Introduction to Machine Learning

CS4375 --- Spring 2016
Decision Tree Learning

Reading:
Sections 18.2-18.3, R&N
Sections 3.1-3.4, Mitchell

Decision Tree Example

- Three variables:
  - Attribute 1: Hair = (blond, dark)
  - Attribute 2: Height = (tall, short)
  - Class: Country = (Gromland, Polvia)

Training data:
(B,T,P)
(B,T,P)
(B,S,G)
(D,S,G)
(D,T,G)
(B,S,G)

Decision Trees

Decision Trees are classifiers for instances represented as features vectors. Nodes are (equality and inequality) tests for feature values. There is one branch for each value of the feature. Leaves specify the categories (labels). Can categorize instances into multiple disjoint categories.
General Case (Discrete Attributes)

- We have \( R \) observations from training data.
  - Each observation has \( M \) attributes \( X_1, \ldots, X_M \).
  - Each \( X_i \) can take \( N \) distinct discrete values.
  - Each observation has a class attribute \( Y \) with \( C \) distinct (discrete) values.
- Problem: Construct a sequence of tests on the attributes such that, given a new input \( (x'_1, \ldots, x'_M) \), the class attribute \( y \) is correctly predicted.

\[
\begin{array}{c|c|c|c|c|c}
 & X_1 & \cdots & X_M & Y \\
\hline
\text{Input data} & \cdots & \cdots & \cdots & \cdots & \cdots \\
\hline
\text{Data 1} & x_1 & \cdots & x_M & y \\
\hline
\text{Data 2} & \cdots & \cdots & \cdots & \cdots & \cdots \\
\hline
\text{Data } R & \cdots & \cdots & \cdots & \cdots & \cdots \\
\hline
\text{Training Data} & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{array}
\]

\( X = \) attributes of training data \((RxM)\), \( Y = \) Class of training data \((R)\).

Decision Tree Example

The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:
- \((0.2,0.8)\) is classified as \((0.8,0.2)\) is classified as...
**General Case (Continuous Attributes)**

- We have $R$ observations from training data.
- Each observation has $M$ attributes $X_1, \ldots, X_M$.
- Each $X_i$ can take $N$ continuous values.
- Each observation has a class attribute $Y$ with $C$ distinct (discrete) values.
- Problem: Construct a sequence of tests of the form $X_i < t_i$ on the attributes such that, given a new input $(x'_1, \ldots, x'_M)$, the class attribute $Y$ is correctly predicted.

<table>
<thead>
<tr>
<th>$X_i$</th>
<th>$X_{i+1}$</th>
<th>$X_M$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x'_1$</td>
<td>$x'_2$</td>
<td>$\ldots$</td>
<td>$x'_M$</td>
</tr>
</tbody>
</table>

**Basic Questions**

- How to choose the attribute/value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?

**General Decision Tree (Continuous Attributes)**

- How to choose the attribute/value to split on at each level of the tree?

- Two classes (red circles/green crosses)
- Two attributes: $X_1$ and $X_2$
- 11 points in training data
- Goal: Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples.
How to choose the attribute/value to split on at each level of the tree?

We want to find the most compact, smallest size tree (Occam’s razor), that classifies the training data correctly. We want to find the split choices that will get us the fastest to pure nodes.

This node is “pure” because there is only one class left → No ambiguity in the class label

This node is almost “pure” → Little ambiguity in the class label

These nodes contain a mixture of classes → Do not disambiguate between the classes

Entropy

• Entropy is a measure of the impurity of a distribution, defined as:

\[ H = \sum_{i=1}^{n} -P_i \log_2 P_i \]

• \( P_i \) = probability of occurrence of value \( i \)
  – High entropy → All the classes are (nearly) equally likely
  – Low entropy → A few classes are likely; most of the classes are rarely observed
    → assume 0 log₂ 0 = 0
Entropy

The entropy captures the degree of “purity” of the distribution.

Example Entropy Calculation

1. \( N_A = 1 \)
2. \( N_B = 6 \)
3. \( p_A = \frac{N_A}{N_A+N_B} = \frac{1}{7} \)
4. \( p_B = \frac{N_B}{N_A+N_B} = \frac{6}{7} \)
5. \( H_1 = -p_A \log_2 p_A - p_B \log_2 p_B \)
6. \( = 0.59 \)
7. \( H_2 = -p_A \log_2 p_A - p_B \log_2 p_B \)
8. \( = 0.97 \)

\( H_1 < H_2 \Rightarrow (2) \) less pure than (1)

Conditional Entropy

Entropy before splitting: \( H \)

After splitting, a fraction \( P_L \) of the data goes to the left node, which has entropy \( H_L \)

After splitting, a fraction \( P_R \) of the data goes to the right node, which has entropy \( H_R \)

The average entropy (or “conditional entropy”) after splitting

\[ H_L \times P_L + H_R \times P_R \]

Information Gain

We want nodes as pure as possible

\( \rightarrow \) We want to reduce the entropy as much as possible

\( \rightarrow \) We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

Maximize:

\[ IG = H - (H_L \times P_L + H_R \times P_R) \]
Notations

• Entropy: $H(Y) = \text{Entropy of the distribution of classes at a node}$

• Conditional Entropy:
  – *Discrete*: $H(Y|X_j) = \text{Entropy after splitting with respect to variable } j$
  – *Continuous*: $H(Y|X_j, t) = \text{Entropy after splitting with respect to variable } j \text{ with threshold } t$

• Information gain:
  – *Discrete*: $IG(Y|X_j) = H(Y) - H(Y|X_j)$
  – *Continuous*: $IG(Y|X_j, t) = H(Y) - H(Y|X_j, t)$

Another Illustrative 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>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Another Illustrative Example

```
Outlook
  Sunny  Overcast  Rain
     1,2,8,9,11  3,7,12,13  4,5,6,10,14
     2+,3-      4+,0-      3+,2-
Humidity  Wind
   High     No     Normal  Yes
   Strong   No     Weak   Yes
```

Basic Questions

- How to choose the attribute/value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

Pure and Impure Leaves and When to Stop Splitting

- All the data in the node comes from a single class. We declare the node to be a leaf and stop splitting. This leaf will output the class of the data it contains.
- Several data points have exactly the same attributes even though they are not from the same class. We cannot split any further. We still declare the node to be a leaf, but it will output the class that is the majority of the classes in the node (in this example, ‘B’).

Decision Tree Algorithm (Discrete Attributes)

- \texttt{LearnTree}(X, Y)
  - Input:
    - Set \( X \) of \( R \) training vectors, each containing the values \((x_1, \ldots, x_M)\) of \( M \) attributes \((X_1, \ldots, X_M)\)
    - A vector \( Y \) of \( R \) elements, where \( y_j = \text{class of the } j\text{th datapoint} \)
  - If all the datapoints in \( X \) have the same class value \( y \)
    - Return a leaf node that predicts \( y \) as output
  - If all the datapoints in \( X \) have the same attribute value \((x_1, \ldots, x_M)\)
    - Return a leaf node that predicts the majority of the class values in \( Y \) as output
  - Try all the possible attributes \( X_j \) and choose the one, \( j^* \), for which \( \text{IG}(Y|X_j) \) is maximum
  - For every possible value \( v \) of \( X_j \):
    - \( X_v, Y_v = \) set of datapoints for which \( x_j = v \) and corresponding classes
    - Child, \( \leftarrow \text{LearnTree}(X_v, Y_v) \)
Decision Tree Algorithm (Continuous Attributes)

• LearnTree(X, Y)
  – Input:
    • Set $X$ of $R$ training vectors, each containing the values $(x_{1},...,x_{M})$ of $M$ attributes $(X_{1},...,X_{M})$
    • A vector $Y$ of $R$ elements, where $y_{j}$ = class of the $j$th datapoint
  – If all the datapoints in $X$ have the same class value $y$
    • Return a leaf node that predicts $y$ as output
  – If all the datapoints in $X$ have the same attribute value $(x_{1},...,x_{M})$
    • Return a leaf node that predicts the majority of the class values in $Y$ as output
  – Try all the possible attributes $X_{j}$ and threshold $t$ and choose the one, $j^*$, for which IG($Y | X_{j}, t$) is maximum
    – $X_{L}, Y_{L}$ = set of datapoints for which $x_{j}^* < t$ and corresponding classes
    – $X_{H}, Y_{H}$ = set of datapoints for which $x_{j}^* >= t$ and corresponding classes
    – Left Child $\leftarrow$ LearnTree($X_{L}, Y_{L}$)
    – Right Child $\leftarrow$ LearnTree($X_{H}, Y_{H}$)

Expressiveness of Decision Trees

Can represent any Boolean function.
Can be rewritten as rules in Disjunctive Normal Form (DNF)

Decision Trees So Far

• Given $R$ observations from training data, each with $M$ attributes $X$ and a class attribute $Y$, construct a sequence of tests (decision tree) to predict the class attribute $Y$ from the attributes $X$
• Basic strategy for defining the tests ("when to split") $\rightarrow$ maximize the information gain on the training data set at each node of the tree
• Problem (next):
  – Evaluating the tree on training data is dangerous $\rightarrow$ overfitting

The Overfitting Problem (Example)

• Suppose that, in an ideal world, class B is everything such that $X_{2} >= 0.5$ and class A is everything with $X_{2} < 0.5$
• Note that attribute $X_{1}$ is irrelevant
• Seems like generating a decision tree would be trivial
The Overfitting Problem (Example)

- However, we collect training examples from the perfect world through some imperfect observation device.
- As a result, the training data is corrupted by *noise*.

The Overfitting Problem: Example

- Because of the noise, the resulting decision tree is far more complicated than it should be.
- This is because the learning algorithm tries to classify *all of the training set perfectly*. This is a fundamental problem in learning: *overfitting*.

The Overfitting Problem: Example

- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’.

The Overfitting Problem: Example

- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’.

It would be nice to identify automatically that splitting this node is stupid. Possible criterion: figure out that splitting this node will lead to a “complicated” tree suggesting noisy data.
The Overfitting Problem: Example

- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6, 0.9) is classified as ‘A’

Note that, even though the attribute $X_i$ is completely irrelevant in the original distribution, it is used to make the decision at that node.

Possible Overfitting Solution

- Grow tree based on training data (unpruned tree)
- Prune the tree by removing useless nodes based on additional test data (also known as validation data) not used for training

Unpruned decision tree from training data

Training data with the partitions induced by the decision tree (Notice the tiny regions at the top necessary to correctly classify the ‘A’ outliers!)
Unpruned decision tree from training data
Performance (% correctly classified)
Training: 100%
Test: 77.5%

Pruned decision tree from training data
Performance (% correctly classified)
Training: 95%
Test: 80%

Pruned decision tree from training data
Performance (% correctly classified)
Training: 80%
Test: 97.5%

Tree with best performance on test set
Performance on training set
Performance on test set
Locating the Overfitting Point

- General principle: As the complexity of the classifier increases (depth of the decision tree), the performance on the training data increases and the performance on the test data decreases when the classifier overfits the training data.

Decision Tree Pruning

- Construct the entire tree as before
- Starting at the leaves, recursively eliminate splits:
  - Evaluate performance of the tree on additional test data (also known as validation data)
  - Prune the tree if the classification performance increases by removing the split

Inductive Learning

- The decision tree approach is one example of an inductive learning technique:
- Suppose that data x is related to output y by a unknown function $y = f(x)$
- Suppose that we have observed training examples $\{(x_1, y_1), ..., (x_n, y_n)\}$
- **Inductive learning problem**: Recover a function $h$ (the “hypothesis”) such that $h(x) = f(x)$
- $y = h(x)$ predicts y from the input data x
- **The challenge**: The hypothesis space (the space of all hypothesis $h$ of a given form; for example the space of all of the possible decision trees for a set of $M$ attributes) is huge + many different hypotheses may agree with the training data.
Two stupid hypotheses that fit the training data perfectly
- What property should $h$ have?
- It should agree with the training data…
- But that can lead to arbitrarily complex hypotheses
  and there are many of them; which one should we choose?…

- Problems with a complex hypothesis:
  - It leads to completely wrong prediction on new test data…
  - It does not generalize beyond the training data…it overfits
    the training data

- Simplicity principle (Occam’s razor): “entities are not to be multiplied
  beyond necessity”
- The simpler hypothesis is preferred
- Compromise between:
  - Error on data under hypothesis $h$
  - Complexity of hypothesis $h$

- Different illustration, same concept….
Inductive Learning

- Decision tree is one example of inductive learning
- In many supervised learning algorithms, the goal is to minimize:
  \[ \text{Error on data} + \text{complexity of model} \]

Summary: Decision Trees

- Information Gain (IG) criterion for choosing splitting criteria at each level of the tree.
- Versions with continuous attributes and with discrete (categorical) attributes
- Basic tree learning algorithm leads to overfitting of the training data
- Pruning with validation data (not used for training)
- Example of inductive learning

Decision Trees on Real Problems

Must consider the following issues:
1. Assessing the performance of a learning algorithm
2. Inadequate attributes
3. Noise in the data
4. Missing values
5. Attributes with numeric values
6. Bias in attribute selection

Assessing the Performance of a Learning Algorithm

**Performance task:** predict the classifications of unseen examples

**Assessing prediction quality after tree construction:** check the classifier’s predictions on a test set.

But this requires that we get more data after we have trained the classifier.
Evaluation Methodology

1. Collect a large set of examples
2. Divide it into two disjoint sets: the training set and the test set
3. Use the learning algorithm with the training set to generate a hypothesis H
4. Measure the percentage of examples in the test set that are classified correctly by H
5. Repeat steps 1 to 4 for different sizes of training sets and different randomly selected training sets of each size.

Inadequate Attributes

- Cause inconsistent instances. Cannot find a classifier that is consistent with all of the training instances.

Solutions:
1. have each leaf node report the majority class
2. have each leaf report the estimated probabilities of each classification using the relative frequencies

Learning Curve

Noisy Data

Incorrect attribute values. Incorrect class labels.

Noise can be caused by many factors, such as:
1. Faulty measurements
2. Subjective interpretation

A further complication: may or may not know whether data is noisy.

Solution: pruning
Unknown Attribute Values

1. Throw away instances during training; during testing, try all paths, letting leaves vote.
2. Take most common value.
3. Take fractional value.

<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>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>???</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Attributes with Numeric Values

Look for best splits.

1. Sort values
2. Create Boolean variables out of mid points
3. Evaluate all of these using information gain formula

• Example:
  Length (L): 10 15 21 28 32 40 50
  Class:       - + + - + + -

Bias in Attribute Selection

Problem: Metric chooses higher branching attributes

Solution: Take into account the branching factor

<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>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>