Decision Tree Learning

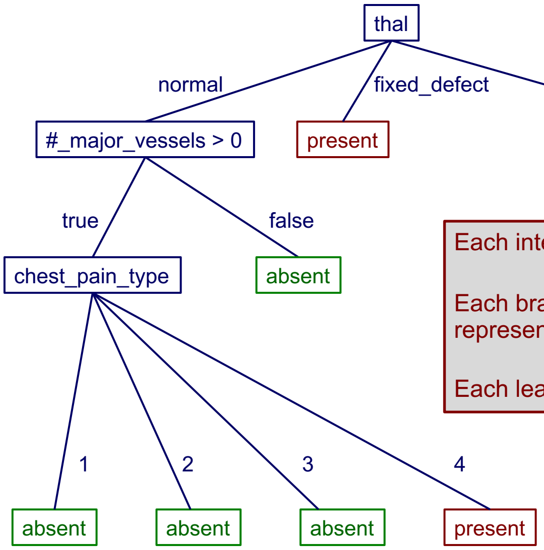
Mark Craven and David Page
Computer Sciences 760
Spring 2019

Goals for the 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
- test sets and unbiased estimates of accuracy
- overfitting
- early stopping and pruning
- tuning (validation) sets
- regression trees

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 $P(y \mid \boldsymbol{x})$

Decision tree exercise

Suppose $x_1 \ldots x_5$ are Boolean features, and y is also Boolean

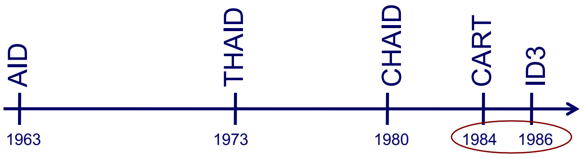
How would you represent the following with decision trees?

$$y = x_2 x_5 \quad \text{(i.e. } y = x_2 \wedge x_5 \text{)}$$

$$y = x_2 \vee x_5$$

$$y = x_2 x_5 \vee x_3 \neg x_1$$

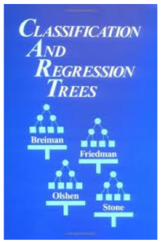
History of decision tree learning



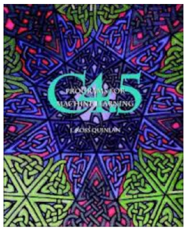
many DT variants have been developed since CART and ID3

dates of seminal publications: work on these 2 was contemporaneous

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

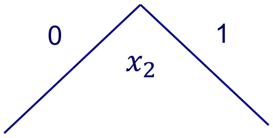


ID3, C4.5, C5.0 developed by Ross Quinlan



Key idea in DT learning: recursively split training data

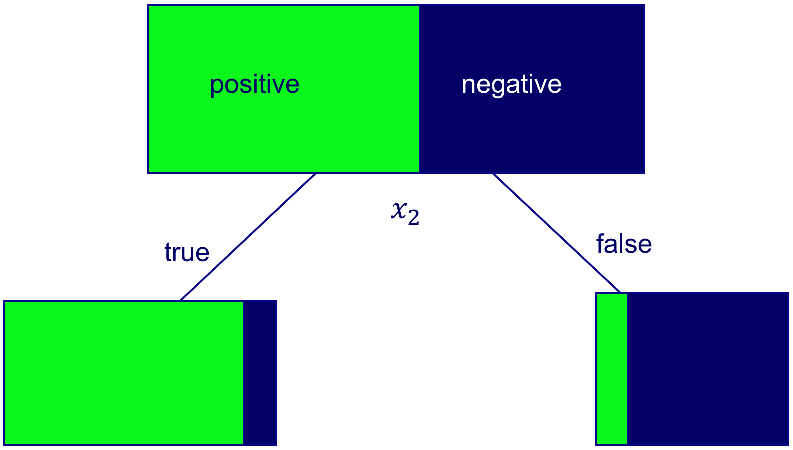
1, 1, 0, 0 +
0, 0, 1, 0 +
1, 1, 1, 0 +
0, 1, 1, 0 +
1, 1, 1, 1 +
0, 0, 1, 1 -
1, 0, 1, 0 -
0, 0, 0, 0 -
1, 0, 1, 1 -
0, 1, 1, 1 -



0, 0, 1, 0 +
0, 0, 1, 1 -
1, 0, 1, 0 -
0, 0, 0, 0 -
1, 0, 1, 1 -

1, 1, 0, 0 +
1, 1, 1, 0 +
0, 1, 1, 0 +
1, 1, 1, 1 +
0, 1, 1, 1 -

Key idea in DT learning: recursively split training data

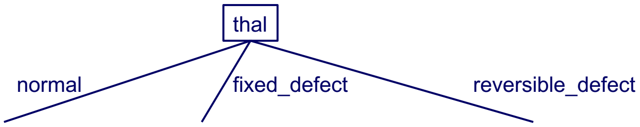


Top-down decision tree learning

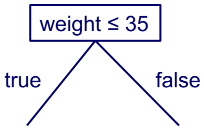
```
MakeSubtree(set of training instances $D$)
    $C$ = DetermineCandidateSplits($D$)
    if stopping criteria met
        make a leaf node $N$
        determine class label/probabilities for $N$
    else
        make an internal node $N$
        $S$ = FindBestSplit($D$, $C$)
        for each outcome $k$ of $S$
            $D_k$ = subset of instances that have outcome $k$
            $k^{th}$ child of $N$ = MakeSubtree($D_k$)
    return subtree rooted at $N$
```

Candidate splits in ID3, C4.5

- splits on nominal features have one branch per value



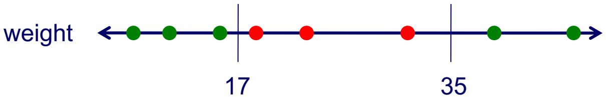
- splits on continuous features use a threshold

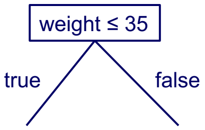


Candidate splits on continuous features

given a set of training instances D and a specific feature F

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

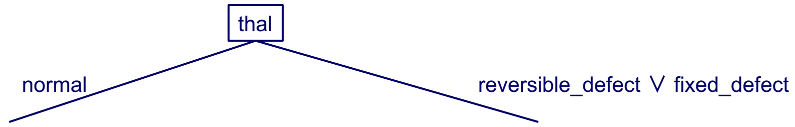


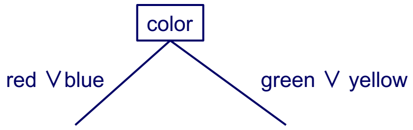


- could use midpoint of each considered interval as the threshold
- C4.5 instead picks the largest value of F in the entire training set that does not exceed the midpoint

Candidate splits

- instead of using k-way splits for k-valued features, could require binary splits on all discrete features (CART does this)





- Breiman et al. proved for the 2-class case, the optimal binary partition can be found considered only $O(k)$ possibilities instead of $O(2^k)$

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 examples accurately will work well on previously unseen examples

Occam’s razor

- attributed to 14th century William of Ockham
- “Nunquam ponenda est pluralitis sin necesitate”

↓

- “Entities should not be multiplied beyond necessity”
- “should proceed to simpler theories until simplicity can be traded for greater explanatory power”
- “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

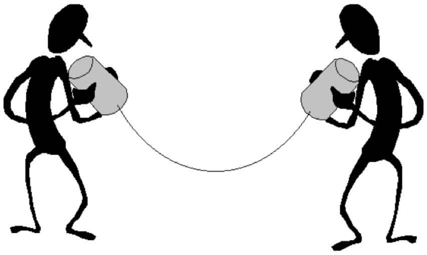
- Can we return the smallest possible decision tree that accurately classifies the training set?

  NO! This is an NP-hard problem
  [Hyafil & Rivest, *Information Processing Letters, 1976]*

- 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

type	code
Trek	11
Specialized	10
Cervelo	01
Serrota	00

- 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 $P(y)$

Type/probability	# bits	code
$P($Trek$) = 0.5$	1	1
$P($Specialized$) = 0.25$	2	01
$P($Cervelo$) = 0.125$	3	001
$P($Serrota$) = 0.125$	3	000

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

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

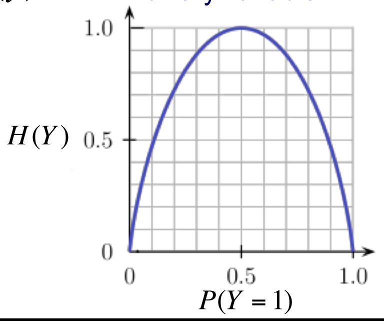
Entropy

- 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?

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

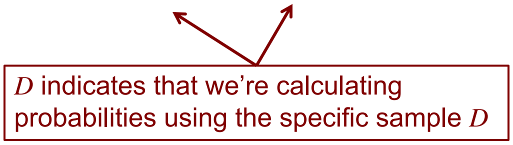
where

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

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

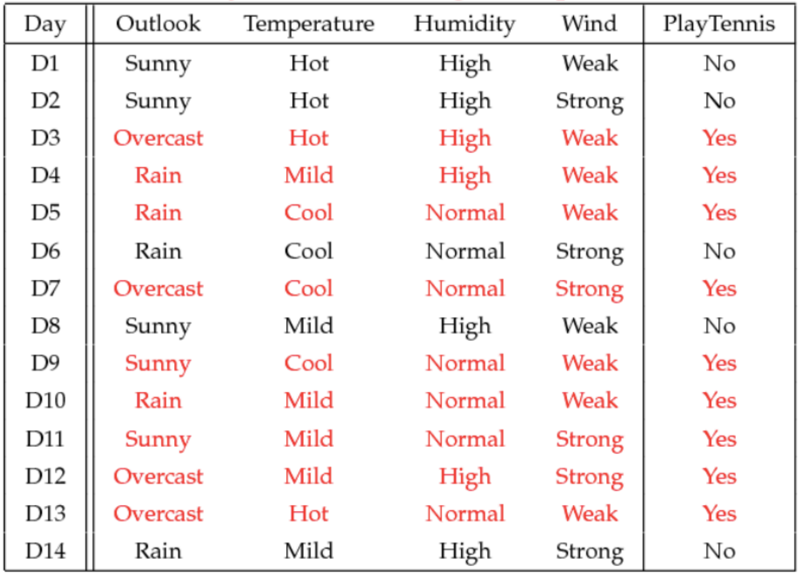
$$\text{InfoGain}(D,S) = H_D(Y) - H_D(Y \mid S)$$



D indicates that we're calculating probabilities using the specific sample D

Information gain example

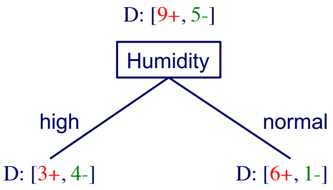
PlayTennis: training examples



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Information gain example

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



D: [9+, 5-]

Humidity

high

normal

D: [3+, 4-]

D: [6+, 1-]

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

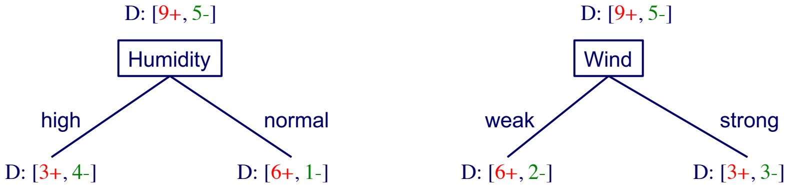
$$H_D(Y \mid \text{high}) = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$
$$= 0.985$$

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

$$\begin{aligned}\text{InfoGain}(D,\text{Humidity}) &= H_D(Y) - H_D(Y \mid \text{Humidity})\\ &= 0.940 - \left[\frac{7}{14}(0.985) + \frac{7}{14}(0.592)\right]\\ &= 0.151\end{aligned}$$

Information gain example

- Is it better to split on Humidity or Wind?



D: [9+, 5-]

Humidity

high

normal

D: [3+, 4-]

D: [6+, 1-]

D: [9+, 5-]

Wind

weak

strong

D: [6+, 2-]

D: [3+, 3-]

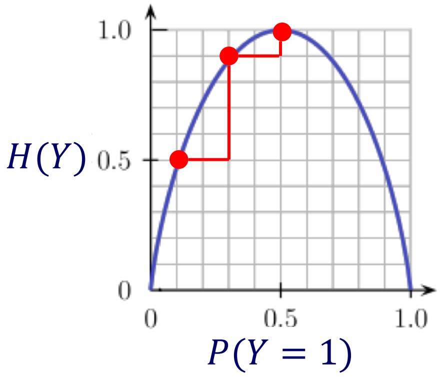
$H_D(Y \mid \text{weak}) = 0.811$ $\quad H_D(Y \mid \text{strong}) = 1.0$

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

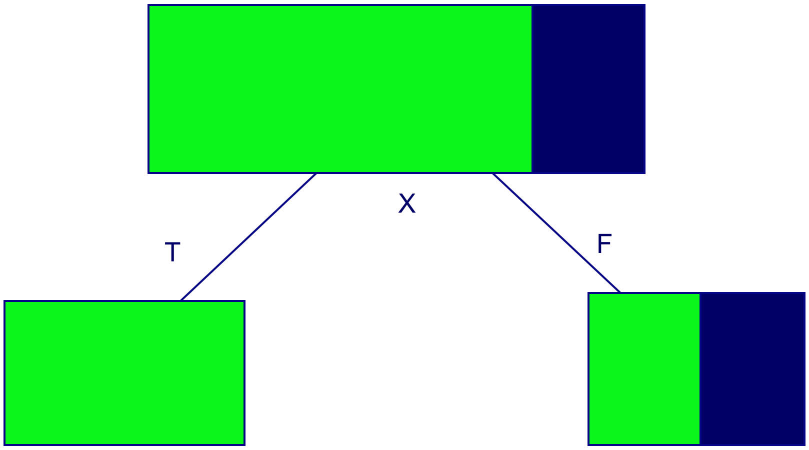
$$\text{InfoGain}(D, \text{Wind}) = 0.940 - \left[\frac{8}{14}(0.811) + \frac{6}{14}(1.0)\right]$$
$$= 0.048$$

Key property of entropy function

Equal change in $P(Y)$ yields bigger change in entropy if toward an extreme

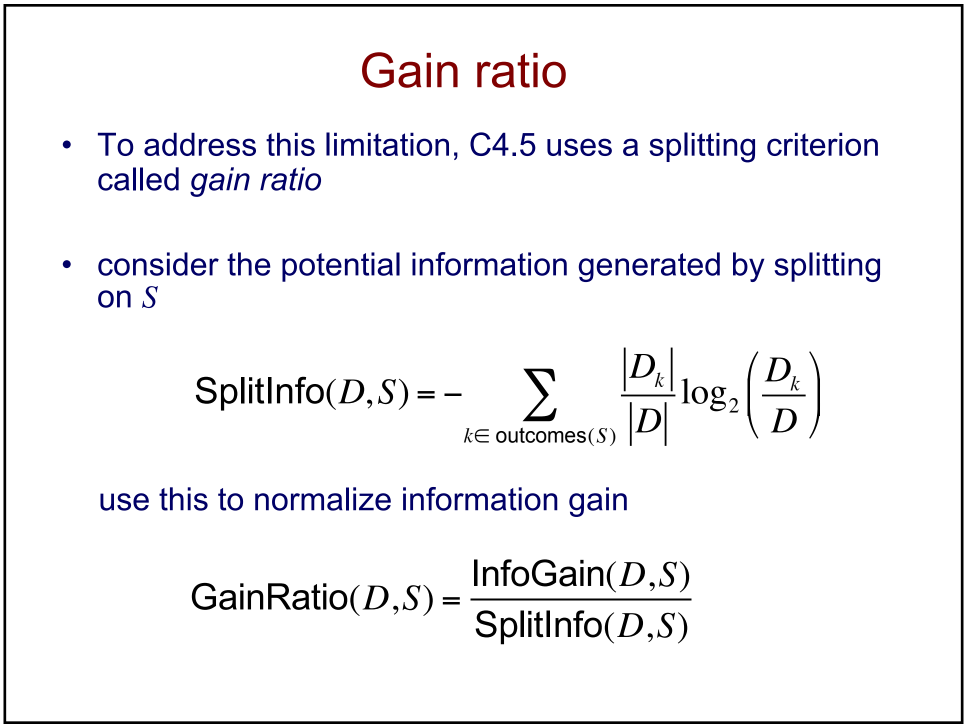


Means there is InfoGain in this split, though no gain in accuracy



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*
- consider the potential information generated by splitting on S

$$\text{SplitInfo}(D,S) = -\sum_{k \in \text{ outcomes}(S)} \frac{|D_k|}{|D|} \log_2\left(\frac{D_k}{D}\right)$$

use this to normalize information gain

$$\text{GainRatio}(D,S) = \frac{\text{InfoGain}(D,S)}{\text{SplitInfo}(D,S)}$$