Today 11/2

• Machine learning
  - Review Naïve Bayes
  - Decision Trees
  - Decision Lists
**Where we are**

- Agents can
  - Search
  - Represent stuff
  - Reason logically
  - Reason probabilistically
- Left to do
  - Learn
  - Communicate

**Connections**

- As we’ll see there’s a strong connection between
  - Search
  - Representation
  - Uncertainty
- You should view the ML discussion as a natural extension of these previous topics
Connections

- More specifically
  - The representation you choose defines the space you search
  - How you search the space and how much of the space you search introduces uncertainty
  - That uncertainty is captured with probabilities

Supervised Learning: Induction

- General case:
  - Given a set of pairs \((x, f(x))\) discover the function \(f\).
- Classifier case:
  - Given a set of pairs \((x, y)\) where \(y\) is a label, discover a function that correctly assigns the correct labels to the \(x\).
Supervised Learning: Induction

• Simpler Classifier Case:
  - Given a set of pairs (x, y) where x is an object and y is either a + if x is the right kind of thing or a - if it isn’t. Discover a function that assigns the labels correctly.

Learning as Search

• Everything is search...
  - A hypothesis is a guess at a function that can be used to account for the inputs.
  - A hypothesis space is the space of all possible candidate hypotheses.
  - Learning is a search through the hypothesis space for a good hypothesis.
What Are These Objects

• By object, we mean a logical representation.
  - Normally, simpler representations are used that consist of fixed lists of feature-value pairs.
• A set of such objects paired with answers, constitutes a training set.

Naïve-Bayes Classifiers

• Argmax \( P(\text{Label} \mid \text{Object}) \)

\[
P(\text{Label} \mid \text{Object}) = \frac{P(\text{Object} \mid \text{Label}) \cdot P(\text{Label})}{P(\text{Object})}
\]

• Where Object is a feature vector.
Naïve Bayes

- Ignore the denominator
- $P(\text{Label})$ is just the prior for each class. I.e., The proportion of each class in the training set
- $P(\text{Object}|\text{Label}) = ???$
  - The number of times this object was seen in the training data with this label divided by the number of things with that label.

Nope

- Too sparse, you probably won’t see enough examples to get numbers that work.
- Answer
  - Assume the parts of the object are independent so $P(\text{Object}|\text{Label})$ becomes

$$
\prod P(\text{Feature} = \text{Value} | \text{Label})
$$
Training Data

<table>
<thead>
<tr>
<th>#</th>
<th>F1 (In/Out)</th>
<th>F2 (Meat/Veg)</th>
<th>F3 (Red/Green/Blue)</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>Meat</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Out</td>
<td>Meat</td>
<td>Red</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Out</td>
<td>Veg</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>

Example

- \( P(Yes) = \frac{3}{4}, P(No) = 1/4 \)
- \( P(F1=\text{In}|Yes) = \frac{4}{6} \)
- \( P(F1=\text{Out}|Yes) = \frac{2}{6} \)
- \( P(F2=\text{Meat}|Yes) = \frac{3}{6} \)
- \( P(F2=\text{Veg}|Yes) = \frac{3}{6} \)
- \( P(F3=\text{Red}|Yes) = \frac{4}{6} \)
- \( P(F3=\text{Green}|Yes) = \frac{2}{6} \)
- \( P(F1=\text{In}|No) = 0 \)
- \( P(F1=\text{Out}|No) = 1 \)
- \( P(F2=\text{Meat}|No) = 1/2 \)
- \( P(F2=\text{Veg}|No) = 1/2 \)
- \( P(F3=\text{Red}|No) = 1/2 \)
- \( P(F3=\text{Green}|No) = 1/2 \)
Example

- In, Meat, Green
  - First note that you’ve never seen this before
  - So you can’t use stats on In, Meat, Green since you’ll get a zero for both yes and no.

Example: In, Meat, Green

- \( P(\text{Yes} | \text{In, Meat, Green}) = \frac{P(\text{In} | \text{Yes}) P(\text{Meat} | \text{Yes}) P(\text{Green} | \text{Yes}) P(\text{Yes})}{P(\text{Yes})} \)

- \( P(\text{No} | \text{In, Meat, Green}) = \frac{P(\text{In} | \text{No}) P(\text{Meat} | \text{No}) P(\text{Green} | \text{No}) P(\text{No})}{P(\text{No})} \)

Remember we’re dumping the denominator since it can’t matter.
Naïve Bayes

- This technique is always worth trying first.
  - It's easy
  - Sometimes it works well enough
  - When it doesn't, it gives you a baseline to compare more complex methods to

Decision Trees

- A decision tree is a tree where
  - Each internal node of the tree tests a single feature of an object
  - Each branch follows a possible value of each feature
  - The leaves correspond to the possible labels on the objects
  - DTs easily handle multiclass labeling problems.
Decision Tree Learning

- Given a training set find a tree that correctly assigns labels (classifies) the elements of the training set.
- Sort of... there might be lots of such trees. In fact some of them look a lot like tables.
Training Set

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Goal</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_1</td>
<td>Yes No No Yes Some $$$$ No Yes French 0±10</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>X_2</td>
<td>Yes No No Yes Full $ No No Thai 30±60</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_3</td>
<td>No Yes No No Some $ No No Burger 0±10</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>X_4</td>
<td>Yes No Yes Yes Full $ No No Thai 10±30</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>X_5</td>
<td>Yes No Yes No Full $$$$ No Yes French &gt;60</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_6</td>
<td>No Yes No Yes Some $$ Yes Yes Italian 0±10</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>X_7</td>
<td>No Yes No No None $ Yes No Burger 0±10</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_8</td>
<td>No No No Yes Some $$ Yes Yes Thai 0±10</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>X_9</td>
<td>No Yes Yes No Full $ Yes No Burger &gt;60</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_10</td>
<td>Yes Yes Yes Yes Full $$$$ No Yes Italian 10±30</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_11</td>
<td>No No No No None $ No No Thai 0±10</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>X_12</td>
<td>Yes Yes Yes Yes Full $ No No Burger 30±60</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Decision Tree Learning

- Start with a null tree.
- Select a feature to test and put it in tree.
- Split the training data according to that test.
- Recursively build a tree for each branch.
- Stop when a test results in a uniform label or you run out of tests.
Well

- What makes a good tree?
  - Trees that cover the training data
  - Trees that are small...
- How should features be selected?
  - Choose features that lead to small trees.
  - How do you know if a feature will lead to a small tree?

Search

- What’s that as a search?
- We want a small tree that covers the training data.
- So... search through the trees in order of size for a tree that covers the training data.
- No need to worry about bigger trees that also cover the data.
Small Trees?

• Small trees are good trees...
  - More precisely, all things being equal we prefer small trees to larger trees.

• Why?
  - Well how many small trees are there compared with larger trees?
  - Lots of big trees, not many small trees.

Small Trees

• Not many small trees, lots of big trees.
  - So odds are less
    • that you'll run across a good looking small tree that turns out bad
    • then a bigger tree that looks good but turns out bad...

CSCI 5582 Fall 2006
What?

• What does looks good, turns out bad mean?
  - It means doing well on the training data and not well on the testing data
• We want trees that work well on both.

Finding Small Trees

• What stops the recursion?
  - Running out of tests (bad).
  - Uniform samples at the leaves
    • To get uniform samples at the leaves, choose features that maximally separate the training instances
Information Gain

• Roughly...
  - Start with a pure guess the majority strategy. If I have a 60/40 split (y/n) in the training, how well will I do if I always guess yes?
  - Ok so now iterate through all the available features and try each at the top of the tree.

Information Gain

• Then guess the majority label in each of the buckets at the leaves. How well will I do?
  - Well it’s the weighted average of the majority distribution at each leaf.
• Pick the feature that results in the best predictions.
Patrons

• Picking Patrons at the top takes the initial 50/50 split and produces three buckets
  - None: 0 Yes, 2 No
  - Some: 4 Yes, 0 No
  - Full: 2 Yes, 4 No
    • That’s 10 right out of 12

Training and Evaluation

• Given a fixed size training set, we need a way to
  - Organize the training
  - Assess the learned system’s likely performance on unseen data
Test Sets and Training Sets

- Divide your data into three sets:
  - Training set
  - Development test set
  - Test set
1. Train on the training set
2. Tune using the dev-test set
3. Test on withheld data

Cross-Validation

- What if you don’t have enough training data for that?
  1. Divide your data into N sets and put one set aside (leaving N-1)
  2. Train on the N-1 sets
  3. Test on the set aside data
  4. Put the set aside data back in and pull out another set
  5. Go to 2
  6. Average all the results

CSCI 5582 Fall 2006
Performance Graphs

- It's useful to know the performance of the system as a function of the amount of training data.

![Performance Graphs](image)

Break

- Quiz is pushed back to Tuesday, November 28.
  - So you can spend Thanksgiving studying.
Decision Lists

- Key parameters:
  - Maximum allowable length of the list
  - Maximum number of elements in a test
  - Logical connectives allowed in the test
- The longer the lists, and the more complex the tests, the larger the hypothesis space.
Decision List Learning

function DECISION-LIST-LEARNING(examples) returns a decision list, No or failure

if examples is empty then return the value No

\( t \leftarrow \) a test that matches a nonempty subset examples, of examples

\( \text{such that the members of examples, are all positive or all negative} \)

if there is no such \( t \) then return failure

if the examples in examples, are positive then \( o \leftarrow \text{Yes} \)

else \( o \leftarrow \text{No} \)

return a decision list with initial test \( t \) and outcome \( o \)

and remaining elements given by DECISION-LIST-LEARNING(examples \( - \) examples )

---

### Training Data

<table>
<thead>
<tr>
<th>#</th>
<th>F1 (In/Out)</th>
<th>F2 (Meat/Veg)</th>
<th>F3 (Red/Green/Blue)</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>Meat</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Out</td>
<td>Meat</td>
<td>Red</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Out</td>
<td>Veg</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Lists

- Let's try
  \[ F_1 = \text{In} \Rightarrow \text{Yes} \]

### Training Data

<table>
<thead>
<tr>
<th>#</th>
<th>F1 (In/Out)</th>
<th>F2 (Meat/Veg)</th>
<th>F3 (Red/Green/Blue)</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>Meat</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Out</td>
<td>Meat</td>
<td>Red</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Out</td>
<td>Veg</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Lists

- \([F1 = \text{In}] \rightarrow \text{Yes}\)
- \([F2 = \text{Veg}] \rightarrow \text{No}\)

Training Data

<table>
<thead>
<tr>
<th>#</th>
<th>F1 (In/Out)</th>
<th>F2 (Meat/Veg)</th>
<th>F3 (Red/Green/Blue)</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>Meat</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Out</td>
<td>Meat</td>
<td>Red</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Out</td>
<td>Veg</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Lists

- \([F_1 = \text{In}] \rightarrow \text{Yes}\)
- \([F_2 = \text{Veg}] \rightarrow \text{No}\)
- \([F_3 = \text{Green}] \rightarrow \text{Yes}\)

Training Data

<table>
<thead>
<tr>
<th>#</th>
<th>(F_1) (In/Out)</th>
<th>(F_2) (Meat/Veg)</th>
<th>(F_3) (Red/Green/Blue)</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>Meat</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>Veg</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Out</td>
<td>Meat</td>
<td>Green</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Out</td>
<td>Meat</td>
<td>Red</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Out</td>
<td>Veg</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Lists

- [F1 = In] → Yes
- [F2 = Veg] → No
- [F3=Green] → Yes
- No

Covering and Splitting

- The decision tree learning algorithm is a splitting approach.
  - The training set is split apart according to the results of a test
  - Until all the splits are uniform
- Decision list learning is a covering algorithm
  - Tests are generated that uniformly cover a subset of the training set
  - Until all the data are covered
Choosing a Test

• What tests should be put at the front of the list?
  - Tests that are simple?
  - Tests that uniformly cover large numbers of examples?
  - Both?

Choosing a Test

• What about choosing tests that only cover small numbers of examples?
  - Would that ever be a good idea?
    • Sure, suppose that you have a large heterogeneous group with one label.
    • And a very small homogeneous group with a different label.
    • You don’t need to characterize the big group, just the small one.
### Decision Lists

- The flexibility in defining the tests and the length of the lists is a big advantage to decision lists.
  - (Decision trees can end up being a bit unwieldy)

### What Does Matter?

- I said that in practical applications the choice of ML technique doesn’t really matter.
- They will all result in the same error rate (give or take)
- So what does matter?
What Matters

• Having the right set of features in the training set
• Having enough training data