Lecture Slides for

INTRODUCTION TO

Machine Learning

ETHEM ALPAYDIN
© The MIT Press, 2004

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml
CHAPTER 9: Decision Trees
Tree Uses Nodes, and Leaves
Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, $x_i$
    - Numeric $x_i$: Binary split: $x_i > w_m$
    - Discrete $x_i$: $n$-way split for $n$ possible values
  - Multivariate: Uses all attributes, $x$

- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; $r$ average, or local fit

- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)
Classification Trees
(ID3, CART, C4.5)

- For node $m$, $N_m$ instances reach $m$, $N_m^i$ belong to $C_i$

$$\hat{P}(C_i|x, m) = p_m^i = \frac{N_m^i}{N_m}$$

- Node $m$ is pure if $p_m^i$ is 0 or 1
- Measure of impurity is entropy

$$I_m = - \sum_{i=1}^{K} p_m^i \log_2 p_m^i$$
Best Split

- If node $m$ is pure, generate a leaf and stop, otherwise split and continue recursively.

- Impurity after split: $N_{mj}$ of $N_m$ take branch $j$. $N^i_{mj}$ belong to $C_i$

$$\hat{P}(C_i|x, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)
\textbf{GenerateTree}(\mathcal{X}')

\textbf{If} \text{NodeEntropy}(\mathcal{X}') < \theta_I \quad /* \text{eq. 9.3} \\
\text{Create leaf labelled by majority class in } \mathcal{X}' \\
\text{Return} \\
i \leftarrow \text{SplitAttribute}(\mathcal{X}') \\
\text{For each branch of } x_i \\
\text{Find } \mathcal{X}_i \text{ falling in branch} \\
\text{GenerateTree}(\mathcal{X}_i) \\

\textbf{SplitAttribute}(\mathcal{X}')

\text{MinEnt} \leftarrow \text{MAX} \\
\text{For all attributes } i = 1, \ldots, d \\
\text{If } x_i \text{ is discrete with } n \text{ values} \\
\quad \text{Split } \mathcal{X} \text{ into } \mathcal{X}_1, \ldots, \mathcal{X}_n \text{ by } x_i \\
\quad e \leftarrow \text{SplitEntropy}(\mathcal{X}_1, \ldots, \mathcal{X}_n) \quad /* \text{eq. 9.8} */ \\
\quad \text{If } e < \text{MinEnt} \text{ MinEnt} \leftarrow e; \text{ bestf } \leftarrow i \\
\text{Else } /* x_i \text{ is numeric } */ \\
\quad \text{For all possible splits} \\
\quad \text{Split } \mathcal{X} \text{ into } \mathcal{X}_1, \mathcal{X}_2 \text{ on } x_i \\
\quad e \leftarrow \text{SplitEntropy}(\mathcal{X}_1, \mathcal{X}_2) \\
\quad \text{If } e < \text{MinEnt} \text{ MinEnt} \leftarrow e; \text{ bestf } \leftarrow i \\
\text{Return } \text{bestf}
Regression Trees

- Error at node $m$:

$$b_m(x) = \begin{cases} 1 & \text{if } x \in X_m: x \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

$$E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(x^t) \quad g_m = \frac{\sum_t b_m(x^t) r^t}{\sum_t b_m(x^t)}$$

- After splitting:

$$b_{mj}(x) = \begin{cases} 1 & \text{if } x \in X_{mj}: x \text{ reaches node } m \text{ and takes branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$E'_m = \frac{1}{N_m} \sum_j \sum_t (r^t - g_{mj})^2 b_{mj}(x^t) \quad g_{mj} = \frac{\sum_t b_{mj}(x^t) r^t}{\sum_t b_{mj}(x^t)}$$
Model Selection in Trees:

- $\theta_1 = 0.5$
- $\theta_1 = 0.2$
- $\theta_1 = 0.05$
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)
Rule Extraction from Trees

C4.5 Rules
(Quinlan, 1993)

\[ x_1 > 38.5 \]

- Yes
- No

\[ x_2 > 2.5 \]

- Yes
- No

\[ x_4 \]

- 'A'
- 'B'
- 'C'

\[
\begin{array}{ccc}
0.8 & 0.6 & 0.4 & 0.3 & 0.2 \\
\end{array}
\]

R1: IF (age > 38.5) AND (years-in-job > 2.5) THEN \( y = 0.8 \)
R2: IF (age > 38.5) AND (years-in-job \( \leq \) 2.5) THEN \( y = 0.6 \)
R3: IF (age \( \leq \) 38.5) AND (job-type = 'A') THEN \( y = 0.4 \)
R4: IF (age \( \leq \) 38.5) AND (job-type = 'B') THEN \( y = 0.3 \)
R5: IF (age \( \leq \) 38.5) AND (job-type = 'C') THEN \( y = 0.2 \)
Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule covers an example if all terms of the rule evaluate to true for the example
- Sequential covering: Generate rules one at a time until all positive examples are covered
- IREP (Fürnkranz and Widmer, 1994), Ripper (Cohen, 1995)
Ripper(Pos, Neg, k)
   RuleSet ← LearnRuleSet(Pos, Neg)
   For k times
      RuleSet ← OptimizeRuleSet(RuleSet, Pos, Neg)
   LearnRuleSet(Pos, Neg)
   RuleSet ← Ø
   DL ← DescLen(RuleSet, Pos, Neg)
   Repeat
      Rule ← LearnRule(Pos, Neg)
      Add Rule to RuleSet
      DL' ← DescLen(RuleSet, Pos, Neg)
      If DL’ > DL + 64
         PruneRuleSet(RuleSet, Pos, Neg)
         Return RuleSet
      If DL’ < DL DL ← DL’
         Delete instances covered from Pos and Neg
   Until Pos = Ø
   Return RuleSet
PruneRuleSet(RuleSet, Pos, Neg)
  For each Rule ∈ RuleSet in reverse order
    DL ← DescLen(RuleSet, Pos, Neg)
    DL′ ← DescLen(RuleSet-Rule, Pos, Neg)
    IF DL′<DL Delete Rule from RuleSet
  Return RuleSet

OptimizeRuleSet(RuleSet, Pos, Neg)
  For each Rule ∈ RuleSet
    DL0 ← DescLen(RuleSet, Pos, Neg)
    DL1 ← DescLen(RuleSet-Rule+ ReplaceRule(RuleSet, Pos, Neg), Pos, Neg)
    DL2 ← DescLen(RuleSet-Rule+ ReviseRule(RuleSet, Rule, Pos, Neg), Pos, Neg)
    If DL1=min(DL0, DL1, DL2)
      Delete Rule from RuleSet and
      add ReplaceRule(RuleSet, Pos, Neg)
    Else If DL2=min(DL0, DL1, DL2)
      Delete Rule from RuleSet and
      add ReviseRule(RuleSet, Rule, Pos, Neg)
  Return RuleSet
Multivariate Trees

\[ w_{11}x_1 + w_{12}x_2 + w_{10} = 0 \]

Yes

No

\[ C_2 \]

\[ C_1 \]