Why Learning?

- Learning is essential for unknown environments e.g., when designer lacks omniscience
- Learning is necessary in dynamic environments Agent can adapt to changes in environment not foreseen at design time
- Learning is useful as a system construction method Expose the agent to reality rather than trying to approximate it through equations etc.
- Learning modifies the agent's decision mechanisms to improve performance

Types of Learning

- **Supervised learning**: correct answers for each input is provided
  - E.g., decision trees, backprop neural networks
- **Unsupervised learning**: correct answers not given, must discover patterns in input data
  - E.g., clustering, principal component analysis
- **Reinforcement learning**: occasional rewards (or punishments) given
  - E.g., Q learning, MDPs

Inductive learning

A form of **Supervised Learning**:
Learn a function from examples

\( f \) is the target function. Examples are pairs \((x, f(x))\)

Problem: learn a function ("hypothesis") \( h \) such that \( h \approx f \) \((h\) approximates \( f \) as best as possible) given a training set of examples

(This is a highly simplified model of real learning:
Ignores prior knowledge
Assumes examples are given)
Inductive learning example

- Construct \( h \) to agree with \( f \) on training set.
  - \( h \) is consistent if it agrees with \( f \) on all training examples.
- E.g., curve fitting (regression):

![Graph showing curve fitting](image)

What about a quadratic function?

![Graph showing quadratic function](image)

Finally, a function that satisfies all!

![Graph showing function that satisfies all](image)
Inductive learning example

But so does this one...

Ockham’s razor principle

Ockham’s razor: prefer the simplest hypothesis consistent with data
Smooth blue function preferable over wiggly yellow one
If noise known to exist in this data, even linear might be better (the lowest x might be due to noise)

Decision Trees

Input: Description of an object or a situation through a set of attributes.
Output: a decision, that is the predicted output value for the input.
Both, input and output can be discrete or continuous.
Discrete-valued functions lead to classification problems.
Learning a continuous function is called regression.
Experience: “Good day for tennis”

Day Outlook | Temp | Humid | Wind | PlayTennis?
---|---|---|---|---
d1 | s | h | h | w | n

d2 | s | h | h | s | n

d3 | o | h | h | w | y

d4 | r | m | h | w | y

d5 | r | c | n | w | y

d6 | r | c | n | s | y

d7 | o | c | n | s | y

d8 | s | m | h | w | n

d9 | s | c | n | w | y

d10 | r | m | n | w | y

d11 | s | m | n | s | y

d12 | o | m | h | s | y

d13 | o | h | n | w | y

d14 | r | m | h | s | n

Decision Tree Representation

Good day for tennis?
Leaves = classification
Arcs = choice of value
for parent attribute

Decision tree is equivalent to logic in disjunctive normal form
G-Day ⇔ (Sunny ∧ Normal) ∨ Overcast ∨ (Rain ∧ Weak)

DT Learning as Search

• Nodes
  Decision Trees

• Operators
  Tree Refinement: Sprouting the tree

• Initial node
  Smallest tree possible: a single leaf

• Heuristic?
  Information Gain

• Goal?
  Best tree possible (???)

How good?

[10+, 4-] Means:
correct on 10 examples
incorrect on 4 examples
Successors

To be decided:
- How to choose best attribute?
  - Information gain
  - Entropy (disorder)
- When to stop growing tree?

Which attribute should we use to split?

Disorder is bad
Homogeneity is good

Using information theory to quantify uncertainty
- Entropy measures the amount of uncertainty in a probability distribution
- Entropy (or Information Content) of an answer to a question with possible answers $v_1, \ldots, v_n$:
  \[
  I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} P(v_i) \log_2 P(v_i)
  \]
Entropy (disorder) is bad
Homogeneity is good

- Let $S$ be a set of examples
- $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$
  where $P$ is proportion of pos example
  and $N$ is proportion of neg examples
  and $0 \log 0 = 0$
- Example: $S$ has 10 pos and 4 neg
  $\text{Entropy}([10+, 4-]) = -(10/14) \log_2(10/14) - (4/14)\log_2(4/14)
  = 0.863$

Information Gain
- Measure of expected reduction in entropy
- Resulting from splitting along an attribute

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

Where $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$

Gain of Splitting on Wind

<table>
<thead>
<tr>
<th>Day</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d3</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d4</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d5</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d6</td>
<td>s</td>
<td>yes</td>
</tr>
<tr>
<td>d7</td>
<td>s</td>
<td>yes</td>
</tr>
<tr>
<td>d8</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d10</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>yes</td>
</tr>
<tr>
<td>d12</td>
<td>s</td>
<td>yes</td>
</tr>
<tr>
<td>d13</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d14</td>
<td>s</td>
<td>n</td>
</tr>
</tbody>
</table>

Values($\text{wind}$)=weak, strong
$S = [10+, 4-]$  
$S_{\text{weak}} = [6+, 2-]$  
$S_s = [3+, 3-]$

Gain($S, \text{wind}$)  
= Entropy($S$) - $\sum_{v \in \{\text{weak, s}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$

= Entropy($S$) - 8/14 Entropy($S_{\text{weak}}$) - 6/14 Entropy($S_s$)
= 0.863 - (8/14) 0.811 - (6/14) 1.00
= -0.029

Entropy of a set $S$: $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$

Gain of a split on $S$: $\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$
Evaluating Attributes

Gain(S, Humid) = 0.375
Gain(S, Outlook) = 0.401
Gain(S, Temp) = 0.029
Gain(S, Wind) = -0.029

Resulting Tree ....

Good day for tennis?

Outlook

Sunny

Overcast

Rain

No

Yes

No

[2+, 3-]

[4+]

[2+, 3-]

One Step Later...

Day | Temp | Humid | Wind | Tennis?
---|---|---|---|---
 1 | h | h | weak | n
 2 | h | h | s | n
 8 | m | h | weak | n
 9 | c | n | weak | yes
11 | m | n | s | yes

Outlook

Humidity

Normal

High

Yes

No

[2+]

[3-]
### Decision Tree Algorithm

**BuildTree**(TrainingData)

```
Split(TrainingData)
```

**Split**(D)

- If (all points in D are of the same class) Then Return
- For each attribute A
  - Evaluate splits on attribute A
  - Use best split to partition D into D1, D2
  - Split (D1)
  - Split (D2)

### Overfitting

- **DT is overfit** when exists another DT' and DT has **smaller** error on training examples, but DT has **bigger** error on test examples

- **Causes of overfitting**
  - Noisy data, or
  - Training set is too small

---

**Figure from scw.cohen**
Avoiding Overfitting

How can we avoid overfitting?

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

How to select “best” tree:

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

Reduced-Error Pruning

Split data into training and validation 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 one that most improves validation set accuracy

Effect of Reduced-Error Pruning

Other Decision Tree Features

- Can handle continuous data
  Input: Use threshold to split
  Output: Estimate linear function at each leaf
- Can handle missing values
  Use expectation taken from other samples