Classification: Decision Trees

These slides were assembled by Byron Boots, with grateful acknowledgement to Eric Eaton and the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution.
Last Time

- Common decomposition of machine learning, based on differences in inputs and outputs
  - **Supervised Learning:** Learn a function mapping inputs to outputs using labeled training data (you get instances/examples with both inputs and ground truth output)
  - **Unsupervised Learning:** Learn something about just data without any labels (harder!), for example clustering instances that are “similar”
  - **Reinforcement Learning:** Learn how to make decisions given a sparse reward

- ML is an interaction between:
  - the input data
  - input to the function (features/attributes of data)
  - the function (model) you choose, and
  - the optimization algorithm you use to explore space of functions

- We are new at this, so let’s start by learning if/then rules for classification!
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 $f$

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

Output: Hypothesis $h \in H$ that best approximates $f$
Sample Dataset (was Tennis Played?)

- Columns denote features $X_i$
- Rows denote labeled instances $\langle \mathbf{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Tree

- A possible decision tree for the data:

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

- Each internal node: test one attribute $X_i$
- Each branch from a node: selects one value for $X_i$
- Each leaf node: predict $Y$

Based on slide by Tom Mitchell
Decision Tree

• A possible decision tree for the data:

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

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

• Given a particular space of functions, you may not be able to represent everything.

• What **functions** can decision trees represent?

• Decision trees can represent any function of the input attributes!
  – Boolean operations (and, or, xor, etc.)?
    – Yes!
  – All boolean functions?
    – Yes!
Stages of (Batch) Machine Learning

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

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

**Train the model:**

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

**Apply the model to new data:**

- Given: new unlabeled instance \( x \sim D(X) \)

\[ y_{\text{prediction}} \leftarrow \text{model}.\text{predict}(x) \]
Basic Algorithm for Top-Down Learning of Decision Trees
[ID3, C4.5 by Quinlan]

node = root of decision tree

Main loop:
1. A ← the “best” decision attribute for the next node.
2. Assign A as decision attribute for node.
3. For each value of A, create a new descendant of 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

- Some possibilities are:
  - **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
Which test is more informative?

Split over whether Balance exceeds 50K

Split over whether applicant is employed

Based on slide by Pedro Domingos
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 = \( \sum_i - p_i \log_2 p_i \)

  \( p_i \) is the probability of class \( i \)
  Compute it as the proportion of class \( i \) in the set.

• Entropy comes from information theory. The higher the entropy the more the information content.

What does that mean for learning from examples?

Based on slide by Pedro Domingos
2-Class Cases:

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

- 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
Sample 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 $S$

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$
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.
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 is the expected reduction in entropy of target variable Y for data sample S, due to sorting.
Information Gain

$\text{Information Gain} = \text{entropy(parent)} - \left[\text{average entropy(children)}\right]

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

\[
\text{child entropy} = -\left(\frac{13}{17} \cdot \log_2 \frac{13}{17}\right) - \left(\frac{4}{17} \cdot \log_2 \frac{4}{17}\right) = 0.787
\]

\[
\text{child entropy} = -\left(\frac{1}{13} \cdot \log_2 \frac{1}{13}\right) - \left(\frac{12}{13} \cdot \log_2 \frac{12}{13}\right) = 0.391
\]

\[
\text{Entire population (30 instances)}
\]

\[
\text{17 instances}
\]

\[
\text{13 instances}
\]

(Weighted) Average Entropy of Children

\[
\text{(Weighted) Average Entropy of Children} = \left(\frac{17}{30} \cdot 0.787\right) + \left(\frac{13}{30} \cdot 0.391\right) = 0.615
\]

Information Gain = $0.996 - 0.615 = 0.38$

Based on slide by Pedro Domingos
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?

- **Humidity**
  - **High**
    - [3, 4-]
    - \( E = 0.985 \)
  - **Normal**
    - [6, 1-]
    - \( E = 0.592 \)

- **Wind**
  - **Weak**
    - [6, 2-]
    - \( E = 0.811 \)
  - **Strong**
    - [3, 3-]
    - \( E = 1.00 \)

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

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

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

\[ Gain(S_{sunny}, \text{Humidity}) = 0.970 - \left( \frac{3}{5} \right) 0.0 - \left( \frac{2}{5} \right) 0.0 = 0.970 \]

\[ Gain(S_{sunny}, \text{Temperature}) = 0.970 - \left( \frac{2}{5} \right) 0.0 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{1}{5} \right) 0.0 = 0.570 \]

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

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

ID3: ID3 is a decision tree learning algorithm designed to create a model for classifying data. It is a predictive model that is primarily used for classification problems. ID3 uses an information-theoretic criterion known as the Shannon entropy to select the best attribute for splitting the data, which is calculated using the formula:

\[ H(S) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \left( \frac{|S_i|}{|S|} \right) \]

where:
- \( H(S) \) is the entropy of the data set \( S \)
- \( c \) is the number of classes
- \( |S_i| \) is the number of instances of class \( i \)
- \( |S| \) is the total number of instances

Entropy measures the impurity of the data set. The lower the entropy, the purer the data set. ID3 selects the attribute with the lowest entropy for splitting the data, as this attribute provides the most information about the target variable. This process is repeated recursively until the tree is fully grown or until a stopping criterion is met.
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
Overfitting in Decision Trees

• 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 in Decision Trees

• 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:

![Decision Tree Diagram]

What would be the effect of adding:

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

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 Trees

![Graph showing accuracy of decision trees vs. size of tree. The graph plots two curves: one for training data and one for test data. The accuracy increases as the size of the tree increases, but the test data curve starts to decline after a certain size, indicating overfitting.](image)
Avoiding Overfitting in Decision Trees

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 (heuristic: simpler is better)
Reduced-Error Pruning

Split training data further 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

```
<table>
<thead>
<tr>
<th>Color</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Validation

```
<table>
<thead>
<tr>
<th>Color</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>blue</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

If we had simply predicted the majority class (negative), we make 2 errors instead of 4.

Based on Example from M. desJardins & T. Finin
Effect of Reduced-Error Pruning

On training data it looks great

But that’s not the case for the test (validation) data

Based on Slide by Pedro Domingos
Effect of Reduced-Error Pruning

The tree is pruned back to the red line where it gives more accurate results on the test data.
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)
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