Decision Trees
Playing Tennis

- Is it raining out?
  - Probably shouldn’t play
- Is it really hot?
  - Yes: Maybe, is it also windy?
    - If yes, sure!
    - Otherwise, I’ll pass
  - No: Sounds like a nice day, let’s play!

Image Credit: Will Povell (TA ‘17)
Decision Trees

- Modelled after flowcharts
- Main idea: Ask Yes/No questions until you learn enough to make a decision
- 20 questions: Is it bigger than a breadbox?
- Strategy: What questions should you ask first?
Which is the best predictor?

<table>
<thead>
<tr>
<th>Rainy?</th>
<th>Temp</th>
<th>Windy?</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cold</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
The data say never play tennis on Rainy days, and usually play on Sunny days.

Decision Heuristic: **Majority vote**
- None of the Rainy observations are classified incorrectly.
- But 33\% of the Sunny observations are classified incorrectly.
The data say sometimes play tennis on Hot days, and don’t usually play on Cold days.

Decision Heuristic: Majority vote
- 50% of the Hot observations are classified incorrectly.
- 33% of the Cold observations are classified incorrectly.
The data say sometimes play tennis on Windy days, and don’t usually play on non-Windy days.

Decision Heuristic: **Majority vote**
- 50% of the Windy observations are classified incorrectly.
- 33% of the Not Windy observations are classified incorrectly.
Measure of Impurity

We can calculate the misclassification error at a node after a split, assuming majority vote, using the mode:

- \( \max(p_1, p_2) \) are classified correctly
- \( 1 - \max(p_1, p_2) \) are classified incorrectly

Here, \( p_1 \) and \( p_2 \) are the percentages in class 1 (YES) and 2 (NO), respectively.

In binary classification, assuming majority vote, \( 1 - \max(p_1, p_2) \leq 0.5 \).
To ask good questions, minimize impurity!

- $p_1 = \%$ belonging to class 1
- $p_2 = \%$ belonging to class 2

GOOD:
- Error = $1 - \max(100\%, 0\%) = 1 - 1 = 0$

BAD:
- Error = $1 - \max(50\%, 50\%) = 1 - 0.5 = 0.5$

Note that the error at both GOOD nodes and the error at both BAD nodes are equal, although of course the errors at the GOOD nodes and the errors at the BAD nodes differ.
To ask good questions, minimize impurity!

Error(Left) = 1 - max(67%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0

Weight(Left) = 60%
Weight(Right) = 40%

Weighted Error = (0.33 x 0.6) + (0.0 x 0.4) = 0.2
<table>
<thead>
<tr>
<th>Weather</th>
<th>Temp</th>
<th>Windy?</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cold</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Which is the best predictor?**

<table>
<thead>
<tr>
<th></th>
<th>Play</th>
<th>Don’t Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>67%</td>
<td>33%</td>
</tr>
<tr>
<td>Rainy</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Hot</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Cold</td>
<td>33%</td>
<td>67%</td>
</tr>
<tr>
<td>Windy</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Not Windy</td>
<td>33%</td>
<td>67%</td>
</tr>
</tbody>
</table>
Error(Left) = 1 - max(66%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0
Weight(Left) = 0.6, Weight(Right) = 0.4
I = (0.6 * 0.33) + (0.4 * 0.0) = 0.2
Error(Left) = 1 - max(66%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0
Weight(Left) = 0.4, Weight(Right) = 0.6
I = (0.4 * 0.33) + (0.6 * 0.0) = 0.2

Error(Left) = 1 - max(50%, 50%) = 0.5
Error(Right) = 1 - max(33%, 66%) = 0.33
Weight(Left) = 0.6, Weight(Right) = 0.4
I = (0.6 * 0.33) + (0.4 * 0.0) = 0.4

Error(Left) = 1 - max(50%, 50%) = 0.5
Error(Right) = 1 - max(33%, 66%) = 0.33
Weight(Left) = 0.4, Weight(Right) = 0.6
I = (0.4 * 0.5) + (0.6 * 0.33) = 0.4
The Algorithm

- Start at the root of the tree, with all observations
- Score all the questions using current set of observations
- Split current set of observations by the question with the best score
- Repeat until all sets of observations are contained in just one class, or all observations’ feature values are identical
- Classify by walking the tree according to the feature values of a new observation
Pros and Cons of Decision Trees

Pros:
- Interpretable
- Suitable for both quantitative and categorical features
- Suitable for both quantitative and categorical labels (i.e., regression)
  (Regression trees are up next!)

Cons:
- Low bias
- High variance: a small change in the training data leads to a very different tree
- Easy to overfit!
## Classifying Mammals vs. Non-mammals

### Training data

<table>
<thead>
<tr>
<th>Name</th>
<th>Body Temperature</th>
<th>Gives Birth</th>
<th>Four-legged</th>
<th>Hibernates</th>
<th>Class Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>porcupine</td>
<td>warm-blooded</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>cat</td>
<td>warm-blooded</td>
<td>yes</td>
<td>no</td>
<td>no*</td>
<td>no*</td>
</tr>
<tr>
<td>bat</td>
<td>warm-blooded</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no*</td>
</tr>
<tr>
<td>whale</td>
<td>warm-blooded</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>salamander</td>
<td>cold-blooded</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>komodo dragon</td>
<td>cold-blooded</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>python</td>
<td>cold-blooded</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>salmon</td>
<td>cold-blooded</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>eagle</td>
<td>warm-blooded</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>guppy</td>
<td>cold-blooded</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

### Test data

<table>
<thead>
<tr>
<th>Name</th>
<th>Body Temperature</th>
<th>Gives Birth</th>
<th>Four-legged</th>
<th>Hibernates</th>
<th>Class Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>warm-blooded</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>pigeon</td>
<td>warm-blooded</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>elephant</td>
<td>warm-blooded</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>leopard shark</td>
<td>cold-blooded</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>turtle</td>
<td>cold-blooded</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>penguin</td>
<td>cold-blooded</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>eel</td>
<td>cold-blooded</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>dolphin</td>
<td>warm-blooded</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>spiny anteater</td>
<td>warm-blooded</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>gila monster</td>
<td>cold-blooded</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

*Note: mislabeled data marked with an asterisk.*
Overfitting

Fits training data perfectly; accuracy is poor on test data!

Accuracy is less than perfect on training data; but tree is simpler, and accuracy is better on test data!
Overfitting
Key Design Decisions

What is the best size for the tree?

● Pre-pruning
  ○ Stop growing the tree when:
    ■ its depth reaches some threshold
    ■ there are fewer than some threshold number of observations at a node
    ■ when the impurity measure no longer decreases by “enough”
      ● But how much is “enough”? It is difficult, if not impossible, to know.

● Post-pruning
  ○ Replace small subtrees with leaf nodes
  ○ Determine class by majority vote among observations in the subtree
Model Selection

Find a model that appropriately balances complexity and generalization capabilities: i.e., that optimizes the bias-variance tradeoff.

- **High bias, low variance**
  - Set some minimum node size, only adding predictors whose children are big enough
  - Set some minimum improvement threshold, only adding predictors that are good enough

- **Low bias, high variance**
  - Trees of unlimited depth and (no minimum) node size
Other complications

- How to fill in missing values
- How to split on numerical values
  - Temperature is in degrees rather than hot vs cold
  - One option is to bin values (＞75 vs. ＜75)
- How to handle noisy labels: identical observations with different labels
Decision Trees in R
Decision Trees in R

- R provides the `rpart` package for decision trees
- The package create trees as follows:
  ```r
  > library(rpart)
  > fit <- rpart(city ~ elevation + beds + bath + sqft)
  ```
- Trees can be visualized using the `rpart.plot` library:
  ```r
  > library(rpart.plot)
  > rpart.plot(fit, type = "class")
  ```
Controlling \texttt{rpart} models

- Often, we can improve performance by tweaking parameters

- \texttt{rpart.control} provides these parameters for decision trees
  - \texttt{minsplit} is the minimum number of observations that must exist to split
  - \texttt{minbucket} is the minimum number of observations that must exist in each leaf
  - \texttt{maxdepth} is the maximum depth of the decision tree
  - \texttt{xval} is the number of cross validations to perform

- Usage:
  
  \begin{verbatim}
  fit <- rpart(y ~ x, data = frame,
               control = rpart.control(maxdepth = 10))
  \end{verbatim}
Missing data

- Sometimes certain features will be missing in the training data
- \texttt{rpart} automatically handles missing data using surrogate splits
- Surrogates are fake values that \texttt{rpart} substitutes for NAs
- They are used on missing test data, as well as missing training data
- It is sometimes difficult to find appropriate surrogates for missing data
A decision tree for *iris*
A decision tree for *iris*
Misclassification error for the decision tree for *iris*

- 22 irises that have been misclassified
- 22/150 irises misclassified = 14.67% misclassification rate
Extras

Adapted from Visual Intro to Machine Learning
San Francisco and New York

- Data on NYC and SF apartments
- **Green** = SF & **Blue** = NYC
- We can look at a scatterplot matrix, a set of scatterplots comparing different features
- We can already see some patterns in the data
- Let’s look at elevation on its own first
Elevation

- NYC is lower than SF
- We could pick a convenient point, like the highest NYC house at 73m, and classify using that
- Houses above 73m are in SF, below 73m are in NYC
Elevation

- NYC < 73m, SF > 73m

\[
preds <- \text{ifelse}(\text{apartments}\$elevation <= \text{breakpoint}, \ "NYC", \ "SF")
\]

- Accuracy on training data is only 63%
- Barely better than guessing
• We classify all houses about 73m correctly, but misclassify a lot below that height, called “false negatives”
• If we split on a lower height, we then misclassify many NYC homes, but we could get better accuracy overall
Elevation

- We accept some false positives, incorrectly classified NYC homes, in order to get a better overall error rate
- We can still improve our accuracy
After the split

- Here are histograms for each side of the split, lower elevation on the left and higher elevation on the right
- We can see more patterns arising in these additional features
- What if we kept splitting?
Split all the things!

- We’ve partitioned our data once, why not split on different features?
- For low elevation houses, splitting on price per square foot gives the best results; same as price for high elevation houses
Just keep splitting

- For each split, we keep splitting and eventually make a decision tree