Plan for today:

- Supervised Learning and Decision Trees
- Decision Trees and Overfitting,
- k-Nearest Neighbor and Instance-based Learning
Function Approximation

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{ h \mid h : \mathcal{X} \rightarrow \mathcal{Y} \}$

Input: Training examples of unknown target function $f$

$$\left\{ \langle x_i, y_i \rangle \right\}_{i=1}^{n} = \left\{ \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle \right\}$$

Output: Hypothesis $h \in H$ that best approximates $f$
Sample Dataset

- Columns denote features $X_i$
- Rows denote labeled instances $\langle x_i, y_i \rangle$
- Class label denotes whether a tennis game was played
Decision Tree

- A possible decision tree for the data:

- Each internal node: test one attribute $X_i$
- Each branch from a node: selects one value for $X_i$
- Each leaf node: predict $Y$ (or $p(Y \mid \mathbf{x} \in \text{leaf})$)

Based on slide by Tom Mitchell
Decision Tree

- A possible decision tree for the data:

```
   Outlook
   /    |
Sunny  Overcast  Rain
   |
Humidity
   /   |
High  Normal
   |
No    Yes
   |
   |
Wind
   /   |
Strong  Weak
   |
No    Yes
   |
```

- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak>?
Decision Tree

- If features are continuous, internal nodes can test the value of a feature against a threshold.
Decision Tree Learning

Problem Setting:

• Set of possible instances $X$
  – each instance $x$ in $X$ is a feature vector
  – e.g., $<\text{Humidity}=\text{low}, \text{Wind}=\text{weak}, \text{Outlook}=\text{rain}, \text{Temp}=\text{hot}>$

• Unknown target function $f : X \rightarrow Y$
  – $Y$ is discrete valued

• Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$
  – each hypothesis $h$ is a decision tree
  – trees sorts $x$ to leaf, which assigns $y$
Stages of (Batch) Machine Learning

**Given:** labeled training data $X, Y = \{\langle x_i, y_i \rangle \}_{i=1}^n$

- Assumes each $x_i \sim D(X)$ with $y_i = f_{\text{target}}(x_i)$

Train the model:

$$model \leftarrow \text{classifier}.\text{train}(X, Y)$$

Apply the model to new data:

- Given: new unlabeled instance $x \sim D(X)$
  
  $$y_{\text{prediction}} \leftarrow \text{model}.\text{predict}(x)$$
Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles.
- Each rectangular region is labeled with one label – or a probability distribution over labels.
Expressiveness

• Decision trees can represent any boolean function of the input attributes

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

• In the worst case, the tree will require exponentially many nodes

Truth table row → path to leaf
Expressiveness

Decision trees have a variable-sized hypothesis space

• As the #nodes (or depth) increases, the hypothesis space grows
  – Depth 1 ("decision stump"): can represent any boolean function of one feature
  – Depth 2: any boolean fn of two features; some involving three features (e.g., \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)\))
  – etc.

Based on slide by Pedro Domingos
Another Example: Restaurant Domain (Russell & Norvig)

Model a patron’s decision of whether to wait for a table at a restaurant

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

~7,000 possible cases
A Decision Tree from Introspection

Is this the best decision tree?
Preference bias: Ockham’s Razor

• Principle stated by William of Ockham (1285-1347)
  – “*non sunt multiplicanda entia praeter necessitatem*”
  – entities are not to be multiplied beyond necessity
  – AKA Occam’s Razor, Law of Economy, or Law of Parsimony

Idea: The simplest consistent explanation is the best

• Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  • Finding the provably smallest decision tree is NP-hard
  • ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small
Basic Algorithm for Top-Down Induction of Decision Trees
[ID3, C4.5 by Quinlan]

\[\text{node} = \text{root of decision tree}\]

Main loop:
1. \(A \leftarrow\) the “best” decision attribute for the next node.
2. Assign \(A\) as decision attribute for \(\text{node}\).
3. For each value of \(A\), create a new descendant of \(\text{node}\).
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop. Else, recurse over new leaf nodes.

How do we choose which attribute is best?
Choosing the Best Attribute

**Key problem:** choosing which attribute to split a given set of examples

- **Random:** Select any attribute at random
- **Least-Values:** Choose the attribute with the smallest number of possible values
- **Most-Values:** Choose the attribute with the largest number of possible values
- **Max-Gain:** Choose the attribute that has the largest expected *information gain*
  - i.e., attribute that results in smallest expected size of subtrees rooted at its children

- The ID3 algorithm uses the Max-Gain method of selecting the best attribute
Choosing an Attribute

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

Which split is more informative: *Patrons?* or *Type*?

Based on Slide from M. desJardins & T. Finin
ID3-induced Decision Tree

Based on Slide from M. desJardins & T. Finin
Compare the Two Decision Trees

Based on Slide from M. desJardins & T. Finin
Information Gain

Which test is more informative?

**Split over whether Balance exceeds 50K**

<table>
<thead>
<tr>
<th>Less or equal 50K</th>
<th>Over 50K</th>
</tr>
</thead>
</table>

**Split over whether applicant is employed**

<table>
<thead>
<tr>
<th>Unemployed</th>
<th>Employed</th>
</tr>
</thead>
</table>
**Information Gain**

**Impurity/Entropy (informal)**

– Measures the level of **impurity** in a group of examples
Impurity

Very impure group

Less impure

Minimum impurity

Based on slide by Pedro Domingos
Entropy: a common way to measure impurity

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of $X$ (under most efficient code)
Entropy: a common way to measure impurity

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of $X$ (under most efficient code)

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random $X$ is:
  $$\sum_{i=1}^{n} P(X = i)(- \log_2 P(X = i))$$
2-Class Cases:

Entropy  \[ H(x) = - \sum_{i=1}^{n} P(x = i) \log_2 P(x = i) \]

• What is the entropy of a group in which all examples belong to the same class?
  – entropy = - 1 \( \log_2 1 = 0 \)

  not a good training set for learning

• What is the entropy of a group with 50% in either class?
  – entropy = -0.5 \( \log_2 0.5 \) – 0.5 \( \log_2 0.5 \) = 1

  good training set for learning

Based on slide by Pedro Domingos
Information Gain

• We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

• Information gain tells us how important a given attribute of the feature vectors is.

• We will use it to decide the ordering of attributes in the nodes of a decision tree.

Based on slide by Pedro Domingos
From Entropy to Information Gain

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$
From Entropy to Information Gain

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Specific conditional entropy $H(X|Y=v)$ of $X$ given $Y=v$:

$$H(X|Y = v) = - \sum_{i=1}^{n} P(X = i|Y = v) \log_2 P(X = i|Y = v)$$
From Entropy to Information Gain

Entropy \( H(X) \) of a random variable \( X \)

\[
H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)
\]

Specific conditional entropy \( H(X|Y=v) \) of \( X \) given \( Y=v \) :

\[
H(X|Y = v) = - \sum_{i=1}^{n} P(X = i|Y = v) \log_2 P(X = i|Y = v)
\]

Conditional entropy \( H(X|Y) \) of \( X \) given \( Y \) :

\[
H(X|Y) = \sum_{v \in \text{values}(Y)} P(Y = v) H(X|Y = v)
\]
From Entropy to Information Gain

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Specific conditional entropy $H(X|Y=v)$ of $X$ given $Y=v$:

$$H(X|Y = v) = - \sum_{i=1}^{n} P(X = i|Y = v) \log_2 P(X = i|Y = v)$$

Conditional entropy $H(X|Y)$ of $X$ given $Y$:

$$H(X|Y) = \sum_{v \in \text{values}(Y)} P(Y = v) H(X|Y = v)$$

Mutual information (aka Information Gain) of $X$ and $Y$:

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$
Information Gain

Information Gain is the mutual information between input attribute A and target variable Y.

Information Gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A.

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$
Calculating Information Gain

**Information Gain** = entropy(parent) – [average entropy(children)]

Entire population (30 instances)

- Parent entropy
  \[-\left(\frac{14}{30} \cdot \log_2 \frac{14}{30}\right) - \left(\frac{16}{30} \cdot \log_2 \frac{16}{30}\right) = 0.996\]

Child entropy

17 instances

- \(-\left(\frac{13}{17} \cdot \log_2 \frac{13}{17}\right) - \left(\frac{4}{17} \cdot \log_2 \frac{4}{17}\right) = 0.787\)

13 instances

- \(-\left(\frac{1}{13} \cdot \log_2 \frac{1}{13}\right) - \left(\frac{12}{13} \cdot \log_2 \frac{12}{13}\right) = 0.391\)

(Weighted) Average Entropy of Children

\[
\frac{17}{30} \cdot 0.787 + \frac{13}{30} \cdot 0.391 = 0.615
\]

**Information Gain** = 0.996 - 0.615 = 0.38

Based on slide by Pedro Domingos
Using Information Gain to Construct a Decision Tree

Full Training Set $X$

Attribute $A$

Set $X$ such that $X_{v1} = \{ x \in X | \text{value}(A) = v1 \}$

Construct child nodes for each value of $A$. Each has an associated subset of vectors in which $A$ has a particular value.

Choose the attribute $A$ with highest information gain for the full training set at the root of the tree.

Disadvantage of information gain:

- It prefers attributes with large number of values that split the data into small, pure subsets
- Quinlan’s gain ratio uses normalization to improve this
### Training Examples

<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>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Selecting the Next Attribute

Which attribute is the best classifier?

$S: [9+,5-]$
$E = 0.940$

Humidity

High: $[3+,4-]$  Normal: $[6+,1-]$

Wind

Weak: $[6+,2-]$  Strong: $[3+,3-]$
Selecting the Next Attribute

Which attribute is the best classifier?

Gain \( S, \text{ Humidity } \)
\[
= 0.940 - \left(\frac{7}{14}\right)0.985 - \left(\frac{7}{14}\right)0.592 \\
= 0.151
\]

Gain \( S, \text{ Wind } \)
\[
= 0.940 - \left(\frac{8}{14}\right)0.811 - \left(\frac{6}{14}\right)1.0 \\
= 0.048
\]
Which attribute should be tested here?

\[ S_{\text{sunny}} = \{D1,D2,D8,D9,D11\} \]

\[
\text{Gain (} S_{\text{sunny}}, \text{Humidity)} = 0.970 - \left( \frac{3}{5} \right) 0.0 - \left( \frac{2}{5} \right) 0.0 = 0.970
\]
\[
\text{Gain (} S_{\text{sunny}}, \text{Temperature)} = 0.970 - \left( \frac{2}{5} \right) 0.0 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{1}{5} \right) 0.0 = 0.570
\]
\[
\text{Gain (} S_{\text{sunny}}, \text{Wind)} = 0.970 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{3}{5} \right) 0.918 = 0.019
\]
Which Tree Should We Output?

- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Occam’s razor: prefer the simplest hypothesis that fits the data
How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

– A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct.

– British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system.

– Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example.
Summary of Decision Trees (so far)

- Decision tree induction \(\rightarrow\) choose the best attribute
  - Choose split via information gain
  - Build tree greedily, recursing on children of split
  - Stop when we achieve homogeny
    - i.e., when all instance in a child have the same class
Summary of Decision Trees (so far)

Information Gain: Mutual information of attribute $A$ and the class variable of data set $X$

$$\text{InfoGain}(X, A) = H(X) - H(X | A)$$

$$= H(X) - \sum_{v \in \text{values}(A)} \frac{|\{x \in X | x_A = v\}|}{|X|} \times H(\{x \in X | x_A = v\})$$

Entropy:

$$H(X) = - \sum_{c \in \text{Classes}} \frac{|\{x \in X | \text{class}(x) = c\}|}{|X|} \log_2 \frac{|\{x \in X | \text{class}(x) = c\}|}{|X|}$$
Restaurant Example

Random: Patrons or Wait-time; Least-values: Patrons; Most-values: Type; Max-gain: ???

<table>
<thead>
<tr>
<th>Type variable</th>
<th>Empty</th>
<th>Some</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>French</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Italian</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Thai</td>
<td>N</td>
<td>Y</td>
<td>N Y</td>
</tr>
<tr>
<td>Burger</td>
<td>N</td>
<td>Y</td>
<td>N Y</td>
</tr>
</tbody>
</table>

Patrons variable
# Computing information gain

<table>
<thead>
<tr>
<th></th>
<th>French</th>
<th>Italian</th>
<th>Thai</th>
<th>Burger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Some</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N Y</td>
</tr>
<tr>
<td>Full</td>
<td></td>
<td></td>
<td></td>
<td>N Y</td>
</tr>
</tbody>
</table>

\[ I(X) = ? \]

\[ I(Pat, X) = ? \]

\[ I(Type, X) = ? \]

\[ \text{Gain (Pat, X)} = ? \]
\[ \text{Gain (Type, X)} = ? \]

Based on Slide from M. desJardins & T. Finin
### Computing information gain

\[ I(X) = \]
\[ = .5 + .5 = 1 \]

\[ I(Pat, X) = \]
\[ I(\text{Type}, X) = \]

\[ \text{Gain (Pat, X)} = ? \]
\[ \text{Gain (Type, X)} = ? \]

Based on Slide from M. desJardins & T. Finin
Computing information gain

\[ I(X) = - (.5 \log .5 + .5 \log .5) = .5 + .5 = 1 \]

\[ I(\text{Pat}, X) = \frac{2}{12} (0) + \frac{4}{12} (0) + \frac{6}{12} (- (\frac{4}{6} \log 4/6 + \frac{2}{6} \log 2/6)) = \frac{1}{2} (2/3*.6 + 1/3*1.6) = .47 \]

\[ I(\text{Type}, X) = ? \]

Gain (\text{Pat}, X) = ?
Gain (\text{Type}, X) = ?

Based on Slide from M. desJardins & T. Finin
Computing information gain

$I(X) =$
- $(.5 \log .5 + .5 \log .5)$
= $.5 + .5 = 1$

$I (\text{Pat}, X) =$
$2/12 \ (0) + 4/12 \ (0) +$
$6/12 \ (- (4/6 \log 4/6 +$
$2/6 \log 2/6))$
= $1/2 \ (2/3*.6 +$
$1/3*1.6)$
= $.47$

$I (\text{Type}, X) =$
$2/12 \ (1) + 2/12 \ (1) +$
$4/12 \ (1) + 4/12 \ (1) = 1$

Gain (Pat, X) = ?
Gain (Type, X) = ?
Computing information gain

\[
I(X) = \left( -0.5 \log 0.5 + 0.5 \log 0.5 \right)
= 0.5 + 0.5 = 1
\]

\[
I(\text{Pat}, X) = \frac{2}{12} (0) + \frac{4}{12} (0) + \frac{6}{12} \left( - \left( \frac{4}{6} \log \frac{4}{6} + \frac{2}{6} \log \frac{2}{6} \right) \right)
= \frac{1}{2} (2/3 \times 0.6 + 1/3 \times 1.6)
= 0.47
\]

\[
I(\text{Type}, X) = \frac{2}{12} (1) + \frac{2}{12} (1) + \frac{4}{12} (1) + \frac{4}{12} (1) = 1
\]

Gain (Pat, X) = 1 - 0.47 = 0.53
Gain (Type, X) = 1 - 1 = 0
Attributes with Many Values

• Problem
  – If attribute has many values, InfoGain() will select it
  – e.g., imagine using date = Jan_28_2011 as an attribute

• Alternative approach: use GainRatio() instead

\[
GainRatio(X, A) = \frac{InfoGain(X, A)}{SplitInformation(X, A)}
\]
\[
SplitInformation(X, A) = -\sum_{v \in values(A)} \frac{|X_v|}{|X|} \log_2 \frac{|X_v|}{|X|}
\]

where \( X_v \) is a subset of \( X \) for which \( A \) has value \( v \)
## Computing Gain Ratio

### Already computed:
- $I(X) = 1$
- $I(Pat, X) = 0.47$
- $I(Type, X) = 1$
- $Gain(Pat, X) = 0.53$
- $Gain(Type, X) = 0$

<table>
<thead>
<tr>
<th></th>
<th>French</th>
<th>Italian</th>
<th>Thai</th>
<th>Burger</th>
<th>Empty</th>
<th>Some</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N Y</td>
</tr>
</tbody>
</table>

### SplitInfo
- $SplitInfo(Pat, X) = -(1/6 \log 1/6 + 1/3 \log 1/3 + 1/2 \log 1/2)$
- $= 1/6*2.6 + 1/3*1.6 + 1/2*1 = 1.47$

- $SplitInfo(Type, X) = 1/6 \log 1/6 + 1/6 \log 1/6 + 1/3 \log 1/3 + 1/3 \log 1/3$
- $= 1/6*2.6 + 1/6*2.6 + 1/3*1.6 + 1/3*1.6 = 1.93$

Based on Slide from M. desJardins & T. Finin
## Computing Gain Ratio

Already computed:

- $I(X) = 1$
- $I(Pat, X) = 0.47$
- $I(Type, X) = 1$
- $Gain(Pat, X) = 0.53$
- $Gain(Type, X) = 0$

### SplitInfo

- $SplitInfo(Pat, X) = -\left(\frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{2}\log\frac{1}{2}\right) = \frac{1}{6}\times2.6 + \frac{1}{3}\times1.6 + \frac{1}{2}\times1 = 1.47$

- $SplitInfo(Type, X) = \frac{1}{6}\log\frac{1}{6} + \frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{3}\log\frac{1}{3} = \frac{1}{6}\times2.6 + \frac{1}{6}\times2.6 + \frac{1}{3}\times1.6 + \frac{1}{3}\times1.6 = 1.93$

### GainRatio

- $GainRatio(Pat, X) = \frac{Gain(Pat, X)}{SplitInfo(Pat, X)} = \frac{0.53}{1.47} = 0.36$

- $GainRatio(Type, X) = \frac{Gain(Type, X)}{SplitInfo(Type, X)} = \frac{0}{1.93} = 0$

---

Based on Slide from M. desJardins & T. Finin
Extensions of ID3

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Based on Slide from M. desJardins & T. Finin
Real-Valued Features

• Change to binary splits by choosing a threshold
• One method:
  – Sort instances by value, identify adjacencies with different classes
    Temperature: 40 48 60 72 80 90
    PlayTennis: No No Yes Yes Yes No
    candidate splits
  – Choose among splits by InfoGain()
Unknown Attribute Values

What if some examples are missing values of $A$?

Use training example anyway, sort through tree:
• If node $n$ tests $A$, assign most common value of $A$ among other examples sorted to node $n$
• Assign most common value of $A$ among other examples with same class label
• Assign probability $p_i$ to each possible value $v_i$ of $A$.
  Assign fraction $p_i$ of example to each descendent of tree

Classify new examples in same fashion
Noisy Data

• Many kinds of “noise” can occur in the examples:
  – Two examples have same attribute/value pairs, but different classifications
  – Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
  – The instance was labeled incorrectly (+ instead of -)

• Also, some attributes are irrelevant to the decision-making process
  – e.g., color of a die is irrelevant to its outcome
Overfitting

• Irrelevant attributes can result in overfitting the training example data
  
  – If hypothesis space has many dimensions (large number of attributes), we may find meaningless regularity in the data that is irrelevant to the true, important, distinguishing features

• If we have too little training data, even a reasonable hypothesis space will ‘overfit’
Overfitting in Decision Trees

Consider adding a noisy training example to the following tree:

What would be the effect of adding:

<outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No>?
Overfitting

Consider error of hypothesis $h$ over

- training data: $\text{error}_{\text{train}}(h)$
- entire distribution $\mathcal{D}$ of data: $\text{error}_{\mathcal{D}}(h)$

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

$$\text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h')$$
Overfitting in Decision Tree Learning

![Graph showing overfitting in decision tree learning](image)

- **Accuracy**
- **Size of tree (number of nodes)**
- **On training data**
- **On test data**
Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)
- 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

Based on Slide by Pedro Domingos
Reduced-Error Pruning

Split data into training and validation sets

Grow tree based on training set

Do until further pruning is harmful:
1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the node that most improves validation set accuracy
Pruning Decision Trees

• Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
• The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
• For example,

Training

Validation

If we had simply predicted the majority class (negative), we make 2 errors instead of 4.
Effect of Reduced-Error Pruning

Accuracy vs. Size of tree (number of nodes)

- On training data
- On test data
- On test data (during pruning)
The tree is pruned back to the red line where it gives more accurate results on the test data.
Converting a Tree to Rules

IF (Outlook = Sunny) AND (Humidity = High)
THEN PlayTennis = No

IF (Outlook = Sunny) AND (Humidity = Normal)
THEN PlayTennis = Yes

...
Converting Decision Trees to Rules

• It is easy to derive rules from a decision tree: write a rule for each path from the root to a leaf

  \((Outlook = Sunny) \text{ AND } (Humidity = High) \rightarrow PlayTennis = No\)

• To simplify the resulting rule set:
  – Let LHS be the left-hand side of a rule
  – LHS’ obtained from LHS by eliminating some conditions
  – Replace LHS by LHS' in this rule if the subsets of the training set satisfying LHS and LHS' are equal
  – A rule may be eliminated by using meta-conditions such as “if no other rule applies”
Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)
Scaling Up

• ID3, C4.5, etc.: assumes that data fits in memory (OK for up to hundreds of thousands of examples)

• SPRINT, SLIQ: multiple sequential scans of data (OK for up to millions of examples)

• VFDT: at most one sequential scan (OK for up to billions of examples)
Comparison of Learning Methods

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Neural Nets</th>
<th>SVM</th>
<th>Trees</th>
<th>MARS</th>
<th>k-NN, Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural handling of data of “mixed” type</td>
<td>▼▼</td>
<td>▼</td>
<td>▲▲</td>
<td>▲</td>
<td>▼</td>
</tr>
<tr>
<td>Handling of missing values</td>
<td>▼▼</td>
<td>▼</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>Robustness to outliers in input space</td>
<td>▼▼</td>
<td>▼</td>
<td>▲</td>
<td>▼</td>
<td>▲</td>
</tr>
<tr>
<td>Insensitive to monotone transformations of inputs</td>
<td>▼▼</td>
<td>▼</td>
<td>▲</td>
<td>▼</td>
<td>▼</td>
</tr>
<tr>
<td>Computational scalability (large $N$)</td>
<td>▼▼</td>
<td>▼</td>
<td>▲</td>
<td>▲</td>
<td>▼</td>
</tr>
<tr>
<td>Ability to deal with irrelevant inputs</td>
<td>▼▼</td>
<td>▼</td>
<td>▲</td>
<td>▲</td>
<td>▼</td>
</tr>
<tr>
<td>Ability to extract linear combinations of features</td>
<td>▲</td>
<td>▲</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
</tr>
<tr>
<td>Interpretability</td>
<td>▼▼</td>
<td>▼</td>
<td>▼</td>
<td>▲</td>
<td>▼</td>
</tr>
<tr>
<td>Predictive power</td>
<td>▲</td>
<td>▲</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
</tr>
</tbody>
</table>

[Table 10.3 from Hastie, et al. Elements of Statistical Learning, 2nd Edition]
Summary: Decision Tree Learning

• Representation: decision trees
• Bias: prefer small decision trees
• Search algorithm: greedy
• Heuristic function: information gain or information content or others
• Overfitting / pruning
Summary: Decision Tree Learning

• Widely used in practice

• Strengths include
  – Fast and simple to implement
  – Can convert to rules
  – Handles noisy data

• Weaknesses include
  – Univariate splits/partitioning using only one attribute at a time --- limits types of possible trees
  – Large decision trees may be hard to understand
  – Requires fixed-length feature vectors
  – Non-incremental (i.e., batch method)
  – Sacrifices predictive power
1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point
1-Nearest Neighbor

- A type of instance-based learning
  - Also known as “memory-based” learning
- Forms a Voronoi tessellation of the instance space
Distance Metrics

- Different metrics can change the decision surface

![Distance metrics example](image)

**Standard Euclidean distance metric:**

- Two-dimensional: \( \text{Dist}(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \)
- Multivariate: \( \text{Dist}(a,b) = \sqrt{\sum (a_i - b_i)^2} \)

Adapted from “Instance-Based Learning” lecture slides by Andrew Moore, CMU.
Four Aspects of an Instance-Based Learner:

1. A distance metric
2. How many nearby neighbors to look at?
3. A weighting function (optional)
4. How to fit with the local points?

Adapted from “Instance-Based Learning” lecture slides by Andrew Moore, CMU.
1-NN’s Four Aspects as an Instance-Based Learner:

1. A distance metric
   - *Euclidian*

2. How many nearby neighbors to look at?
   - *One*

3. A weighting function (optional)
   - *Unused*

4. How to fit with the local points?
   - *Just predict the same output as the nearest neighbor.*

Adapted from “Instance-Based Learning” lecture slides by Andrew Moore, CMU.
k – Nearest Neighbor

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its $k$ nearest neighbors

Label it red, when $k = 3$

Label it blue, when $k = 7$
KNN and Overfitting

- $K$ increases $\rightarrow$ smoother Predictions
- $K = N$ $\rightarrow$ majority label of the dataset
Today

- Decision Trees
- k-Nearest Neighbor