Data Mining

Dr. Lutz Hamel
Dept. of Computer Science & Statistics

http://homepage.cs.uri.edu/faculty/hamel/
Have You Ever Wondered…

● How does Amazon.com know exactly which books to recommend?
● How do music services like iTunes and Rhapsody figure out what to suggest?
● How do stores that have a membership card know exactly what coupons to print when you pay?

☞ Data Mining!
Machine Learning + Databases = Data Mining

- Machine learning is the discovery and extraction of patterns from data.
- In order for data mining to be useful the patterns need to be actionable, that is, they should tell us something useful about the state of the world.
- It is even better if the patterns are predictive, that is, the knowledge gained from the patterns can be applied to objects that are not currently in our database.
## Patterns

<table>
<thead>
<tr>
<th>(Day)</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</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>

### Some Patterns:

1) If Outlook = Overcast then Yes

2) If Outlook = Sunny and Humidity = High then No

3) If Outlook = Rain and Wind = Weak then Yes

♫ It is very difficult for us to find patterns.
♫ We are easily overwhelmed by large amounts of data.
Patterns & Machine Learning

- Computers are very good at looking at large amounts of data.
- Machine learning is a sub-discipline of AI and deals with algorithms that can detect patterns in data.
- One of the most straightforward machine learning algorithms is the decision tree learner.
If we apply the decision tree algorithm to the Play Tennis data we obtain the following tree:
Interpreting a Decision Tree

A decision tree uses the attributes of a data table as nodes and the attribute values as links.

All attribute values of a particular attribute need to be represented as links.

The target attribute is special - its values show up as leaf nodes in the decision tree.
Interpreting a Decision Tree

Each path from the root of the decision tree to a leaf can be interpreted as a pattern.

IF Outlook = Sunny AND Humidity = Normal THEN Playtennis = Yes
IF Outlook = Overcast THEN Playtennis = Yes
IF Outlook = Rain AND Wind = Strong THEN Playtennis = No
**Explanation & Prediction**

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

**Explanation:** the DT summarizes (explains) all the observations in the table perfectly ⇒ 100% Accuracy

**Prediction:** once we have a DT (or model) we can use it to make predictions on observations that are not in the original training table, consider:

Outlook = Sunny, Temperature = Mild, Humidity = Normal, Windy = False, PlayTennis = ?
Entropy

- $S$ is a sample of training examples
- $p^+$ is the proportion of positive examples in $S$
- $p^-$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$\text{Entropy}(S) \equiv -p^+ \log_2 p^+ - p^- \log_2 p$$
The Algorithm

Partition(Examples, TargetAttribute, Attributes)
Examples are the training examples. TargetAttribute is a binary (+/-) categorical dependent variable and Attributes is the list of independent variables which are available for testing at this point. This function returns a decision tree.

• Create a Root node for the tree.
• If all Examples are positive then return Root as a leaf node with label = +.
• Else if all Examples are negative then return Root as a leaf node with label = -.
• Else if Attributes is empty then return Root as a leaf node with label = most common value of TargetAttribute in Examples.
• Otherwise
  ○ A := the attribute from Attributes that reduces entropy the most on the Examples.
  ○ Root := A
  ○ F or each v \in values(A)
    ▪ Add a new branch below the Root node with value A = v
    ▪ L et Examples_v be the subset of Examples where A = v
    ▪ If Examples_v is empty then add new leaf node to branch with label = most common value of TargetAttribute in Examples.
    ▪ Else add new subtree to branch
      Partition(Examples_v, TargetAttribute, Attributes − {A})
• Return Root

Artificial Neural Networks (ANNs)

Biologically inspired computational model:

1. **Simple** computational units (neurons).
2. **Highly interconnected** - connectionist view
3. **Vast parallel** computation, consider:
   - Human brain has $\sim 10^{11}$ neurons
   - Slow computational units, switching time $\sim 10^{-3}$ sec (compared to the computer $>10^{-10}$ sec)
   - Yet, you can recognize a face in $\sim 10^{-1}$ sec
   - This implies only about 100 sequential, computational neuron steps - this seems too low for something as complicated as recognizing a face

$\Rightarrow$ **Parallel processing**

ANNs are naturally parallel - each neuron is a self-contained computational unit that depends only on its inputs.
Artificial Neural Networks

\[
\begin{align*}
\text{Input Layer} & \quad \text{Hidden Layer} & \quad \text{Output Layer} \\
X_n & \quad X_{n-1} & \quad \ldots & \quad X_2 & \quad X_1 & \quad X_0 \\
\end{align*}
\]

\[\sum_{i=0}^{n} w_i x_i \]

\[o = \sigma(\text{net}) = \frac{1}{1 + e^{-\text{net}}}\]
Artificial Neural Networks

Feed-forward with Backpropagation

Input Layer  Hidden Layer  Output Layer

Signal Feed-forward  Error Backpropagation
Artificial Neural Networks

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

Signal Feed-forward

Error Backpropagation

- Outlook
- Temperature
- Humidity
- Windy
- PlayTennis = Yes
- PlayTennis = No
Hidden Layer Representations

Target Function:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>00000100</td>
</tr>
<tr>
<td>00000010</td>
<td>00000010</td>
</tr>
<tr>
<td>00000001</td>
<td>00000001</td>
</tr>
</tbody>
</table>

Can this be learned?
Hidden Layer Representations

Hidden layers allow a network to invent appropriate internal representations.
Representational Power

- Every bounded continuous function can be approximated with arbitrarily small error by a network with one hidden layer.
- Any function can be approximated to arbitrary accuracy by a network with two hidden layers.
ANN Models

• ANNs are called *sub-symbolic* learners:
  • We do not get a nice representation like a tree
  • All we obtain is a trained network that can make classification decisions
Model Selection

- Typically we have a choice of many different models.
- Most often we choose the *simplest one* that *fits* the data:
  - Occam’s Razor
  - Law of parsimony
  - Invented by 14th century logician William of Ockham
- For decision trees we choose the tree with the least number of nodes.
- For artificial neural networks we choose the network with the least number of hidden layer elements.
Experiments

- You will be constructing both decision trees and artificial neural networks.
- You will need the following data sets:
  - weather.nominal.arff
  - iris.arff
- We will use Weka - a machine learning/data mining tool set.

Weka is available here: http://www.cs.waikato.ac.nz/ml/weka

Data sets are available at my homepage: http://homepage.cs.uri.edu/faculty/hamel/
WEKA

Weka GUI Chooser

Program Visualization Tools Help

Applications

Explorer

Experimenter

KnowledgeFlow

Simple CLI
Reading a Data File
Building a Decision Tree

Selecting a tree model: Choose → trees → J48
Building a Decision Tree
Building an ANN

- Load the data as before (this step is not necessary if you still have the data loaded)
- Then select MultilayerPerceptrons from the functions menu point
- To see the options of ANNs you need to click on the text box of the ANN itself, then set the GUI selector to true and hit OK
Building an ANN

Choose → functions → MultilayerPerceptron
- Set GUI to True
- Set hiddenLayers to the number of nodes in the hidden layer
- Set nominalToBinaryFilter to False
- Press ‘OK’
- The press ‘Start’
- This will show you the ANN
Press ‘Start’ to start training the ANN.
• Press ‘Accept’ to accept the trained network
• You can change the training by increasing or reducing the number of training epochs
Building an ANN
Experiments

- **Weather.nominal.arff**
  - Build tree
  - Build ANN with default settings
  - Build ANN with only 1 node in the hidden layer

- **Iris.arff**
  - Build tree
  - Build ANN with default settings
  - Build ANN with 10,000 training epochs
  - Build ANN with a single hidden layer node

- In all the above experiments note the misclassifications of the models.
- Which model(s) would you choose to represent each data set?