Machine Learning

Decision Trees
Decision Trees

- Classifiers covered so far have been
  - Non-parametric (KNN)
  - Probabilistic with independence (Naïve Bayes)
  - Linear in features (Logistic regression, SVM)
- Decision trees use a different representation that is inherently non-linear
  - Hypothesis space contains logic sentences over boolean variables and feature functions (e.g. =, >)
    - Intuitive representation
    - Can be converted into rules relatively easily
Decision Trees

- Hypotheses are represented as trees
  - Nodes are variables (features)
  - Leaves are class predictions (or class probabilities)
    - Leaves can also contain subsets of the training set
  - Arcs represent node evaluations

Example from Tom Mitchell
Decision Trees

- Decision trees represent logical sentences in disjunctive normal form (DNF)
  - Consecutive elements in a branch form conjuncts
  - Branches with same leaf label form disjuncts
- Variables/features are assumed to be independent
- Can be converted into rules by making the sentences into implications with class as the right hand side

A Decision tree for $F$: <Outlook, Humidity, Wind, Temp>

```
        Outlook
         /    \
       Sunny Overcast Rain
         /    \
       Humidity   \\
     /    \
   High Normal Yes
   /    \
  No    Yes
```

© Manfred Huber 2015
Decision Trees

- Decision trees work with continuous and discrete variables and form complex decision boundaries
  - Non-continuous
  - Non-differentiable
- Decision trees can dramatically overfit noisy data
Decision Trees

- Variables can occur multiple times on a branch of the tree if they are non-boolean
  - Decision trees with “predicates” = , < represent all possible DNF sentences with arbitrary thresholds
    - Trees can have very different depths
    - Unlimited hypothesis space
    - Generally many decision trees for same classification
- Decision trees can be extended
  - Use additional features
  - Including additional “predicates” for evaluation
Learning Decision Trees

- There is no continuous interpretation of either the probability or the decision boundary
  - No derivative with respect to tree parameters
- To optimize performance, decision tree algorithms generally use search
  - Since the space of all trees is intractable they typically use greedy, fixed lookahead search
    - Each greedy search adds one node to the tree
    - The resulting tree is generally not optimal
Learning Decision Trees

- Performance function
  - Classification performance
    - “Purity” of the prediction
- Tree construction criterion
  - Tree simplicity
    - Usually terms of number of nodes or depth of the tree
- Basic algorithms iterates as follows
  - Do lookahead search to find “best” feature node
    - Terminate if feature does not improve performance
  - Add node and split parent data set on node values
Learning Decision Trees

- How can we pick the best feature?
  - Error in terms of number of misclassified data points
  - Does not consider whether the result of the split might allow for a better split later
Learning Decision Trees

- How can we pick the best feature?
  - Information gain (reduction in the Entropy)
    - Performance function is entropy
      \[ H_D(Y) = \sum_{y \in C} P_D(y) \log P_D(y) \]
    - Best feature is the one that reduces entropy the most (has the highest information gain)
      \[
      Gain(D, A) = I_D(A, Y) = H_D(Y) - H_D(Y \mid A)
      = H_D(Y) - \sum_{a \in \text{range}(A)} P_D(A = a) \log H_D(Y \mid A = a)
      \]
    - Prefer “purer” sets
Example

<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 from Tom Mitchell
Example

\[ \text{Gain (S, Humidity)} = 0.940 - (5/14) \times 0.971 - (4/14) \times 0 - (5/14) \times 0.971 = 0.246 \]

Gain (S, Wind) = 0.588

Gain (S, Outlook) = 0.971 - (4/14) \times 0 - (5/14) \times 0.971 = 0.246
Multi-Valued Features

- Features with multiple values have to be treated differently
  - Many-way splits result in very small sets and thus unreliable estimates of the entropy
    - Gain ratio provides one possible way to overcome this
      - Gain ratio looks at the information gain relative to the intrinsic information of the feature

\[
GainRatio(D, A) = \frac{Gain(D, A)}{-\sum_{a \in \text{range}(A)} \frac{|D_a|}{|D|} \log \frac{|D_a|}{|D|}}
\]

- Tries to compensate for the benefits of multiple options
Continuous features

- Continuous Features have to be converted into boolean (or multi-valued)
  - For continuous variables a threshold for the < "predicate" has to be picked
    - Sort data elements and evaluate every possible threshold at which the classification changes
      - Other points can not result in maximum
    - Use the threshold with the highest gain
Overfitting in Decision Trees

- If there is noise in the data, decision trees can significantly overfit
  - To determine overfitting, we need to use test data that was not used for training
Overfitting in Decision Trees

- There are two basic mechanisms to address overfitting
  - Early stopping when split is not significant
  - Post-pruning after complete learning
  - Evaluate benefit of pruning on accuracy in test set and remove the node (branch) with highest accuracy gain

![Graph showing accuracy over tree size](image)
Decision Trees and Rules

- Decision trees can be easily converted into a set of rules (one rule per branch)

  IF \( (\text{Outlook} = \text{Sunny}) \text{ AND } (\text{Humidity} = \text{High}) \) THEN \( \text{PlayTennis} = \text{No} \)

  IF \( (\text{Outlook} = \text{Sunny}) \text{ AND } (\text{Humidity} = \text{Normal}) \) THEN \( \text{PlayTennis} = \text{Yes} \)

- In C4.5 (and a number of other decision tree algorithms) pruning is performed on the rules
  - Prune each rule individually
  - Sort rules into desired sequence for faster use
Decision Trees

- Decision trees are a very frequently used type of classifier in particular when discrete features are present
  - Can represent highly non-linear classes
  - Can be translated into rules
  - Result is easy to understand and interpret
  - Does not represent continuous relationships
    - Can be augmented by including logistic regression “predicates” and multi-variate feature functions
- Many practical algorithms: ID3, C4.5, ...