CSCE 478/878 Lecture 3: Learning Decision Trees

Stephen Scott

(Adapted from Ethem Alpaydin and Tom Mitchell)

sscott@cse.unl.edu
**Decision trees** form a simple, easily-interpretable, hypothesis

- Interpretability useful in independent validation and explanation

Quick to train

Quick to evaluate new instances

Effective “off-the-shelf” learning method

Can be combined with boosting, including using “stumps”
Outline

- Decision tree representation
- Learning trees (ID3/C4.5/CART)
  - Entropy as a splitting criterion
  - Example run of algorithm
  - Regression trees
  - Variations
- Inductive bias
- Overfitting and pruning
- Deriving rules from tree
Decision Tree for *PlayTennis* (Mitchell)

**Outlook**
- **Sunny**
  - **Humidity**
    - **High**
      - No
    - **Normal**
      - Yes
  
- **Overcast**
  - **Yes**

- **Rain**
  - **Wind**
    - **Strong**
      - No
    - **Weak**
      - Yes
With Numeric Attributes
Decision Tree Representation

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\land$, $\lor$, XOR
- $(A \land B) \lor (C \land \neg D \land E)$
High-Level Learning Algorithm
(ID3, C4.5, CART)

Main loop:
1. $A \leftarrow$ the “best” decision attribute for next node $m$
2. Assign $A$ as decision attribute for $m$
3. For each value of $A$, create new descendant of $m$
4. Sort (partition) training examples over children based on $A$’s value
5. If training examples perfectly classified, Then STOP, Else recursively iterate over new child nodes

Which attribute is best?

$$[29+, 35-] \quad A1=?$$

$$[21+, 5-] \quad t$$

$$[8+, 30-] \quad f$$

$$[29+, 35-] \quad A2=?$$

$$[18+, 33-] \quad t$$

$$[11+, 2-] \quad f$$
Entropy

- $\mathcal{X}_m$ is a sample of training examples reaching node $m$
- $p_m^\oplus$ is the proportion of positive examples in $\mathcal{X}_m$
- $p_m^\ominus$ is the proportion of negative examples in $\mathcal{X}_m$
- *Entropy* $\mathcal{I}_m$ measures the impurity of $\mathcal{X}_m$

$$
\mathcal{I}_m \equiv -p_m^\oplus \log_2 p_m^\oplus - p_m^\ominus \log_2 p_m^\ominus
$$

or for $K$ classes,

$$
(9.3) \quad \mathcal{I}_m \equiv - \sum_{i=1}^{K} p_m^i \log_2 p_m^i
$$
Total Impurity

Now can look for an attribute $A$, when used to partition $\mathcal{X}_m$ by value, produces the most pure (lowest-entropy) subsets

- Weight each subset by relative size
- E.g., size-3 subsets should carry less influence than size-300 ones

Let $N_m = |\mathcal{X}_m| = \text{number of instances reaching node } m$

Let $N_{mj} = \text{number of these instances with value } j \in \{1, \ldots, n\} \text{ for attribute } A$

Let $N_{mj}^i = \text{number of these instances with label } i \in \{1, \ldots, K\}$

Let $p_{mj}^i = N_{mj}^i / N_{mj}$

Then the total impurity is

\[
\mathcal{I}_m'(A) \equiv - \sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log_2 p_{mj}^i
\]
Learning Algorithm

\[
\text{GenerateTree}(X) \\
\text{If } \text{NodeEntropy}(X) < \theta \text{ /* equation 9.3 */} \\
\text{Create leaf labelled by majority class in } X \\
\text{Return} \\
i \leftarrow \text{SplitAttribute}(X) \\
\text{For each branch of } x_i \\
\text{Find } X_i \text{ falling in branch} \\
\text{GenerateTree}(X_i)
\]

\[
\text{SplitAttribute}(X) \\
\text{MinEnt} \leftarrow \text{MAX} \\
\text{For all attributes } i = 1, \ldots, d \\
\text{If } x_i \text{ is discrete with } n \text{ values} \\
\text{Split } X \text{ into } X_1, \ldots, X_n \text{ by } x_i \\
e \leftarrow \text{SplitEntropy}(X_1, \ldots, X_n) \text{ /* equation 9.8 */} \\
\text{If } e < \text{MinEnt} \text{ MinEnt } \leftarrow e; \text{ bestf } \leftarrow i \\
\text{Else /* } x_i \text{ is numeric */} \\
\text{For all possible splits} \\
\text{Split } X \text{ into } X_1, X_2 \text{ on } x_i \\
e \leftarrow \text{SplitEntropy}(X_1, X_2) \\
\text{If } e < \text{MinEnt} \text{ MinEnt } \leftarrow e; \text{ bestf } \leftarrow i \\
\text{Return } \text{bestf}
\]
### Example Run

**Training Examples**

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</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>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Example Run
Selecting the First Attribute

Comparing *Humidity* to *Wind*:

\[
\mathcal{I}_m^{'}(\text{Humidity}) = \frac{7}{14}0.985 + \frac{7}{14}0.592 = 0.789 \\
\mathcal{I}_m^{'}(\text{Wind}) = \frac{8}{14}0.811 + \frac{6}{14}1.000 = 0.892 \\
\mathcal{I}_m^{'}(\text{Outlook}) = \frac{5}{14}0.971 + \frac{4}{14}0.0 + \frac{5}{14}0.971 = 0.694 \\
\mathcal{I}_m^{'}(\text{Temp}) = \frac{4}{14}1.000 + \frac{6}{14}0.918 + \frac{4}{14}0.811 = 0.911
\]
Example Run
Selecting the Next Attribute

{D1, D2, ..., D14}
[9+,5−]

\( \text{Outlook} \)

\( \text{Sunny} \)
\[ \text{Yes} \]
\[ ? \]

\( \text{Overcast} \)
\[ ? \]

\( \text{Rain} \)
\[ ? \]

\( \chi_m = \{D_1, D_2, D_8, D_9, D_{11}\} \)

\( \mathcal{I}_m'(\text{Humidity}) = (3/5)0.0 + (2/14)0.0 = 0.0 \)

\( \mathcal{I}_m'(\text{Wind}) = (2/5)1.0 + (3/5)0.918 = 0.951 \)

\( \mathcal{I}_m'(\text{Temp}) = (2/5)0.0 + (2/5)1.0 + (1/5)0.0 = 0.400 \)
Regression Trees

- A *regression tree* is similar to a decision tree, but with real-valued labels at the leaves.
- To measure impurity at a node \( m \), replace entropy with variance of labels:

\[
E_m \equiv \frac{1}{N_m} \sum_{(x^t, r^t) \in \mathcal{X}_m} (r^t - g_m)^2 ,
\]

where \( g_m \) is the mean (or median) label in \( \mathcal{X}_m \).
Regression Trees (cont’d)

Now can adapt Eq. (9.8) from classification to regression:

\[
E'_m(A) \equiv \frac{1}{N_m} \sum_{j=1}^{n} \frac{N_{mj}}{N_m} \left( \frac{1}{N_{mj}} \sum_{(x^t, r^t) \in X_{mj}} (r^t - g_{mj})^2 \right)
\]

(9.14) \quad = \frac{1}{N_m} \sum_{j=1}^{n} \sum_{(x^t, r^t) \in X_{mj}} (r^t - g_{mj})^2,

where \( j \) iterates over the values of attribute \( A \)

- When variance of a subset is sufficiently low, insert leaf with mean or median label as constant value
Continuous-Valued Attributes

Use threshold to map continuous to boolean, e.g. 
\((Temperature > 72.3) \in \{t,f\}\)

<table>
<thead>
<tr>
<th>Temperature:</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis:</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- Can show that threshold minimizing impurity must lie between two adjacent attribute values in \(X\) such that label changed, so try all such values, e.g., 
  \((48 + 60)/2 = 54\) and \((80 + 90)/2 = 85\)
- Now (dynamically) replace continuous attribute with boolean attributes \(Temperature > 54\) and \(Temperature > 85\) and run algorithm normally
- Other options: Split into multiple intervals rather than two; use thresholded linear combinations of continuous attributes (Sec 9.6)
Attributes with Many Values

Problem:

- If attribute $A$ has many values, it might artificially minimize $I'_m(A)$
- E.g., if $Date$ is attribute, $I'_m(A)$ will be low because several very small subsets will be created

One approach: penalize $A$ with a measure of *split information*, which measures how broadly and uniformly attribute $A$ splits data:

$$S(A) \equiv - \sum_{j=1}^{n} \frac{N_{mj}}{N_m} \log_2 \frac{N_{mj}}{N_m} \in [0, \log_2 n]$$
Unknown Attribute Values

What if a training example is missing a value of $A$?

Use it anyway (sift it through tree)

- If node $m$ tests $A$, assign most common value of $A$ among other training examples sifted to $m$
- Assign most common value of $A$ among other examples with same target value (either overall or at $m$)
- Assign probability $p_j$ to each possible value $v_j$ of $A$
  - Assign fraction $p_j$ of example to each descendant in tree

Classify new examples in same fashion
Hypothesis space $\mathcal{H}$ is complete, in that any function can be represented.

Thus inductive bias does not come from restricting $\mathcal{H}$, but from preferring some trees over others.

- Tends to prefer shorter trees.
- Computationally intractable to find a guaranteed shortest tree, so heuristically apply greedy approach to locally minimize impurity.
Consider adding noisy training example #15:

\[ \text{Sunny, Hot, Normal, Strong, PlayTennis} = \text{No} \]

What effect on earlier tree?

Expect old tree to generalize better since new one fits noisy example
Consider error of hypothesis $h$ over
- training data (empirical error): $\text{error}_{\text{train}}(h)$
- entire distribution $D$ of data (generalization error): $\text{error}_D(h)$

Hypothesis $h \in \mathcal{H}$ overfits training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

and

$$\text{error}_D(h) > \text{error}_D(h')$$
Overfitting (cont’d)

![Graph showing accuracy vs size of tree]

- On training data
- On test data

**Accuracy** vs **Size of tree (number of nodes)**
Pruning to Avoid Overfitting

- To prevent trees from growing too much and overfitting the data, we can prune them
  - In spirit of Occam’s Razor, minimum description length
- In prepruning, we allow skipping a recursive call on set $\mathcal{X}_m$ and instead insert a leaf, even if $\mathcal{X}_m$ is not pure
  - Can do this when entropy (or variance) is below a threshold ($\theta_I$ in pseudocode)
  - Can do this when $|\mathcal{X}_m|$ is below a threshold, e.g., 5
- In postpruning, we grow the tree until it has zero error on training set and then prune it back afterwards
  - First, set aside a pruning set not used in initial training
  - Then repeat until 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
Pruning Example

![Graph showing accuracy vs. size of tree (number of nodes).](graph.png)

- On training data
- On test data
- On test data (during pruning)
Rule Postpruning

- Convert tree to equivalent set of rules
- Prune each rule independently of others by removing selected preconditions (the ones that improve accuracy the most)
- Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g. C4.5)
Converting A Tree to Rules

\[
\text{IF} \quad (\text{Outlook} = \text{Sunny}) \land (\text{Humidity} = \text{High}) \\
\text{THEN} \quad \text{PlayTennis} = \text{No}
\]

\[
\text{IF} \quad (\text{Outlook} = \text{Sunny}) \land (\text{Humidity} = \text{Normal}) \\
\text{THEN} \quad \text{PlayTennis} = \text{Yes}
\]

\ldots