Decision Tree Learning

- Blue slides: Mitchell
- Turquoise slides: Alpaydin

Learn to approximate **discrete-valued** target functions.

Step-by-step decision making: It can learn **disjunctive expressions**: Hypothesis space is **completely expressive**, avoiding problems with restricted hypothesis spaces.

Inductive bias: **small trees** over large trees.

Decision Trees

- A popular **inductive inference** algorithm.
- Algorithms: ID3, ASSISTANT, C4.5, etc.
- Applications: medical diagnosis, assess credit risk of loan applicants, etc.

Each instance holds attribute values.

Instances are classified by filtering the attribute values down the decision tree, down to a leaf which gives the final answer.

Internal nodes: attribute names or attribute values. Branching occurs at attribute nodes.
Tree Uses Nodes, and Leaves

Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, $x_i$
    - Numeric $x_i$: Binary split: $x_i > w_m$
    - Discrete $x_i$: $n$-way split for $n$ possible values
  - Multivariate: Uses all attributes, $x$

- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; $r$ average, or local fit
  - Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

Decision Trees: What They Represent

- Each path from root to leaf is a conjunctions of constraints on the attribute values.

\[
(\text{Outlook} = \text{Sunny} \land \text{Humidity} = \text{Normal})
\]
\[\lor\]
\[
(\text{Outlook} = \text{Overcast})
\]
\[\lor\]
\[
(\text{Outlook} = \text{Rain} \land \text{Wind} = \text{Weak})
\]

Appropriate Tasks for Decision Trees

Good at classification problems where:

- Instances are represented by attribute-value pairs.
- The target function has discrete output values.
- Disjunctive descriptions may be required.
- The training data may contain errors.
- The training data may contain missing attribute values.
Constructing Decision Trees from Examples

- Given a set of examples (training set), both positive and negative, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to classify new instances of examples (either as yes or no).

Constructing Decision Trees: Trivial Solution

- A trivial solution is to explicitly construct paths for each given example. In this case, you will get a tree where the number of leaves is the same as the number of training examples.
- The problem with this approach is that it is not able to deal with situations where, some attribute values are missing or new kinds of situations arise.
- Consider that some attributes may not count much toward the final classification.

Finding a Concise Decision Tree

- Memorizing all cases may not be the best way.
- We want to extract a decision pattern that can describe a large number of cases in a concise way.
- In terms of a decision tree, we want to make as few tests as possible before reaching a decision, i.e. the depth of the tree should be shallow.

Finding a Concise Decision Tree (cont’d)

- Basic idea: pick up attributes that can clearly separate positive and negative cases.
- These attributes are more important than others: the final classification heavily depend on the value of these attributes.
Decision Tree Learning Algorithm: ID3

Main loop:
1. $A \leftarrow$ the “best” decision attribute for next node
2. Assign $A$ as decision attribute for node
3. For each value of $A$, create new descendant of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

ID3 stands for Iterative Dichotomizer 3

Choosing the Best Attribute

A1 or A2?

- With initial and final number of positive and negative examples based on the attribute just tested, we want to decide which attribute is better.
- How to quantitatively measure which one is better?

Information Theory (Informal Intro)

Given two events, $H$ and $T$ (Head and Tail):

- Rare (uncertain) events give more surprise:
  $H$ more surprising than $T$ if $P(H) < P(T)$
  $H$ more uncertain than $T$ if $P(H) < P(T)$
- How to represent “more surprising”, or “more uncertain”? $\text{Surprise}(H) > \text{Surprise}(T)$ if
  $$P(H) < P(T) \iff \frac{1}{P(H)} > \frac{1}{P(T)}$$
  $$\iff \log \left( \frac{1}{P(H)} \right) > \log \left( \frac{1}{P(T)} \right)$$
  $$\iff -\log (P(H)) > -\log (P(T))$$
Information Theory (Cont’d)

- $S$ 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 average uncertainty in $S$

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Uncertainty and Information

- By performing some query, if you go from state $S_1$ with entropy $E(S_1)$ to state $S_2$ with entropy $E(S_2)$, where $E(S_1) > E(S_2)$, your uncertainty has decreased.
- The amount by which uncertainty decreased, i.e., $E(S_1) - E(S_2)$, can be thought of as information you gained (information gain) through getting answers to your query.

Entropy and Code Length

- $Entropy(S) = \sum_{i \in C} -P_i \log_2(P_i)$
- Information theory: optimal length code assigns $-\log_2 p$ bits to a message having probability $p$.
  - Encode with short string for frequent messages (less surprising), and long string for rarely occurring messages (more surprising).
- So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of $S$:
  $$p_\oplus (-\log_2 p_\oplus) + p_\ominus (-\log_2 p_\ominus)$$
  $$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Entropy and Information Gain

- $Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} |S_v| \frac{|S_v|}{|E|} Entropy(S_v)$
- $C$: categories (classifications)
- $S$: set of examples
- $A$: a single attribute
- $S_v$: set of examples where attribute $A = v$
- $|X|$: cardinality of arbitrary set $X$. 
### Example

<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>Mid</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>Mid</td>
<td>High</td>
<td>Weak</td>
<td>Yes</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>Mid</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mid</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mid</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>Mid</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- Which attribute to test first?

### Choosing the Best Attribute

Which attribute is the best classifier?

- **Humidity**
  - $[3+,4-] [6+,1-]$
  - Gain ($S$, Humidity) = $0.940 - \frac{8}{14} 0.985 - \frac{6}{14} 1.0 = 0.048$

- **Wind**
  - $[6+,2-] [3+,3-]$
  - Gain ($S$, Wind) = $0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1.0 = 0.048$

- Initial entropy $= -\frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} = 0.94$.

- You can calculate the rest.

- Note: $0.0 \times \log 0.0 \equiv 0.0$ even though $\log 0.0$ is not defined.

### Partially Learned Tree

```
Sunny: [D1, D2, D8, D9, D11]
Overcast: [D3, D7, D12, D13]
Rain: [D4, D5, D6, D10, D14]
```

### Hypothesis Space Search in ID3

- Select next attribute, based on the remaining examples.

- At each branch, we make a decision regarding a particular attribute. Choice of an attribute directs the search toward a certain final hypothesis.
Hypothesis Space Search in ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can’t play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx “prefer shortest tree”

Inductive Bias in ID3

- ID3 is biased:
  - Not because of the restriction on the hypothesis space, but
  - Because of the preference for a particular hypothesis.
- Such an inductive bias is called Occam’s razor: The most likely hypothesis is the simplest one that is consistent with all observations.

Accuracy of Decision Trees

- Divide examples into training and test sets.
- Train using the training set.
- Measure accuracy of resulting decision tree on the test set.

Overfitting:

- Given a hypothesis space $H$, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$ such that $h'$ is worse than $h$ on the training set but $h'$ is better than $h$ over the entire distribution of instances.
- Can be due to noise in data.
**Issue: Noise**

![Decision Tree Diagram]

- What if \(\{\text{Outlook} = \text{Sunny}, \text{Temp} = \text{Hot}, \text{Humidity} = \text{Normal}, \text{Wind} = \text{Strong}, \text{Play} = \text{No}\}\) was added as a training example?

- Further elaboration of the above tree becomes necessary.

- The resulting tree will fit the training data plus the noise, but it may perform poorly on the true instance distribution.

**Overcoming Overfitting**

- Stop early.
- Allow overfitting, then post-prune tree.
- Use separate set of examples not used in training to monitor performance on unobserved data (validation set).
- Use all available data, but perform statistical test to estimate chance of improving.
- Use explicit measure of complexity of encoding, and put a bound on tree size.

**Regression Trees**

- Error at node \(m\):
  \[
  b_m(x) = \begin{cases} 
  1 & \text{if} \ x \in X_m : x \ \text{reaches node} \ m \\
  0 & \text{otherwise}
  \end{cases}
  \]
  \[
  E_m = \frac{1}{N_m} \sum_{t} (r^t - g_m) b_m(x^t) 
  g_m = \frac{\sum_{t} b_m(x^t) r^t}{\sum_{t} b_m(x^t)}
  \]

- After splitting:
  \[
  b_{mj}(x) = \begin{cases} 
  1 & \text{if} \ x \in X_{mj} : x \ \text{reaches node} \ m \ \text{and branch} \ j \\
  0 & \text{otherwise}
  \end{cases}
  \]
  \[
  E_m = \frac{1}{N_m} \sum_{j} \sum_{t} (r^t - g_{mj}) b_{mj}(x^t) 
  g_{mj} = \frac{\sum_{t} b_{mj}(x^t) r^t}{\sum_{t} b_{mj}(x^t)}
  \]

**Model Selection in Trees**

- Error at node \(m\):
  \[
  b_m(x) = \begin{cases} 
  1 & \text{if} \ x \in X_m : x \ \text{reaches node} \ m \\
  0 & \text{otherwise}
  \end{cases}
  \]
  \[
  E_m = \frac{1}{N_m} \sum_{t} (r^t - g_m) b_m(x^t) 
  g_m = \frac{\sum_{t} b_m(x^t) r^t}{\sum_{t} b_m(x^t)}
  \]

- After splitting:
  \[
  b_{mj}(x) = \begin{cases} 
  1 & \text{if} \ x \in X_{mj} : x \ \text{reaches node} \ m \ \text{and branch} \ j \\
  0 & \text{otherwise}
  \end{cases}
  \]
  \[
  E_m = \frac{1}{N_m} \sum_{j} \sum_{t} (r^t - g_{mj}) b_{mj}(x^t) 
  g_{mj} = \frac{\sum_{t} b_{mj}(x^t) r^t}{\sum_{t} b_{mj}(x^t)}
  \]
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule covers an example if all terms of the rule evaluate to true for the example
- Sequential covering: Generate rules one at a time until all positive examples are covered
- IREP (Fürnkranz and Widmer, 1994), Ripper (Cohen, 1995)

Other Issues

- Continuous-valued attributes: dynamically define new discrete-valued attributes
- Multi-valued attributes with large number of possible values: Use measures other than information gain.
- Training examples with missing attribute values: Assign most common value, or assign with the occurring frequency.
- Attributes with different cost/weighting: Scale using the cost.