Decision Trees

some slides/drawings thanks to Carlos Guestrin@CMU
• two basic supervised learning algorithms
  - decision trees
  - linear regression
• two simple datasets
  - housing
  - spam emails
Module 1 Objectives / Decision Trees

- Decision Trees
- Splitting Criteria
  - decision stumps
  - how to look for the best splits
- Regression Trees
  - regression criteria
- Run a Decision Tree in practice
- Pruning
**Data Partition Rules**

- \( x_1, x_2 = \) data features
- Each path in the tree corresponds to a region
- Deeper paths correspond to smaller regions
Data Partition Rules
Data Partition Rules
Decision Trees

• Goal: Learn from training set a decision tree
  – initially all training datapoints at root
• iterative splits:
  – pick a terminal node (leaf) with inconsistent labels
  – use a split criteria to branch data so that each resulting child node has [more] consistent labels
  – until no terminal nodes are inconsistent
• Use learned tree for prediction on the test set
Walkthrough Decision Tree Example

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>modelyear</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>75to78</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>8</td>
<td>high</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>medium</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>medium</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
</tr>
</tbody>
</table>

40 Records

- Data (matrix) example: automobiles
- Target: mpg ∈ \{good, bad\} - 2 class /binary problem
• Split by feature “cylinders”, using feature values for branches
each terminal leaf is labeled by majority (at that leaf). This leaf-label is used for prediction.
Decision Tree Splits

The final tree

mpg values: bad good

root
22 18
pchance = 0.001

- cylinders = 3
  - 0 0
  - Predict bad
- cylinders = 4
  - 4 17
  - pchance = 0.135
  - Predict bad
- cylinders = 5
  - 1 0
  - Predict bad
- cylinders = 6
  - 8 0
  - Predict bad
- cylinders = 8
  - 9 1
  - pchance = 0.085

- maker = america
  - 0 10
  - Predict good
  - pchance = 0.317
  - Predict bad
- maker = asia
  - 2 5
  - Predict good
  - pchance = 0.717
  - Predict bad
- maker = europe
  - 2 2
  - Predict good
  - pchance = 0.717
  - Predict bad

- horsepower = low
  - 0 4
  - Predict good
  - pchance = 0.894
  - Predict bad
- horsepower = medium
  - 2 1
  - Predict good
  - pchance = 0.894
  - Predict bad
- horsepower = high
  - 0 0
  - Predict bad
  - pchance = 0.717
  - Predict good

- acceleration = low
  - 1 0
  - Predict bad
  - (unexpandable)
- acceleration = medium
  - 1 1
  - Predict bad
  - (unexpandable)
- acceleration = high
  - 0 0
  - Predict bad
  - pchance = 0.717
  - Predict good
  - Predict bad
  - Predict bad
  - Predict bad
Splitting criteria: entropy-based gain

\[ H(Y) = \sum_j P(y_j) \log_2 \left( \frac{1}{P(y_j)} \right) \]

Entropy after split by \( X \) feature

\[ H(Y|X) = \sum_i P(x_i) \sum_j P(y_j|x_i) \log_2 \left( \frac{1}{P(y_j|x_i)} \right) \]

Mutual information (or Information Gain).

\[ IG(X) = H(Y) - H(Y|X) \]

- \( Y = \) labels random variable, \( H(Y) \) its entropy
- \( X \) is a feature of the data used for splitting
Entropy gain toy example

At each split we are going to choose the feature that gives the highest information gain.

\[
\begin{array}{c|c|c|c|c|c}
 & x^1 & x^2 & Y \\
 T & T & T & T \\
 T & F & T & T \\
 T & T & T & T \\
 T & F & T & T \\
 F & T & T & T \\
 F & F & F & F \\
 F & T & F & F \\
 F & F & F & F \\
\end{array}
\]

Figure 6: 2 possible features to split by

\[
H(Y|X^1) = \frac{1}{2} H(Y|X^1 = T) + \frac{1}{2} H(Y|X^1 = F) = 0 + \frac{1}{2} \left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) \approx 0.405
\]

\[
IG(X^1) = H(Y) - H(Y|X^1) = 0.954 - 0.405 = 0.549
\]

\[
H(Y|X^2) = \frac{1}{2} H(Y|X^2 = T) + \frac{1}{2} H(Y|X^2 = F) = \frac{1}{2} \left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) + \frac{1}{2} \left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) \approx 0.905
\]

\[
IG(X^2) = H(Y) - H(Y|X^2) = 0.954 - 0.905 = 0.049
\]
compute the information gain for f=cylinders and for f=displacement

once a split by f=cylinders is performed, for the branch “cylinders=4” compute the information gain for f=displacement and for f=maker
Regression Tree

- same tree structure, split criteria
- assume numerical labels
- for each terminal node compute the node label (predicted value) and the mean square error

Estimate a predicted value per tree node

\[ g_m = \frac{\sum_{t \in \chi_m} y_t}{|\chi_m|} \]

Calculate mean square error

\[ E_m = \frac{\sum_{t \in \chi_m} (y_t - g_m)^2}{|\chi_m|} \]

- choose a split criteria to minimize the weighted error at children nodes
Regression Tree

• choose a split criteria to minimize the weighted or total error at children nodes
  - in the example total error after the split is $14.75 + \frac{2}{2} = 16.75$
Prediction with a tree

- for each test datapoint $x=(x^1,x^2,...,x^d)$ follow the corresponding path to reach a terminal node $n$
- predict the value/label associated with node $n$
Prediction with a tree

- cylinder=4
- maker=asia
- horsepower=low
- weight=low
- displacement=medium
- modelyear=75to78
Overfitting

- decision trees can overfit quite badly
  - in fact they are designed to do so due to high complexity of the produced model
  - if a decision tree training error doesn't approach zero, it means that data is inconsistent

- some ideas to prevent overfitting:
  - create more than one tree, each using a different subset of features; average/vote predictions
  - do not split nodes in the tree that have very few datapoints (for example less than 10)
  - only split if the improvement is massive
Pruning

• done also to prevent overfitting
• construct a full decision tree
• then walk back from the leaves and decide to “merge” overfitting nodes
  – when split complexity overwhelms the gain obtained by the split
tree implementation

- perl/python: easy to use a hash
- matlab: use a vector/matrix
- C/Java: use a struct/object with pointers to children nodes.
Decision Tree Screencast

- http://www.screencast.com/t/J0jLmCdBW0M6