Machine Learning

Approach based on Decision Trees
• **Decision Tree Learning**
  – Practical *inductive inference* method
  – Same goal as *Candidate-Elimination algorithm*
    • Find *Boolean function* of attributes
    • Decision trees can be extended to *functions with more than two output values*.
  – Widely used
  – Robust to noise
  – Can handle disjunctive (OR’s) expressions
  – Completely expressive hypothesis space
  – Easily interpretable (tree structure, if-then rules)
Shall we play tennis today?

<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>
**Decision Tree for PlayTennis**

- **Classification**
  - Classifies instances into one of a discrete set of possible categories
  - Learned function represented by tree
  - Each node in tree is a test on some attribute of an instance
  - Branches represent values of attributes
  - Follow the tree from root to leaves to find the output value.

**Shall we play tennis today?**
• The tree itself forms hypothesis
  – Disjunction (OR’s) of conjunctions (AND’s)
  – Each path from root to leaf forms conjunction
    of constraints on attributes
  – Separate branches are disjunctions

• Example from *PlayTennis* decision tree:

  (Outlook=Sunny ∧ Humidity=Normal)
  \[ \vee \]
  (Outlook=Overcast)
  \[ \vee \]
  (Outlook=Rain ∧ Wind=Weak)
Types of problems decision tree learning is good for:

- Instances represented by attribute-value pairs
  - For algorithm in book, attributes take on a small number of discrete values
  - Can be extended to real-valued attributes
    - (numerical data)
- Target function has discrete output values
- Algorithm in book assumes Boolean functions
- Can be extended to multiple output values
– **Hypothesis space** can include **disjunctive expressions**.
   
   • In fact, hypothesis space is complete space of **finite discrete-valued functions**

– **Robust** to imperfect training data

   • **classification errors**
   
   • errors **in attribute values**
   
   • **missing attribute values**

• **Examples:**

  – **Equipment diagnosis**
  
  – **Medical** diagnosis
  
  – Credit card risk analysis
  
  – Robot movement
  
  – **Pattern Recognition**

     • face recognition
     
     • hexapod walking gates
• **Algorithms used:**
  
  – ID3 Quinlan (1986)
  – C4.5 Quinlan (1993)
  – C5.0 Quinlan
  – Cubist Quinlan
  – CART Classification and regression trees Breiman (1984)

• **ID3 is algorithm discussed in textbook**
  – Simple, but representative
  – Source code publicly available
• **ID3 Algorithm**
  – Top-down, greedy search through space of possible decision trees
    • Remember, decision trees represent hypotheses, so this is a search through hypothesis space.
  – What is top-down?
    • How to start tree?
      – What attribute should represent the root?
    • As you proceed down tree, choose attribute for each successive node.
  • **No backtracking:**
    – So, algorithm proceeds from top to bottom
– What is a greedy search?
  • At each step, make decision which makes greatest improvement in whatever you are trying optimize.
  • Do not backtrack (unless you hit a dead end)
  • This type of search is likely not to be a globally optimum solution, but generally works well.

– What are we really doing here?
  • At each node of tree, make decision on which attribute best classifies training data at that point.
  • Never backtrack (in ID3)
  • Do this for each branch of tree.
  • End result will be tree structure representing a hypothesis which works best for the training data.
Question?
How do you determine which attribute best classifies data?

Answer: Entropy!

• **Information gain:**
  – Statistical quantity measuring how well an attribute classifies the data.
  • Calculate the information gain for each attribute.
  • Choose attribute with greatest information gain.
But how do you measure information?

- Claude Shannon in 1948 at Bell Labs established the field of information theory.
- Mathematical function, Entropy, measures information content of random process:
  - Takes on largest value when events are equiprobable.
  - Takes on smallest value when only one event has non-zero probability.
- For two states:
  - Positive examples and Negative examples from set $S$:
    $$H(S) = -p_+\log_2(p_+) - p_-\log_2(p_-)$$

Entropy of set $S$ denoted by $H(S)$
Largest entropy

Boolean functions with the same number of ones and zeros have largest entropy

- $S$ is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in $S$
- $p_\ominus$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_\ominus \log_2 p_\ominus$$
In general:
– For an ensemble of random events: \{A_1,A_2,\ldots,A_n\}, occurring with probabilities: \(z = \{P(A_1), P(A_2), \ldots, P(A_n)\}\)

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

(Note: \(l = \sum_{i=1}^{n} P(A_i)\) and \(0 \leq P(A_i) \leq 1\))

If you consider the self-information of event, \(i\), to be: \(-\log_2(P(A_i))\)
Entropy is **weighted average** of information carried by each event.

**Does this make sense?**
– If an event conveys information, **that means it’s a surprise.**

– If an event always occurs, $P(A_i)=1$, then it carries no information. $-\log_2(1) = 0$

– If an event rarely occurs (e.g. $P(A_i)=0.001$), it carries a lot of info. $-\log_2(0.001) = 9.97$

– **The less likely the event, the more the information it carries** since, for $0 \leq P(A_i) \leq 1$, $-\log_2(P(A_i))$ increases as $P(A_i)$ goes from 1 to 0.

– (Note: ignore events with $P(A_i)=0$ since they never occur.)
What about entropy?

– Is it a good measure of the information carried by an ensemble of events?
– If the events are equally probable, the entropy is maximum.

1) For N events, each occurring with probability $1/N$.

$$H = -\sum (1/N) \log_2(1/N) = -\log_2(1/N)$$

This is the maximum value.

(e.g. For $N=256$ (ascii characters) $-\log_2(1/256) = 8$ number of bits needed for characters. Base 2 logs measure information in bits.)

This is a good thing since an ensemble of equally probable events is as uncertain as it gets. (Remember, information corresponds to surprise - uncertainty.)
2) $H$ is a continuous function of the probabilities.
   - That is always a good thing.

3) If you sub-group events into compound events, the entropy calculated for these compound groups is the same.
   - That is good since the uncertainty is the same.

- It is a remarkable fact that the equation for entropy shown above (up to a multiplicative constant) is the only function which satisfies these three conditions.
Choice of base 2 log corresponds to choosing units of information (BIT’s).

Another remarkable thing:

- This is the same definition of entropy used in statistical mechanics for the measure of disorder.
- Corresponds to macroscopic thermodynamic quantity of Second Law of Thermodynamics.
The concept of a **quantitative measure for information content** plays an important role in many areas:

- **Data communications** (channel capacity)
- **Data compression** (limits on error-free encoding)

Entropy in a message corresponds to *minimum number of bits needed to encode that message*.

In our case, for a set of training data, the entropy measures the number of bits needed to encode classification for an instance.

- Use probabilities found from entire set of training data.
- \( \text{Prob(Class=Pos)} = \frac{\text{Num. of positive cases}}{\text{Total case}} \)
- \( \text{Prob(Class=Neg)} = \frac{\text{Num. of negative cases}}{\text{Total cases}} \)
(Back to the story of ID3)

- **Information gain** is our metric for how well one attribute $A_i$ classifies the training data.

- **Information gain** for a particular attribute $A_i$ = 
  Information about target function, given the value of that attribute. 
  (conditional entropy)

- Mathematical expression:

$$Gain(S, A_i) = H(S) - \sum_{v \in Values(A_i)} P(A_i = v) H(S_v)$$

Information gain

Entropy
• **ID3 algorithm (for boolean-valued function)**
  
  – Calculate the entroy for all training examples
    
    • positive and negative cases
    • \( p_+ = \frac{\text{#pos}}{\text{Tot}} \quad p_- = \frac{\text{#neg}}{\text{Tot}} \)
    • \( H(S) = -p_+\log_2(p_+) - p_-\log_2(p_-) \)
  
  – Determine which single attribute best classifies the training examples using information gain.
    
    • For each attribute find:
      
      \[
      \text{Gain}(S, A_i) = H(S) - \sum_{v \in \text{Values}(A_i)} P(A_i = v) H(S_v)
      \]

    • Use attribute with greatest information gain as a root
• **Example: PlayTennis**
  – Four attributes used for classification:
    • *Outlook* = {Sunny, Overcast, Rain}
    • *Temperature* = {Hot, Mild, Cool}
    • *Humidity* = {High, Normal}
    • *Wind* = {Weak, Strong}
  – One predicted (target) attribute (binary)
    • *PlayTennis* = {Yes, No}
  – Given 14 Training examples
    • 9 positive
    • 5 negative
## 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>
Step 1: Calculate entropy for all cases:

\[ H(S) = -(\frac{9}{14}) \cdot \log_2(\frac{9}{14}) - (\frac{5}{14}) \cdot \log_2(\frac{5}{14}) = 0.940 \]

Day | Outlook | Temperature | Humidity | Wind | PlayTennis
---|---------|-------------|----------|------|------------
D1 | Sunny   | Hot         | High     | Weak | No         
D2 | Sunny   | Hot         | High     | Strong| No         
D3 | Overcast| Hot         | High     | Weak | Yes        
D4 | Rain    | Mild        | High     | Weak | Yes        
D5 | Rain    | Cool        | Normal   | Weak | Yes        
D6 | Rain    | Cool        | Normal   | Strong| No         
D7 | Overcast| Cool        | Normal   | Strong| Yes        
D8 | Sunny   | Mild        | High     | Weak | No         
D9 | Sunny   | Cool        | Normal   | Weak | Yes        
D10| Rain    | Mild        | Normal   | Weak | Yes        
D11| Sunny   | Mild        | Normal   | Strong| Yes        
D12| Overcast| Mild        | High     | Strong| Yes        
D13| Overcast| Hot         | Normal   | Weak | Yes        
D14| Rain    | Mild        | High     | Strong| No         

14 cases

9 positive cases
• **Step 2:** Loop over all attributes, calculate gain:

  – **Attribute = Outlook**

  • Loop over values of *Outlook*

    *Outlook = Sunny*

    \[
    N_{\text{Pos}} = 2, \quad N_{\text{Neg}} = 3, \quad N_{\text{Tot}} = 5
    \]

    \[
    H(\text{Sunny}) = -(2/5) \log_2(2/5) - (3/5) \log_2(3/5) = 0.971
    \]

    *Outlook = Overcast*

    \[
    N_{\text{Pos}} = 4, \quad N_{\text{Neg}} = 0, \quad N_{\text{Tot}} = 4
    \]

    \[
    H(\text{Sunny}) = -(4/4) \log_2(4/4) = 0.00
    \]
Outlook = Rain

N_{Pos} = 3 \hspace{1cm} N_{Neg} = 2 \hspace{1cm} N_{Tot} = 5

H(Sunny) = -(3/5)*\log_2(3/5) - (2/5)*\log_2(2/5) = 0.971

• Calculate **Information Gain** for attribute Outlook

Gain(S, Outlook) = H(S) - \frac{N_{Sunny}}{N_{Tot}}*H(Sunny) \\
- \frac{N_{Over}}{N_{Tot}}*H(Overcast) \\
- \frac{N_{Rain}}{N_{Tot}}*H(Rainy)

Gain(S, Outlook) = 9.40 - (5/14)*0.971 - (4/14)*0 - (5/14)*0.971

Gain(S, Outlook) = 0.246

– **Attribute = Temperature**

• (Repeat process looping over \{Hot, Mild, Cool\})

Gain(S, Temperature) = 0.029
– Attribute = *Humidity*
  - (Repeat process looping over \{High, Normal\})
    \[
    \text{Gain}(S, \text{Humidity}) = 0.029
    \]

– Attribute = *Wind*
  - (Repeat process looping over \{Weak, Strong\})
    \[
    \text{Gain}(S, \text{Wind}) = 0.048
    \]

**Find attribute with greatest information gain:**

\[
\begin{align*}
\text{Gain}(S, \text{Outlook}) &= 0.246, \\
\text{Gain}(S, \text{Temperature}) &= 0.029 \\
\text{Gain}(S, \text{Humidity}) &= 0.029, \\
\text{Gain}(S, \text{Wind}) &= 0.048
\end{align*}
\]

\[\therefore \text{Outlook is root node of tree}\]
– Iterate algorithm to find attributes which best classify training examples under the values of the root node

– **Example continued**
  
  • Take three subsets:
    
    – *Outlook* = Sunny \( (N_{Tot} = 5) \)
    
    – *Outlook* = Overcast \( (N_{Tot} = 4) \)
    
    – *Outlook* = Rainy \( (N_{Tot} = 5) \)
  
  • For each subset, repeat the above calculation looping over all attributes other than *Outlook*
– For example:

- **Outlook** = Sunny  \( (N_{Pos} = 2, N_{Neg} = 3, N_{Tot} = 5) \)  \( H = 0.971 \)
  - **Temp** = Hot  \( (N_{Pos} = 0, N_{Neg} = 2, N_{Tot} = 2) \)  \( H = 0.0 \)
  - **Temp** = Mild  \( (N_{Pos} = 1, N_{Neg} = 1, N_{Tot} = 2) \)  \( H = 1.0 \)
  - **Temp** = Cool  \( (N_{Pos} = 1, N_{Neg} = 0, N_{Tot} = 1) \)  \( H = 0.0 \)

  \[
  \text{Gain}(S_{Sunny}, \text{Temperature}) = 0.971 - (2/5)*0 - (2/5)*1 - (1/5)*0
  \]

  \[
  \text{Gain}(S_{Sunny}, \text{Temperature}) = 0.571
  \]

  Similarly:

  \[
  \text{Gain}(S_{Sunny}, \text{Humidity}) = 0.971
  \]

  \[
  \text{Gain}(S_{Sunny}, \text{Wind}) = 0.020
  \]

  \[\therefore \text{Humidity classifies } \text{Outlook} = \text{Sunny} \] instances best and is placed as the node under Sunny outcome.

– Repeat this process for **Outlook** = Overcast & Rainy
– **Important:**
  - Attributes are excluded from consideration if they appear higher in the tree

– Process **continues for each new leaf node until:**
  - Every attribute **has already been included along path through the tree**

  *or*

  - Training examples associated with this leaf all **have same target attribute value.**
– End up with tree:

Decision Tree for *PlayTennis*

```
Outlook
  Sunny
    Humidity
      High
        No
      Normal
        Yes
  Overcast
  Rain
    Wind
      Strong
        No
        Yes
      Weak
```
– **Note:** In this example data was perfect.
  • No contradictions
  • Branches led to unambiguous *Yes, No* decisions
  • If there are contradictions take the majority vote
    – This handles *noisy data*.

– **Another note:**
  • Attributes are eliminated when they are assigned to a node and never reconsidered.
    – e.g. You would not go back and reconsider *Outlook* under *Humidity*

– **ID3 uses all of the training data at once**
  • Contrast to Candidate-Elimination
  • Can handle noisy data.
• What is the **hypothesis space** for decision tree learning?
  – Search through space of **all possible decision trees**
    • from simple to more complex guided by a heuristic: *information gain*
  – The space searched is complete space of finite, discrete-valued functions.
    • Includes disjunctive and conjunctive expressions
  – Method only **maintains one current hypothesis**
    • In contrast to Candidate-Elimination
  – **Not necessarily global optimum**
    • attributes eliminated when assigned to a node
    • No backtracking
    • Different trees are possible
• **Inductive Bias:** (restriction vs. preference)
  
  – ID3
    • searches *complete hypothesis space*
    • But, *incomplete search* through this space looking for simplest tree
    • This is called a **preference** (or search) bias
  
  – **Candidate-Elimination**
    • Searches an *incomplete hypothesis space*
    • But, does a *complete search* finding all valid hypotheses
    • This is called a **restriction** (or language) bias
  
  – Typically, preference bias is better since you do not limit your search up-front by restricting hypothesis space considered.
• **Summary of ID3 Inductive Bias**

  – **Short trees** are preferred over long trees
    • It accepts the first tree it finds
  – Information gain heuristic
    • Places high information gain attributes near root
    • Greedy search method is an approximation to finding the **shortest tree**
  – Why would short trees be preferred?
    • Example of **Occam’s Razor**: Prefer simplest hypothesis consistent with the data.
      (Like Copernican vs. Ptoleemic view of Earth’s motion)
– Homework Assignment
  • Tom Mitchell’s software
  
  See:
  • http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html
  • Assignment #2 (on decision trees)
  • Software is at: http://www.cs.cmu.edu/afs/cs/project/theo-3/mlc/hw2/
    – Compiles with gcc compiler
    – Unfortunately, README is not there, but it’s easy to figure out:
      » After compiling, to run:
        dt [-s <random seed> ] <train %> <prune %> <test %> <SSV-format data file>
      » %train, %prune, & %test are percent of data to be used for training, pruning &
        testing. These are given as decimal fractions. To train on all data, use 1.0 0.0 0.0
    – Data sets for PlayTennis and Vote are include with code.
    – Also try the Restaurant example from Russell & Norvig
    – Also look at www.kdnuggets.com/    (Data Sets)
      Machine Learning Database Repository at UC Irvine  - (try “zoo” for fun)
Questions and Problems

1. Think how the method of finding best variable order for decision trees that we discussed here be adopted for:
   - ordering variables in binary and multi-valued decision diagrams
   - finding the bound set of variables for Ashenhurst and other functional decompositions

2. Find a more precise method for variable ordering in trees, that takes into account special function patterns recognized in data

3. Write a Lisp program for creating decision trees with entropy based variable selection.
– Sources
  • Tom Mitchell
  • Machine Learning, Mc Graw Hill 1997
  • Allan Moser