Administration

- Registration [Ask NOW]
- **Hw0**: Solution is out; make sure you understand it.
- **Hw1** is out.
  - Please start working on it as soon as possible;
    - Discussion sessions will start next week; come ready with questions
- Projects
  - Small (2-3) groups; your choice of a topic.
  - 25% of the grade → needs to be a substantial project
  - Extra credit for undergrads
- Quiz 1: Avg. score: 4.51/5
  - Only 165 of you attempted it (???)
  - Check out the solution.
- No Quiz in the coming week.
What Did We Learn?

- Learning problem:
  - Find a function that best separates the data
- What function?
- What’s best?
- How to find it?

**Linear:**
- \( x = \text{data representation}; \ w = \text{the classifier} \)
- \( Y = \text{sgn} \{ w^T x \} \)

- A possibility: Define the learning problem to be:
  - Find a (linear) function that best separates the data
Introduction - Summary

- We introduced the technical part of the class by giving two (very important) examples for learning approaches to linear discrimination.
- There are many other solutions.
- **Question 1:** Our solution learns a linear function; in principle, the target function may not be linear, and this will have implications on the performance of our learned hypothesis.
  - Can we learn a function that is more flexible in terms of what it does with the feature space?
- **Question 2:** Can we say something about the quality of what we learn (sample complexity, time complexity; quality)
Earlier, we decoupled the generation of the feature space from the learning.

Argued that we can map the given examples into another space, in which the target functions are linearly separable.

Do we always want to do it?

How do we determine what are good mappings?

The study of decision trees may shed some light on this.

Learning is done directly from the given data representation.

The algorithm `transforms” the data itself.
This Lecture

- Decision trees for (binary) classification
  - Non-linear classifiers

- Learning decision trees (ID3 algorithm)
  - Greedy heuristic (based on information gain)
    Originally developed for discrete features
  - Some extensions to the basic algorithm

- Overfitting
  - Some experimental issues
Think about a large table, $N$ attributes, and assume you want to know something about the people represented as entries in this table.

E.g. own an expensive car or not;

Simplest way: Histogram on the first attribute – own

Then, histogram on first and second (own & gender)

But, what if the # of attributes is larger: $N=16$

How large are the 1-d histograms (contingency tables)? 16 numbers

How large are the 2-d histograms? 16-choose-2 = 120 numbers

How many 3-d tables? 560 numbers

With 100 attributes, the 3-d tables need 161,700 numbers

- We need to figure out a way to represent data in a better way, and figure out what are the important attributes to look at first.
- Information theory has something to say about it – we will use it to better represent the data.
Decision Trees

- A hierarchical data structure that represents data by implementing a divide and conquer strategy.
- Can be used as a non-parametric classification and regression method.
- Given a collection of examples, learn a decision tree that represents it.
- Use this representation to classify new examples.
Decision Trees

The Representation

Decision Trees

Evaluation of a Decision Tree

Learning a Decision Tree
Expressivity of Decision Trees

As Boolean functions they can represent any Boolean function. Can be rewritten as rules in Disjunctive Normal Form (DNF):

- Green^square \rightarrow positive
- Blue^circle \rightarrow positive
- Blue^square \rightarrow positive

The disjunction of these rules is equivalent to the Decision Tree.

What did we show? What is the hypothesis space here?

2 dimensions, 3 values each \( \rightarrow |X| = 9; |Y| = 2; |H| = 2^9 \)
Decision Trees

- Output is a discrete category. Real valued outputs are possible (regression trees).
- There are efficient algorithms for processing large amounts of data (but not too many features).
- There are methods for handling noisy data (classification noise and attribute noise) and for handling missing attribute values.
Decision Boundaries

- Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes
- In this case, the tree divides the features space into axis-parallel rectangles, each labeled with one of the labels

<table>
<thead>
<tr>
<th></th>
<th>Y&lt;5</th>
<th>Y&gt;7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X&lt;1</th>
<th>X&lt;3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Decision Trees
Decision Trees

- Can represent any Boolean Function
- Can be viewed as a way to compactly represent a lot of data.
- Natural representation: (20 questions)
- The evaluation of the Decision Tree Classifier is easy

- Clearly, given data, there are many ways to represent it as a decision tree.
- Learning a good representation from data is the challenge.
Will I play tennis today?

**Features**
- Outlook: \{Sun, Overcast, Rain\}
- Temperature: \{Hot, Mild, Cool\}
- Humidity: \{High, Normal, Low\}
- Wind: \{Strong, Weak\}

**Labels**
- Binary classification task: \(Y = \{+, -\}\)
Will I play tennis today?

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>

**Outlook:** S(unny), O(vercast), R(ainy)

**Temperature:** H(ot), M(edium), C(ool)

**Humidity:** H(igh), N(ormal), L(ow)

**Wind:** S(trong), W(eak)
Registration [Ask NOW]

Hw1 is out. Due on Friday.
- You should be working on it already.
  - You have noticed that the goal of the Hw is to teach you something.
  - Discussion sessions will start next week; come ready with questions.

Projects
- Small (2-3) groups; your choice of a topic.
  - Anything with a significant Machine Learning component works.
  - More details will come.
- 25% of the grade → needs to be a substantial project
- Extra credit for undergrads

Quiz 2: will be made available over the weekend.

Check the website for office hours, discussion sessions etc.
Basic Decision Trees Learning Algorithm

- Data is processed in Batch (i.e. all the data available)
- Recursively build a decision tree top down.

### Data Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Outlook</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>
Basic Decision Tree Algorithm

- Let $S$ be the set of Examples
  - Label is the target attribute (the prediction)
  - Attributes is the set of measured attributes

- ID3($S$, Attributes, Label)

  If all examples are labeled the same return a single node tree with Label

  Otherwise Begin

  $A = \text{attribute in Attributes that } best \text{ classifies } S$ (Create a Root node for tree)

  for each possible value $v$ of $A$

  Add a new tree branch corresponding to $A=v$

  Let $S_v$ be the subset of examples in $S$ with $A=v$

  if $S_v$ is empty: add leaf node with the common value of Label in $S$

  Else: below this branch add the subtree ID3($S_v$, Attributes - {a}, Label)

  End

Return Root
Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam’s Razor)
  - But, finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.
Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

- < (A=0,B=0), - >: 50 examples
- < (A=0,B=1), - >: 50 examples
- < (A=1,B=0), - >: 0 examples
- < (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes.

1  A  0
+  -

1  B  0
+  -

Splitting on B: we don’t get purely labeled nodes.

1  A  0
+  -

What if we have: <(A=1,B=0), - >: 3 examples
Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

- (A=0, B=0), - : 50 examples
- (A=0, B=1), - : 50 examples
- (A=1, B=0), - : 0 examples, 3 examples
- (A=1, B=1), + : 100 examples

Trees looks structurally similar; which attribute should we choose?

Advantage A. But...

Need a way to quantify things
Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam’s Razor).
- The main decision in the algorithm is the selection of the next attribute to condition on.
- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.
- The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.
Entropy (impurity, disorder) of a set of examples, $S$, relative to a binary classification is:

$$\text{Entropy}(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

where $p_+$ is the proportion of positive examples in $S$ and $p_-$ is the proportion of negatives.

- If all the examples belong to the same category: Entropy = 0
- If all the examples are equally mixed (0.5, 0.5): Entropy = 1
- Entropy = Level of uncertainty.

In general, when $p_i$ is the fraction of examples labeled $i$:

$$\text{Entropy}\{p_1, p_2, \ldots, p_k\} = -\sum_{i=1}^{k} p_i \log(p_i)$$

Entropy can be viewed as the number of bits required, on average, to encode the class of labels. If the probability for $+$ is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.
Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy}(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

where $p_+$ is the proportion of positive examples in S and $p_-$ is the proportion of negatives.

- If all the examples belong to the same category: Entropy = 0
- If all the examples are equally mixed (0.5, 0.5): Entropy = 1
- Entropy = Level of uncertainty.
Entropy (impurity, disorder) of a set of examples, $S$, relative to a binary classification is:

$$\text{Entropy}(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

where $p_+$ is the proportion of positive examples in $S$ and $p_-$ is the proportion of negatives.

If all the examples belong to the same category: Entropy = 0
If all the examples are equally mixed (0.5, 0.5): Entropy = 1
Information Gain

The information gain of an attribute $a$ is the expected reduction in entropy caused by partitioning on this attribute

$$\text{Gain}(S, a) = \text{Entropy}(S) - \sum_{v \in \text{values}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

where $S_v$ is the subset of $S$ for which attribute $a$ has value $v$, and the entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set.

- Partitions of low entropy (imbalanced splits) lead to high gain
- Go back to check which of the A, B splits is better

High Entropy – High level of Uncertainty
Low Entropy – No Uncertainty.
Will I play tennis today?

<table>
<thead>
<tr>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>

**Outlook:**  S(unny),
               O(vercast),
               R(ainy)

**Temperature:**  H(ot),
                   M(edium),
                   C(ool)

**Humidity:**  H(igh),
               N(ormal),
               L(ow)

**Wind:**  S(trong),
           W(eak)
Will I play tennis today?

<table>
<thead>
<tr>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>

Current entropy:

\[
p = \frac{9}{14} \\
n = \frac{5}{14}
\]

\[
H(Y) = -(\frac{9}{14}) \log_2(\frac{9}{14}) \quad -(\frac{5}{14}) \log_2(\frac{5}{14}) \\
\approx 0.94
\]
Information Gain: Outlook

<table>
<thead>
<tr>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>

**Outlook = sunny:**

\[ p = \frac{2}{5} \quad n = \frac{3}{5} \]

\[ H_S = 0.971 \]

**Outlook = overcast:**

\[ p = \frac{4}{4} \quad n = 0 \]

\[ H_o = 0 \]

**Outlook = rainy:**

\[ p = \frac{3}{5} \quad n = \frac{2}{5} \]

\[ H_R = 0.971 \]

**Expected entropy:**

\[ \frac{5}{14} \times 0.971 \left( \frac{4}{14} \times 0 \right) + \frac{5}{14} \times 0.971 = 0.694 \]

**Information gain:**

\[ 0.940 - 0.694 = 0.246 \]
Information Gain: Humidity

<table>
<thead>
<tr>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>

**Humidity = high:**

\[ p = \frac{3}{7} \quad n = \frac{4}{7} \]

\[ H_h = 0.985 \]

**Humidity = Normal:**

\[ p = \frac{6}{7} \quad n = \frac{1}{7} \]

\[ H_o = 0.592 \]

**Expected entropy:**

\[ \frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592 = 0.7785 \]

**Information gain:**

\[ 0.940 - 0.151 = 0.1515 \]
Which feature to split on?

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>

Information gain:
- Outlook: 0.246
- Humidity: 0.151
- Wind: 0.048
- Temperature: 0.029

→ Split on Outlook
An Illustrative Example (III)

Gain(S,Humidity) = 0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246
An Illustrative Example (III)

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>
An Illustrative Example (III)

### Outlook

- Sunny: 1, 2, 8, 9, 11
  - 2+, 3-
  - ?
- Overcast: 3, 7, 12, 13
  - 3+, 2
  - Yes
- Rain: 4, 5, 6, 10, 14
  - 3+, 2-  
  - ?

Continue until:
- Every attribute is included in path, or,
- All examples in the leaf have same label

#### Table

<table>
<thead>
<tr>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
</tr>
</tbody>
</table>
An Illustrative Example (IV)

### Outlook

- **Sunny**: 1,2,8,9,11
- **Overcast**: 3,7,12,13
- **Rain**: 4,5,6,10,14

### Gain Calculations

- \( \text{Gain}(S_{\text{sunny}}, \text{Humidity}) = 0.97 - (3/5) 0 - (2/5) 0 = 0.97 \)
- \( \text{Gain}(S_{\text{sunny}}, \text{Temp}) = 0.97 - 0 - (2/5) 1 = 0.57 \)
- \( \text{Gain}(S_{\text{sunny}}, \text{Wind}) = 0.97 - (2/5) 1 - (3/5) 0.92 = 0.02 \)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>
An Illustrative Example (V)

```
Outlook
  Sunny 1,2,8,9,11
    2+,3- ?
  Overcast 3,7,12,13
    4+,0- Yes
  Rain 4,5,6,10,14
    3+,2- ?
```
An Illustrative Example (V)

Decision Tree:

- **Outlook**
  - Sunny: 1,2,8,9,11, 2+,3-
  - Overcast: 3,7,12,13, 4+,0-
  - Rain: 4,5,6,10,14, 3+,2-

- **Humidity**
  - High: No
  - Normal: Yes

- **Result**:
  - Yes
  - ?
induceDecisionTree(S)

1. Does S uniquely define a class?
   \[
   \text{if all } s \in S \text{ have the same label } y: \text{return } S;
   \]

2. Find the feature with the most information gain:
   \[
   i = \arg\max_i \text{Gain}(S, X_i)
   \]

3. Add children to S:
   \[
   \text{for } k \text{ in \ Values}(X_i):
   \]
   \[
   S_k = \{s \in S \mid x_i = k\}
   \]
   \[
   \text{addChild}(S, S_k)
   \]
   \[
   \text{induceDecisionTree}(S_k)
   \]
   \[
   \text{return } S;
   \]
An Illustrative Example (VI)

- **Outlook**
  - Sunny
    - Humidity
      - High: No
      - Normal: Yes
  - Overcast
    - Wind: Yes
  - Rain
    - Wind: No

- **Humidity**
  - High: No
  - Normal: Yes

- **Wind**
  - Strong: No
  - Weak: Yes
Hypothesis Space in Decision Tree Induction

- Conduct a search of the space of decision trees which can represent all possible discrete functions. *(pros and cons)*

- Goal: to find the *best* decision tree
  - Best could be “smallest depth”
  - Best could be “minimizing the expected number of tests”

- Finding a minimal decision tree consistent with a set of data is *NP-hard*.

- Performs a greedy heuristic search: hill climbing without backtracking

- Makes statistically based decisions using all data
Today’s key concepts

- Learning decision trees (ID3 algorithm)
  - Greedy heuristic (based on information gain)
  - Originally developed for discrete features

- Overfitting
  - What is it? How do we deal with it?

- Some extensions of DTs

- Principles of Experimental ML

How can this be avoided with linear classifiers?
History of Decision Tree Research

- Hunt and colleagues in Psychology used full search decision tree methods to model human concept learning in the 60s.
- Quinlan developed ID3, with the information gain heuristics in the late 70s to learn expert systems from examples.
- Breiman, Friedman and colleagues in statistics developed CART (classification and regression trees simultaneously).
- A variety of improvements in the 80s: coping with noise, continuous attributes, missing data, non-axis parallel etc.
- Quinlan’s updated algorithm, C4.5 (1993) is commonly used (New: C5).
- Boosting (or Bagging) over DTs is a very good general purpose algorithm.
Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO
Overfitting - Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO

- Sunny
  - Humidity: Normal, Wind: No
    - Strong No
    - Weak Yes

- Overcast
  - Wind: Yes

- Rain
  - Wind: No

This can always be done – may fit noise or other coincidental regularities
The instance space
Learning a tree that classifies the training data perfectly may not lead to the tree with the **best generalization performance**.

- There may be noise in the training data the tree is fitting
- The algorithm might be making decisions based on very little data

A hypothesis $h$ is said to overfit the training data if there is another hypothesis $h'$, such that $h$ has a smaller error than $h'$ on the training data but $h$ has larger error on the test data than $h'$. 

![Graph showing accuracy of training and testing data with complexity of tree as x-axis](image)
Reasons for overfitting

- **Too much variance in the training data**
  - Training data is not a representative sample of the instance space
  - We split on features that are actually irrelevant

- **Too much noise in the training data**
  - Noise = some feature values or class labels are incorrect
  - We learn to predict the noise

In both cases, it is a result of our will to minimize the empirical error when we learn, and the ability to do it (with DTs)
Pruning a decision tree

- Prune = remove leaves and assign majority label of the parent to all items
- Prune the children of S if:
  - all children are leaves, and
  - the accuracy on the validation set does not decrease if we assign the most frequent class label to all items at S.
Avoiding Overfitting

- Two basic approaches
  - Pre-pruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Post-pruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

- Methods for evaluating subtrees to prune
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions?

- This is related to the notion of regularisation that we will see in other contexts – keep the hypothesis simple.

- How can this be avoided with linear classifiers?

Next: a brief detour into explaining generalization and overfitting
The i.i.d. assumption

- Training and test items are independently and identically distributed (i.i.d.):
  - There is a distribution $P(X, Y)$ from which the data $\mathcal{D} = \{(x, y)\}$ is generated.
    - Sometimes it’s useful to rewrite $P(X, Y)$ as $P(X)P(Y|X)$
      - Usually $P(X, Y)$ is unknown to us (we just know it exists)
  - Training and test data are samples drawn from the same $P(X, Y)$: they are identically distributed
  - Each $(x, y)$ is drawn independently from $P(X, Y)$
A decision tree **overfits the training data** when its accuracy on the training data goes up but its accuracy on unseen data goes down.

Why this shape of curves?
Empirical error (= on a given data set):
The percentage of items in this data set are misclassified by the classifier $f$. 
Model complexity (informally): How many parameters do we have to learn?
- Decision trees: complexity = #nodes
Expected error:
What percentage of items drawn from $P(x,y)$ do we expect to be misclassified by $f$?

(That’s what we really care about – generalization)
Variance of a learner (informally)

- How susceptible is the learner to minor changes in the training data?
  - (i.e. to different samples from $P(X, Y)$)

- Variance increases with model complexity
  - Think about extreme cases: a hypothesis space with one function vs. all functions.
  - The larger the hypothesis space is, the more flexible the selection of the chosen hypothesis is as a function of the data.
  - More accurately: for each data set $D$, you will learn a different hypothesis $h(D)$, that will have a different true error $e(h)$; we are looking here at the variance of this random variable.
Bias of a learner (informally)

- How likely is the learner to identify the target hypothesis?
- Bias is **low** when the model is expressive (low empirical error)
- Bias is **high** when the model is (too) simple
  - The larger the hypothesis space is, the easiest it is to be close to the true hypothesis.
  - More accurately: for each data set $D$, you learn a different hypothesis $h(D)$, that has a different true error $e(h)$; we are looking here at the difference of the mean of this random variable from the true error.
Impact of bias and variance

\[ \text{Expected error} \approx \text{bias} + \text{variance} \]
Model complexity

- **Simple models:** High bias and low variance
- **Complex models:** High variance and low bias
Underfitting and Overfitting

- **Simple models:** High bias and low variance
- **Complex models:** High variance and low bias

- This can be made more accurate for some loss functions.
- We will develop a more precise and general theory that trades expressivity of models with empirical error.
Avoiding Overfitting

- Two basic approaches
  - Pre-pruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Post-pruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

- Methods for evaluating subtrees to prune
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions?

- This is related to the notion of regularization that we will see in other contexts – keep the hypothesis simple.

How can this be avoided with linear classifiers?

Hand waving, for now.

Next: a brief detour into explaining generalization and overfitting.
Trees and Rules

- Decision Trees can be represented as Rules
  - (outlook = sunny) and (humidity = normal) then YES
  - (outlook = rain) and (wind = strong) then NO

- Sometimes Pruning can be done at the rules level
  - Rules are generalized by erasing a condition (different!)

```
Decision Trees

DECISION TREES

CS446 Spring’17

64
```
Continuous Attributes

- Real-valued attributes can, in advance, be discretized into ranges, such as *big, medium, small*

- Alternatively, one can develop splitting nodes based on thresholds of the form $A < c$ that partition the data into examples that satisfy $A < c$ and $A \geq c$. The information gain for these splits is calculated in the same way and compared to the information gain of discrete splits.

- How to find the *split with the highest gain*?
  - For each continuous feature $A$:
    - Sort examples according to the value of $A$
    - For each ordered pair $(x, y)$ with different labels
      - Check the mid-point as a possible threshold, i.e.
        $$S_a \leq x, S_a \geq y$$
Continuous Attributes

- Example:
  - Length (L): 10 15 21 28 32 40 50
  - Class: - + + - + + -
  - Check thresholds: L < 12.5; L < 24.5; L < 45
  - Subset of Examples= {...}, Split= k+, j-

- How to find the split with the highest gain?
  - For each continuous feature A:
    - Sort examples according to the value of A
    - For each ordered pair (x,y) with different labels
      - Check the mid-point as a possible threshold. I.e,
        \[ S_a \leq x, S_a \geq y \]
Missing Values

- Diagnosis = < fever, blood_pressure, ..., blood_test=?, ...>

- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)

- Training: evaluate $\text{Gain}(S,a)$ where in some of the examples a value for $a$ is not given
Missing Values

\[ \text{Gain}(S_{sunny}, \text{Temp}) = 0.97 \times 0 - (2/5) = 0.57 \]

\[ \text{Gain}(S_{sunny}, \text{Humidity}) = \]

- Fill in: assign the most likely value of \( X_i \) to \( s \):
  \[ \arg\max_k P(X_i = k): \text{Normal} \]
- \( 0.97 \times (3/5) \times \text{Ent}[0,-3] \times (2/5) \times \text{Ent}[2,0] = 0.97 \)
- Assign fractional counts \( P(X_i = k) \) for each value of \( X_i \) to \( s \)
- \( 0.97 \times (2.5/5) \times \text{Ent}[0,-2.5] \times (2.5/5) \times \text{Ent}[2,-.5] < 0.97 \)

Other suggestions?

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>???</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

DECISION TREES

CS446 Spring’17
Missing Values

- Diagnosis = < fever, blood_pressure,..., blood_test=?,...>

- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)

- **Training**: evaluate $\text{Gain}(S,a)$ where in some of the examples a value for $a$ is not given

- **Testing**: classify an example without knowing the value of $a$
Missing Values

Outlook = Sunny, Temp = Hot, Humidity = ???, Wind = Strong, label = ?? Normal/High

Outlook = ???, Temp = Hot, Humidity = Normal, Wind = Strong, label = ??

1/3 Yes + 1/3 Yes + 1/3 No = Yes

Outlook

Sunny
1,2,8,9,11
2+,3-
Humidity
High
No
Normal
Yes

Overcast
3,7,12,13
4+,0-
Yes

Rain
4,5,6,10,14
3+,2-
Wind
Strong
No
Weak
Yes

Other suggestions?
Other Issues

- **Attributes with different costs**
  - Change information gain so that low cost attribute are preferred
    - Dealing with features with different # of values

- **Alternative measures for selecting attributes**
  - When different attributes have different number of values
    - Information gain tends to prefer those with many values

- **Oblique Decision Trees**
  - Decisions are not axis-parallel

- **Incremental Decision Trees induction**
  - Update an existing decision tree to account for new examples incrementally (Maintain consistency?)
Decision Trees as Features

- Rather than using decision trees to represent the target function, it is becoming common to use small decision trees as features.
- When learning over a large number of features, learning decision trees is difficult and the resulting tree may be very large → (over fitting)
- Instead, learn small decision trees, with limited depth.
- Treat them as “experts”; they are correct, but only on a small region in the domain. (what DTs to learn? same every time?)
- Then, learn another function, typically a linear function, over these as features.
- **Boosting** (but also other linear learners) are used on top of the small decision trees. (Either Boolean, or real valued features)
Experimental Machine Learning

- Machine Learning is an Experimental Field and we will spend some time (in Problem sets) learning how to run experiments and evaluate results
  - First hint: be organized; write scripts

- Basics:
  - Split your data into two (or three) sets:
    - Training data (often 70-90%)
    - Test data (often 10-20%)
    - Development data (10-20%)

- You need to report performance on test data, but you are not allowed to look at it.
  - You are allowed to look at the development data (and use it to tweak parameters)
N-fold cross validation

- Instead of a single test-training split:

- Split data into N equal-sized parts

- Train and test N different classifiers

- Report average accuracy and standard deviation of the accuracy
Evaluation: significance tests

- You have two different classifiers, A and B
- You train and test them on the same data set using N-fold cross-validation
- For the \( n \)-th fold:
  \[
  p_n = \text{accuracy}(A, n) - \text{accuracy}(B, n)
  \]
- Is the difference between A and B’s accuracies significant?
Hypothesis testing

- You want to show that hypothesis H is true, based on your data
  - (e.g. H = “classifier A and B are different”)

- Define a null hypothesis $H_0$
  - ($H_0$ is the contrary of what you want to show)

- $H_0$ defines a distribution $P(m \mid H_0)$ over some statistic
  - e.g. a distribution over the difference in accuracy between A and B

- Can you refute (reject) $H_0$?
Rejecting $H_0$

- $H_0$ defines a distribution $P(M \mid H_0)$ over some statistic $M$
  - (e.g. $M =$ the difference in accuracy between A and B)
- Select a significance value $S$
  - (e.g. 0.05, 0.01, etc.)
  - You can only reject $H_0$ if $P(m \mid H_0) \leq S$
- Compute the test statistic $m$ from your data
  - e.g. the average difference in accuracy over your N folds
- Compute $P(m \mid H_0)$
- Refute $H_0$ with $p \leq S$ if $P(m \mid H_0) \leq S$
Paired t-test

- Null hypothesis ($H_0$; to be refuted):
  - There is no difference between A and B, i.e. the expected accuracies of A and B are the same
  - That is, the expected difference (over all possible data sets) between their accuracies is 0:
    \[ H_0: E[p_D] = 0 \]
- We don’t know the true $E[p_D]$
- $N$-fold cross-validation gives us $N$ samples of $p_D$
Paired t-test

- Null hypothesis $H_0: E[\text{diff}_D] = \mu = 0$
- $m$: our estimate of $\mu$ based on $N$ samples of $\text{diff}_D$
  \[ m = \frac{1}{N} \sum_{n} \text{diff}_n \]
- The estimated variance $S^2$:
  \[ S^2 = \frac{1}{(N-1)} \sum_{1,N} (\text{diff}_n - m)^2 \]
- Accept Null hypothesis at significance level $\alpha$ if the following statistic lies in $(-t_{\alpha/2, N-1}, +t_{\alpha/2, N-1})$
  \[ \frac{\sqrt{N}m}{S} \sim t_{N-1} \]
Decision Trees - Summary

- **Hypothesis Space:**
  - Variable size (contains all functions)
  - Deterministic; Discrete and Continuous attributes

- **Search Algorithm**
  - ID3 - batch
  - Extensions: missing values

- **Issues:**
  - What is the goal?
  - When to stop? How to guarantee good generalization?

- **Did not address:**
  - How are we doing? (Correctness-wise, Complexity-wise)