Decision Tree

R Greiner

Cmput 466 / 551
Learning Decision Trees

- Def'n: Decision Trees
- Algorithm for Learning Decision Trees
  - Entropy, Inductive Bias (Occam's Razor)
- Overfitting
  - Def'n, MDL, $\chi^2$, PostPruning
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...
DecisionTree Hypothesis Space

- Internal nodes labeled with some feature $x_j$
- Arc (from $x_j$) labeled with results of test $x_j$
- Leaf nodes specify class $h(x)$

Instance:
- Outlook = Sunny
- Temperature = Hot
- Humidity = High
- Wind = Strong

classified as “No”
- (Temperature, Wind: irrelevant)

Easy to use in Classification
- Answer short series of questions…
Decision Trees

**Hypothesis space** is. . .
- Variable Size: Can represent any boolean function
- Deterministic
- Discrete and Continuous Parameters

**Learning algorithm** is. . .
- Constructive Search: Build tree by adding nodes
- Eager
- Batch (although ∃ online algorithms)
Continuous Features

- If feature is continuous:
  
  internal nodes may test value against threshold
Decision trees divide feature space into axis-parallel rectangles, labeling each rectangle with one class.
Using Decision Trees

- Instances represented by Attribute-Value pairs
  - “Bar = Yes”, “Size = Large”, “Type = French”, “Temp = 82.6”, ...
  - (Boolean, Discrete, Nominal, Continuous)

- Can handle:
  - Arbitrary DNF
  - Disjunctive descriptions

- Our focus:
  - Target function output is discrete
  - (DT also work for continuous outputs [regression])

- Easy to EXPLAIN

- Uses:
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Equipment or medical diagnosis
Learned Decision Tree
Meaning

- Concentration of \( \alpha \)-catenin in nucleus is very important:
  - If >0, probably relapse
  - If =0, then #lymph_nodes is important:
    - If >0, probably relapse
    - If =0, then concentration of pten is important:
      - If <2, probably relapse
      - If >2, probably NO relapse
      - If =2, then concentration of \( \beta \)-catenin in nucleus is important:
        - If =0, probably relapse
        - If >0, probably NO relapse
Can Represent Any Boolean Fn

- v, &, ⊕, MofN
  
  \[(A \lor B) \land (C \lor \neg D \lor E)\]

  ... but may require exponentially many nodes. . .

- **Variable-Size Hypothesis Space**
  - Can "grow" hypothesis space by increasing number of nodes
  - **depth 1** ("decision stump"): represent any boolean function of one feature
  
  **depth 2**: Any boolean function of two features; 
  + some boolean functions involving three features
  
  \[(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_3)\]

  ...
May require $>2$-ary splits

- Cannot represent using Binary Splits
Regression (Constant) Tree

- Represent each region as CONSTANT
Learning Decision Tree

**Classify:** $DecisionTree \times Instance \rightarrow ClassLabel$

Given Decision Tree

can classify (new) instance $\langle 1, 0.2, -1, \text{false}, \ldots, 1 \rangle$ as false

**Learn:** $Data \rightarrow DecisionTree$

\[
\begin{array}{c}
\begin{pmatrix}
  + & - & + & - & + & + & + \\
  - & + & - & + & + & + & + \\
  - & - & - & - & + & + & + \\
  - & - & + & + & + & + & + \\
\end{pmatrix}
\end{array}
\Rightarrow
\]

\[
\begin{array}{c}
\begin{pmatrix}
  + & + & + & + & + & + & + \\
  - & - & - & - & + & + & + \\
\end{pmatrix}
\end{array}
\]
Training Examples

- 4 discrete-valued attributes
- "Yes/No" classification

Want: Decision Tree

$$DT_{PT} (Out, Temp, Humid, Wind) \in \{ \text{Yes, No} \}$$
Learning Decision Trees – Easy?

- **Learn:** Data $\mapsto$ Decision Tree

  $\begin{array}{c|c|c|c|c} \text{A1} & \text{A2} & \text{A3} & \text{LABEL} \\ \hline + & + & + & 1 \\ + & - & + & 0 \\ + & - & - & 0 \end{array}$ $\mapsto$

- **Option 1:** Just store training data

- But …
Learn ?Any? Decision Tree

- Just produce “path” for each example
  - May produce large tree
  - Any generalization? (what of other instances?) \( \langle -, ++ \rangle \) ?
  - Noise in data
    \[
    \langle \langle +, -, - \rangle, 0 \rangle
    \]
    mis-recorded as
    \[
    \langle \langle +, +, - \rangle, 0 \rangle
    \]
    \[
    \langle \langle +, -, - \rangle, 1 \rangle
    \]
- Intuition:
  Want SMALL tree
  ... to capture “regularities” in data ...
  ... easier to understand, faster to execute, ...
First Split?

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>P. Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>d₂</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d₃</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>d₅</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d₆</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d₇</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d₈</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>d₉</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>d₁₀</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d₁₁</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d₁₂</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d₁₃</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d₁₄</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d₁₅</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
First Split: Outlook

Outlook
\[ [d_1, \ldots, d_{14}] \]

- Sunny
  - \( N_{\text{Sunny}} \) \( [d_1, d_2, d_8, d_9, d_{11}] \)
- Rain
  - \( N_{\text{Rain}} \) \( [d_4, d_5, d_6, d_{10}, d_{14}] \)
  - \( N_{\text{OC}} \) \( [d_3, d_7, d_{12}, d_{13}] \)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>P. Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>d_2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d_8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>d_9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_10</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>d_6</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>d_12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>d_13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>d_10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>d_14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Onto $N_{OC}$ ...

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>P. Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_3$</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$d_7$</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>$d_{12}$</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>$d_{13}$</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>
What about $N_{\text{Sunny}}$?
(Simplified) Algorithm for Learning Decision Tree

```
GrowTree(S)
    if (y = 0) for all ⟨x,y⟩ ∈ S return new leaf(0)
    if (y = 1) for all ⟨x,y⟩ ∈ S return new leaf(1)
    /* else */

    x_j = ChooseBestAttribute(S)
    S_0 = all ⟨x,y⟩ ∈ S with x_j = 0
    S_1 = all ⟨x,y⟩ ∈ S with x_j = 1
    return new node(x_j, GrowTree(S_0), GrowTree(S_1))
```

- Many fields independently discovered this learning alg...
- Issues
  - no more attributes
  - > 2 labels
  - continuous values
  - oblique splits
  - pruning
Alg for Learning Decision Trees

function DECISION-TREE-LEARNING(examples, attributes, default) returns a decision tree
inputs: examples, set of examples
        attributes, set of attributes
        default, default value for the goal predicate

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 MAJORITY-VALUE(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 ← DECISION-TREE-LEARNING(examples_i, attributes – best,
                                         MAJORITY-VALUE(examples))
    add a branch to tree with label $v_i$ and subtree subtree
  end
return tree
Search for Good Decision Tree

- Local search
  - expanding one leaf at-a-time
  - no backtracking
- Trivial to find tree that perfectly “fits” training data*
- but... this is NOT necessarily best tree
- Prefer small tree
  - NP-hard to find smallest tree that fits data

* noise-free data
Issues in Design of Decision Tree Learner

- What attribute to split on?

- Avoid Overfitting
  - When to stop?
  - Should tree be pruned?

- How to evaluate classifier (decision tree)?
Choosing Best Splitting Test

- How to choose best feature to split?

- After Gender split, still some uncertainty

- After Smoke split, no more uncertainty

  $\Rightarrow$ NO MORE QUESTIONS!

  (Here, Smoke is a great predictor for Cancer)

- Want a “measure” that prefers Smoke over Gender
If split on $x_i$, produce 2 children:

- $(x_i = t)$ follow TRUE branch
  $\Rightarrow$ data: [ $(x_i = t, Y = +)$, $(x_i = t, Y = -)$ ]

- $(x_i = f)$ follow FALSE branch
  $\Rightarrow$ data: [ $(x_i = f, Y = +)$, $(x_i = f, Y = -)$ ]
Desired Properties

- Score for split $M(S, x_i)$ related to

$$S \left( \frac{\#(x_i = t, Y = +)}{\#(x_i = t, Y = -)} \right) \quad S \left( \frac{\#(x_i = f, Y = +)}{\#(x_i = f, Y = -)} \right)$$

- Score $S(.)$ should be
  - Score is BEST for $[+0, -200]$
  - Score is WORST for $[+100, -100]$
  - Score is “symmetric"
    - Same for $[+19, -5]$ and $[+5, -19]$
  - Deals with any number of values

| $v_1$ | 7 |
| $v_2$ | 19 |
| ... | ...
| $v_k$ | 2 |
Play 20 Questions

- I'm thinking of integer $\in \{1, \ldots, 100\}$
- Questions
  - Is it 22?
  - More than 90?
  - More than 50?
- Why?
- $Q(r) = \#$ of additional questions wrt set of size $r$
  - $\geq 22$? $\frac{1}{100} \times Q(1) + \frac{99}{100} \times Q(99)$
  - $\geq 90$? $\frac{11}{100} \times Q(11) + \frac{89}{100} \times Q(89)$
  - $\geq 50$? $\frac{50}{100} \times Q(50) + \frac{50}{100} \times Q(50)$

Want this to be small...
Desired Measure: Entropy

- Entropy of $V = [p(V = 1), p(V = 0)]$:
  $$H(V) = - \sum_{v_i} P(V = v_i) \log_2 P(V = v_i)$$
  \(\equiv\) \# of bits needed to obtain full info
  ...average surprise of result of one “trial” of $V$

- Entropy \(\approx\) measure of uncertainty
Examples of Entropy

- Fair coin:
  - $H(\frac{1}{2}, \frac{1}{2}) = -\frac{1}{2} \log_2(\frac{1}{2}) - \frac{1}{2} \log_2(\frac{1}{2}) = 1$ bit
  - ie, need 1 bit to convey the outcome of coin flip

- Biased coin:
  - $H(\frac{1}{100}, \frac{99}{100}) = -\frac{1}{100} \log_2(\frac{1}{100}) - \frac{99}{100} \log_2(\frac{99}{100}) = 0.08$ bit

- As $P(\text{heads}) \mapsto 1$, info of actual outcome $\mapsto 0$
  - $H(0, 1) = H(1, 0) = 0$ bits
  - ie, no uncertainty left in source
  - $(0 \times \log_2(0) = 0)$
Entropy in a Nut-shell

...the values (locations of soup) sampled entirely from within the soup bowl

...the values (locations of soup) unpredictable... almost uniformly sampled throughout Andrew’s dining room
Prefer Low Entropy Leaves

- Use decision tree \( h(.) \) to classify (unlabeled) test example \( x \)
  ... Follow path down to leaf \( r \)
  ... What classification?
- Consider training examples that reached \( r \):
  - If all have same class \( c \)
    \[ \Rightarrow \text{label } x \text{ as } c \] (ie, entropy is 0)
  - If \( \frac{1}{2} \) are +; \( \frac{1}{2} \) are -
    \[ \Rightarrow \text{label } x \text{ as } ??? \] (ie, entropy is 1)
- On reaching leaf \( r \) with entropy \( H_r \), uncertainty w/label is \( H_r \)
  (ie, need \( H_r \) more bits to decide on class)
  \[ \Rightarrow \text{prefer leaf with LOW entropy} \]
Entropy of Set of Examples

- Don't have **exact** probabilities...
  ...but training data provides estimates of probabilities:

- Given training set with \( \{ p \) positive, \( n \) negative \} examples:

\[
H\left( \frac{p}{p+n}, \frac{n}{p+n} \right) = - \frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}
\]

- Eg: wrt 12 instances, \( S \):
  \( p = n = 6 \Rightarrow H( \frac{1}{2}, \frac{1}{2} ) = 1 \) bit
  ...so need 1 bit of info to classify example randomly picked from \( S \)
**Remaining Uncertainty**

\[ \text{Uncert}(S, A) = \text{remaining expected entropy after splitting on } A \]

\[ \equiv \sum_{v_i \in \text{Values}(A)} \frac{|S_{v_i}|}{|S|} \text{Entropy}(S_{v_i}) \]

\[ \equiv \sum_{i=1}^{v} \frac{p_i(A) + n_i(A)}{p + n} \text{H} \left( \frac{p_i(A)}{p(A) + n_i(A)}, \frac{n_i(A)}{p(A) + n_i(A)} \right) \]

\[ S: \ [9+, \ 5-] \]

[9+, 2-] \quad U = 0.683

[0+, 3-] \quad U = 0.0

\[ \text{Uncert}(S, \text{Rain}) = \frac{11}{14} \cdot 0.683 + \frac{3}{14} \cdot 0.0 = 0.536 \]

[6+, 2-] \quad U = 0.811

[3+, 3-] \quad U = 1.00

\[ \text{Uncert}(S, \text{Wind}) = \frac{8}{14} \cdot 0.811 + \frac{6}{14} \cdot 1.00 = 0.892 \]
... as tree is built ...

Which attribute should be tested here?

\[ S_{\text{Sunny}} = \{D1, D2, D8, D9, D11\} \]

\[ \text{Rem}(S_{\text{Sunny}}, \text{Humidity}) = (3/5) 0.0 - (2/5) 0.0 = .000 \]

\[ \text{Rem}(S_{\text{Sunny}}, \text{Temperature}) = (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .400 \]

\[ \text{Rem}(S_{\text{Sunny}}, \text{Wind}) = (2/5) 1.0 - (3/5) .918 = .951 \]
Assume \([p,n]\) reach node

Feature \(A\) splits into \(A_1, \ldots, A_v\)
- \(A_i\) has \(\{p_i^{(A)}\) positive, \(n_i^{(A)}\) negative\}

Entropy of each is …

\[
H \left( \frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}} \right)
\]

So for \(A_2\):
\[
H(\frac{28}{40}, \frac{12}{40})
\]

Uncert\((A)\) = \(\sum_{i=1}^{v} \frac{p_i^{(A)} + n_i^{(A)}}{p + n} \cdot H\left( \frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}} \right)\)
Minimize Remaining Uncertainty

- Greedy: Split on attribute that leaves **least entropy** wrt class … over training examples that reach there
- Assume A divides training set E into E₁, …, Eᵥ

- Eᵢ has { pᵢ⁽A⁾ positive, nᵢ⁽A⁾ negative } examples
- Entropy of each Eᵢ is
  \[ H\left(\frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}\right) \]
- Uncert(A) = expected information content
  \[ \Rightarrow \text{weighted contribution of each } E_i \]

\[
\text{Uncert}(A) = \sum_{i=1}^{v} \frac{p_i^{(A)} + n_i^{(A)}}{p + n} \cdot H\left(\frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}\right)
\]

- Often worded as **Information Gain**
  \[ \text{Gain}(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{Uncert}(A) \]
Notes on Decision Tree Learner

- Hypothesis space is complete!
  \[ \Rightarrow \text{contains target function...} \]

- No back tracking
  - Local minima...

- Statistically-based search choices
  - Robust to noisy data...

- Inductive bias: \( \approx \) “prefer shortest tree”
Inductive Bias in C4.5

- $H = \text{DecisionTreeClassifiers}$
  $\approx \text{power set of instances } X$
  $\Rightarrow$ Unbiased?

- Not really...
  - Preference for short trees,
    [trees w/ high info gain attributes near root]
  - Here: Bias is **preference** for some hypotheses,
    rather than restriction of hypothesis space $H$

- Occam's razor:
  Prefer shortest hypothesis that fits data
Occam's Razor

Q: Why prefer short hypotheses?

Argument in favor:
- Fewer short hyps. than long hyps.
  ⇒ a short hyp that fits data *unlikely* to be coincidence
  ⇒ a long hyp that fits data *might be* coincidence

Argument opposed:
- ∃ many ways to define small sets of hyps
  Eg, all trees with prime number of nodes
  whose attributes all begin with "Z"
- What's so special about small sets based on size of hypothesis??
Perceptron vs Decision Tree

Majority Function (11 inputs)  WillWait Predicate
Learning Decision Trees

- Defn: Decision Tree
- Algorithm for Learning Decision Trees
- Overfitting
  - Def’n
  - MDL, $\chi^2$
  - PostPruning
- Topics:
Example of Overfitting

- 25% have *butterfly-itis*
- \( \frac{1}{2} \) of patients have \( F_1 = 1 \)
  - Eg: “odd birthday”
- \( \frac{1}{2} \) of patients have \( F_2 = 1 \)
  - Eg: “even SSN”
- … for 10 features
- Decision Tree results
  - over 1000 patients (using these silly features) …
Decision Tree Results

- Standard decision tree learner:
  - Error Rate:
    - Train data: 0%
    - New data: 37%

- Optimal decision tree:
  - Error Rate:
    - Train data: 25%
    - New data: 25%
Overfitting

- Often “meaningless regularity” in data due to coincidences in the noise
  ⇒ bad generalization behavior

“Overfitting”

- Consider error in hypothesis $h$ over ...
  - training data $S$: $err_S(h)$
  - entire distribution $D$ of data: $err_{D,f}(h)$

- Hypothesis $h \in H$ overfits training data if
  $\exists$ alternative hypothesis $h' \in H$ s.t.
  $err_S(h) < err_S(h')$
  but
  $err_{D,f}(h) > err_{D,f}(h')$
Fit-to-Data ≠ Generalization

- \( h_k = \text{hyp after } k \text{ updates} \)
  - \( \text{err}_S(h_{20000}) < \text{err}_S(h_{10000}) \)
  - but
  - \( \text{err}_{D,f}(h_{20000}) > \text{err}_{D,f}(h_{10000}) \)

- “Overfitting"
  - Best “fit-to-data" will often find meaningless regularity in data
    (coincidences in the noise)
  - \( \Rightarrow \) bad generalization behavior
Example of Overfitting

- Spse 10 binary attributes (uniform),
  but class is random: \[ \begin{aligned} N \text{ w/prob } p &= 0.75 \\
Y \text{ w/prob } 1-p &= 0.25 \end{aligned} \]

- C4.5 builds nonsensical tree w/ 119 nodes!
  ⇒ Should be SINGLE NODE!
- Error rate (hold-out): 35%
  ⇒ Should be 25% (just say “No”)
- Why? Tree assigns leaf “N” w/prob 1-p, “Y” w/prob p
  - Tree sends instances to arbitrary leaves
  - Mistake if
    - Y-instance reaches N-leaf: \( p \times (1-p) \)
    - N-instance reaches Y-leaf: \( (1-p) \times p \)
  - Total prob of mistake = 2 \times p \times (1-p) = 0.375

- Overfitting happens for EVERY learner … not just DecTree!!
How to Avoid Overfitting
(Decision Trees)

- When to act
  - Use more stringent STOPPING criterion while growing tree
    - ... only allow statistically significant splits ...
  - Grow full tree, then post-prune

- To evaluate tree, measure performance over ...
  - training data
  - separate validation data set

- How to represent classifier?
  - as Decision Tree
  - as Rule Set
Avoid Overfitting #1

( StopEARYL, Training-Data, DecTree )

- Add more stringent STOPPING criterion while growing tree
  - At leaf \( n_r \) (w/ instances \( S_r \))
    - spse optimal proposed split is based on attribute \( A \)
  
  A. Use \( \chi^2 \) test, on data \( S_r \)
    - Apply statistical test to compare
      - \( T_0 \): leaf at \( r \) (majority label) vs
      - \( T_A \): split using \( A \)
    - Is error of \( T_A \) statistically better than \( T_0 \)?

  B. MDL: minimize

\[
\text{size(tree)} + \text{size( misclassifications(tree) )}
\]
Spse A is irrelevant
  - \([p_i, n_i] \propto [p, n]\)
  - So if \([p,n] = 3:2\), then \([p_i, n_i] = 3:2\)

Not always so clear-cut:
- Is this significant?
- Or this??

Test for Significance

\[
\begin{align*}
A_{1} & : p_1 = 10^+ & n_1 = 20^- \\
A_{2} & : p_2 = 30^+ & n_2 = 10^- \\
A_{3} & : p_3 = 20^+ & n_3 = 10^- \\
\end{align*}
\]
$\chi^2$ Test for Significance

- Null hypothesis $H_0$: Attribute $A$ is irrelevant in context of $r$.
  I.e., distr of class labels at node $n_r \equiv$ distr after splitting on $A$.
- Observe some difference between these distr's.
  What is prob (under $H_0$) of observing this difference, given $m = |S_r|$ iid samples?

- Defn: $\left\{ \begin{array}{c} p \nonumber \\ n \end{array} \right\}$ of $S_r$ are $\left\{ \begin{array}{c} \text{positive} \\ \text{negative} \end{array} \right\}$.

  After splitting on $A$, get $k$ subsets
  wrt $A = i: \rho_i$ positives, $n_i$ negatives.

- If $H_0$ ($A$ irrelevant), would have
  - $p_i^\sim = p \times (p_i + n_i)/p + n$ positives
  - $n_i^\sim = n \times (p_i + n_i)/p + n$ negatives.
\( \chi^2 \) Test – con't

- If \( H_0 \) (A irrelevant), would have
  \[
  \hat{p}_i = p \times \frac{p_i + n_i}{p + n} \quad \text{positives}
  \]
  \[
  \hat{n}_i = n \times \frac{p_i + n_i}{p + n} \quad \text{negatives}
  \]

- Measure of deviation:
  \[
  D = \sum_{i=1}^{k} \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}
  \]

  Under \( H_0 \), \( D \sim \chi^2_{(k-1)(2-1)} \)
  \( (k = \text{size of A's domain}; 2 = \# \text{ classes}) \)

  Q? Is \( D \) value within tolerances?
  **Don't add iff** \( D < T_{\alpha,d} \)

  \( T_{\alpha,d} \) is threshold:
  \( \alpha = \text{prob of making mistake} \)
  \( \text{(rejecting null hyp. when A is irrelevant)} \)
  \( d = \text{degree of freedom} \)

- Obvious extension when > 2 classes

Results: Empirically, not reliable...
### $\chi^2$ Table

For $\chi^2$ test, with

\[ \alpha = 0.05 \]

various “degrees of freedom”

<table>
<thead>
<tr>
<th>DOF</th>
<th>$T_{\alpha,DOF}$</th>
<th>DOF</th>
<th>$T_{\alpha,DOF}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.841</td>
<td>16</td>
<td>26.296</td>
</tr>
<tr>
<td>2</td>
<td>5.991</td>
<td>17</td>
<td>27.587</td>
</tr>
<tr>
<td>3</td>
<td>7.815</td>
<td>18</td>
<td>28.869</td>
</tr>
<tr>
<td>4</td>
<td>9.488</td>
<td>19</td>
<td>30.144</td>
</tr>
<tr>
<td>5</td>
<td>11.070</td>
<td>20</td>
<td>31.410</td>
</tr>
<tr>
<td>6</td>
<td>12.592</td>
<td>21</td>
<td>32.671</td>
</tr>
<tr>
<td>7</td>
<td>14.067</td>
<td>22</td>
<td>33.924</td>
</tr>
<tr>
<td>8</td>
<td>15.507</td>
<td>23</td>
<td>35.172</td>
</tr>
<tr>
<td>9</td>
<td>16.919</td>
<td>24</td>
<td>36.415</td>
</tr>
<tr>
<td>10</td>
<td>18.307</td>
<td>25</td>
<td>37.652</td>
</tr>
<tr>
<td>11</td>
<td>19.675</td>
<td>26</td>
<td>38.885</td>
</tr>
<tr>
<td>12</td>
<td>21.026</td>
<td>27</td>
<td>40.113</td>
</tr>
<tr>
<td>13</td>
<td>22.362</td>
<td>28</td>
<td>41.337</td>
</tr>
<tr>
<td>14</td>
<td>23.685</td>
<td>29</td>
<td>42.557</td>
</tr>
<tr>
<td>15</td>
<td>24.996</td>
<td>30</td>
<td>43.773</td>
</tr>
</tbody>
</table>
A wants to transmit to B classification function \( c() \) simplified to:

A and B agree on instances \( \{x_1, \ldots, x_M\} \)

What should A send, to allow B to determine \( M \) bits:

Option #1: A can send \( M \) "bits"

Option #2: A sends "perfect" decision tree \( d \), s.t. \( c(x_i) = d(x_i) \) for each \( x_i \)

Option #3: A sends "imperfect" decision tree \( d' \) + set of indices of \( K \) exceptions \( B = \{x_{i1}, \ldots, x_{iK}\} \)

- If (significant) reduction in \# exceptions

\[
c(x_i) = \begin{cases} 
-d(x_i) & \text{if } x_i \in B \\
 d(x_i) & \text{otherwise}
\end{cases}
\]
Avoid overfitting#2: PostPruning

- Grow tree, to “purity”, then PRUNE it back!

  Build “complete” decision tree $h$, from train
  For each penultimate node: $n_i$
  Let $h_i$ be tree formed by “collapsing” subtree under $n_i$, into single node
  If $h_i$ better than $h$
    Reset $h \leftarrow h_i$, . . .

- How to decide if $h_i$ better than $h$?
  1. Test on Hold-Out data?
     - 3 sets: training, $VALIDATION$, testing
       Problematic if small total # of samples
  2. Pessimistic Pruning
     . . . re-use training samples . . .
Using Validation Set

Train

Validate

Test

1,000

800

Learn

Compare

Eval?

Evaluate?
Avoid Overfitting #2.1
“Reduced-Error Pruning"

- Split data into *training* and *validation* set

Alg: 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 node that most improves accuracy on *validation* set

- Produces small version of accurate subtree
- What if data is limited?
Avoid Overfitting #2.2

“Pessimistic Pruning”

- Assume $N$ training samples reach leaf $r$; ... which makes $E$ mistakes so (resubstitution) error is $E/N$

- For confidence level (eg, 1-sided $\alpha = 0.25$), can estimate upper bound on # of errors:
  \[
  \text{Number Of Errors} = N \times \left[ \frac{E}{N} \pm \text{EB}_\alpha(N, E) \right]
  \]

- Let $U_\alpha(N, E) = \frac{E}{N} + \text{EB}_\alpha(N, E)$
  
  $\text{EB}_\alpha(E; N)$ based on binomial distribution
  
  $\sim$ Normal distribution: $z_\alpha \times \sqrt{\frac{p(1-p)}{N}}$
  
  $p \approx \frac{E}{N}$

- Laplacian correction... to avoid "divide by 0" problems:
  
  Use $p = \frac{(E+1)}{(N+2)}$ not $\frac{E}{N}$

- For $\alpha = 0.25$, use $z_{0.25} = 1.53$
  (recall 1-sided)
Pessimistic Pruning (example)

- Eg, spse A has 3 values: \{ v_1 \; v_2 \; v_3 \}
- If split on A, get
  - A= v_1  \rightarrow \text{return } Y \quad (6 \text{ cases, 0 errors})
  - A= v_2  \rightarrow \text{return } Y \quad (9 \text{ cases, 0 errors})
  - A= v_3  \rightarrow \text{return } N \quad (1 \text{ case, 0 errors})
- So 0 errors if split on A

W/prob \( \alpha \),

For \( A = \{ v_1 \; v_2 \; v_3 \} \), error rate is under

\[
\begin{pmatrix}
U_\alpha(6,0) \\
U_\alpha(9,0) \\
U_\alpha(1,0)
\end{pmatrix}
\]

- For \( \alpha = 0.25 \):
  #errors \leq 6 \times U_{0.25}(6,0) + 9 \times U_{0.25}(9,0) + 1 \times U_{0.25}(1,0)
  = 6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75 = 3.273
- If replace A-subtree w/ simple “Y”-leaf: (16 cases, 1 error)
  #errors \leq 16 \times U_{0.25}(16,1) = 16 \times 0.1827 = 2.923
- As 2.923 < 3.273, prune A-subtree to single “Y” leaf
  Then recur – going up to higher node
Pessimistic Pruning: Notes

Results: Pruned trees tend to be
- more accurate
- smaller
- easier to understand than original tree

Notes:
- Goal: to remove irrelevant attributes
- Seems inefficient to grow subtree, only to remove it
- This is VERY ad hoc, and WRONG statistically
  but works SO WELL in practice it seems essential
- Resubstitution error goes UP; but generalization error, down...
- Could replace $n_i$ with single node,
  or with most-frequently used branch
Avoid Overfitting #3
Using Rule Post-Pruning

1. Grow decision tree.
   Fit data as well as possible.
   Allow overfitting.

2. Convert tree to equivalent set of rules:
   - One rule for each path from root to leaf.

3. Prune each rule independently of others.
   - ie, delete preconditions that improve its accuracy

4. Sort final rules into desired sequence for use depending on accuracy.

5. Use ordered sequence for classification.
Converting Trees to Rules

- Every decision tree corresponds to a set of rules:
  - IF (Patrons = None)
    THEN WillWait = No
  - IF (Patrons = Full) & (Hungry = No) & (Type = French)
    THEN WillWait = Yes
  - ...

- Why? (Small) RuleSet MORE expressive
  small DecTree ≈ small RuleSet
  (DecTree is subclass of ORTHOGONAL DNF)
Learning Decision Trees

- Def’n: Decision Trees
- Algorithm for Learning Decision Trees
- Overfitting

Topics:
- k-ary attribute values
- Real attribute values
- Other splitting criteria
- Attribute Cost
- Missing Values
- …
Attributes with Many Values

- Problem: *Gain* prefers attribute with many values.
  - Entropy \( \approx \ln(k) \ldots \)
  - Eg, imagine using
    - Date = Jun 3 1996
    - Name = Russ

- One approach: use *GainRatio* instead
  \[
  \text{GainRatio}(S,A) = \frac{\text{Gain}(S,A)}{\text{SplitInformation}(S,A)}
  \]
  \[
  \text{SplitInformation}(S,A) \equiv -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
  \]
  where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \)

- Issues:
  - Construct a multiway split?
  - Test one value versus all of the others?
  - Group values into two disjoint subsets?
Continuous Valued Attributes

Create a discrete attribute to test continuous

- \( \text{Temperature} = 82.5 \)
- \( (\text{Temperature} > 72.3) \in \{ \text{t, f} \} \)

Note: need only consider splits between "class boundaries"

Eg, between 48 / 60; 80 / 90
Finding Split for Real-Valued Features

- Best threshold \( \theta_j \) for attribute \( j \)
  - Sort training instance by \( x_{i,j} \)
  - For each \( \theta \in \mathbb{R} \)
    
    \[
    \begin{array}{c|cc}
    \text{labeled} & < \theta & > \theta \\
    \hline
    A & n_{A,\ell}(\theta) & n_{A,r}(\theta) \\
    B & n_{B,\ell}(\theta) & n_{B,r}(\theta) \\
    \text{(either)} & n_{\ell}(\theta) & n_{r}(\theta)
    \end{array}
    \]

    \[
    y_i \quad A \quad A \quad B \quad A \quad B \quad B \quad B \quad B
    \]

    \[
    x_{i,j} \quad 0.2 \quad 0.4 \quad 0.7 \quad 1.1 \quad 1.3 \quad 1.7 \quad 1.9 \quad 2.4 \quad 2.9
    \]

    \[
    n_{A,\ell} = 3 \quad n_{A,r} = 1 \quad n_{B,\ell} = 1 \quad n_{B,r} = 4
    \]

    Mutual information = 0.2294

  - Sequentially set \( \theta \) to value of \( \frac{x_{i,j} + x_{i,j+1}}{2} \)
    
    \[
    \text{Compute mutual information}
    \frac{n_\ell(\theta)}{n_\ell(\theta) + n_\ell(\theta)} \left[ \frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta) + n_{B,\ell}(\theta)} \ln \frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta) + n_{B,\ell}(\theta)} + \frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta) + n_{B,\ell}(\theta)} \ln \frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta) + n_{B,\ell}(\theta)} \right] + \\
    \frac{n_r(\theta)}{n_\ell(\theta) + n_\ell(\theta)} \left[ \frac{n_{A,r}(\theta)}{n_{A,r}(\theta) + n_{B,r}(\theta)} \ln \frac{n_{A,r}(\theta)}{n_{A,r}(\theta) + n_{B,r}(\theta)} + \frac{n_{B,r}(\theta)}{n_{A,r}(\theta) + n_{B,r}(\theta)} \ln \frac{n_{B,r}(\theta)}{n_{A,r}(\theta) + n_{B,r}(\theta)} \right]
    \]

  - Just need to consider \( j \) s.t. \( y_j \neq y_{j+1} \)
Desired Properties

- Score for split $M(D, x_i)$ related to
  
  $$S \left( \frac{\#(x_i = t, Y = +)}{\#(x_i = t, Y = -)} \right)$$

- Score $S(.)$ should be
  
  - Score is BEST for $[+0, -200]$
  - Score is WORST for $[+100, -100]$
  - Score is "symmetric"
    
    Same for $[+19, -5]$ and $[+5, -19]$
  - Deals with any number of values

| $v_1$ | 7 |
| $v_2$ | 19 |
| ...  |   |
| $v_k$ | 2 |
Other Splitting Criteria

- Why use Gain as splitting criterion?
  Want: Large "use me" value if split is
  \[
  \langle 85, 0, 0, \ldots, 0 \rangle
  \]
  Small "avoid me" value if split is
  \[
  \langle 5, 5, 5, \ldots, 5 \rangle
  \]
  True of Gain, GainRatio… also for.
- Statistical tests: \( \chi^2 \)
  For each attr A, compute deviation:
  \[
  D(A) = \sum_{i=1}^{k} \frac{(p_i^{(A)} - \hat{p}_i^{(A)})^2}{\hat{p}_i^{(A)}} + \frac{(n_i^{(A)} - \hat{n}_i^{(A)})^2}{\hat{n}_i^{(A)}}
  \]
  
- GINI index: \( \text{GINI}(A) = \sum_i \sum_{j \neq i} p_i p_j = 1 - \sum_i p_i^2 \)
  
- Others: “Marshall Correction”
  “G” statistic
  Probabilities (rather than statistic)
Node Impurity Measures

- **Node impurity measures for 2-class classification**
  - function of the proportion $p$ in class 2.
  - Scaled cross-entropy has been scaled to pass through $(0.5, 0.5)$.  

HTF 2009
Example of $\chi^2$

- Attributes $T_1$, $T_2$ class $c$

<table>
<thead>
<tr>
<th>Data:</th>
<th>$T_1 = c =</th>
<th>$T_2 = c =</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>$Y$</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>$N$</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

$\chi^2$: For $T_1$: 
\[
\frac{(0-1.5)^2}{1.5} + \frac{(3-1.5)^2}{1.5} + \frac{(2-3.5)^2}{3.5} + \frac{(5-3.5)^2}{3.5} = 4.29
\]

For $T_2$: 
\[
\frac{(3.0-2.5)^2}{2.5} + \frac{(2-2.5)^2}{2.5} + \frac{(1-1.5)^2}{1.5} + \frac{(2-1.5)^2}{1.5} + \frac{(1-1)^2}{1} + \frac{(1-1)^2}{1} = 0.533
\]

- As $4.29$ ($T_1$) $> 0.533$ ($T_2$), ... use $T_1$
  (less likely to be irrelevant)
Example of GINI

- Attributes $T_1, T_2$ class $c$

Data:

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$c$ =</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Yes</td>
</tr>
<tr>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
</tr>
</tbody>
</table>

Scores: $GINI(T_1) = 1 - \left( \frac{5}{10} \right)^2 - \left( \frac{5}{10} \right)^2 - \left( \frac{3}{10} \right)^2 - \left( \frac{3}{10} \right)^2 + \frac{7}{10} \left( 1 - \left( \frac{5}{7} \right)^2 - \left( \frac{2}{7} \right)^2 \right) = \frac{1}{10} \left( \frac{3^2}{3} + \frac{3^2}{3} + \frac{5^2}{7} + \frac{2^2}{7} \right) - \left( \frac{5^2}{10} + \frac{5^2}{10} \right) = 0.21429$

$GINI(T_2) = \frac{1}{10} \left( \frac{3^2}{5} + \frac{2^2}{5} + \frac{1^2}{3} + \frac{2^2}{3} + \frac{1^2}{2} + \frac{1^2}{2} \right) - \left( \frac{5^2}{10} + \frac{5^2}{10} \right) = 0.026667$

- As $GINI(T_1) > GINI(T_2)$, split on $T_1$
- As $\left( \frac{5^2}{10} + \frac{5^2}{10} \right)$ is subtracted from both, can just use first term... and ignore $\frac{1}{10}$ multiplier
Cost-Sensitive Classification … Learning

- So far, only considered **ACCURACY**
  
  In gen'l, may want to consider **COST** as well
  - medical diagnosis: **BloodTest** costs $150
  - robotics: **Width_from_1ft** costs 23 sec

- Learn a consistent tree with low expected cost?
  . . . perhaps replace $\text{InfoGain}(S,A)$ by

  $\frac{\text{Gain}^2(S,A)}{\text{Cost}(A)} \quad [\text{Tan/Schlimmer'90}]$

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

  where $w \in [0, 1]$ determines importance of cost [Nunez'88]

- **General utility (arb rep'n)**

  $E[ \sum_i \text{cost}(A_i) + \text{Misclass penalty} ] \quad [\text{Greiner/Grove/Roth'96}]$
Dealing with Missing Information

Training Instances

\[
\begin{array}{ccccccc}
1 & * & 0 & \ldots & 1 & T \\
* & \nu & * & \ldots & 0 & T \\
0 & 1 & 1 & \ldots & * & F \\
\vdots & \ddots & \vdots & & \vdots & \vdots \\
1 & * & * & \ldots & 0 & T
\end{array}
\]

\[\Downarrow\]

Learning Algorithm

\[
\begin{array}{ccccccc}
0 & * & * & \ldots & 1 & \Rightarrow \quad \text{Classifier} & \Rightarrow & T
\end{array}
\]
Formal Model

- Default Concept returns Categorical Label \{T, F\} even when given partial instance
  - ... even \langle *,*, \ldots, * \rangle!
- Blocker \( \beta : \{0, 1\}^n \rightarrow \text{Stoch} \{0, 1, *\} \)
- N.b., \( \beta \)
  - does NOT map 0 to 1
  - does NOT change class label
  - may reveal different attributes on different instances (on same instance, different times)
Unknown Attribute Values

Q: What if some examples are **incomplete**
   . . . missing values of some attributes?

When learning:

A1: Throw out all incomplete examples?
   ... May throw out too many. . .

A2: Fill in most common value ("imputation")
   - May miss correlations with other values
   - If impute wrt attributes: may require high order statistics

A3: Follow all paths, w/ appropriate weights
   - Huge computational cost if missing MANY values

When classifying

- Similar ideas . . .
- ISSUE: Why are values missing?
  - Transmission Noise
  - "Bald men wear hats"
  - "You don't care"
See [Schuurmans/Greiner'94]
Handling Missing Values: Proportional Distribution

- Associate weight $w_i$ with example $\langle x_i, y_i \rangle$
  At root, each example has weight 1.0
- Modify mutual information computations:
  use weights instead of counts
- When considering test on attribute $j$,
  only consider examples that include $x_{ij}$
- When splitting examples on attribute $j$:
  - $p_L = \text{prob. non-missing example sent left}$
  - $p_R = \text{prob. non-missing example sent right}$
  - For each example $\langle x_i, y_i \rangle$ missing attribute $j$:
    send it to both children;
    - to left w/ $w_i := w_i \times p_L$
    - to right w/ $w_i := w_i \times p_R$
- To classify example missing attribute $j$:
  - Send it down left subtree; $P( y_{L}^{-} | x ) = \text{resulting prediction}$
  - Send it down left subtree; $P( y_{R}^{-} | x ) = \text{resulting prediction}$
  - Return $p_L \times P( y_{L}^{-} | x ) + p_R \times P( y_{R}^{-} | x )$
Handling Missing Values:
Surrogate Splits

- Choose attribute $j$ and splitting threshold $\theta_j$
  using all examples that include $j$

\[
u_i = \begin{cases} 
L & \text{if } \langle x_i, y_i \rangle \text{ sent to LEFT subtree} \\
R & \text{if } \langle x_i, y_i \rangle \text{ sent to RIGHT subtree}
\end{cases}
\]

- For each other attribute $q$, find splitting threshold $\theta_q$
  that best predicts $u_i$
  Sort $q$ by predictive power
  Called "surrogate splits"

- Sort via "surrogate splits"
  To handle $\langle x_i, y_i \rangle$ where $x_{ij} = *$
    - go thru surrogate splits $q$ until finding one NOT missing
    - Use $q$, $\theta_q$ to decide which child gets $x_i$
Questions

1. How to represent default concept?
2. When is best default concept learnable?
3. If so, how many samples are required?
4. Is it better to learn from …
   - Complete Samples, or
   - Incomplete Samples?
Learning Task

Input:

P#1: \( \langle + - - + + \rangle + \)
P#2: \( \langle - - + + \rangle - \)
P#3: \( \langle - - - \rangle - \)
P#4: \( \langle - - + - + \rangle + \)
P#5: \( \langle - + + \rangle - \)

Output: (small) decision tree consistent with data
Learning Decision Trees ... with “You Don't Care” Omissions

- No known algorithm for PAC-learning general Decision Trees given all attribute values
- ... but Decision Trees are TRIVIAL to learn, if superfluous values are omitted:

**Algorithm GrowDT**

Collect “enough” labeled (blocked) instances

Let root = never-blocked instance $x_i$

Split instances by $x_i = 1$ vs $x_i = 0$, and recur (until purity)
Motivation

- Most learning systems work best when
  - few attribute values are missing
  - missing values randomly distributed
- but. . . [Porter, Bareiss, Holte'90]
  - many datasets missing $> \frac{1}{2}$ values!
  - not randomly missing but . . .
    - "[missing] when they are known to be irrelevant for classification or redundant with features already present in the case description"

⇒ Our Situation!!

- Why Learn? . . . when experts
  - not available, or
  - unable to articulate classification process
## Decision Tree Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>LMS</th>
<th>Logistic</th>
<th>LDA</th>
<th>DecTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed data</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Missing values</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Outliers</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Monotone transformations</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Irrelevant inputs</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Linear combinations</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Interpretable</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Predictive power</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Comments on Decision Trees

- "Decision Stumps" (1-level DT) seem to work surprisingly well.

- Efficient algorithms for learning optimal "depth-k decision trees" ... even if continuous variables.

- Oblique Decision Trees
  Not just "$x_3 > 5$", but "$x_4 + x_8 > 91$"

- Use of prior knowledge
  - Incremental Learners ("Theory Revision")
  - "Relevance" info

- Software Systems:
  - C5.0 (from ID3, C4.5) [Quinlan'93]
  - CART
  - ...

- Applications:
  - Gasoil $\approx 2500$ rules
  - designing gas-oil separation for offshore oil platforms
  - Learning to fly Cessna plane
What we haven’t discussed...

- Real-valued outputs – Regression Trees
- Bayesian Decision Trees
  - a different approach to preventing overfitting
- How to choose MaxPchance automatically
- Boosting: a simple way to improve accuracy
What you should know

- Information gain:
  - What is it? Why use it?
- Recursive algorithm for building an unpruned decision tree
- Why pruning can reduce test set error
- How to exploit real-valued inputs
- Computational complexity
  - straightforward, cheap
- Coping with Missing Data
- Alternatives to Information Gain for splitting nodes
For more information

- Two nice books
  - *C4.5: Programs for Machine Learning* (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan

  Both started \( \approx \) 1983, in Bay Area...done independently -- CS vs Stat

- Dozens of nice papers, including

  Dozens of software implementations available on the web for free and commercially for prices ranging between $50 - $300,000
Conclusions

- Classification: predict a categorical output from categorical and/or real inputs
- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Need to avoid overfitting