Today

- Formalizing Learning
  - Consistency
  - Simplicity

- Decision Trees
  - Expressiveness
  - Information Gain
  - Overfitting
Inductive Learning (Science)

- Simplest form: learn a function from examples
  - A target function: \( f \)
  - Examples: input-output pairs \((x, f(x))\)
  - E.g. \( x \) is an email and \( f(x) \) is spam / ham
  - E.g. \( x \) is a house and \( f(x) \) is its selling price

- Problem:
  - Given a hypothesis space \( H \)
  - Given a training set of examples \( x_i \)
  - Find a hypothesis \( h(x) \) such that \( h \sim f \)

- Includes:
  - Classification (multinomial outputs)
  - Regression (real outputs)

- How do perceptron and naïve Bayes fit in? \((H, f, h, \text{etc.})\)

---

Inductive Learning

- Curve fitting (regression, function approximation):

- Consistency vs. simplicity
- Ockham’s razor
Consistency vs. Simplicity

- Fundamental tradeoff: bias vs. variance, etc.
- Usually algorithms prefer consistency by default (why?)
- Several ways to operationalize “simplicity”
  - Reduce the hypothesis space
    - Assume more: e.g. independence assumptions, as in naïve Bayes
    - Have fewer, better features / attributes: feature selection
    - Other structural limitations (decision lists vs trees)
  - Regularization
    - Smoothing: cautious use of small counts
    - Many other generalization parameters (pruning cutoffs today)
    - Hypothesis space stays big, but harder to get to the outskirts

Reminder: Features

- Features, aka attributes
  - Sometimes: TYPE=French
  - Sometimes: \( f_{\text{TYPE=French}}(x) = 1 \)

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>( E_{\text{at}} )</th>
<th>Will/Won</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 )</td>
<td>0-10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>( X_2 )</td>
<td>30-60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_3 )</td>
<td>0-10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>( X_4 )</td>
<td>10-30</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>( X_5 )</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_6 )</td>
<td>0-10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>( X_7 )</td>
<td>0-10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_8 )</td>
<td>0-10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>( X_9 )</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_{10} )</td>
<td>10-30</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_{11} )</td>
<td>0-10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>( X_{12} )</td>
<td>30-60</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Decision Trees

- Compact representation of a function:
  - Truth table
  - Conditional probability table
  - Regression values

- True function
  - Realizable: in $H$

Expressiveness of DTs

- Can express any function of the features

\[ P(C|A, B) \]

- However, we hope for compact trees
Comparison: Perceptrons

- What is the expressiveness of a perceptron over these features?

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$T$ $F$ $F$ $T$ $\text{Some}$ $SSS$ $F$ $T$ French $0-10$</td>
<td>$T$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$T$ $F$ $F$ $T$ $\text{Full}$ $$ $F$ $F$ Thai $30-60$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

- DTs automatically conjoin features / attributes
  - Features can have different effects in different branches of the tree!
- For a perceptron, a feature’s contribution is either positive or negative
  - If you want one feature’s effect to depend on another, you have to add a new conjunction feature
  - E.g., adding “PATRONS=full ∧ WAIT = 60” allows a perceptron to model the interaction between the two atomic features
- Difference between modeling relative evidence weighting (NB) and complex evidence interaction (DTs)
  - Though if the interactions are too complex, may not find the DT greedily

Hypothesis Spaces

- How many distinct decision trees with $n$ Boolean attributes?
  - number of Boolean functions over $n$ attributes
  - number of distinct truth tables with $2^n$ rows
  - $2^{2^n}$
  - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

- How many trees of depth 1 (decision stumps)?
  - number of Boolean functions over 1 attribute
  - number of truth tables with 2 rows, times $n$
  - $4n$
  - E.g. with 6 Boolean attributes, there are 24 decision stumps

- More expressive hypothesis space:
  - Increases chance that target function can be expressed (good)
  - Increases number of hypotheses consistent with training set (bad, why?)
  - Means we can get better predictions (lower bias)
  - But we may get worse predictions (higher variance)
Decision Tree Learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

```python
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default;
    else if all examples have the same classification then return the classification
    else if attributes is empty then return Mode(examples)
    else
        best = CHOOSE-ATTRIBUTE(attributes, examples)
        tree = a new decision tree with root test best
        for each value v_i of best do
            examples_i = {elements of examples with best = v_i}
            subtree = DTL(examples_i, attributes - best, Mode(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”

- So: we need a measure of how “good” a split is, even if the results aren’t perfectly separated out
Entropy and Information

- **Information answers questions**
  - The more uncertain about the answer initially, the more information in the answer
  - Scale: bits
    - Answer to Boolean question with prior <1/2, 1/2>?
    - Answer to 4-way question with prior <1/4, 1/4, 1/4, 1/4>?
    - Answer to 4-way question with prior <0, 0, 1>?
    - Answer to 3-way question with prior <1/2, 1/4, 1/4>?

- **A probability p is typical of:**
  - A uniform distribution of size 1/p
  - A code of length log 1/p

Entropy

- **General answer: if prior is \(<p_1, \ldots, p_n>\):**
  - Information is the expected code length
  \[
  H(\langle p_1, \ldots, p_n \rangle) = E_p \log_2 \frac{1}{p_i} = \sum_{i=1}^{n} -p_i \log_2 p_i
  \]

  - Also called the entropy of the distribution
    - More uniform = higher entropy
    - More values = higher entropy
    - More peaked = lower entropy
    - Rare values almost “don’t count”
Information Gain

- Back to decision trees!
- For each split, compare entropy before and after
  - Difference is the information gain
  - Problem: there’s more than one distribution after split!

Solution: use expected entropy, weighted by the number of examples
- Note: hidden problem here! Gain needs to be adjusted for large-domain splits – why?

Next Step: Recurse

- Now we need to keep growing the tree
- Two branches are done (why?)
- What to do under “full”?  
  - See what examples are there…
Example: Learned Tree

- Decision tree learned from these 12 examples:

  ![Decision Tree Diagram]

- Substantially simpler than “true” tree
  - A more complex hypothesis isn’t justified by data
  - Also: it’s reasonable, but wrong

Example: Miles Per Gallon

<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>high</td>
<td>low</td>
<td>75078</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>70074</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75078</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>high</td>
<td>70074</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>70074</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70074</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75078</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>79083</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>79083</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79083</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>70074</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>75078</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75078</td>
<td>europe</td>
</tr>
</tbody>
</table>

40 Examples
Find the First Split

- Look at information gain for each attribute
- Note that each attribute is correlated with the target!
- What do we split on?

Result: Decision Stump

mpg values: bad good

root

22 18

pchange = 0.001

cylinders = 3

0 0

Predict bad

Predict good

cylinders = 4

4 17

Predict bad

Predict good

cylinders = 5

1 0

Predict bad

Predict good

cylinders = 6

8 0

Predict bad

Predict good

cylinders = 8

9 1

Predict bad
Reminder: Overfitting

- Overfitting:
  - When you stop modeling the patterns in the training data (which generalize)
  - And start modeling the noise (which doesn’t)

- We had this before:
  - Naïve Bayes: needed to smooth
  - Perceptron: didn’t really say what to do about it (stay tuned!)

---

**MPG Training Error**

<table>
<thead>
<tr>
<th>Num Errors</th>
<th>Set Size</th>
<th>Percent Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Test Set</td>
<td>74</td>
<td>352</td>
</tr>
</tbody>
</table>

The test set error is much worse than the training set error... ...why?
Consider this split

Significance of a Split

- Starting with:
  - Three cars with 4 cylinders, from Asia, with medium HP
  - 2 bad MPG
  - 1 good MPG

- What do we expect from a three-way split?
  - Maybe each example in its own subset?
  - Maybe just what we saw in the last slide?

- Probably shouldn't split if the counts are so small they could be due to chance

- A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance (details in the book)

- Each split will have a significance value, \( p_{\text{CHANCE}} \)
Keeping it General

- **Pruning:**
  - Build the full decision tree
  - Begin at the bottom of the tree
  - Delete splits in which $p_{\text{CHANCE}} > \text{MaxP}_{\text{CHANCE}}$
  - Continue working upward until there are no more prunable nodes
  - Note: some chance nodes may not get pruned because they were "redeemed" later

Pruning example

- With MaxP$_{\text{CHANCE}} = 0.1$:

<table>
<thead>
<tr>
<th>mpg values:</th>
<th>bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinders = 3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cylinders = 4</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>cylinders = 5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cylinders = 6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>cylinders = 8</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Num Errors</th>
<th>Set Size</th>
<th>Percent Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>5</td>
<td>40</td>
<td>12.50</td>
</tr>
<tr>
<td>Test Set</td>
<td>56</td>
<td>352</td>
<td>15.91</td>
</tr>
</tbody>
</table>

Note the improved test set accuracy compared with the unpruned tree
Regularization

- $\text{MaxP}_{\text{CHANCE}}$ is a regularization parameter
- Generally, set it using held-out data (as usual)

Two Ways of Controlling Overfitting

- Limit the hypothesis space
  - E.g. limit the max depth of trees
  - Easier to analyze (coming up)

- Regularize the hypothesis selection
  - E.g. chance cutoff
  - Disprefer most of the hypotheses unless data is clear
  - Usually done in practice
Learning Curves

- Another important trend:
  - More data is better!
  - The same learner will generally do better with more data
  - (Except for cases where the target is absurdly simple)

Summary

- Formalization of learning
  - Target function
  - Hypothesis space
  - Generalization

- Decision Trees
  - Can encode any function
  - Top-down learning (not perfect!)
  - Information gain
  - Bottom-up pruning to prevent overfitting