Decision Trees, kNN Classifier

Machine Learning 10-601B
Seyoung Kim

Many of these slides are derived from Tom Mitchell and William Cohen. Thanks!
Beyond linearity

• Decision tree
  – What decision trees are
  – How to learn them

• Nearest neighbor classifier
Decision Tree Learning

Problem Setting:
• Set of possible instances $X, Y$
  – each instance $x$ in $X$ is a feature vector
    \[ x = \langle x_1, x_2 \ldots x_n \rangle \]
  – $Y$ is discrete-valued
• Unknown target function $f : X \rightarrow Y$
• Set of function hypotheses $H=\{ h | h : X \rightarrow Y \}$
  – each hypothesis $h$ is a decision tree

Input:
• Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function $f$

Output:
• Hypothesis $h \in H$ that best approximates target function $f$
A Decision tree for

\( f: \langle \text{Outlook}, \text{Humidity}, \text{Wind}, \text{Temp} \rangle \rightarrow \text{PlayTennis?} \)

Each internal node: test one discrete-valued attribute \( X_i \)
Each branch from a node: selects one value for \( X_i \)
Each leaf node: predict \( Y \) (or \( P(Y|X \in \text{leaf}) \))
A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

\[[833+,167-] \cdot 83+ \cdot 17-\]
Fetal_Presentation = 1: \[[822+,116-] \cdot 88+ \cdot 12-\]
  | Previous_Csection = 0: \[[767+,81-] \cdot 90+ \cdot 10-\]
  |   | Primiparous = 0: \[[399+,13-] \cdot 97+ \cdot 03-\]
  |   | Primiparous = 1: \[[368+,68-] \cdot 84+ \cdot 16-\]
  |   |   | Fetal_Distress = 0: \[[334+,47-] \cdot 88+ \cdot 12-\]
  |   |   |   | Birth_Weight < 3349: \[[201+,10.6-] \cdot 95+ \cdot \]
  |   |   |   | Birth_Weight >= 3349: \[[133+,36.4-] \cdot 78+ \]
  |   |   | Fetal_Distress = 1: \[[34+,21-] \cdot 62+ \cdot 38-\]
  | Previous_Csection = 1: \[[55+,35-] \cdot 61+ \cdot 39-\]
Fetal_Presentation = 2: \[[3+,29-] \cdot 11+ \cdot 89-\]
Fetal_Presentation = 3: \[[8+,22-] \cdot 27+ \cdot 73-\]
Decision tree learning

Induction of decision trees
JR Quinlan - Machine learning, 1986 - Springer
Abstract The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of...
Motivations for Decision Trees

• Often you can find a fairly accurate classifier which is **small** and **easy to understand**.
  – Sometimes this gives you useful **insight** into a problem

• Sometimes features **interact** in complicated ways
  – Trees can find interactions (e.g., “sunny and humid”) that linear classifiers can’t

• Trees are very **inexpensive at test time**
  – You don’t always **even need to compute all the features** of an example.
  – You can even build classifiers that take this into account....
  – Sometimes that’s important (e.g., “bloodPressure<100” vs “MRIScan=normal” might have different costs to compute).
Decision Tree Learning Algorithm
1. Given dataset $D$:
   - return $leaf(y)$ if all examples are in the same class $y$ ... or nearly so
   - pick the best split, on the best attribute $a$
     - $a$ or $not(a)$
     - $a=c_1$ or $a=c_2$ or ...
     - $a<\theta$ or $a \geq \theta$
     - $a$ in $\{c_1, ..., c_k\}$ or not
   - split the data into $D_1, D_2, \ldots, D_k$ and recursively build trees for each subset

2. “Prune” the tree
Most decision tree learning algorithms

1. Given dataset D:
   - return $\text{leaf}(y)$ if all examples are in the same class $y$ ... or nearly so...
   - pick the best split, on the best attribute $a$
     - $a = c_1$ or $a = c_2$ or ...
     - $a < \theta$ or $a \geq \theta$
     - $a$ or $\neg(a)$
     - $a$ in $\{c_1, ..., c_k\}$ or not
   - split the data into $D_1, D_2, ..., D_k$ and recursively build trees for each subset

2. “Prune” the tree

Popular splitting criterion: try to lower entropy of the $y$ labels on the resulting partition
- i.e., prefer splits that have very skewed distributions of labels
Picking the Best Split

- Which attribute is the best?

```
[29+, 35-]  A1=?
  t  f
[21+, 5-]  [8+, 30-]
```

```
[29+, 35-]  A2=?
  t  f
[18+, 33-]  [11+, 2-]
```
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 $S$

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

Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of $X$ (under most efficient code)

Why? Information theory:

- Most efficient possible code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random $X$ is:
  $$\sum_{i=1}^{n} P(X = i)(-\log_2 P(X = i))$$
Entropy

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 mutual information between input attribute $A$ and target variable $Y$

Information Gain is the expected reduction in entropy of target variable $Y$ for data sample $S$, due to sorting on variable $A$

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$

\[ [29+, 35-] \quad A1=? \]

\[ t \quad f \]

\[ [21+, 5-] \quad [8+, 30-] \]

\[ A2=? \]

\[ t \quad f \]

\[ [18+, 33-] \quad [11+, 2-] \]
## 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?

Gain \( (S, \text{ Humidity}) \)
\[
= .940 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 \\
= .151
\]

Gain \( (S, \text{ Wind}) \)
\[
= .940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.0 \\
= .048
\]
Which attribute should be tested here?

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

\[
\text{Gain} (S_{sunny}, \text{Humidity}) = .970 - \left( \frac{3}{5} \right) 0.0 - \left( \frac{2}{5} \right) 0.0 = .970
\]

\[
\text{Gain} (S_{sunny}, \text{Temperature}) = .970 - \left( \frac{2}{5} \right) 0.0 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{1}{5} \right) 0.0 = .570
\]

\[
\text{Gain} (S_{sunny}, \text{Wind}) = .970 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{3}{5} \right) .918 = .019
\]
Overfitting in Decision Trees

Consider adding noisy training example #15:
Sunny, Hot, Normal, Strong, PlayTennis = No
What effect on earlier tree?
Consider a hypothesis \( h \) and its

- Error rate over training data: \( error_{\text{train}}(h) \)
- True error rate over all data: \( error_{\text{true}}(h) \)

We say \( h \) overfits the training data if

\[
error_{\text{true}}(h) > error_{\text{train}}(h)
\]

Amount of overfitting =

\[
error_{\text{true}}(h) - error_{\text{train}}(h)
\]
Overfitting in Decision Tree Learning

![Graph showing accuracy vs. size of tree (number of nodes)]

On training data
On test data
Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune
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
- MDL: minimize
  \[
  \text{size}(\text{tree}) + \text{size}(\text{misclassifications}(\text{tree}))
  \]
Reduced-Error Pruning

Split data into training and validation set

Create tree that classifies training set correctly

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

• produces smallest version of most accurate subtree

• What if data is limited?
Effect of Reduced-Error Pruning
Continuous Valued Attributes

Create a discrete attribute to test continuous

- \( Temperature = 82.5 \)
- \( (Temperature > 72.3) = t, f \)

<table>
<thead>
<tr>
<th>Temperature</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>No</td>
</tr>
<tr>
<td>48</td>
<td>No</td>
</tr>
<tr>
<td>60</td>
<td>No</td>
</tr>
<tr>
<td>72</td>
<td>Yes</td>
</tr>
<tr>
<td>80</td>
<td>Yes</td>
</tr>
<tr>
<td>90</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
Attributes with Many Values

Problem:

- If attribute has many values, Gain will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where $S_i$ is subset of $S$ for which $A$ has value $v_i$
Decision Tree and Linear Classifier

• Decision trees don’t (typically) improve over linear classifiers when you have lots of features
• Sometimes fail badly on problems that linear classifiers perform well on
  – here’s an example
Another view of a decision tree
Another view of a decision tree

- Sepal length < 5.7
- Sepal width > 2.8
Another view of a decision tree
Another view of a decision tree
Questions to think about (1)

- Consider target function $f: \langle x_1, x_2 \rangle \rightarrow y$, where $x_1$ and $x_2$ are real-valued, $y$ is boolean. What is the set of decision surfaces describable with decision trees that use each attribute at most once?
Questions to think about (2)

• What is the relationship between learning decision trees, and learning IF-THEN rules

One of 18 learned rules:

If No previous vaginal delivery, and
   Abnormal 2nd Trimester Ultrasound, and
   Malpresentation at admission
Then Probability of Emergency C-Section is 0.6

Over training data: 26/41 = .63,
Over test data: 12/20 = .60
Questions to think about (3)

if O=sunny and H<= 70 then PLAY
else if O=sunny and H>70 then DON’T_PLAY
else if O=overcast then PLAY
else if O=rain and windy then DON’T_PLAY
else if O=rain and !windy then PLAY

One rule per leaf in the tree

Simpler rule set

if O=sunny and H> 70 then DON’T_PLAY
else if O=rain and windy then DON’T_PLAY
else PLAY
Nearest Neighbor Learning
Given a test example $\mathbf{x}$:
1. Find the $k$ training-set examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_k, y_k)$ that are closest to $\mathbf{x}$.
2. Predict the most frequent label in that set.
Breaking it down:

• To train:
  – save the data

• To test:
  – For each test example \( x \):

  1. Find the \( k \) training-set examples \((x_1,y_1),\ldots,(x_k,y_k)\) that are closest to \( x \).
  2. Predict the \textit{most frequent} label in that set.

Very fast!

...you might build some indices....

Prediction is relatively slow (compared to a linear classifier or decision tree)
What is the decision boundary for 1-NN?

Voronoi Diagram

Each cell $C_i$ is the set of all points that are closest to a particular example $x_i$. 
What is the decision boundary for 1-NN?

Voronoi Diagram
Effect of k on decision boundary

k=1

Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)
Some common variants

• Distance metrics:
  – Euclidean distance: $||x_1 - x_2||$
  – Cosine distance: $1 - \frac{<x_1,x_2>}{||x_1||*||x_2||}$
    • this is in $[0,1]$

• Weighted nearest neighbor:
  – Instead of most frequent $y$ in k-NN predict

$$\arg\max_y \sum_{(x_i,y)\in kNN(x)} sim(x_i, x)$$
You should know:

• Well posed function approximation problems:
  – Instance space, $X$
  – Sample of labeled training data $\{ <x^{(i)}, y^{(i)}> \}$
  – Hypothesis space, $H = \{ f: X \rightarrow Y \}$
  – Learning is a search/optimization problem over $H$

• Decision tree learning
  – Greedy top-down learning of decision trees (ID3, C4.5, ...)
  – Overfitting and tree/rule post-pruning
  – Extensions…

• kNN classifier
  – Non-linear decision boundary
  – Low-cost training, high-cost prediction