DECISION TREES

Slide examples from Andrew Moore and David Sontag
A LEARNING EXAMPLE: PREDICT FUEL EFFICIENCY

- 8 features / attributes
- we want to predict the “mpg” variable
- This example is taken from the UCI Machine Learning Repository

<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>make</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>high</td>
<td>low</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>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>medium</td>
<td>high</td>
<td>low</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>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>70to74</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>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>medium</td>
<td>high</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>low</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</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>
BRANCHING ON FEATURES

- Each non-leaf node tests a feature $x_i$.
- Each branch corresponds to a feature value $x_i = v$.
- Each leaf node is a class label assignment.
- The labeling function that is learned, i.e., $f: X \rightarrow Y$, is the tree itself!
- ... and each labeling instance, i.e., $f(x) = y$ is a tree traversal from the root node to a leaf.
- The function is easily interpretable by humans!

Diagram:

- Cylinders
  - 3 good
  - 4 bad
  - 5 bad
  - 6 bad

- Maker
  - America: bad
  - Asia: good
  - Europe: good

- Horsepower
  - Low: bad
  - Med: good
  - High: bad
HYPOTHESIS SPACE

- How many total possible hypotheses are there?
- What type of functions can be represented by such decision trees?
THE XOR PROBLEM IN MACHINE LEARNING

Linear classifiers cannot deal with the XOR distribution across two or more classes.
WHAT FUNCTIONS CAN BE REPRESENTED BY DECISION TREES?

• Short answer … ANY function!

• For boolean functions, this is easy to interpret, since a path to a leaf node is simply a conjunction (think of it as a specific row in the truth table)

• But this may lead to a tree with exponentially many nodes.
  • $M$ features with $k$ possible values leads to a tree with $k^M$ nodes!
• How many total possible hypotheses are there?
• What type of functions can be represented by such decision trees?
• Which functions are consistent with the given data?
• Which one is the best??
• Is this a good tree?
• It’s correct on 22 points, and incorrect on 18 (the actual dataset contains 40 points)
mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
cylinders = 4
cylinders = 5
cylinders = 6
cylinders = 8

0 0
4 17
1 0
8 0
9 1

Predict bad
Predict good
Predict bad
Predict bad
Predict bad
• Take the original dataset
• Partition it according to the values of the attribute we split on
RECURSION

mpg values: bad good

root

22 18
pchance = 0.001

cylinders = 3
0 0 Predict bad

cylinders = 4
4 17 Predict good

cylinders = 5
1 0 Predict bad

cylinders = 6
8 0 Predict bad

cylinders = 8
9 1 Predict bad

Build tree from These records..

Build tree from These records..

Build tree from These records..

Build tree from These records.

Records in which cylinders = 4

Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8
• Recursively build the tree from the points that have 4 cylinders and the maker is Asian.
• … and similarly recursively build for other cases.
mpg values:  bad  good

A full tree

- Root: mpg values
  - Cylinders = 3: Predict bad (p = 0.135)
  - Cylinders = 4: Predict bad (p = 0.085)
  - Cylinders = 5: Predict bad
  - Cylinders = 6: Predict bad
  - Cylinders = 8: Predict bad

- Maker: America
  - Horsepower = low: Predict good (p = 0.894)
  - Horsepower = medium: Predict bad
  - Horsepower = high: Predict bad

- Maker: Asia
  - Horsepower = low: Predict good (p = 0.317)
  - Horsepower = medium: Predict bad
  - Horsepower = high: Predict bad

- Maker: Europe
  - Horsepower = low: Predict good (p = 0.717)
  - Horsepower = medium: Predict bad
  - Horsepower = high: Predict bad

- Model Year: 70-74
  - Acceleration = low: Predict bad
  - Acceleration = medium: Predict bad
  - Acceleration = high: Predict good (p = 0.717)

- Model Year: 75-78
  - Acceleration = low: Predict bad
  - Acceleration = medium: Predict good
  - Acceleration = high: Predict bad

- Model Year: 79-83
  - Acceleration = low: Predict bad
  - Acceleration = medium: Predict bad
  - Acceleration = high: Predict bad
WHICH TREE TO CHOOSE?

- Remember, each decision tree represents a function.
- And there may be multiple functions $f_1, f_2, \ldots, f_k$ such that $f_1(x_i) = f_2(x_i) = \cdots = f_k(x_i) \forall i$
- But not all the corresponding trees will have the same size, of course. For example, consider the function $f(A, B, C) = (A \land B) \lor (\neg A \land C)$
FINDING THE SMALLEST DECISION TREE

• ... is NP-complete, so good luck!!

• Practically speaking, we adopt a greedy heuristic
  • Start from the empty decision tree
  • Split on the next best feature
  • Continue recursively

What's this?
CHOOSING THE “BEST” FEATURE

- Should we split on $X_1$ or $X_2$?

- We want to reduce uncertainty as quickly as possible.

- So use the counts at the leaf nodes to define the distributions. That way, we can quantify the uncertainty.
HOW TO MEASURE UNCERTAINTY?

- A split is considered good if we are more certain about the classification after the split than we were before.
- Completely deterministic (i.e., completely non-probabilistic) is good.
  - This can happen if everything is labeled with one class (say, true) after a split.
- Uniform distribution is the worst that can happen.
  - Think back to the example of an unbiased coin toss.
- Now we need to worry about the distributions that fall somewhere in between these two extremes …

<table>
<thead>
<tr>
<th>P(Y=A) = 1/2</th>
<th>P(Y=B) = 1/4</th>
<th>P(Y=C) = 1/8</th>
<th>P(Y=D) = 1/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(Y=A) = 1/4</td>
<td>P(Y=B) = 1/4</td>
<td>P(Y=C) = 1/4</td>
<td>P(Y=D) = 1/4</td>
</tr>
</tbody>
</table>
\[ P(Y = T) = \frac{5}{6} \]
\[ P(Y = F) = \frac{1}{6} \]
\[ H(Y) = -\frac{5}{6}\log_2 \frac{5}{6} - \frac{1}{6}\log_2 \frac{1}{6} = 0.65 \]
Conditional entropy of a random variable $Y$, given a random variable $X$, is defined as

$$H(Y \mid X) = - \sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

For $X_1$ and $X_2$:

$$H(Y \mid X_1) = - \frac{4}{6} (1 \times \log_2 1 + 0 \times \log_2 0) - \frac{2}{6} \left( \frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2} \right)$$

$$= \frac{2}{6}$$
INFORMATION GAIN

• Information gain is the decrease in entropy (after a split)

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

• In our example, \( IG(X_1) = 0.65 - 0.33 = 0.32 \)

• When information gain is positive, we prefer to make the split in our decision tree.
LEARNING A DECISION TREE

- Start from the empty decision tree
- Split on the next best feature
  - That is,
  $$\text{argmax}_i IG(X_i) = \text{argmax}_i H(Y) - H(Y|X_i)$$
- Continue recursively

Using Information Gain
• Getting back to predicting MPG …
• First, we look at ALL the information gains, i.e., $IG(X_i)$ for each feature $X_i$
• Turns out, “cylinders” has the highest information gain at $\sim 0.507$

So we get this decision stump. But when do we stop??
Don’t split a node if all matching records have the same output value
mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad
pchance = 0.135

Predict bad

maker = america
0 10
Predict good
pchance = 0.317

maker = asia
2 5
Predict good
pchance = 0.894

maker = europe
2 2
Predict bad
pchance = 0.717

horspower = low
0 4
Predict good
pchance = 0.894

horspower = medium
2 1
Predict bad
pchance = 0.085

horspower = high
0 0
Predict bad
pchance = 0.717

acceleration = low
1 0
Predict bad
pchance = 0.085

acceleration = medium
1 1
(unexpandable)

acceleration = high
0 0
Predict bad
pchance = 0.717

modelyear = 70to74
0 1
Predict bad
pchance = 0.717

modelyear = 75to78
1 0
Predict good
pchance = 0.085

modelyear = 79to83
0 0
Predict bad
pchance = 0.717

Base Case
Two

Don’t split a node if data points are identical on remaining attributes
Base Case Two: No attributes can distinguish
BASE CASES: CONCLUSION

• Case I
  • If all records have the same output, then don’t continue with recursion

• Case II
  • If all records have exactly the same set of input features, then don’t continue with recursion

• Case III
  • If all attributes/features have zero information gain, then don’t continue with recursion

This may not be such a good idea!
THE THIRD BASE CASE

\[ y = a \text{ XOR } b \]

The information gains:

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>blue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>blue</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>red</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>red</td>
<td></td>
</tr>
</tbody>
</table>

The resulting decision tree:

y values: 0 1
root
2 2
Predict 0
... AND WITHOUT THE THIRD BASE CASE
BUILDING A DECISION TREE: RECAP

**BuildTree**($D$, Output)

IF all output from dataset $D$ have the same value $v$

RETURN a single node predicting $v$

IF all input values are the same

RETURN a single node predicting the majority output

$X_{max} \leftarrow$ attribute with highest information gain

$x_1, x_2, ..., x_N \leftarrow$ distinct values of $X_{max}$

create a node $NODE$ with $N$ children

FOR $i \leftarrow 1 \ldots N$

$D_i \leftarrow \{\text{records in } D \text{ where value}(X) = x_i\}$

$i^{th}$ child of $NODE \leftarrow \text{BuildTree}(D_i$, Output)
OVERFITTING

- In the MPG dataset, it turns out that the test set error is much higher than the training set error.
- The standard decision tree learning builds a tree by growing each branch just deep enough to perfectly classify the training examples (i.e., training error will be zero).
- When there is noise in the data (always!) or the training sample is too small to represent the actual function, this simple algorithm produces trees that overfit.
- If we introduce a bias towards shorter trees, we can overcome some of this overfitting.
- Strategies for picking simpler, shorter trees:
  - Fixed maximum depth
  - Fixed maximum number of leaves
OVERFITTING

![Accuracy vs Size of Tree](chart.png)

- **On training data**
- **On test data**

Accuracy vs Size of tree (number of nodes)
CONTINUOUS VARIABLES

- If we have continuous variables like, say, real-values inputs …

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>model_year</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4</td>
<td>97</td>
<td>75</td>
<td>2265</td>
<td>18.2</td>
<td>77</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>199</td>
<td>90</td>
<td>2648</td>
<td>15</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>121</td>
<td>110</td>
<td>2600</td>
<td>12.8</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>350</td>
<td>175</td>
<td>4100</td>
<td>13</td>
<td>73</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>198</td>
<td>95</td>
<td>3102</td>
<td>16.5</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>108</td>
<td>94</td>
<td>2379</td>
<td>16.5</td>
<td>73</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>113</td>
<td>95</td>
<td>2228</td>
<td>14</td>
<td>71</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>302</td>
<td>139</td>
<td>3570</td>
<td>12.8</td>
<td>78</td>
<td>america</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>120</td>
<td>79</td>
<td>2625</td>
<td>18.6</td>
<td>82</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>455</td>
<td>225</td>
<td>4425</td>
<td>10</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>107</td>
<td>86</td>
<td>2464</td>
<td>15.5</td>
<td>76</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>131</td>
<td>103</td>
<td>2830</td>
<td>15.9</td>
<td>78</td>
<td>europe</td>
</tr>
</tbody>
</table>

Infinite number of possible split values!!!

Finite dataset, only finite number of relevant splits!
CONTINUOUS VARIABLES

• Should we create one branch for every numeric value?
  • NO! That will overfit on any dataset!!

• A better idea is to split on threshold values:

• Binary tree split
  • Split on a feature $X$ at value $t$
  • One branch for $X < t$
  • Other branch for $X \geq t$

• But how do we find that value?

```
Year
 <78    ≥78
  
<70   ≥70
  
bad    good
```
• There are too many (maybe infinite) possible values of $t$
• But only a relatively small finite number of those values are important.
• In fact, we should look only at those values that split between different classes

![Diagram](https://via.placeholder.com/150)

• Sort the data according to the different values of $X$, say, \{${\color{green}x_1, x_2, \ldots, x_m}$\}
• Consider split points of the type $x_i + \frac{x_{i+1} - x_i}{2}$
SPLITTING THRESHOLD

• Let $IG(Y|t)$ denote the information gain for $Y$ when testing if $X > t$ or $X < t$

• Define the following:

$$H(Y|t) = p(X < t)H(Y|X < t) + p(X \geq t)H(Y|X \geq t)$$

$$IG(Y|t) = H(Y) - H(Y|t)$$

$$IG^*(Y|X) = \max_t IG(Y|t)$$

• When building a decision tree with continuous variables, use $IG^*(Y|X)$
EXAMPLE DECISION TREE WITH CONTINUOUS VARIABLE IN THE MPG DATASET

mpg values: bad  good

root
22  18
pchance = 0.000

cylinders < 5
4  17
pchance = 0.001

cylinders >= 5
18  1
pchance = 0.003

horsepower < 94
1  17
pchance = 0.274
Predict bad

horsepower >= 94
3  0
acceleration < 19
18  0
Predict bad
acceleration >= 19
0  1
Predict good

maker = america
0  10
Predict good

maker = asia
0  5
Predict good

maker = europe
1  2
pchance = 0.270

displacement < 116
0  2
Predict good

displacement >= 116
1  0
Predict bad
MISSING FEATURE VALUES

• In some cases, certain attributes or features may not have values for a few data items
• E.g., in the Emergency Room, patients come in with some symptoms. Some of these patients may have CBC results, others may not.
• If we have an input-output pair \( \langle x, c(x) \rangle \) with \( x \) missing the value of an attribute \( A \)

• Strategy 1
  • assign it the value most common among training examples at the node where we are splitting on \( A \)

• Strategy 2
  • assign it the most common value among training examples that are labeled \( c(x) \)

• Strategy 3
  • assign a probability to each of the possible values
  • the probability is computed based on observed frequencies of the values of \( A \) among the examples at that node