Decision Trees

CS 760@UW-Madison
The lectures ahead

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, active, semi-supervised

intertwined with experimental methodologies, theory, etc.

1. evaluation of learning algorithms
2. learning theory: PAC, bias-variance, mistake-bound
3. feature selection
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
- test sets and unbiased estimates of accuracy
- overfitting
- early stopping and pruning
- validation sets
- regression trees
- probability estimation trees
Decision Tree Representation
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?

1. $Y = X_2 X_5$ (i.e., $Y = X_2 \land X_5$)
2. $Y = X_2 \lor X_5$
3. $Y = X_2 X_5 \lor X_3 \neg X_1$
Decision Tree Learning
History of decision tree learning

- Many DT variants have been developed since CART and ID3.

Dates of seminal publications:
- Work on these 2 was contemporaneous.
- AID (1963)
- THAID (1973)
- CHAID (1980)
- CART (1984)
- ID3 (1986)

- 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 \(D\))

\(C = \text{DetermineCandidateSplits}(D)\)

if stopping criteria met

make a leaf node \(N\)

determine class label/probabilities for \(N\)

else

make an internal node \(N\)

\(S = \text{FindBestSplit}(D, C)\)

for each outcome \(k\) of \(S\)

\(D_k = \text{subset of instances that have outcome } k\)

\(k^{th}\) child of \(N = \text{MakeSubtree}(D_k)\)

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

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

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

$C = \{\}$ \hspace{1cm} // 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

\hspace{1cm} // 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 Splits
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 14\textsuperscript{th} 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

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

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

$$H(Y|X) = \sum_{x \in \text{values}(X)} P(X = x) H(Y|X = x)$$

where

$$H(Y|X = x) = -\sum_{y \in \text{values}(Y)} P(Y = y|X = x) \log_2 P(Y = y|X = x)$$
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 $D$

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

$D$ 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?

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

$$H_D(Y | \text{high}) = - \frac{3}{7} \log_2 \left( \frac{3}{7} \right) - \frac{4}{7} \log_2 \left( \frac{4}{7} \right) = 0.985$$

$$H_D(Y | \text{normal}) = - \frac{6}{7} \log_2 \left( \frac{6}{7} \right) - \frac{1}{7} \log_2 \left( \frac{1}{7} \right) = 0.592$$

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

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

$$= 0.151$$
Information gain example

- Is it better to split on Humidity or Wind?

\[
\begin{align*}
H_D(Y | \text{weak}) &= 0.811 \\
H_D(Y | \text{strong}) &= 1.0 
\end{align*}
\]

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

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

\[
\text{GainRatio}(D, S) = \frac{\text{InfoGain}(D, S)}{H_D(S)} = \frac{H_D(Y) - H_D(Y \mid S)}{H_D(S)}
\]
Stopping criteria

We should form a leaf when

- all of the given subset of instances are of the same class
- we’ve exhausted all of the candidate splits

Is there a reason to stop earlier, or to prune back the tree?
How to assess the accuracy of a tree?

• can we just calculate the fraction of training instances that are correctly classified?

• consider a problem domain in which instances are assigned labels at random with $P(Y = t) = 0.5$
  
  • how accurate would a learned decision tree be on previously unseen instances?
  
  • how accurate would it be on its training set?
How can we assess the accuracy of a tree?

- to get an unbiased estimate of a learned model’s accuracy, we must use a set of instances that are held-aside during learning
- this is called a test set
Overfitting
Overfitting

- consider error of model $h$ over
  - training data: $\text{error}_\mathcal{D}(h)$
  - entire distribution of data: $\text{error}_\mathcal{D}(h)$

- model $h \in H$ overfits the training data if there is an alternative model $h' \in H$ such that

  \[
  \text{error}_\mathcal{D}(h) > \text{error}_\mathcal{D}(h')
  \]

  \[
  \text{error}_\mathcal{D}(h) < \text{error}_\mathcal{D}(h')
  \]
Example 1: overfitting with noisy data

suppose

- the target concept is $Y = X_1 \land X_2$
- there is noise in some feature values
- we’re given the following training set

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>...</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>...</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>...</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>...</td>
<td>f</td>
</tr>
</tbody>
</table>
Example 1: overfitting with noisy data

Correct tree

Tree that fits noisy training data
Example 2: overfitting with **noise-free** data

suppose

- the target concept is $Y = X_1 \land X_2$
- $P(X_3 = t) = 0.5$ for both classes
- $P(Y = t) = 0.67$
- we’re given the following training set

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>...</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>...</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>...</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>...</td>
<td>f</td>
</tr>
</tbody>
</table>
Example 2: overfitting with **noise-free** data

- because the training set is a limited sample, there might be (combinations of) features that are correlated with the target concept by chance
Overfitting in decision trees
Example 3: regression using polynomial

\[ t = \sin(2\pi x) + \epsilon \]

Figure from *Machine Learning and Pattern Recognition*, Bishop
Example 3: regression using polynomial

\[ t = \sin(2\pi x) + \epsilon \]

Regression using polynomial of degree \( M \)
Example 3: regression using polynomial

\[ t = \sin(2\pi x) + \epsilon \]
Example 3: regression using polynomial

\[ t = \sin(2\pi x) + \epsilon \]
Example 3: regression using polynomial

\[ t = \sin(2\pi x) + \epsilon \]
Example 3: regression using polynomial
General phenomenon

Figure from *Deep Learning*, Goodfellow, Bengio and Courville
Prevent overfitting

• cause: training error and expected error are different
  1. there may be noise in the training data
  2. training data is of limited size, resulting in difference from the true distribution
  3. larger the hypothesis class, easier to find a hypothesis that fits the difference between the training data and the true distribution

• prevent overfitting:
  1. cleaner training data help!
  2. more training data help!
  3. throwing away unnecessary hypotheses helps! (Occam’s Razor)
Avoiding overfitting in DT learning

two general strategies to avoid overfitting

1. *early stopping*: stop if further splitting not justified by a statistical test
   - Quinlan’s original approach in ID3

2. *post-pruning*: grow a large tree, then prune back some nodes
   - more robust to myopia of greedy tree learning
1. split given data into training and validation (tuning) sets
2. grow a complete tree
3. do until further pruning is harmful
   • evaluate impact on tuning-set accuracy of pruning each node
   • greedily remove the one that most improves tuning-set accuracy
Validation sets

- A validation set (a.k.a. tuning set) is a subset of the training set that is held aside
  - Not used for primary training process (e.g. tree growing)
  - But used to select among models (e.g. trees pruned to varying degrees)
Regression trees

- In a regression tree, leaves have functions that predict numeric values instead of class labels.
- The form of these functions depends on the method:
  - CART uses constants.
  - Some methods use linear functions.
Regression trees in CART

• CART does *least squares regression* which tries to minimize

$$\sum_{i=1}^{D} (y^{(i)} - \hat{y}^{(i)})^2$$

value predicted by tree for $i^{th}$ training instance (average value of $y$ for training instances reaching the leaf)

• at each internal node, CART chooses the split that most reduces this quantity
Probability estimation trees

- in a PE tree, leaves estimate the probability of each class
- could simply use training instances at a leaf to estimate probabilities, but ...
- smoothing is used to make estimates less extreme (we’ll revisit this topic when we cover Bayes nets)
**m-of-n splits**

- A few DT algorithms have used *m-of-n* splits [Murphy & Pazzani ‘91].
- Each split is constructed using a heuristic search process.
- This can result in smaller, easier to comprehend trees.

```
5 of {ROI[13] > 1.07, 
   vs_Yen_future < 0.39, 
   interest_rate[1] = false, 
   vs_Dem_future[2] > 0.17, 
   vs_Dem_future[3] < 0.17, 
   vs_SFr_future < 0.39, 
   Dem_USD_ex[8] < -1.02, 
   Dem_future[5] > -1.38, 
   Dem_USD_ex[3] < -0.10, 
   vs_SFr_future < 0.16}
```

- Test is satisfied if 5 of 10 conditions are true.

Tree for exchange rate prediction [Craven & Shavlik, 1997]
Searching for $m$-of-$n$ splits

$m$-of-$n$ splits are found via a hill-climbing search

- initial state: best 1-of-1 (ordinary) binary split
- evaluation function: information gain
- operators:

\[ m \text{-of-}n \quad \Rightarrow \quad m \text{-of-}(n+1) \]

1 of \{ $X_1$=t, $X_3$=f \} \quad \Rightarrow \quad 1 \text{ of } \{ X_1$=t, $X_3$=f, $X_7$=t \}

\[ m \text{-of-}n \quad \Rightarrow \quad (m+1)\text{-of-}(n+1) \]

1 of \{ $X_1$=t, $X_3$=f \} \quad \Rightarrow \quad 2 \text{ of } \{ X_1$=t, $X_3$=f, $X_7$=t \}
Lookahead

• most DT learning methods use a hill-climbing search
• a limitation of this approach is myopia: an important feature may not appear to be informative until used in conjunction with other features
• can potentially alleviate this limitation by using a lookahead search [Norton ‘89; Murphy & Salzberg ‘95]
• empirically, often doesn’t improve accuracy or tree size
Choosing best split in ordinary DT learning

\textbf{OrdinaryFindBestSplit}(set of training instances }D\text{, set of candidate splits }C\text{)

\begin{align*}
  \text{maxgain} &= -\infty \\
  \text{for each split } S \text{ in } C \\
  \text{gain} &= \text{InfoGain}(D, S) \\
  \text{if } \text{gain} > \text{maxgain} \\
  \text{maxgain} &= \text{gain} \\
  S_{\text{best}} &= S \\
  \text{return } S_{\text{best}}
\end{align*}
Choosing best split with lookahead (part 1)

LookaheadFindBestSplit(set of training instances $D$, set of candidate splits $C$)

$maxgain = -\infty$

for each split $S$ in $C$

$\text{gain} = \text{EvaluateSplit}(D, C, S)$

if $\text{gain} > maxgain$

$maxgain = \text{gain}$

$S_{best} = S$

return $S_{best}$
Choosing best split with lookahead (part 2)

**EvaluateSplit**(*D, C, S*)

if a split on *S* separates instances by class (i.e. \( H_D(Y \mid S) = 0 \))

// no need to split further
return \( H_D(Y) - H_D(Y \mid S) \)

else

for each outcome *k* of *S*

// see what the splits at the next level would be
\( D_k \) = subset of instances that have outcome *k*
\( S_k = \text{OrdinaryFindBestSplit}(D_k, C - S) \)

// return information gain that would result from this 2-level subtree
return

\[
H_D(Y) - \left( \sum_k \frac{|D_k|}{|D|} H_{D_k}(Y \mid S = k, S_k) \right)
\]
Calculating information gain with lookahead

Suppose that when considering Humidity as a split, we find that Wind and Temperature are the best features to split on at the next level.

We can assess the value of choosing Humidity as our split by

\[
H_D(Y) - \left( \frac{14}{23} H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind}) + \frac{9}{23} H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature}) \right)
\]
Calculating information gain with lookahead

Using the tree from the previous slide:

\[
\frac{14}{23} H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind}) + \frac{9}{23} H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature})
\]

\[
= \frac{5}{23} H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind} = \text{strong}) + \frac{9}{23} H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind} = \text{weak}) + \frac{4}{23} H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature} = \text{high}) + \frac{5}{23} H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature} = \text{low})
\]

\[
H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind} = \text{strong}) = -\frac{2}{5} \log \left( \frac{2}{5} \right) - \frac{3}{5} \log \left( \frac{3}{5} \right)
\]

\[
\vdots
\]
Comments on decision tree learning

• widely used approach
• many variations
• provides humanly comprehensible models when trees not too big
• insensitive to monotone transformations of numeric features
• standard methods learn axis-parallel hypotheses
• standard methods not suited to on-line setting
• usually not among most accurate learning methods

* although variants exist that are exceptions to this
THANK YOU

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