Overview of Learning

Type of Supervision
(eg, Experience, Feedback)

<table>
<thead>
<tr>
<th>What’s Being Learned?</th>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Function</td>
<td>Classification</td>
<td>Clustering</td>
<td></td>
</tr>
<tr>
<td>Continuous Function</td>
<td>Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
<td></td>
</tr>
</tbody>
</table>

Hypothesis space

- How many possible hypotheses?
- What functions can be represented?
- How many will be consistent with a given dataset?
- How will we choose the best one?

Two Questions

Greedy Algorithm:
- Start from empty decision tree
- Split on the best attribute (feature)
- Recurse

1. Which attribute gives the best split?
2. When to stop recursion?
Reduced Error Pruning

Split data into training & validation sets (10-33%)

Train on training set (overfitting)
Do until further pruning is harmful:
1) Evaluate effect on validation set of pruning each possible node (and tree below it)
2) Greedily remove the node that most improves accuracy of validation set

A chi-square test

Suppose that mpg was completely uncorrelated with maker.
What is the chance we’d have seen data of at least this apparent level of association anyway?
By using a particular kind of chi-square test, the answer is 13.5%
Such hypothesis tests are relatively easy to compute, but involved

Using Chi-squared to avoid overfitting

• Build the full decision tree as before
• But when you can grow it no more, start to prune:
  – Beginning at the bottom of the tree, delete splits in which \( p_{\text{chance}} > \text{MaxPchance} \)
  – Continue working you way up until there are no more prunable nodes

MaxPchance is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise

Regularization

• Note for Future: \( \text{MaxPchance} \) is a regularization parameter that helps us bias towards simpler models

We’ll learn to choose the value of magic parameters like this one later!

ML as Optimization

• Greedy search for best scoring hypothesis
• Where score =
  – Fits training data most accurately?
  – Sum: \( \text{training accuracy} - \text{complexity penalty} \)

To be continued...

Advanced Decision Trees

• Attributes with:
  – Numerous Possible Values
  – Continuous (Ordered) Values
  – Missing Values
Information Gain

IG of attribute =
Decrease in entropy (uncertainty) after splitting

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

Many Attribute Values

- What if split on S/N?
- \( H(MPG \mid S/N) \)?

Attributes with many values

- So many values that splitting on attribute...
  - Divides examples into tiny sets
  - Each set likely homogeneous → high info gain
  - But poor predictor...
- Need to penalize these attributes
  - S/N is worst case, but correction is often needed

One Approach: Gain Ratio

\[ \text{GainRatio}(S, A) = \frac{IG(S, A)}{\text{Split Information}(S, A)} \]

\[ \text{Split Information}(S, A) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \left( \frac{|S_i|}{|S|} \right) \]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \)

SplitInfo ≡ entropy of \( S \) wrt values of \( A \)
  (Contrast with entropy of \( S \) wrt target value)

\( \Updownarrow \) Attributes with many evenly distributed values

SplitInformation = \( \log_2(n) \)...
  = 1 for Boolean

Ordinal (Real-Valued) Inputs

What should we do if some of the inputs are real-valued?
One branch for each value?

Hopeless overfitting

Good News: GainRatio will reject these!

Bad News: They might have useful info!
  Eg, weight of car might = mpg

So...?
Threshold Splits

• Binary tree, split on attribute X at value t
  – One branch: X < t
  – Other branch: X ≥ t

Weight ≤ 3200
Weight > 3200

The set of possible thresholds

• Binary tree, split on attribute X
  – One branch: X < t
  – Other branch: X ≥ t
• Search through possible values of t
  – Seems hard!!!
• But only finite number of t’s are important
  – Sort data according to X into \{x_1, ..., x_m\}
  – Consider split points of the form \(x_i + (x_{i+1} - x_i)/2\)
• Consider all such splits?

Only Certain Splits Make Sense

<table>
<thead>
<tr>
<th>weight</th>
<th>MPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>good</td>
</tr>
<tr>
<td>2600</td>
<td>good</td>
</tr>
<tr>
<td>3100</td>
<td>good</td>
</tr>
<tr>
<td>3500</td>
<td>bad</td>
</tr>
<tr>
<td>4200</td>
<td>good</td>
</tr>
<tr>
<td>4400</td>
<td>bad</td>
</tr>
<tr>
<td>4600</td>
<td>bad</td>
</tr>
<tr>
<td>6000</td>
<td>bad</td>
</tr>
<tr>
<td>7200</td>
<td>good</td>
</tr>
<tr>
<td>7800</td>
<td>bad</td>
</tr>
<tr>
<td>7995</td>
<td>bad</td>
</tr>
</tbody>
</table>

Picking the best threshold

• Suppose X is real valued
• Define \(IG(Y|X:t) = H(Y) - H(Y|X:t)\)
• Define \(H(Y|X:t) = H(Y|X < t)P(X < t) + H(Y|X \geq t)P(X \geq t)\)
• \(IG(Y|X:t)\) is the information gain for predicting Y if all you know is whether X is greater than or less than t
• Then define \(IG^*(Y|X) = \max_t IG(Y|X:t)\)
• For each real-valued attribute, use IG*(Y|X) for assessing its suitability as a split
• Note, may split on an attribute multiple times, with different thresholds

Example Tree for MPG treating values as ordinal

Missing Data

What to do??
Options

• Use “missing” as value
• Don’t use attribute at all
• Assign most common value to missing attribute
• Fractional values

Fractional Values

• 66% low and 33% high
• Training
  – Use in gain calculations
  – Further subdivide if other missing attributes
• Test
  – Classification is most probable classification
  – Averaging over leaves where it got divided

What you need to know about decision trees

• Decision trees are one of the most popular ML tools
  – Easy to understand, implement, and use
  – Computationally cheap (to solve heuristically)
• Information gain to select attributes (ID3, C4.5,...)
• Presented for classification, can be used for regression and density estimation too
• Decision trees will overfit!!!
  – Must use tricks to find “simple trees”, e.g.,
    • Fixed depth/Early stopping
    • Pruning
    • Hypothesis testing

Loss Functions

• How measure quality of hypothesis?

Loss Functions

How measure quality of hypothesis?

$L(x, y, \hat{y}) = \text{utility(result of using } y \text{ given input of } x) - \text{utility(result of using } \hat{y} \text{ given input of } x)$

$L(\text{edible, poison})$
$L(\text{poison, edible})$

Common Loss Functions

• 0/1 loss $0$ if $y=\hat{y}$ else $1$
• Absolute value loss $|y-\hat{y}|$
• Squared error loss $|y-\hat{y}|^2$
### Overview of Learning

#### Type of Supervision
(eg, Experience, Feedback)

<table>
<thead>
<tr>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Function</td>
<td>Classification</td>
<td>Clustering</td>
</tr>
<tr>
<td>Continuous Function</td>
<td>Regression</td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
</tr>
</tbody>
</table>

#### Polynomial Curve Fitting

- Hypothesis Space
  \[ p(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

#### Sum-of-Squares Error Function

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x, w) - t_n)^2 \]

#### Examples

- **1st Order Polynomial**

- **3rd Order Polynomial**

- **9th Order Polynomial**
Over-fitting

Root-Mean-Square (RMS) Error:  

\[ E_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} [y_n - \hat{y}_n]^2} \]

Polynomial Coefficients

<table>
<thead>
<tr>
<th>M = 0</th>
<th>M = 1</th>
<th>M = 3</th>
<th>M = 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0 )</td>
<td>( w_1 )</td>
<td>( w_2 )</td>
<td>( w_3 )</td>
</tr>
<tr>
<td>0.19</td>
<td>-1.27</td>
<td>7.99</td>
<td>222.37</td>
</tr>
<tr>
<td>( w_4 )</td>
<td>( w_5 )</td>
<td>( w_6 )</td>
<td>( w_7 )</td>
</tr>
<tr>
<td>-29.43</td>
<td>-5321.83</td>
<td>17.37</td>
<td>48569.31</td>
</tr>
<tr>
<td>( w_8 )</td>
<td>( w_9 )</td>
<td>( w_{10} )</td>
<td>( w_{11} )</td>
</tr>
<tr>
<td>-2316033.3</td>
<td>6400.02.26</td>
<td>-1061280.52</td>
<td>-1042600.18</td>
</tr>
<tr>
<td>( w_{12} )</td>
<td>( w_{13} )</td>
<td>( w_{14} )</td>
<td>( w_{15} )</td>
</tr>
<tr>
<td>1042600.18</td>
<td>-557852.99</td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>

Data Set Size:  \( N = 15 \)  

9th Order Polynomial

Data Set Size:  \( N = 100 \)  

9th Order Polynomial

Regularization

Regularization:  \( \ln \lambda = -18 \)

\[ \bar{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} [y_n(x_n, \mathbf{w}) - t_n]^2 + \frac{\lambda}{2} |\mathbf{w}|^2 \]

Penalize large coefficient values
Regularization: $\ln \lambda = 0$

Polynomial Coefficients

<table>
<thead>
<tr>
<th>$\ln \lambda = -\infty$</th>
<th>$\ln \lambda = -18$</th>
<th>$\ln \lambda = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1$</td>
<td>232.37</td>
<td>4.74</td>
</tr>
<tr>
<td>$w_2$</td>
<td>-53221.83</td>
<td>-0.77</td>
</tr>
<tr>
<td>$w_3$</td>
<td>-49546.31</td>
<td>-0.97</td>
</tr>
<tr>
<td>$w_4$</td>
<td>-231639.30</td>
<td>-3.89</td>
</tr>
<tr>
<td>$w_5$</td>
<td>640042.26</td>
<td>55.28</td>
</tr>
<tr>
<td>$w_6$</td>
<td>-1061800.52</td>
<td>41.32</td>
</tr>
<tr>
<td>$w_7$</td>
<td>1012000.18</td>
<td>-45.05</td>
</tr>
<tr>
<td>$w_8$</td>
<td>-557682.99</td>
<td>-91.53</td>
</tr>
<tr>
<td>$w_9$</td>
<td>125201.43</td>
<td>72.68</td>
</tr>
</tbody>
</table>

Regularization: $E_{\text{RMS}}$ vs. $\ln \lambda$

Acknowledgements

- Some of the material in the decision trees presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:
  - [http://www.cs.cmu.edu/~awm/tutorials](http://www.cs.cmu.edu/~awm/tutorials)
- Improved by
  - Carlos Guestrin &
  - Luke Zettlemoyer