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 $m$, $N^i_m$ belong to $C_i$
  \[ \hat{P}(C_i \mid x, m) \equiv p^i_m = \frac{N^i_m}{N_m} \]

- Node $m$ is pure if $p^i_m$ is 0 or 1
- Measure of impurity is entropy
  \[ I_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m \]
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 $j$. $N_{mj}^i$ belong to $C_i$

$$\hat{P}(C_i \mid x, m, j) \equiv p_{mj}^i = \frac{N_{mj}^i}{N_{mj}}$$

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log_2 p_{mj}^i$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)
GenerateTree($\mathcal{X}$)
    If NodeEntropy($\mathcal{X}$) < $\theta_I$ /* eq. 9.3
    Create leaf labelled by majority class in $\mathcal{X}$
    Return
    $i \leftarrow$ SplitAttribute($\mathcal{X}$)
    For each branch of $x_i$
        Find $\mathcal{X}_i$ falling in branch
        GenerateTree($\mathcal{X}_i$)

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

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

$$E'_m = \frac{1}{N_m} \sum_j \sum_t (r^t - g_{mj})^2 b_{mj}(x^t) \quad g_{mj} = \frac{\sum_t b_{mj}(x^t)r^t}{\sum_t b_{mj}(x^t)}$$
Model Selection in Trees:
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
  x_2 > 2.5
    Yes
    0.8
  No
    0.6

x_4
  'A'
  'B'
  'C'
  0.4
  0.3
  0.2

x_1: Age
x_2: Years in job
x_3: Gender
x_4: Job type
```

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 ← ∅
    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 = ∅
    Return RuleSet
\begin{algorithm}
\caption{PruneRuleSet(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg})}
\begin{algorithmic}
\State For each Rule $\in \texttt{RuleSet}$ in reverse order
\State \hspace{1em} \texttt{DL} $\leftarrow$ DescLen(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \texttt{DL'} $\leftarrow$ DescLen(\texttt{RuleSet-Rule}, \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \textbf{IF} $\texttt{DL'}<\texttt{DL}$ Delete Rule from RuleSet
\State \hspace{1em} Return RuleSet
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{OptimizeRuleSet(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg})}
\begin{algorithmic}
\State For each Rule $\in \texttt{RuleSet}$
\State \hspace{1em} \texttt{DL0} $\leftarrow$ DescLen(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \texttt{DL1} $\leftarrow$ DescLen(\texttt{RuleSet-Rule+} \textbf{ReplaceRule}(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg}), \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \texttt{DL2} $\leftarrow$ DescLen(\texttt{RuleSet-Rule+} \textbf{ReviseRule}(\texttt{RuleSet}, \texttt{Rule}, \texttt{Pos}, \texttt{Neg}), \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \textbf{If} $\texttt{DL1}=\min(\texttt{DL0}, \texttt{DL1}, \texttt{DL2})$
\State \hspace{2em} Delete Rule from RuleSet and 
\State \hspace{3em} add \textbf{ReplaceRule}(\texttt{RuleSet}, \texttt{Pos}, \texttt{Neg})
\State \hspace{1em} \textbf{Else If} $\texttt{DL2}=\min(\texttt{DL0}, \texttt{DL1}, \texttt{DL2})$
\State \hspace{2em} Delete Rule from RuleSet and 
\State \hspace{3em} add \textbf{ReviseRule}(\texttt{RuleSet}, \texttt{Rule}, \texttt{Pos}, \texttt{Neg})
\State Return RuleSet
\end{algorithmic}
\end{algorithm}
Multivariate Trees

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

Yes

No

\( C_2 \)

\( C_1 \)