Decision Tree Learning: Part 1

Yingyu Liang
Computer Sciences 760
Fall 2017

http://pages.cs.wisc.edu/~yliang/cs760/

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, and Pedro Domingos.
Zoo of machine learning models

classification

- SVC
- Ensemble Classifiers
- KNeighbors Classifier
- SGD Classifier
- Naive Bayes
- Text Data
- Linear SVC

regression

- SGD Regressor
- Lasso
- ElasticNet
- RidgeRegression
- SVR(kernel=‘rbf’)
- SVR(kernel=‘linear’)

clustering

- Spectral Clustering
- GMM
- KMeans
- MiniBatch KMeans
- MeanShift
- VBGMM

dimensionality reduction

- Randomized PCA
- Isomap
- Spectral Embedding
- LLE
- kernel approximation
- NOT WORKING

>50 samples

<100K samples

NOT WORKING

get more data

NO

do you have labeled data

YES

predicting a category

number of categories known

<10K samples

>10K samples

NOT WORKING

just looking

<10K samples

tough luck

<10K samples

predicting a quantity

few features should be important

NOT WORKING

Note: only a subset of ML methods
Figure from scikit-learn.org
Even a subarea has its own collection

Figure from asimovinstitute.org
The lectures

organized according to different machine learning models/methods

1. supervised learning
   • non-parametric: decision tree, nearest neighbors
   • parametric
     • discriminative: linear/logistic regression, SVM, NN
     • generative: Naïve Bayes, Bayesian networks
2. unsupervised learning: clustering*, dimension reduction
3. reinforcement learning
4. other settings: ensemble, semi-supervised, active*

intertwined with experimental methodologies, theory, etc.

1. evaluation of learning algorithms
2. learning theory: PAC, bias-variance, mistake-bound
3. feature selection

*: if time permits
Goals for this lecture

you should understand the following concepts

• the decision tree representation
• the standard top-down approach to learning a tree
• Occam’s razor
• entropy and information gain
• types of decision-tree splits
A decision tree to predict heart disease

Each internal node tests one feature $x_i$

Each branch from an internal node represents one outcome of the test

Each leaf predicts $y$ or $P(y \mid x)$
Suppose $X_1 \ldots X_5$ are Boolean features, and $Y$ is also Boolean.

How would you represent the following with decision trees?

\[ Y = X_2 X_5 \]  (i.e., $Y = X_2 \land X_5$)

\[ Y = X_2 \lor X_5 \]

\[ Y = X_2 X_5 \lor X_3 \neg X_1 \]
History of decision tree learning

dates of seminal publications: work on these 2 was contemporaneous

many DT variants have been developed since CART and ID3

CART developed by Leo Breiman, Jerome Friedman, Charles Olshen, R.A. Stone

ID3, C4.5, C5.0 developed by Ross Quinlan
Top-down decision tree learning

\textbf{MakeSubtree}(set of training instances }D)\\
\quad C = \textbf{DetermineCandidateSplits}(D)\\
\quad \text{if stopping criteria met}\\
\quad \quad \text{make a leaf node } N\\
\quad \quad \text{determine class label/probabilities for } N\\
\quad \text{else}\\
\quad \quad \text{make an internal node } N\\
\quad S = \textbf{FindBestSplit}(D, C)\\
\quad \text{for each outcome } k \text{ of } S\\
\quad \quad D_k = \text{subset of instances that have outcome } k\\
\quad k^{th} \text{ child of } N = \textbf{MakeSubtree}(D_k)\\
\quad \text{return subtree rooted at } N
Candidate splits in ID3, C4.5

- splits on nominal features have one branch per value

```
thal
  normal
  fixed_defect
  reversible_defect
```

- splits on numeric features use a threshold

```
weight ≤ 35
  true
  false
```
Candidate splits on numeric features

given a set of training instances $D$ and a specific feature $X_i$

- sort the values of $X_i$ in $D$
- evaluate split thresholds in intervals between instances of different classes

<table>
<thead>
<tr>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
</tr>
<tr>
<td>35</td>
</tr>
</tbody>
</table>

• could use midpoint of each considered interval as the threshold
• C4.5 instead picks the largest value of $X_i$ in the entire training set that does not exceed the midpoint
Candidate splits on numeric features (in more detail)

// Run this subroutine for each numeric feature at each node of DT induction

DetermineCandidateNumericSplits(set of training instances $D$, feature $X_i$)

$C = \{\}$ // initialize set of candidate splits for feature $X_i$

$S = \text{partition instances in } D \text{ into sets } s_1 \ldots s_V \text{ where the instances in each set have the same value for } X_i$

let $v_j$ denote the value of $X_i$ for set $s_j$

sort the sets in $S$ using $v_j$ as the key for each $s_j$

for each pair of adjacent sets $s_j, s_{j+1}$ in sorted $S$

if $s_j$ and $s_{j+1}$ contain a pair of instances with different class labels

    // assume we’re using midpoints for splits

    add candidate split $X_i \leq (v_j + v_{j+1})/2$ to $C$

return $C$
Candidate splits

• instead of using \( k \)-way splits for \( k \)-valued features, could require binary splits on all discrete features (CART does this)
Finding the best split

• How should we select the best feature to split on at each step?

• Key hypothesis: the simplest tree that classifies the training instances accurately will work well on previously unseen instances
Occam’s razor

• attributed to 14th century William of Ockham

• “Nunquam ponenda est pluralitis sin necesitate”

• “Entities should not be multiplied beyond necessity”

• “when you have two competing theories that make exactly the same predictions, the simpler one is the better”
But a thousand years earlier, I said, “We consider it a good principle to explain the phenomena by the simplest hypothesis possible.”
Occam’s razor and decision trees

Why is Occam’s razor a reasonable heuristic for decision tree learning?

• there are fewer short models (i.e. small trees) than long ones
• a short model is unlikely to fit the training data well by chance
• a long model is more likely to fit the training data well coincidentally
Finding the best splits

• Can we find and return the smallest possible decision tree that accurately classifies the training set?

**NO! This is an NP-hard problem**

• Instead, we’ll use an information-theoretic heuristic to greedily choose splits
Information theory background

• consider a problem in which you are using a code to communicate information to a receiver

• example: as bikes go past, you are communicating the manufacturer of each bike
Information theory background

- suppose there are only four types of bikes
- we could use the following code

<table>
<thead>
<tr>
<th>type</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trek</td>
<td>11</td>
</tr>
<tr>
<td>Specialized</td>
<td>10</td>
</tr>
<tr>
<td>Cervelo</td>
<td>01</td>
</tr>
<tr>
<td>Serrota</td>
<td>00</td>
</tr>
</tbody>
</table>

- expected number of bits we have to communicate: 2 bits/bike
Information theory background

- we can do better if the bike types aren’t equiprobable
- optimal code uses $\log_2 P(y)$ bits for event with probability $P(y)$

<table>
<thead>
<tr>
<th>Type/probability</th>
<th># bits</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(\text{Trek}) = 0.5$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P(\text{Specialized}) = 0.25$</td>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>$P(\text{Cervelo}) = 0.125$</td>
<td>3</td>
<td>001</td>
</tr>
<tr>
<td>$P(\text{Serrota}) = 0.125$</td>
<td>3</td>
<td>000</td>
</tr>
</tbody>
</table>

- expected number of bits we have to communicate:
  1.75 bits/bike

$$- \sum_{y \in \text{values}(Y)} P(y) \log_2 P(y)$$
Entropy

- entropy is a measure of uncertainty associated with a random variable
- defined as the expected number of bits required to communicate the value of the variable

\[ H(Y) = - \sum_{y \in \text{values}(Y)} P(y) \log_2 P(y) \]

entropy function for binary variable
Conditional entropy

- What’s the entropy of $Y$ if we condition on some other variable $X$?

\[
H(Y \mid X) = \sum_{x \in \text{values}(X)} P(X = x) H(Y \mid X = x)
\]

where

\[
H(Y \mid X) = - \sum_{y \in \text{values}(Y)} P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x)
\]
Information gain
(a.k.a. mutual information)

- choosing splits in ID3: select the split $S$ that most reduces the conditional entropy of $Y$ for training set $D$

$$\text{InfoGain}(D,S) = H_D(Y) - H_D(Y | S)$$

$D$ indicates that we’re calculating probabilities using the specific sample $D$.
Relations between the concepts

Figure from wikipedia.org
## Information gain example

### PlayTennis: 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>
Information gain example

- What’s the information gain of splitting on Humidity?

\[
H_D(Y) = -\frac{9}{14} \log_2 \left( \frac{9}{14} \right) - \frac{5}{14} \log_2 \left( \frac{5}{14} \right) = 0.940
\]

\[
H_D(Y \mid \text{high}) = -\frac{3}{7} \log_2 \left( \frac{3}{7} \right) - \frac{4}{7} \log_2 \left( \frac{4}{7} \right) = 0.985
\]

\[
H_D(Y \mid \text{normal}) = -\frac{6}{7} \log_2 \left( \frac{6}{7} \right) - \frac{1}{7} \log_2 \left( \frac{1}{7} \right) = 0.592
\]

\[
\text{InfoGain}(D, \text{Humidity}) = H_D(Y) - H_D(Y \mid \text{Humidity})
\]

\[
= 0.940 - \left[ \frac{7}{14} (0.985) + \frac{7}{14} (0.592) \right]
\]

\[
= 0.151
\]
Information gain example

- Is it better to split on Humidity or Wind?

\[
H_D(Y \mid \text{weak}) = 0.811 \\
H_D(Y \mid \text{strong}) = 1.0
\]

\[
\text{InfoGain}(D, \text{Humidity}) = 0.940 - \left[ \frac{7}{14} (0.985) + \frac{7}{14} (0.592) \right] \\
= 0.151
\]

\[
\text{InfoGain}(D, \text{Wind}) = 0.940 - \left[ \frac{8}{14} (0.811) + \frac{6}{14} (1.0) \right] \\
= 0.048
\]
One limitation of information gain

• information gain is biased towards tests with many outcomes

• e.g. consider a feature that uniquely identifies each training instance
  – splitting on this feature would result in many branches, each of which is “pure” (has instances of only one class)
  – maximal information gain!
Gain ratio

• to address this limitation, C4.5 uses a splitting criterion called *gain ratio*

• gain ratio normalizes the information gain by the entropy of the split being considered

\[
\text{GainRatio}(D, S) = \frac{\text{InfoGain}(D, S)}{H_D(S)} = \frac{H_D(Y)}{H_D(Y | S)} \frac{H_D(Y | S)}{H_D(S)}
\]