Supervised Learning

- Decision trees
- Artificial neural nets
- K-nearest neighbor
- Support Vector Machines (SVMs)
- Linear regression
- Logistic regression
- ...

Supervised Learning

- y=F(x): true function (usually not known)
- D: training sample drawn from F(x)

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0      0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0     1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0    0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0  1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0   0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0  1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0    0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0       0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1   1
...
```

Supervised Learning

Train Set:

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0      0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0     1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0    0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0  1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0   0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0  1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0    0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0       0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1   1
...
```

Test Set:

```
71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0       ?
```

Supervised Learning

- F(x): true function (usually not known)
- D: training sample drawn from F(x)

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0      0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0     1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0    0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0  1
```

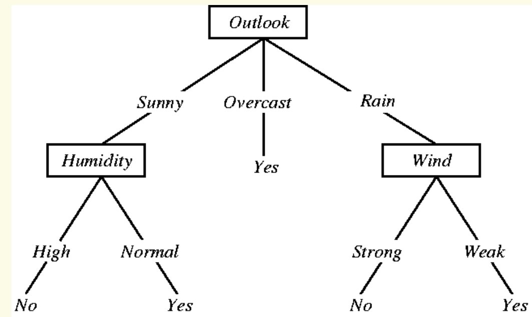
- G(x): model learned from training sample D

```
71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0       ?
```

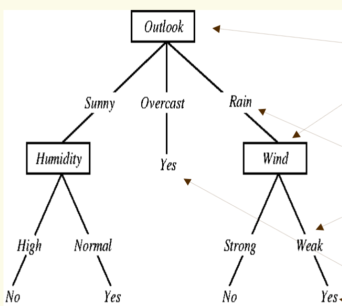
- Goal: $E<(F(x)-G(x))^2>$ is small (near zero) for future test samples drawn from F(x)

Decision Trees

A Simple Decision Tree



Representation

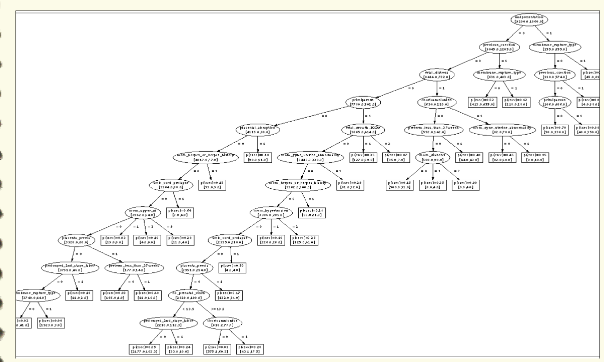


internal node = *attribute test*

branch = *attribute value*

leaf node = *classification*

A Real Decision Tree



A Real Decision Tree

Small Decision Tree Trained on 1000 Patients:

```
+833+167 (tree) 0.8327 0.1673 0
fetal_presentation = 1: +822+116 (tree) 0.8759 0.1241 0
| previous_csection = 0: +767+81 (tree) 0.904 0.096 0
| | primiparous = 0: +399+13 (tree) 0.9673 0.03269 0
| | primiparous = 1: +368+68 (tree) 0.8432 0.1568 0
| | | fetal_distress = 0: +334+47 (tree) 0.8757 0.1243 0
| | | | birth_weight < 3349: +201+10.555 (tree) 0.9482 0.05176 0
| | | | birth_weight >= 3349: +133+36.445 (tree) 0.783 0.217 0
| | | fetal_distress = 1: +34+21 (tree) 0.6161 0.3839 0
| previous_csection = 1: +55+35 (tree) 0.6099 0.3901 0
fetal_presentation = 2: +3+29 (tree) 0.1061 0.8939 1
fetal_presentation = 3: +8+22 (tree) 0.2742 0.7258 1
```

Real Data: C-Section Prediction

Do Decision Tree Demo Now!

collaboration with Magee Hospital, Siemens Research, Tom Mitchell

Real Data: C-Section Prediction

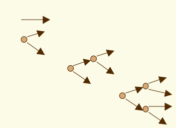
Demo summary:

- Fast
- Reasonably intelligible
- Different training sample => different tree
- Larger training sample => larger tree
- Larger training sample => more accurate predictions
- Accuracy on train set usually better than on test set

collaboration with Magee Hospital, Siemens Research, Tom Mitchell

Search Space

- all possible sequences of all possible tests
- very large search space, e.g., if N binary attributes:
  - 1 null tree
  - N trees with 1 (root) test
  - N*(N-1) trees with 2 tests
  - N*(N-1)*(N-1) trees with 3 tests
  - $\approx N^4$ trees with 4 tests
  - maximum depth is N

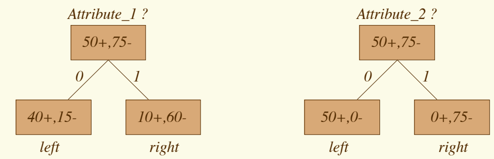


- size of search space is exponential in number of attributes
  - too big to search exhaustively
  - exhaustive search might overfit data (too many models)
  - so what do we do instead?

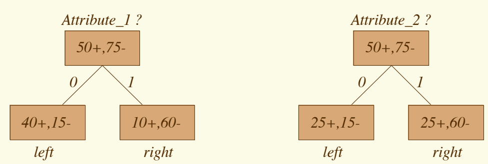
Top-Down Induction of Decision Trees

- TDIDT
- a.k.a. Recursive Partitioning
  - *find "best" attribute test to install at current node*
  - *split data on the installed node test*
  - *repeat until:*
    - all nodes are pure
    - all nodes contain fewer than k cases
    - no more attributes to test
    - tree reaches predetermined max depth
    - distributions at nodes indistinguishable from chance

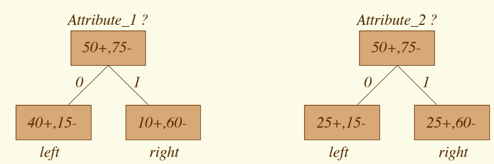
What is a Good Split?



What is a Good Split?



Find "*Best*" Split?



$$\left[\frac{\# Class_1}{\# Class_1 + \# Class_2}\right]_{leftnode} \bullet \left[\frac{\# Class_2}{\# Class_1 + \# Class_2}\right]_{rightnode}$$

0.6234 0.4412

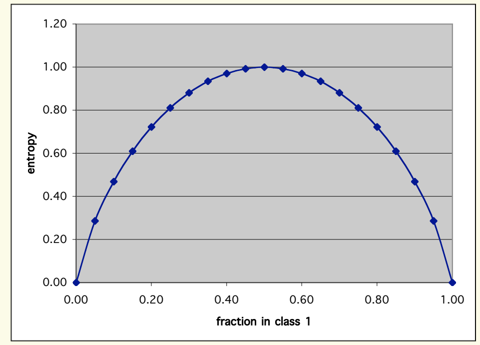
Splitting Rules

- Information Gain = reduction in entropy due to splitting on an attribute
- Entropy = how random the sample looks
- = expected number of bits needed to encode class of a randomly drawn + or – example using optimal information-theory coding

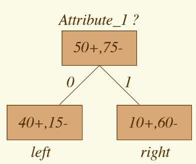
$$Entropy = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

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

Entropy



Information Gain



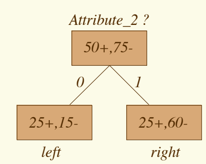
$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- = -\frac{50}{125}\log_2\frac{50}{125} - \frac{75}{125}\log_2\frac{75}{125} = 0.6730$$

$$A1: Entropy(left) = -\frac{40}{55}\log_2\frac{40}{55} - \frac{15}{55}\log_2\frac{15}{55} = 0.5859$$

$$A1: Entropy(right) = -\frac{10}{70}\log_2\frac{10}{70} - \frac{60}{70}\log_2\frac{60}{70} = 0.4101$$

$$Gain(S, A1) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) = 0.6730 - \frac{55}{125}0.5859 - \frac{70}{125}0.4101 = 0.1855$$

Information Gain



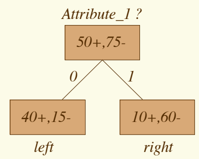
$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- = -\frac{50}{125}\log_2\frac{50}{125} - \frac{75}{125}\log_2\frac{75}{125} = 0.6730$$

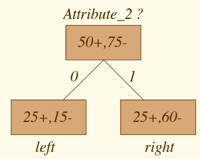
$$A2: Entropy(left) = -\frac{25}{40}\log_2\frac{25}{40} - \frac{15}{40}\log_2\frac{15}{40} = 0.6616$$

$$A2: Entropy(right) = -\frac{25}{85}\log_2\frac{25}{85} - \frac{60}{85}\log_2\frac{60}{85} = 0.6058$$

$$Gain(S, A2) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) = 0.6730 - \frac{40}{125}0.6616 - \frac{85}{125}0.6058 = 0.0493$$

Information Gain



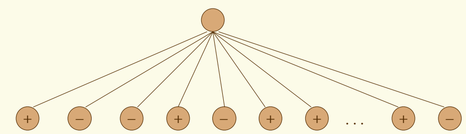


InfoGain = 0.1855

InfoGain = 0.0493

Splitting Rules

- Problem with Node Purity and Information Gain:
  - prefer attributes with many values
  - extreme cases:
    - Social Security Numbers
    - patient ID's
    - integer/nominal attributes with many values (JulianDay)

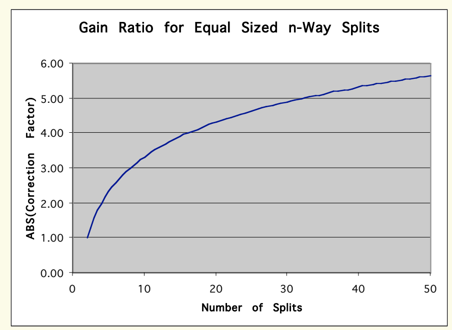


Splitting Rules

$$GainRatio(S,A) = \frac{InformationGain}{CorrectionFactor}$$

$$GainRatio(S,A) = \frac{Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}{\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}}$$

Gain_Ratio Correction Factor



Splitting Rules

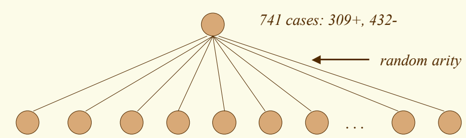
- GINI Index
  - node impurity weighted by node size

$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p_c]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

Experiment

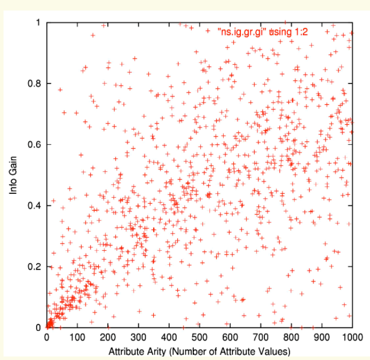
- Randomly select # of training cases: 2-1000
- Randomly select fraction of +'s and -'s: [0.0,1.0]
- Randomly select attribute arity: 2-1000
- Randomly assign cases to branches!!!!!
- Compute IG, GR, GINI



Info_Gain

Good Splits

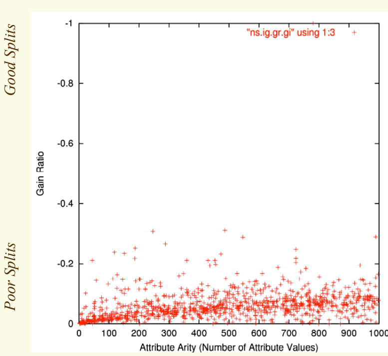
Poor Splits



Gain_Ratio

Good Splits

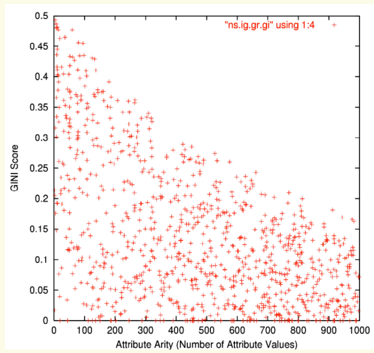
Poor Splits



GINI Score

Poor Splits

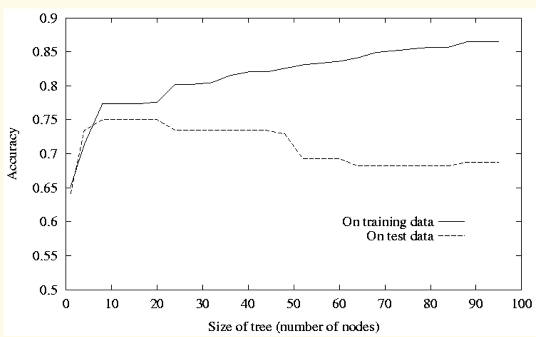
Good Splits



Attribute Types

- Boolean
- Nominal
- Ordinal
- Integer
- Continuous
  - Sort by value, then find best threshold for binary split
  - Cluster into n intervals and do n-way split

Overfitting



Machine Learning LAW #1

Because performance on data used for training often looks optimistically good, you should NEVER use test data for any part of learning.

Pre-Pruning (Early Stopping)

- Evaluate splits before installing them:
  - don't install splits that don't look worthwhile
  - when no worthwhile splits to install, done
- Seems right, but:
  - hard to properly evaluate split without seeing what splits would follow it (use lookahead?)
  - some attributes useful only in combination with other attributes (e.g., diagonal decision surface)
  - suppose no single split looks good at root node?

Post-Pruning

- Grow decision tree to full depth (no pre-pruning)
- Prune-back full tree by eliminating splits that do not appear to be warranted statistically
- Use train set, or an independent prune/test set, to evaluate splits
- Stop pruning when remaining splits all appear to be warranted
- Alternate approach: convert to rules, then prune rules

Converting Decision Trees to Rules

- each path from root to a leaf is a separate rule:

```
fetal_presentation = 1: +822+116 (tree) 0.8759 0.1241 0
|  previous_csection = 0: +767+81 (tree) 0.904 0.096 0
|  |  primiparous = 1: +368+68 (tree) 0.8432 0.1568 0
|  |  |  fetal_distress = 0: +334+47 (tree) 0.8757 0.1243 0
|  |  |  |  birth_weight < 3349: +201+10.555 (tree) 0.9482 0.05176 0
fetal_presentation = 2: +3+29 (tree) 0.1061 0.8939 1
fetal_presentation = 3: +8+22 (tree) 0.2742 0.7258 1
```

if (fp=1 & ¬pc & primip & ¬fd & bw<3349) => 0,
if (fp=2) => 1,
if (fp=3) => 1.

Missing Attribute Values

- Many real-world data sets have missing values
- Will do lecture on missing values later in course
- Decision trees handle missing values easily/well. Cases with missing attribute go down:
  - majority case with full weight
  - probabilistically chosen branch with full weight
  - all branches with partial weight

Greedy vs. Optimal

- Optimal
  - Maximum expected accuracy (test set)
  - Minimum size tree
  - Minimum depth tree
  - Fewest attributes tested
  - Easiest to understand
- XOR problem
- Test order not always important for accuracy
- Sometimes random splits perform well (acts like KNN)

Decision Tree Predictions

- Classification into discrete classes
- Simple probability for each class
- Smoothed probability
- Probability with threshold(s)

Performance Measures

- **Accuracy**
  - High accuracy doesn't mean good performance
  - Accuracy can be misleading
  - What threshold to use for accuracy?
- **Root-Mean-Squared-Error**

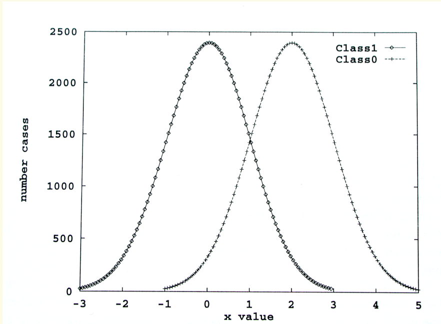
$$\text{RMSE} = \sqrt{\sum_{i=1}^{\#test} (1 - \text{Pred\_Prob}_i(\text{True\_Class}_i))^2 \Big/ \#\,test}$$

- Many other measures: ROC, Precision/Recall, …
- Will do lecture on performance measures later in course

Machine Learning LAW #2

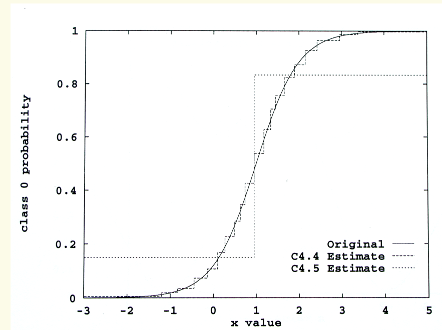
ALWAYS report baseline performance (and how you defined it if not obvious).

A Simple Two-Class Problem



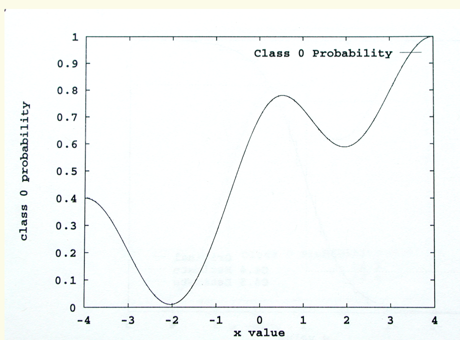
From Provost, Domingos pet-mlj 2002

Classification vs. Predicting Probs



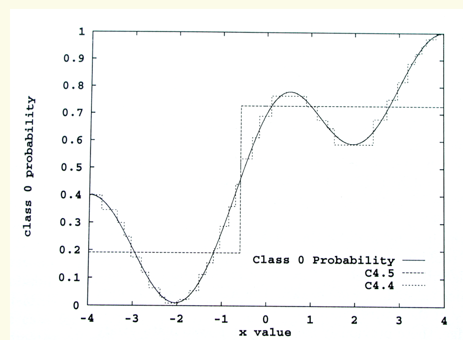
From Provost, Domingos pet-mlj 2002

A Harder Two-Class Problem



From Provost, Domingos pet-mlj 2002

Classification vs. Prob Prediction



From Provost, Domingos pet-mlj 2002

Predicting Probabilities with Trees

- Small Tree
  - few leaves
  - few discrete probabilities
- Large Tree
  - many leaves
  - few cases per leaf
  - few discrete probabilities
  - probability estimates based on small/noisy samples
- What to do?

PET: Probability Estimation Trees

- Smooth large trees
  - correct estimates from small samples at leaves
- Average many trees
  - average of many things each with a few discrete values is more continuous
  - averages improve quality of estimates
- Both

Laplacian Smoothing

- Small leaf count: 4+, 1–
- Maximum Likelihood Estimate: k/N
  - P(+) = 4/5 = 0.8; P(–) = 1/5 = 0.2?
- Could easily be 3+, 2- or even 2+, 3-, or worse
- Laplacian Correction: (k+1)/(N+C)
  - P(+) = (4+1)/(5+2) = 5/7 = 0.7143
  - P(–) = (1+1)/(5+2) = 2/7 = 0.2857
  - If N=0, P(+)=P(–) = 1/2
  - Bias towards P(class) = 1/C

Bagging (Model Averaging)

- Train many trees with different random samples
- Average prediction from each tree

Results

Table II. Summary of experimental results: AUC comparisons.

Systems	Wins-Ties-Losses	Avg. diff. (%)	Sign test	Wilcoxon test
C4.4 vs. C4.5	18 - 1 - 6	2.0	1.0	0.3
C4.4 vs. C4.5-L	13 - 3 - 9	0.2	30.0	30.0
C4.5-L vs. C4.5	21 - 2 - 2	1.7	0.1	0.1
C4.5-B vs. C4.5	24 - 1 - 0	7.3	0.1	0.1
C4.4-B vs. C4.4	23 - 2 - 0	5.3	0.1	0.1
C4.4-B vs. C4.5-B	11 - 5 - 9	−0.1	45.0	50.0

C4.4: no pruning or collapsing
"L": Laplacian Smoothing
"B": bagging

From Provost, Domingos pet-mlj 2002

Decision Tree Methods

- ID3:
  - info gain
  - full tree
  - no pruning
- CART (Classification and Regression Trees):
  - subsetting of discrete attributes (binary tree)
  - GINI criterion
  - "twoing" criterion for splitting continuous attributes ((Pleft*Pright)*SUM$_c$((P$_c$(left)-P$_c$(right))^2)
  - stop splitting when split achieves no gain, or <= 5 cases
  - cost-complexity pruning: minimize tree error + alpha*no-leaves
- C4:
  - subsetting of discrete attributes (binary tree)
  - gain ratio
  - pessimistic pruning

Decision Tree Methods

- MML:
  - splitting criterion?
  - large trees
  - Bayesian smoothing
- SMM:
  - MML tree after pruning
  - much smaller trees
  - Bayesian smoothing
- Bayes:
  - Bayes splitting criterion
  - full size tree
  - Bayesian smoothing

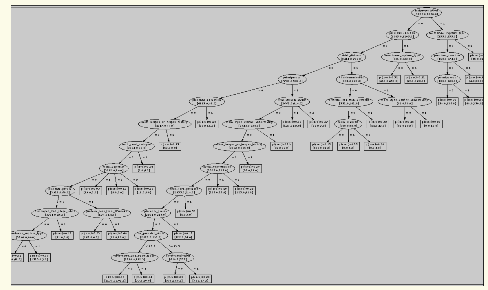
Popular Decision Tree Packages

- ID3 (ID4, ID5, …) [Quinlan]
  - research code with many variations introduced to test new ideas
- CART: Classification and Regression Trees [Breiman]
  - best known package to people outside machine learning
  - 1st chapter of CART book is a good introduction to basic issues
- C4.5 (C5.0) [Quinlan]
  - most popular package in machine learning community
  - both decision trees and rules
- IND (INDuce) [Buntine]
  - decision trees for Bayesians (good at generating probabilities)
  - available from NASA Ames for use in U.S.

Advantages of Decision Trees

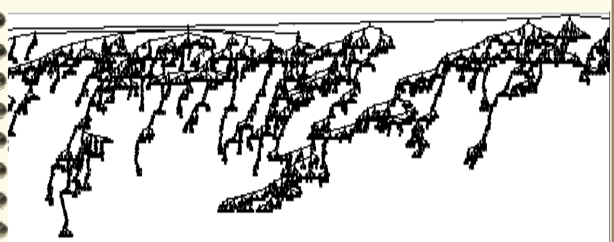
- TDIDT is relatively fast, even with large data sets (10^6) and many attributes (10^3)
  - advantage of recursive partitioning: only process all cases at root
- Can be converted to rules
- TDIDT does feature selection
- TDIDT often yields compact models (Occam's Razor)
- Decision tree representation is understandable
- Small-medium size trees usually intelligible

Decision Trees are Intelligible



Not *ALL* Decision Trees Are Intelligible

Part of Best Performing C-Section Decision Tree



Weaknesses of Decision Trees

- Large or complex trees can be just as unintelligible as other models
- Trees don't easily represent some basic concepts such as M-of-N, parity, non-axis-aligned classes…
- Don't handle real-valued parameters as well as Booleans
- If model depends on summing contribution of many different attributes, DTs probably won't do well
- DTs that look very different can be same/similar
- Usually poor for predicting continuous values (regression)
- Propositional (as opposed to 1st order)
- Recursive partitioning: run out of data fast as descend tree

When to Use Decision Trees

- Regression doesn't work
- Model intelligibility is important
- Problem does not depend on many features
  - Modest subset of features contains relevant info
  - not vision
- Speed of learning is important
- Missing values
- Linear combinations of features not critical
- Medium to large training sets

Current Research

- Increasing representational power to include M-of-N splits, non-axis-parallel splits, perceptron-like splits, …
- Handling real-valued attributes better
- Using DTs to explain other models such as neural nets
- Incorporating background knowledge
- TDIDT on really large datasets
  - >> 10^6 training cases
  - >> 10^3 attributes
- Better feature selection
- Unequal attribute costs
- Decision trees optimized for metrics other than accuracy

Regression Trees vs. Classification

- Split criterion: minimize SSE at child nodes
- Tree yields discrete set of predictions

$$\mathrm{SSE} = \sum_{i=1}^{\#test} (\mathrm{True}_i - \mathrm{Pred}_i)^2$$

Interpreting Results

Mean & Variance

$$Mean(x) = \bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$Variance(x) = S^2 = \frac{\sum_{i=1}^{N} (\bar{x} - x_i)^2}{N}$$

$$StdDev(x) = S = \sqrt{Var(x)}$$

Confidence Interval of Mean

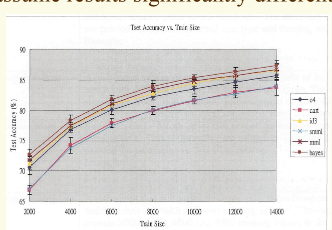
$$StdErr(\bar{x}) = StdDev(\bar{x}) = S / \sqrt{N}$$

$$\pm 1S \approx 68\%$$
$$\pm 2S \approx 95\%$$
$$\pm 3S \approx 99\%$$

$$Confidence\_Interval$$
$$95\% : \bar{X} - 1.96S < true\_mean < \bar{X} + 1.96S$$

Error Bars

- Typically 1 or 2 standard errors about mean
- Always specify what error bars are
- If 1 StdErr error bars do not overlap over regions of graph, typically assume results significantly different in regions



Hypothesis: Two Pops Have Same Mean

- t-test
- Given sample sizes, means, and variances, what are chances of seeing this large a difference in mean by chance?

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{pooled}\sqrt{(1/N_1) + (1/N_2)}}$$

$$S_{pooled} = \sqrt{\frac{(N_1 - 1)S_1^2 + (N_2 - 1)S_2^2}{N_1 + N_2 - 2}}$$

Hypothesis Testing continued (t-test)

- calculate t statistic (see previous slide)
- Find critical values of t in table for alpha = 0.05 (or 0.01, 0.001) with (N_1+N_2-2) degrees of freedom
- One-sided:
  - testing one mean is larger than other
  - E.g., for (alpha=0.05, N_1=N_2=10): t = 1.734
- Two-sided:
  - testing means are different
  - E.g., for (alpha=0.05, N_1=N_2=10): t = 2.101