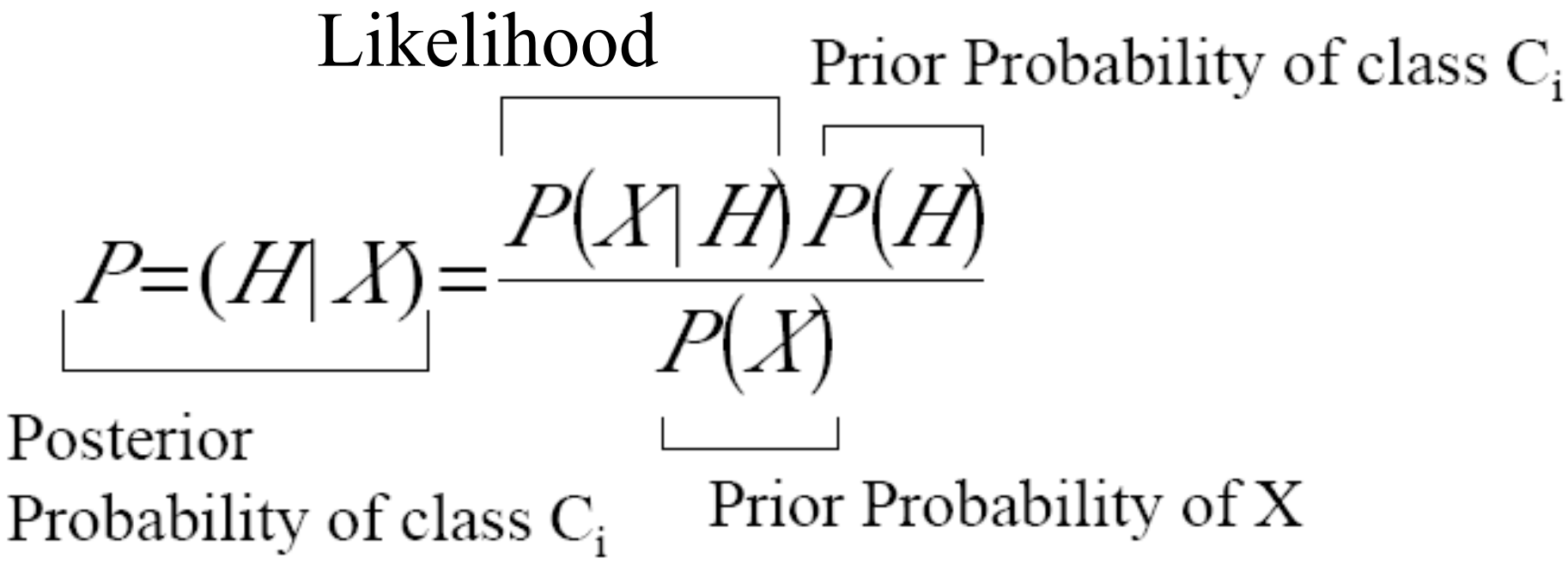
- P(H) Prior probability of hypothesis H
  - Represents the probability of H just happening, regardless of data.
  - E.g. What is the probability of picking a peach from a fruit bin without knowledge of shape and color.

Bayes Theorem - Learning

- P(X|H) *Likelihood* - the evidence X conditioned on hypothesis H
  - Shows the confidence (*probability*) of X given H
  - Given H is true (i.e., X is a peach) calculate the probability that X is round and orange, i.e., x_1=round, x_2=orange.

- P(X) Prior probability of X
  - Represents the probability that sample is round and orange.

Bayes Theorem - Classification



Naïve Bayes Classification

- Hypothesis *H* is the class C_i.
  - Note: *P (X)* can be ignored (for classification) as it is constant for all classes.
- Assuming the independence assumption, $P(X/C_i)$ is:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Therefore:

$$P(C_i \mid X) = P(C_i) \prod_{k=1}^{n} P(x_k \mid C_i)$$

- $P(C_i)$ is the ratio of total samples in class C_i to all samples.

Naïve Bayes Classification

- For categorical attribute:
  - $P(x_k/C_i)$ is the frequency of samples having value x_k for class C_i

- For continuous (numeric) attribute:
  - $P(x_k/C_i)$ is calculated via a Gaussian density function. with a mean μ and standard deviation σ

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X} \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classification

- Having pre-calculated all $P(x_k/C_i)$, an unknown example X is classified as follows:

  1. For all classes calculate $P(C_i/X)$
  2. Assign X to the class with the highest $P(C_i/X)$

Play Tennis?

Incoming sample: X = <sunny, cool, high, true>

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Play Tennis Example: estimating $P(x_i/C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 1/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

Play Tennis Example: estimating $P(C_i/x_i)$

- An incoming sample: X = <sunny, cool, high, true>

- P(p|X) = P(X|p)*P(p) =
  P(p)*P(sunny|p)*P(cool|p)*P(high|p)*p(true|p)
  9/14 * 2/9 * 3/9 * 3/9 * 3/9= 0.0053
  // >>> .0053/(.0053+.026) =0.16932907348242812

- P(n|X) = P(X|p)*P(n) =
  P(n)*P(sunny|n)*P(cool|n)*P(high|n)*p(true|n)
  5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.0206
  // >>> .026/(.0053+.026) = 0.830670926517571 8

  Class *n* (no play) has higher probability than class *p* (play) for example X.

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be ***non-zero***. Otherwise, the likelihood will be zero:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

  - Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian smoothing)
  - *Adding 1 to each case*

    Prob(income = low) = (0+1)/(1000+3) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Avoiding the Zero-Probability Problem

- Standard approach: use logarithms
- Log of products == sum of logs

$$P(C \mid X) \approx \log_2\left(\prod_{k=1}^{n} P(x_k \mid C_i)\right) = \sum \log_2((P(x_k \mid C_i))$$

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Networks

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification 
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

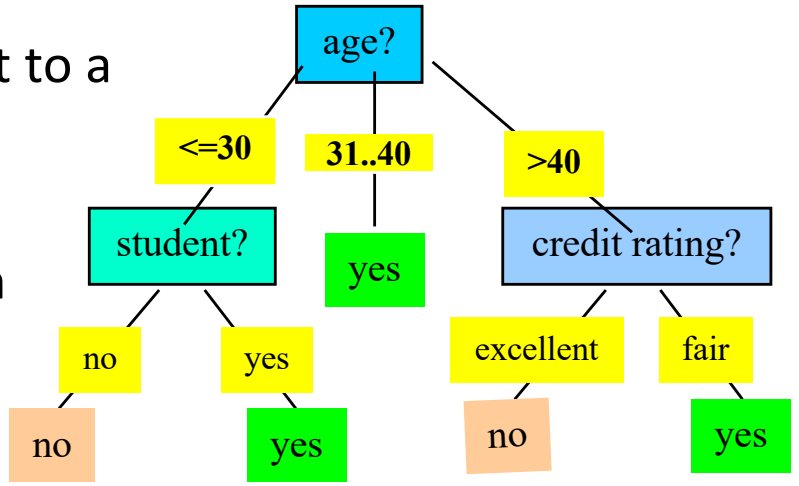
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* (support) and *accuracy* (confidence)
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = n_{covers} /|D| /* D: training data set */

  accuracy(R) = $n_{correct}$ / n_{covers}
- If more than one rule are triggered, need **conflict resolution or priority scheme**
  - Size ordering: assign the highest priority to the triggering rules that has the "toughest" most specific requirement (i.e., *most attribute tests*)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*

IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

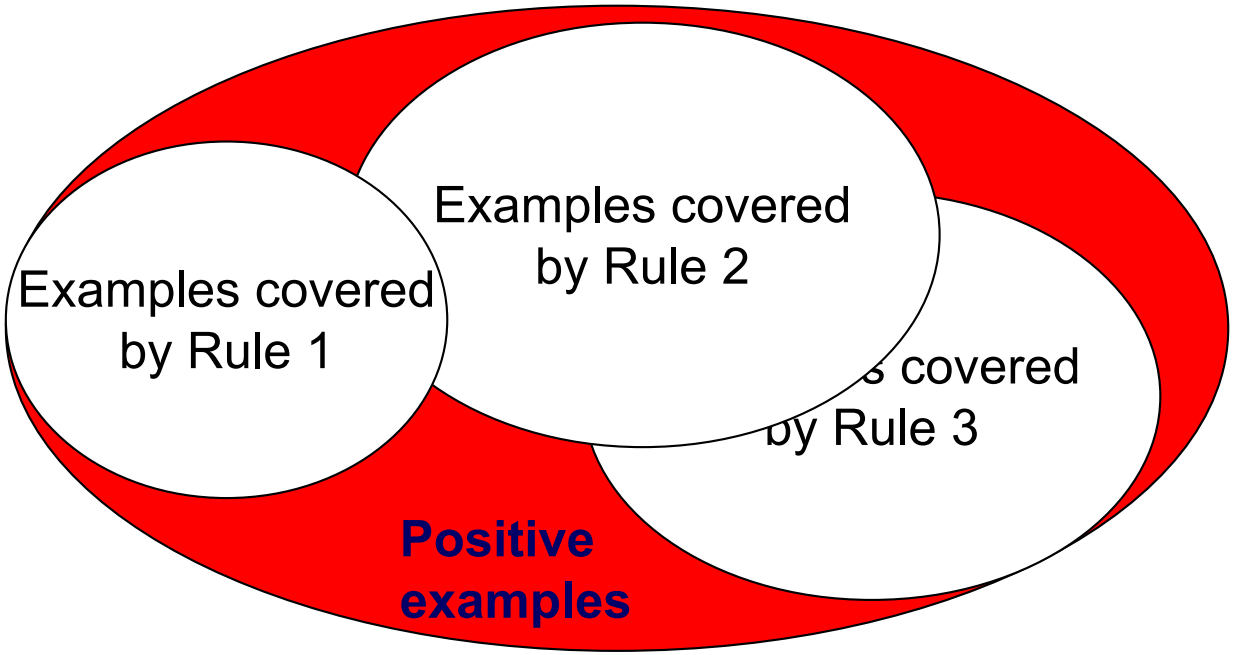
- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - Repeat the process on the remaining tuples until *termination condition*, e.g., when there are no more training examples, or when the quality of a rule returned is below a user-specified threshold

Sequential Covering Algorithm

while (enough target tuples left)

generate a rule

remove positive target tuples satisfying this rule



Rule Generation

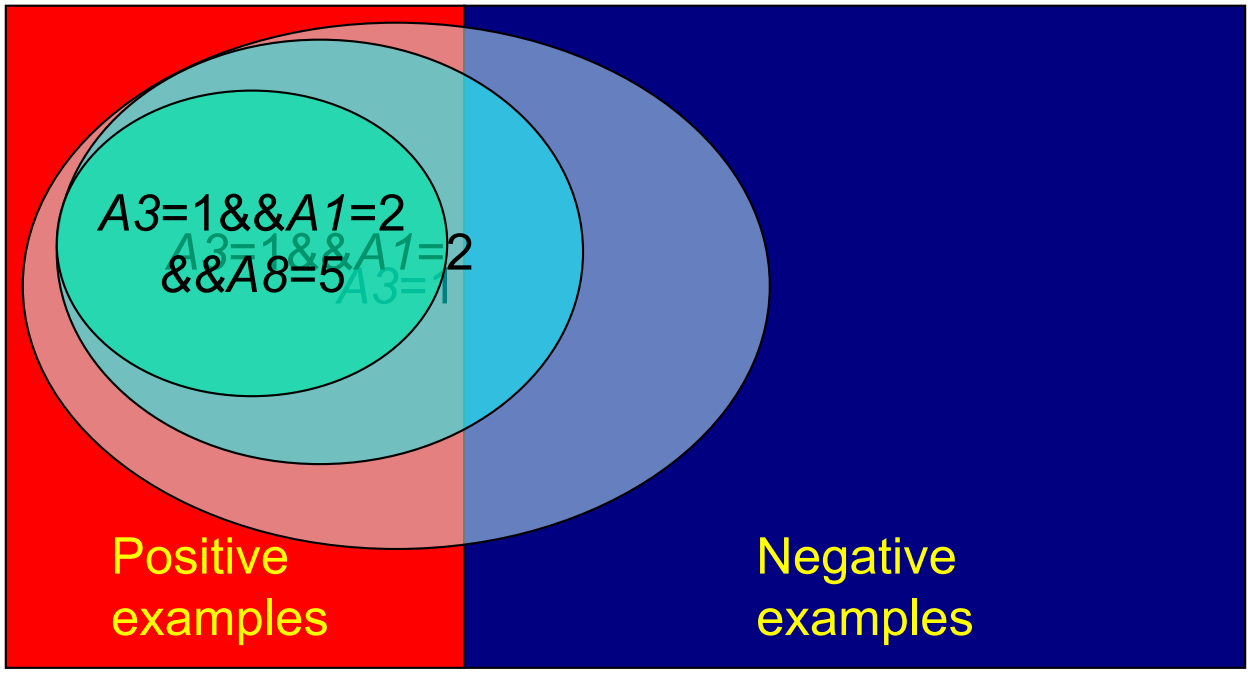
- To generate a rule

  while(true)

    find the *best* predicate *p*

    if *foil-gain(p) > threshold* **then** add *p* to current rule

    else break



How to Learn-One-Rule?

- Start with the *most general rule* possible: *condition = empty*
- *Add new attributes* by adopting a greedy depth-first strategy
  - Picks the attribute that most improves the rule quality
- Rule-Quality measures: *consider both coverage and accuracy*
  - Foil-gain (in FOIL & RIPPER): assesses *info_gain* by extending condition:

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

pos', neg' predicted

    - *favors rules that have high accuracy and cover many positive tuples*
- Rule pruning based on an independent set of test tuples

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection 
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix: **BOLD** red is correct classification

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy**: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity** : True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0

$$recall = \frac{TP}{TP + FN}$$

- Inverse relationship between precision & recall *(show curve)*
- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F_β*:** weighted measure of precision and recall
  - ß = 0.5 weighs precision twice the weight of recall

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity TP/P)*
cancer = no	**140**	**9560**	9700	98.56 (*specificity TN/N)*
Total	230	9770	10000	96.40 (*accuracy TP+TN/All*)

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained

- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class distribution in each fold is approximately the same as the initial data

Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

Note: Training set accuracy
Used in model accuracy

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

- Suppose we have 2 classifiers, M_1 and M_2, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the 2 error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: M_1 & M_2 are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate

Estimating Confidence Intervals: t-test

- If only 1 test set available: **pairwise comparison**
  - For i[th] round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - Average over 10 rounds to get $\overline{err}(M_1)$ and $\overline{err}(M_2)$
  - **t-test** computes **t-statistic** with *k-1* **degrees of freedom:**

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}$$ where

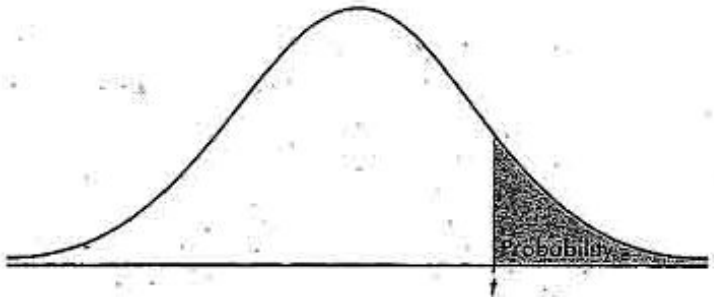
$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))\right]^2$$

- If two test sets available: use **non-paired (2-sample) t-test**

where $$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

where k_1 & k_2 are # of cross-validation samples used for M_1 & M_2, resp.

Estimating Confidence Intervals: Table for t-distribution



- Symmetric
- **Significance level**, e.g., *sig = 0.05* or *5%* means M_1 & M_2 are *significantly different* for 95% of population
- **Confidence limit**, *z = sig/2*

TABLE B: *t*-DISTRIBUTION CRITICAL VALUES

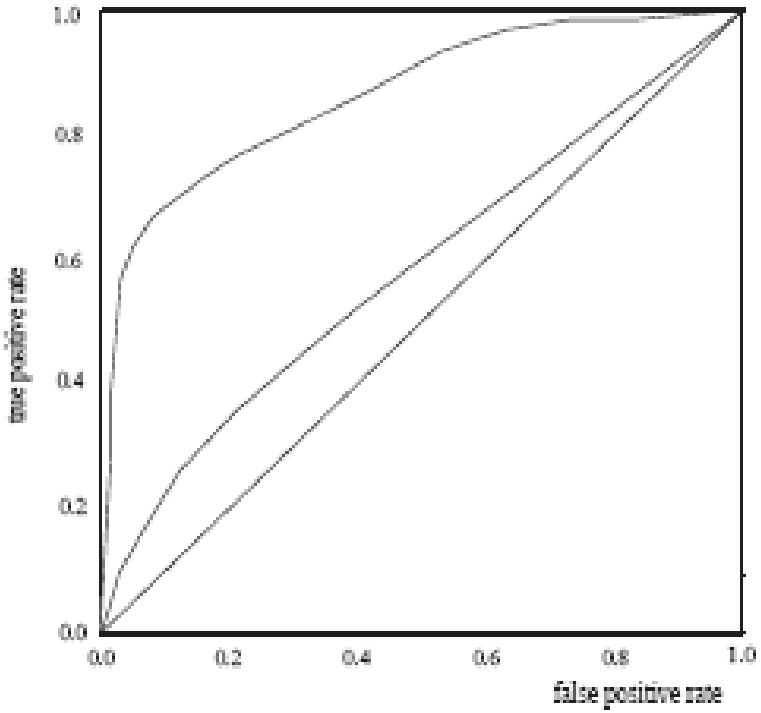
	Tail probability *p*											
df	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	.765	.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	.741	.941	1.190	1.533	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	.727	.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.893	6.869
6	.718	.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	.711	.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	.706	.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	.703	.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	.700	.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	.697	.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	.695	.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318
13	.694	.870	1.079	1.350	1.771	2.160	2.282	2.650	3.012	3.372	3.852	4.221
14	.692	.868	1.076	1.345	1.761	2.145	2.264	2.624	2.977	3.326	3.787	4.140
15	.691	.866	1.074	1.341	1.753	2.131	2.249	2.602	2.947	3.286	3.733	4.073
16	.690	.865	1.071	1.337	1.746	2.120	2.235	2.583	2.921	3.252	3.686	4.015
17	.689	.863	1.069	1.333	1.740	2.110	2.224	2.567	2.898	3.222	3.646	3.965
18	.688	.862	1.067	1.330	1.734	2.101	2.214	2.552	2.878	3.197	3.611	3.922
19	.688	.861	1.066	1.328	1.729	2.093	2.205	2.539	2.861	3.174	3.579	3.883
20	.687	.860	1.064	1.325	1.725	2.086	2.197	2.528	2.845	3.153	3.552	3.850
21	.686	.859	1.063	1.323	1.721	2.080	2.189	2.518	2.831	3.135	3.527	3.819
22	.686	.858	1.061	1.321	1.717	2.074	2.183	2.508	2.819	3.119	3.505	3.792
23	.685	.858	1.060	1.319	1.714	2.069	2.177	2.500	2.807	3.104	3.485	3.768
24	.685	.857	1.059	1.318	1.711	2.064	2.172	2.492	2.797	3.091	3.467	3.745
25	.684	.856	1.058	1.316	1.708	2.060	2.167	2.485	2.787	3.078	3.450	3.725
26	.684	.856	1.058	1.315	1.706	2.056	2.162	2.479	2.779	3.067	3.435	3.707
27	.684	.855	1.057	1.314	1.703	2.052	2.158	2.473	2.771	3.057	3.421	3.690
28	.683	.855	1.056	1.313	1.701	2.048	2.154	2.467	2.763	3.047	3.408	3.674
29	.683	.854	1.055	1.311	1.699	2.045	2.150	2.462	2.756	3.038	3.396	3.659
30	.683	.854	1.055	1.310	1.697	2.042	2.147	2.457	2.750	3.030	3.385	3.646
40	.681	.851	1.050	1.303	1.684	2.021	2.123	2.423	2.704	2.971	3.307	3.551
50	.679	.849	1.047	1.299	1.676	2.009	2.109	2.403	2.678	2.937	3.261	3.496
60	.679	.848	1.045	1.296	1.671	2.000	2.099	2.390	2.660	2.915	3.232	3.460
80	.678	.846	1.043	1.292	1.664	1.990	2.088	2.374	2.639	2.887	3.195	3.416
100	.677	.845	1.042	1.290	1.660	1.984	2.081	2.364	2.626	2.871	3.174	3.390
1000	.675	.842	1.037	1.282	1.646	1.962	2.056	2.330	2.581	2.813	3.098	3.300
∞	.674	.841	1.036	1.282	1.645	1.960	2.054	2.326	2.576	2.807	3.091	3.291
	50%	60%	70%	80%	90%	95%	96%	98%	99%	99.5%	99.8%	99.9%
	Confidence level *C*											

Estimating Confidence Intervals: Statistical Significance

- Are M_1 & M_2 **significantly different**?
  - Compute *t.* Select *significance level* (e.g. *sig = 5%)*
  - Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - t-distribution is symmetric: typically upper % points of distribution shown → look up value for **confidence limit** *z=sig/2* (here, 0.025)
  - **If t > z or t < -z**, then t value lies in rejection region:
    - **Reject null hypothesis** that mean error rates of M_1 & M_2 are same
    - Conclude: statistically significant difference between M_1 & M_2
  - **Otherwise**, conclude that any difference is **chance**

Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is *most likely* to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

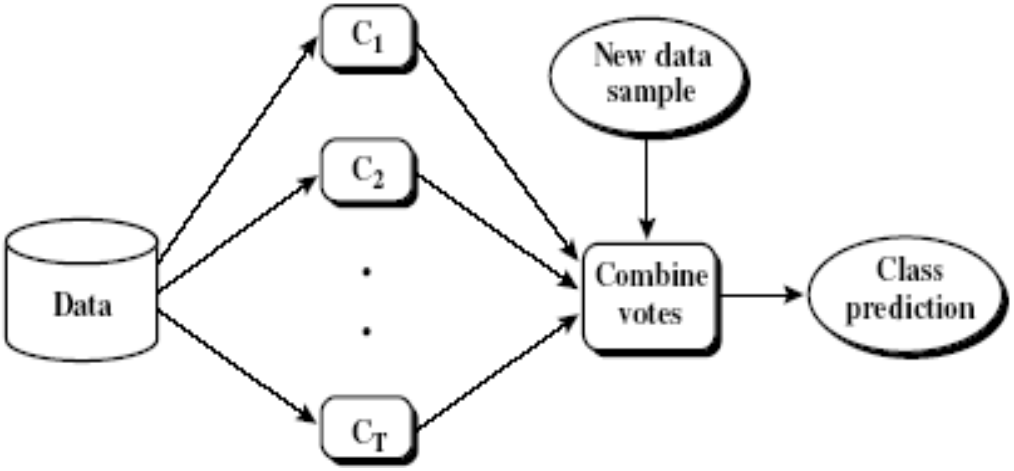
Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - *Use a combination of models to increase accuracy*
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noisy data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Compared with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Note: Weighted Majority

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a *random selection of attributes* at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART (Gini) or Info Gain methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold, **t**, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques**: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and F_β measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation.
- Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, **Discriminative Frequent Pattern Analysis for Effective Classification**, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, **Direct Discriminative Pattern Mining for Effective Classification**, ICDE'08
- W. Cohen. **Fast effective rule induction**. ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05

References (2)

- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96.
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report**. ECML'93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96.

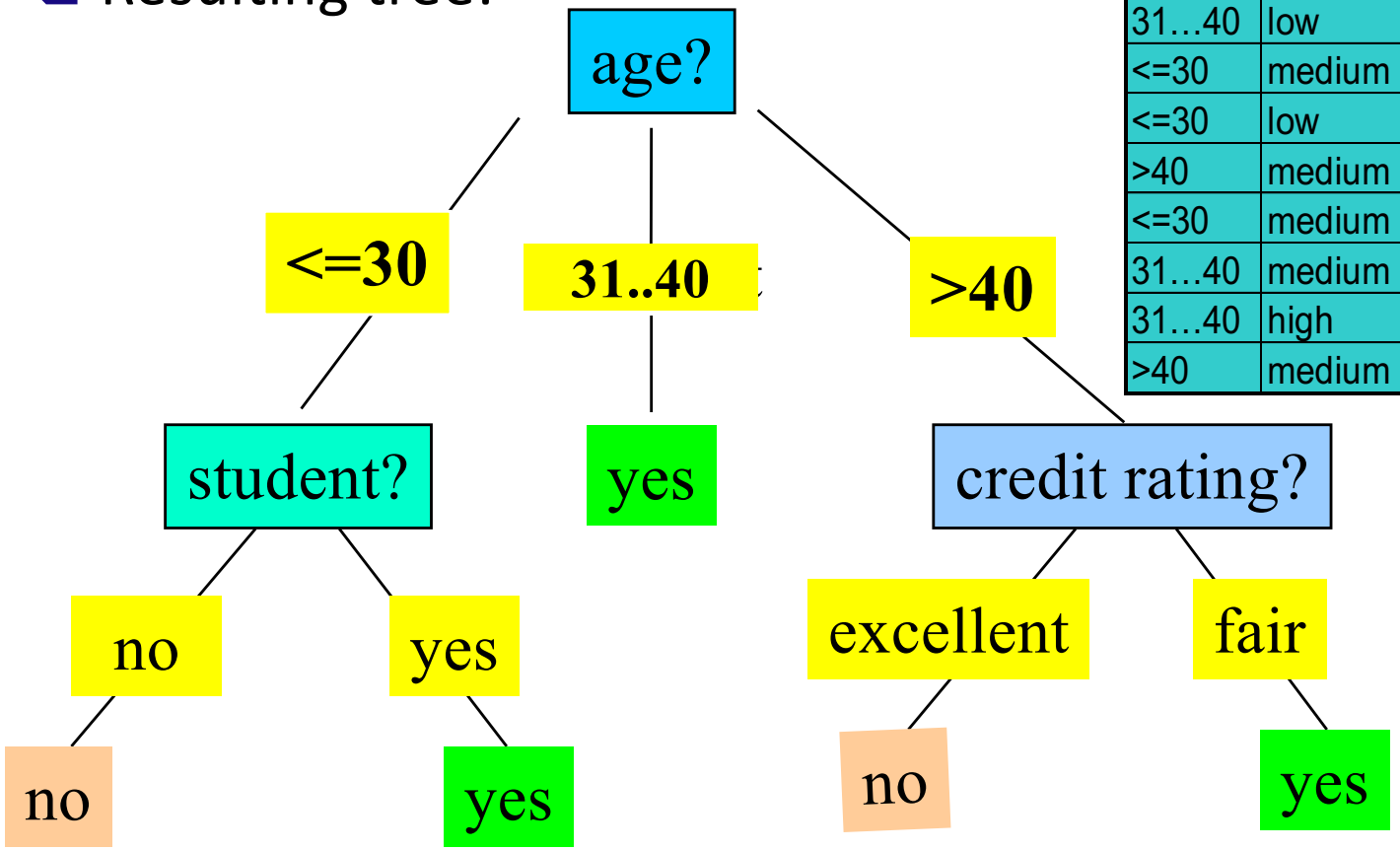
References (4)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules**. SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters**. KDD'03.

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

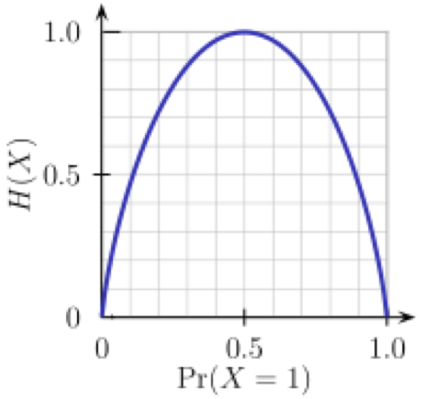


Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes's and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- If a data set *D* contains examples from *n* classes, gini index, *gini*(*D*) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  where p_j is the relative frequency of class *j* in *D*

- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

  $Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Enhancements to Basic Decision Tree Induction

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: **AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X*)
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n*)
  - Set of AVC-sets of all predictor attributes at the node *n*

Rainforest: Training Set and Its AVC Sets

Training Examples

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

AVC-set on *Age*

Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

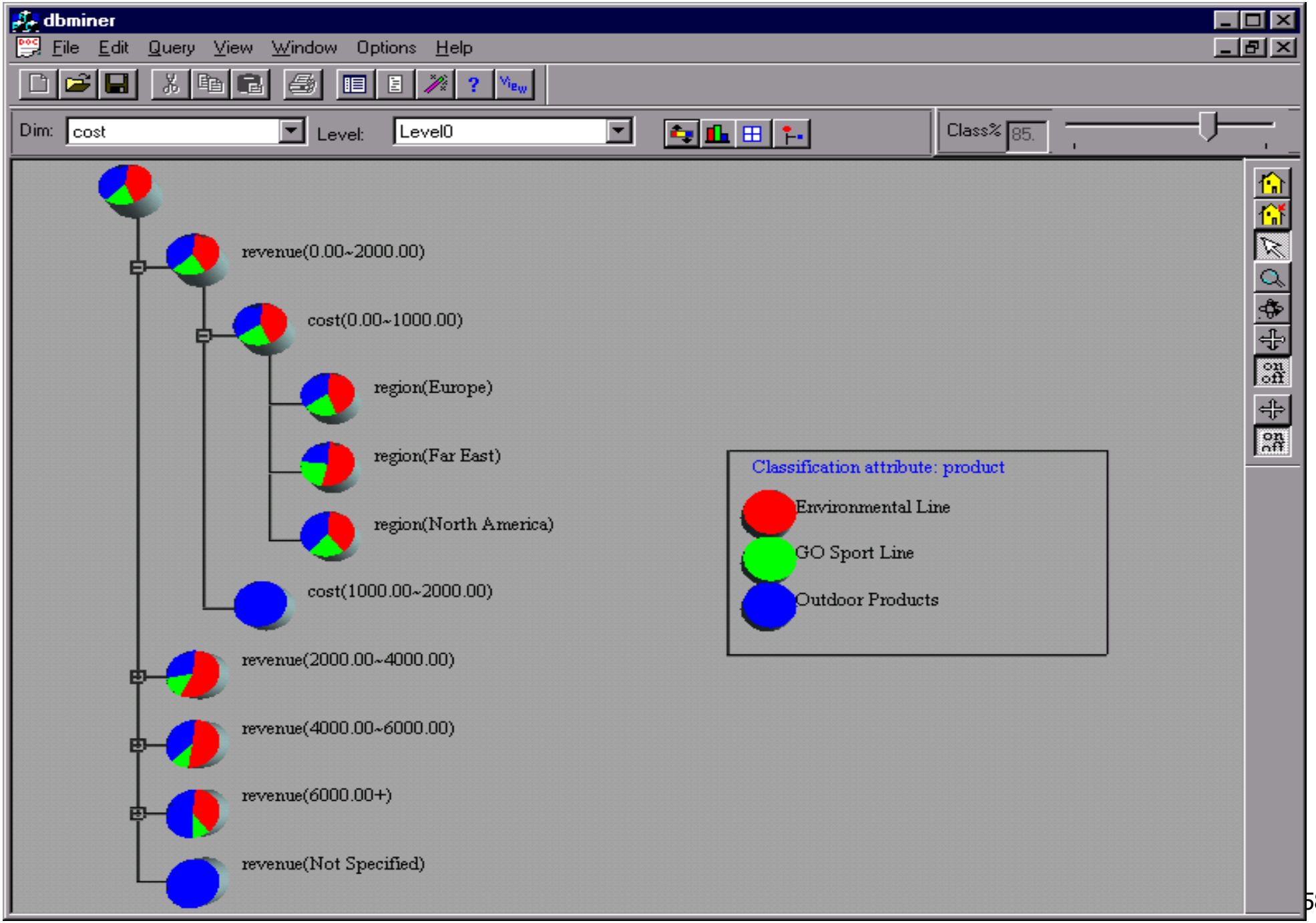
AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

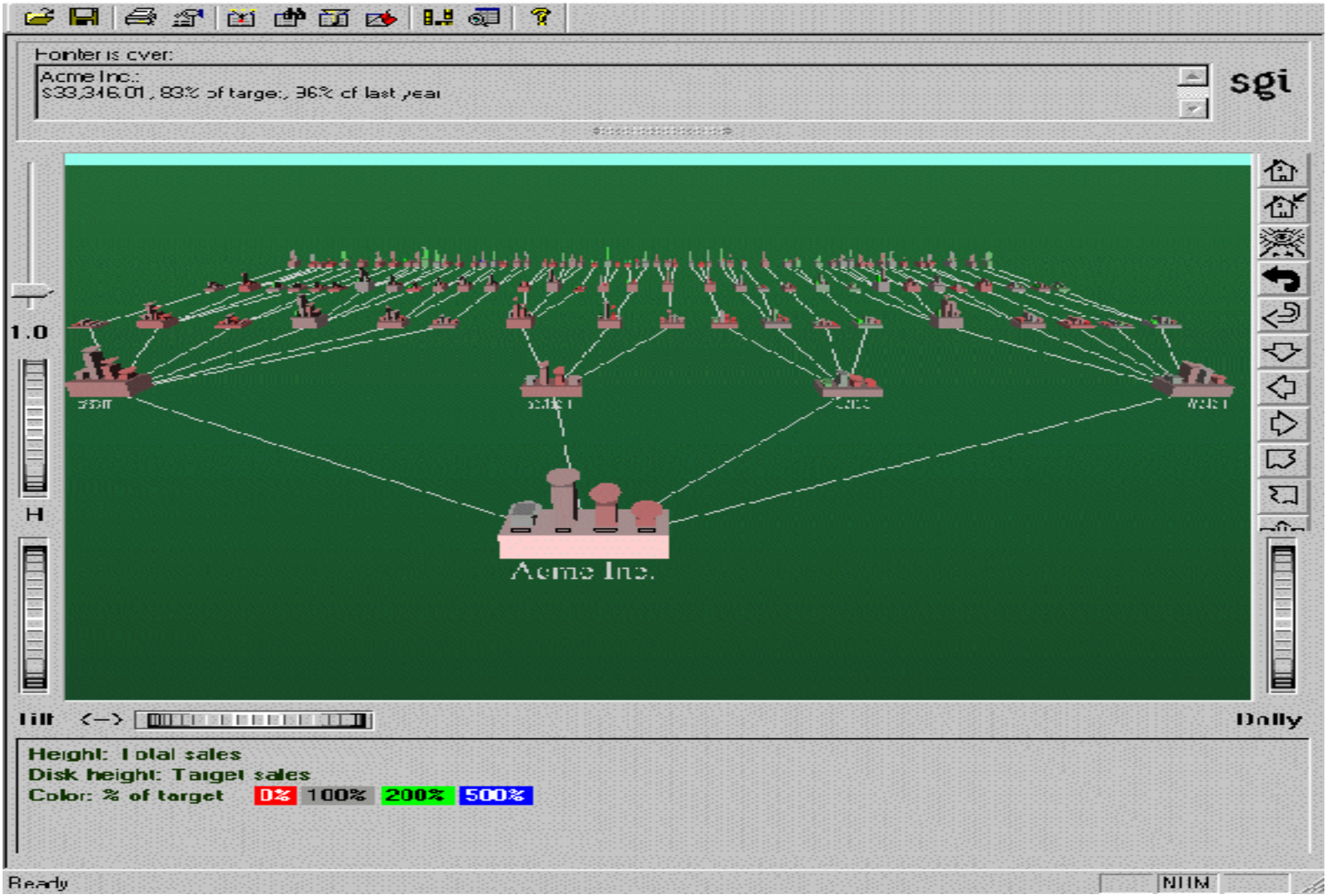
BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

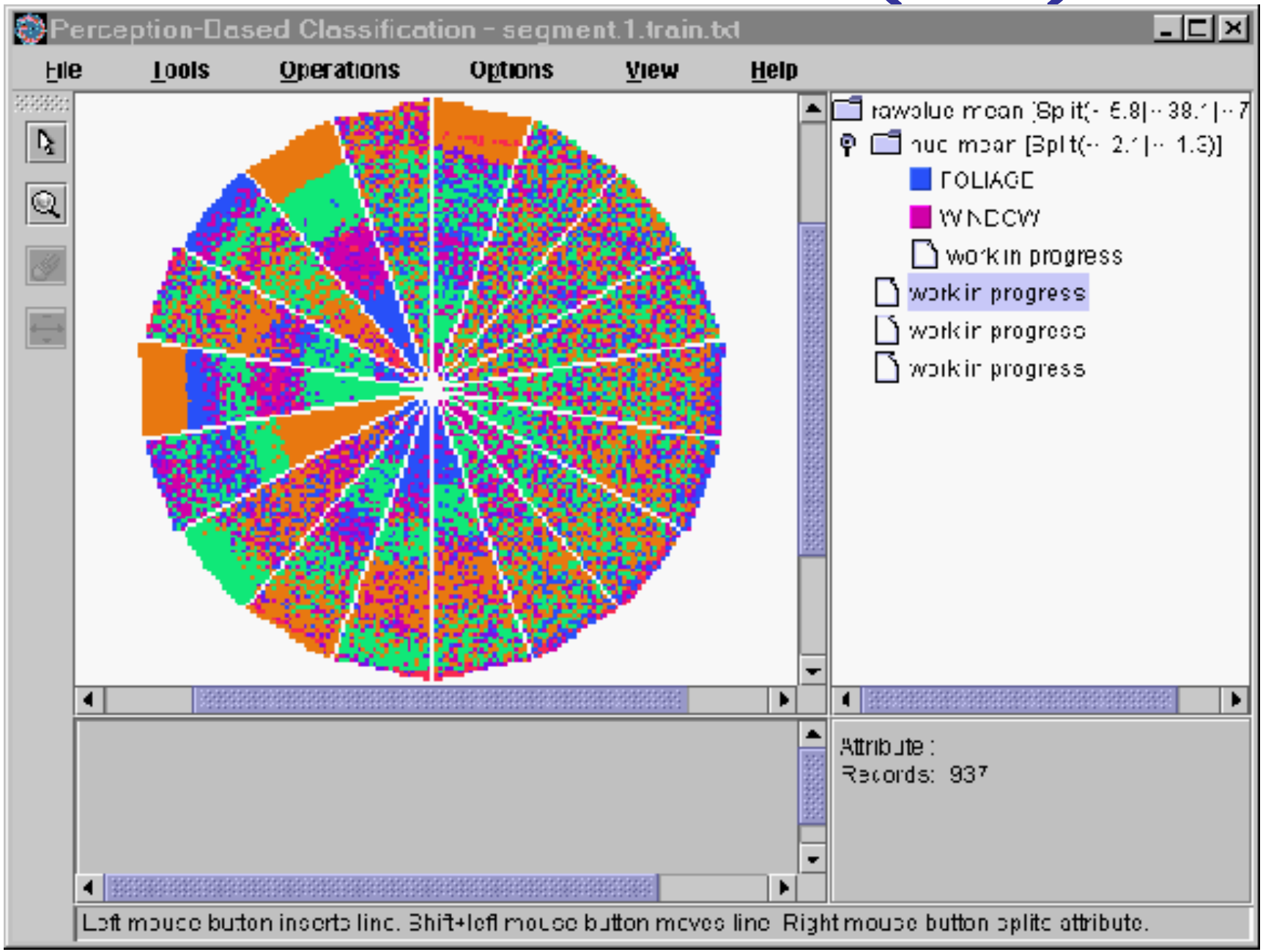
Presentation of Classification Results



Visualization of a Decision Tree in SGI/MineSet 3.0



Interactive Visual Mining by Perception-Based Classification (PBC)



Issues: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error betw. y_i and the predicted value y_i'
  - Absolute error: $| y_i - y_i' |$
  - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{d}$ Mean squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$
  - Relative absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{\sum_{i=1}^{d} | y_i - \bar{y} |}$ Relative squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{\sum_{i=1}^{d} (y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers

Popularly use (square) root mean-square error, similarly, root relative squared error

Scalable Decision Tree Induction Methods

- SLIQ (EDBT'96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - Constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)
- BOAT (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples

Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayes*, makes strong independence assumptions, but can perform well.
- Comparable performance with basic decision trees and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct (online learning) — prior knowledge can be combined with observed data
- Standard: Can provide a standard of optimal decision making against which other methods can be measured

Bayes' Theorem: Basics

- Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$
- Bayes' Theorem: $P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$
  - Let **X** be a data sample ("*evidence*"): class label is unknown
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: It requires initial knowledge of many probabilities, which may involve significant computational cost

Classification Is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

and $P(x_k|C_i)$ is

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = ‘yes’

C2:buys_computer = ‘no’

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)