Decision Trees

(Most slides by Pedro Domingos)

Reading: CIML chapter 1
Note: These slides contain some information not in the readings!

Warm-up and Review

1. Turn to the person next to you and introduce yourself.
2. If your name come first alphabetically, answer the first question for your neighbor. If your name comes second alphabetically, explain the answer to the second question to your neighbor.
3. If you have time, feel free to answer the last two questions as well.

Q1: What is the difference between the data distribution and the training data?
Q2: What is the difference between validation data and test data?
Review: Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Tune hyperparameters on held-out set
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on test data
  - Overfitting: fitting the training data very closely, but not generalizing well

Outline

- Quick introduction to decision trees
- What you should learn:
  - What decision trees can represent compactly
  - How to learn a decision tree recursively, including choosing attributes
  - What decision trees can learn
  - How decision trees handle overfitting
Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features $x_j$ and branch according to the results of the test.
- **Leaf nodes** specify the class $h(x)$.

Suppose the features are **Outlook** ($x_1$), **Temperature** ($x_2$), **Humidity** ($x_3$), and **Wind** ($x_4$). Then the feature vector $x = (\text{Sunny, Hot, High, Strong})$ will be classified as **No**. The **Temperature** feature is irrelevant.

---

Decision Tree Hypothesis Space

If the features are continuous, internal nodes may test the value of a feature against a threshold.

---
**Decision Tree Decision Boundaries**

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.

![Decision Tree Diagram]

**Decision Trees Can Represent Any Boolean Function**

The tree will in the worst case require exponentially many nodes, however.
Exercise

Consider the following dataset:

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Draw a decision tree that classifies all examples correctly.
- How would your tree classify (0, 1, 1)?

Learning Algorithm for Decision Trees

The same basic learning algorithm has been discovered by many people independently:

\[
\text{GROW_TREE}(S) \\
\text{if } (y = 0 \text{ for all } (x, y) \in S) \text{ return new leaf}(0) \\
\text{else if } (y = 1 \text{ for all } (x, y) \in S) \text{ return new leaf}(1) \\
\text{else} \\
\quad \text{choose best attribute } x_j \\
\quad S_0 = \text{all } (x, y) \in S \text{ with } x_j = 0; \\
\quad S_1 = \text{all } (x, y) \in S \text{ with } x_j = 1; \\
\quad \text{return new node}(x_j, \text{GROW_TREE}(S_0), \text{GROW_TREE}(S_1))
\]
Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

**CHOOSEBESTATTRIBUTE(S)**
choose \( j \) to minimize \( J_j \), computed as follows:
- \( S_0 = \) all \((x, y)\) \( \in S \) with \( x_j = 0 \);
- \( S_1 = \) all \((x, y)\) \( \in S \) with \( x_j = 1 \);
- \( y_0 = \) the most common value of \( y \) in \( S_0 \);
- \( y_1 = \) the most common value of \( y \) in \( S_1 \);
- \( J_0 = \) number of examples \((x, y)\) \( \in S_0 \) with \( y \neq y_0 \);
- \( J_1 = \) number of examples \((x, y)\) \( \in S_1 \) with \( y \neq y_1 \);
- \( J_j = J_0 + J_1 \) (total errors if we split on this feature)

return \( j \)

---

Choosing the Best Attribute—An Example

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- \( J=2 \) (4-4)
- \( J=4 \) (4-4, 4-4)
- \( J=4 \) (4-4)
Choosing the Best Attribute (3)

Unfortunately, this measure does not always work well, because it does not detect cases where we are making “progress” toward a good tree.

(These slides are important and not in CIML.)

A Better Heuristic From Information Theory

Let $V$ be a random variable with the following probability distribution:

\[
\begin{array}{ll}
P(V = 0) & P(V = 1) \\
0.2 & 0.8 \\
\end{array}
\]

The surprise, $S(V = v)$ of each value of $V$ is defined to be

\[
S(V = v) = -\log P(V = v).
\]

An event with probability 1 gives us zero surprise.

An event with probability 0 gives us infinite surprise!

It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.

This is also called the description length of $V = v$.

Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).
Entropy

The entropy of $V$, denoted $H(V)$, is defined as follows:

$$H(V) = \sum_{v \in V} -P(H = v) \log P(H = v).$$

This is the average surprise of describing the result of one “trial” of $V$ (one coin toss).

Entropy can be viewed as a measure of uncertainty.

Mutual Information

Now consider two random variables $A$ and $B$ that are not necessarily independent. The mutual information between $A$ and $B$ is the amount of information we learn about $B$ by knowing the value of $A$ (and vice versa—it is symmetric). It is computed as follows:

$$I(A; B) = H(B) - \sum_b P(B = b) \cdot H(A|B = b)$$

In particular, consider the class $Y$ of each training example and the value of feature $x_1$ to be random variables. Then the mutual information quantifies how much $x_1$ tells us about the value of the class $Y$.

$$H(Y) = 0.9183$$

$$P(x_1=0) = 0.6667$$

$$P(x_1=1) = 0.3333$$

$$H(Y|x_1=0) = 0.9710$$

$$H(Y|x_1=1) = 0.7219$$

$$I(Y;x_1) = 0.0304$$
Learning Parity with Noise

When learning exclusive-or (2-bit parity), all splits look equally good. If extra random boolean features are included, they also look equally good. Hence, decision tree algorithms cannot distinguish random noisy features from parity features.

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Overfitting in Decision Trees

Consider adding a noisy training example:

*Sunny, Hot, Normal, Strong, PlayTennis=No*

What effect on tree?
Overfitting

Consider error of hypothesis \( h \) over
- training data: \( \text{error}_{\text{train}}(h) \)
- entire distribution \( D \) of data: \( \text{error}_D(h) \)

Hypothesis \( h \in H \) overfits training data if there is an alternative hypothesis \( h' \in H \) such that

\[
\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')
\]

and

\[
\text{error}_D(h) > \text{error}_D(h')
\]

Overfitting in Decision Tree Learning

![Graph showing overfitting in decision tree learning. The graph plots accuracy against the size of the tree (number of nodes), with separate lines for training and test data.]
Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select “best” tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

Summary: Decision Trees

- Representation
- Tree growth
- Heuristics
- Overfitting and pruning