Decision Tree Learning: Part 1

CS 760@UW-Madison

The lectures

organized according to different machine learning models/methods

1. supervised learning
   - non-parametric-function models: decision tree, nearest neighbors
   - parametric
     - discriminative: linear/logistic regression, SVM, neural networks
     - 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

*: if time permits

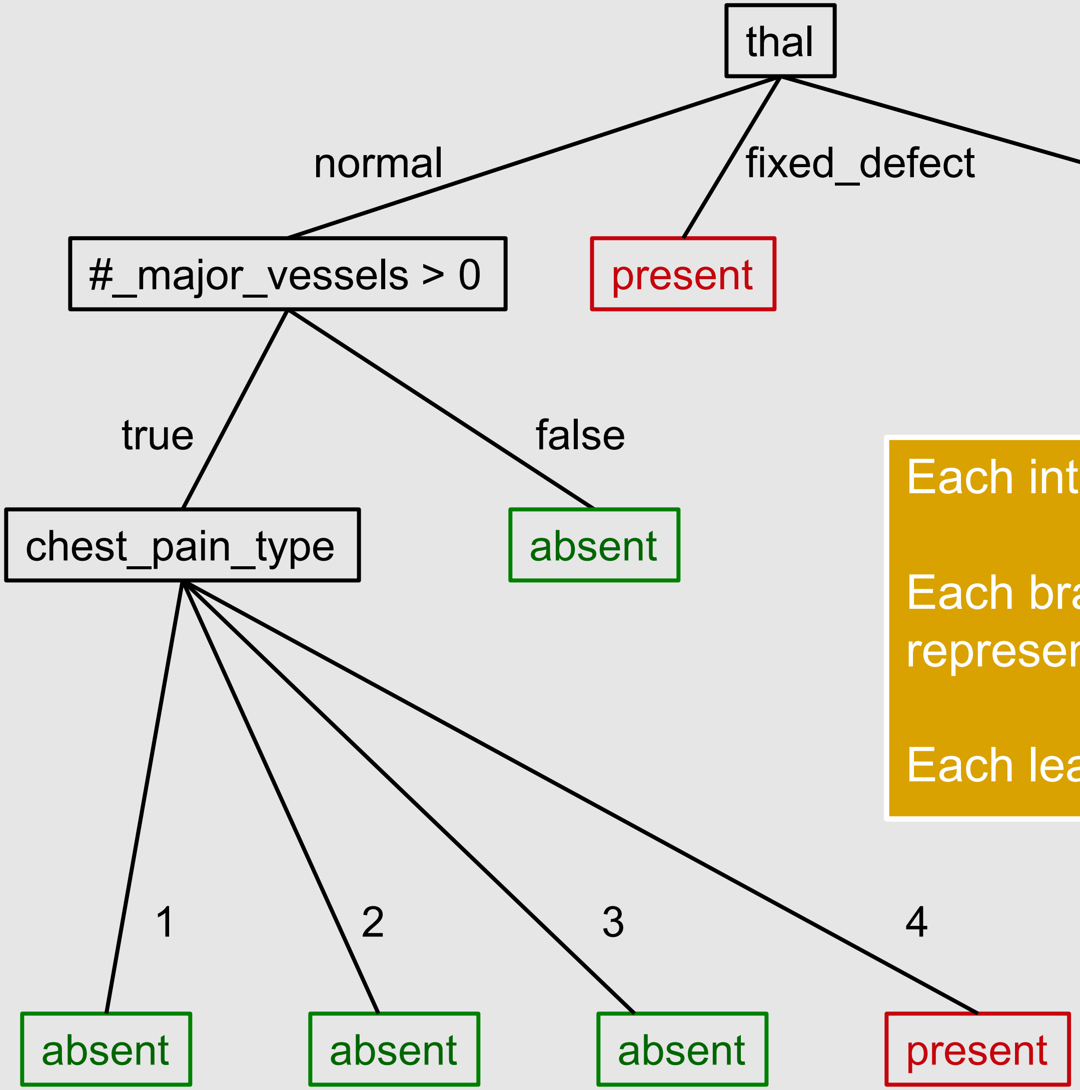
Goals for this lecture

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

Decision Tree Representation

A decision tree to predict heart disease

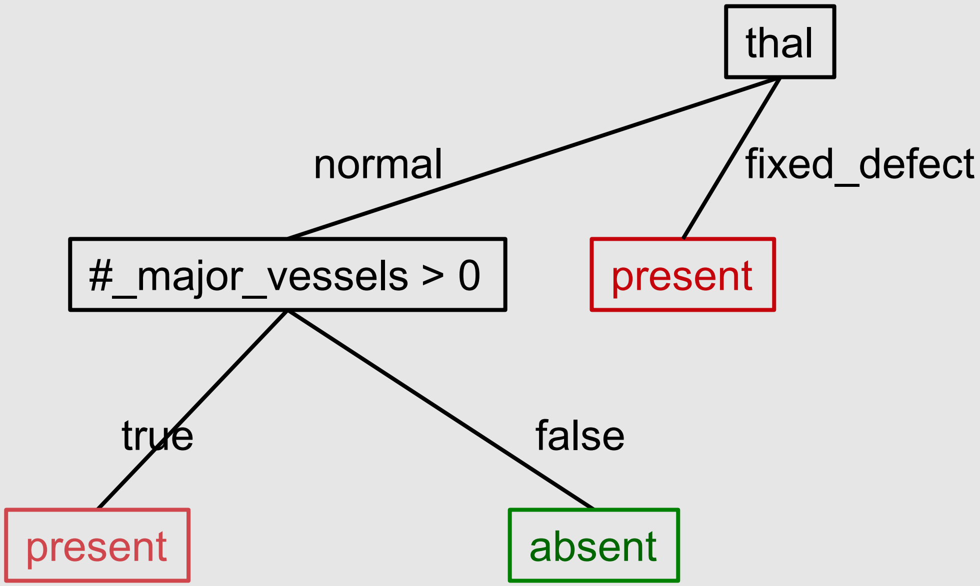


Each internal node tests one feature x_i

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

Each leaf predicts y or $P(y \mid \boldsymbol{x})$

Text description of decision trees



- thal = normal
  - [#_major_vessels > 0] = true: present
  - [#_major_vessels > 0] = false: absent
- thal = fixed_defect: present

Text description of decision trees

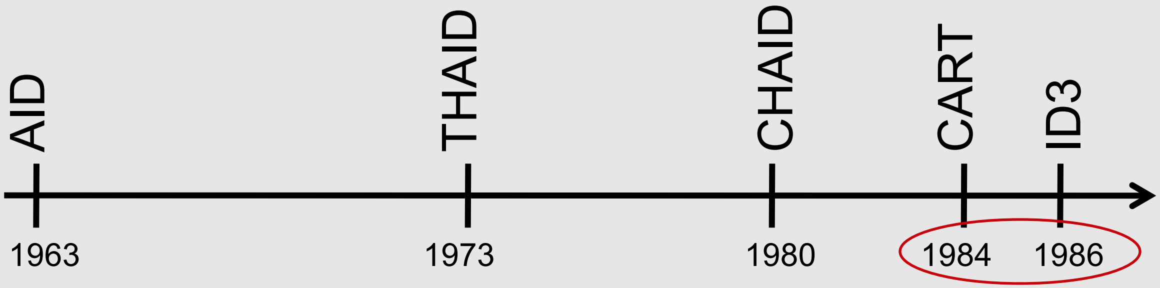
```
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (2160.0)
odor = l: e (400.0)
odor = m: p (36.0)
odor = n
|   spore-print-color = b: e (48.0)
|   spore-print-color = h: e (48.0)
|   spore-print-color = k: e (1296.0)
|   spore-print-color = n: e (1344.0)
|   spore-print-color = o: e (48.0)
|   spore-print-color = r: p (72.0)
|   spore-print-color = u: e (0.0)
|   spore-print-color = w
|   |   gill-size = b: e (528.0)
|   |   gill-size = n
|   |   |   gill-spacing = c: p (32.0)
|   |   |   gill-spacing = d: e (0.0)
|   |   |   gill-spacing = w
|   |   |   |   population = a: e (0.0)
|   |   |   |   population = c: p (16.0)
|   |   |   |   population = n: e (0.0)
|   |   |   |   population = s: e (0.0)
|   |   |   |   population = v: e (48.0)
|   |   |   |   population = y: e (0.0)
|   spore-print-color = y: e (48.0)
odor = p: p (256.0)
odor = s: p (576.0)
odor = y: p (576.0)
```

if odor=almond, predict edible

if odor=none $\wedge$
spore-print-color=white $\wedge$
gill-size=narrow $\wedge$
gill-spacing=crowded,
predict poisonous

Decision Tree Learning

History of decision tree learning

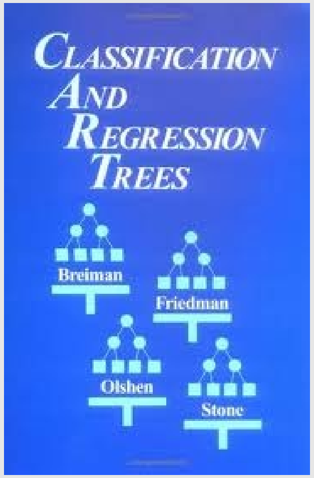


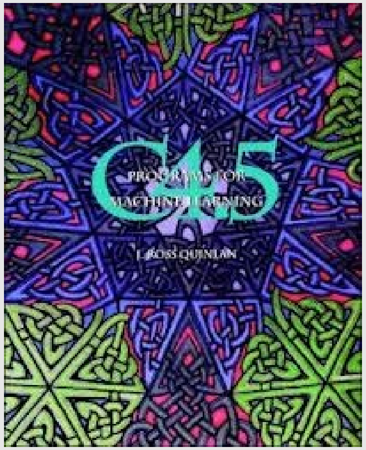
many DT variants have been developed since CART and ID3

dates of seminal publications: work on these 2 was contemporaneous

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

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

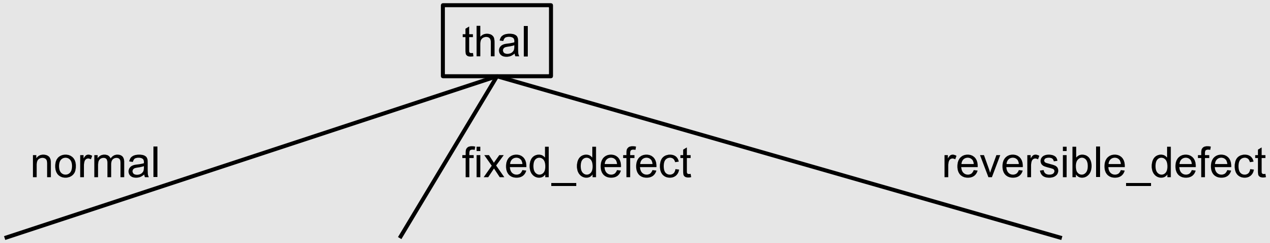
      D_k = subset of instances that have outcome k

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

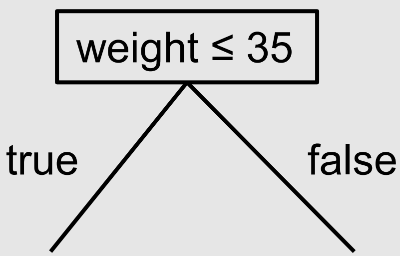
  return subtree rooted at N

Candidate splits in ID3, C4.5

- splits on nominal features have one branch per value



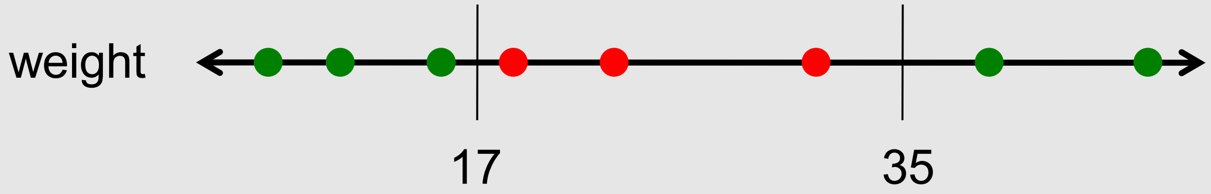
- splits on numeric features use a threshold

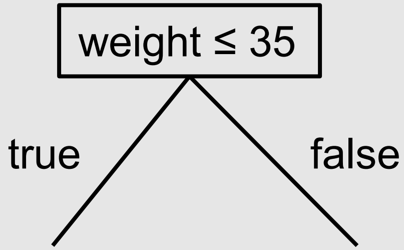


Candidate splits on numeric features

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

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 = partition instances in D into sets $s_1 \ldots s_V$ 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

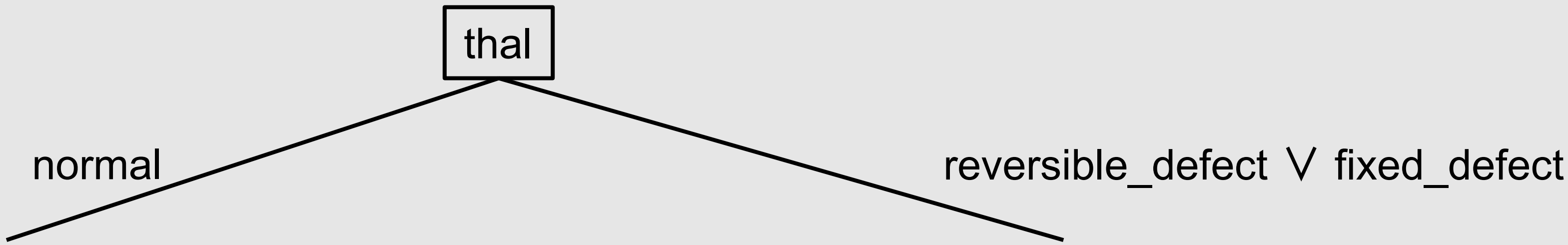
// assume we're using midpoints for splits

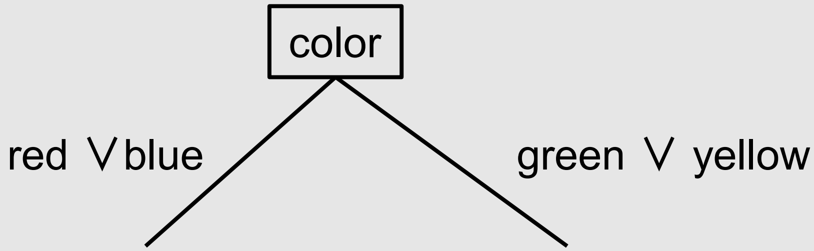
add candidate split $X_i \leq (v_j + v_{j+1})/2$ to C

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 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 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"

Ptolemy

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

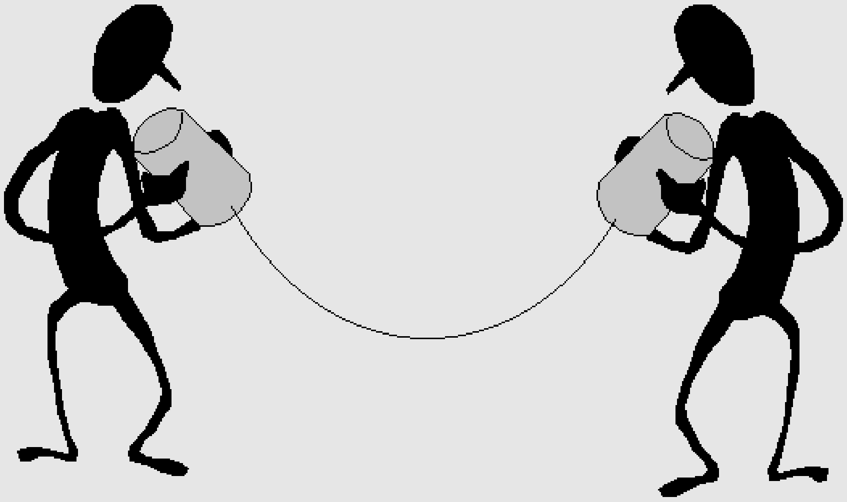
- Can we find and return the smallest possible decision tree that accurately classifies the training set?

  NO! This is an NP-hard problem
  [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
$P(\text{Trek}) = 0.5$	1	1
$P(\text{Specialized}) = 0.25$	2	01
$P(\text{Cervelo}) = 0.125$	3	001
$P(\text{Serrota}) = 0.125$	3	000

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

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

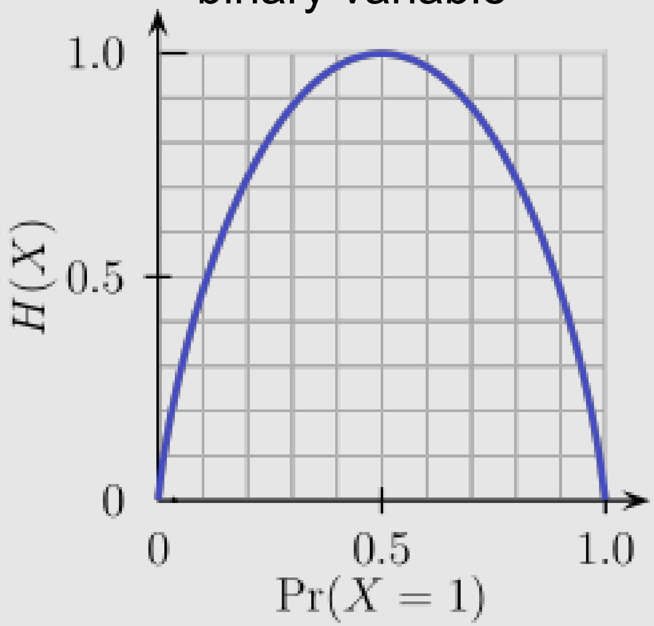
Entropy

- 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?

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

where

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

Conditional entropy: example

Y=Type/X=Color	Black	White
Trek	0.25	0.25
Specialized	0.125	0.125
Cervelo	0.125	0
Serrota	0	0.125

$$H(Y|X = black) = -0.5 \times \log 0.5 - 0.25 \times \log 0.25 - 0.25 \times \log 0.25 - 0 = 1.5$$

$$H(Y|X = white) = -0.5 \times \log 0.5 - 0.25 \times \log 0.25 - 0 - 0.25 \times \log 0.25 = 1.5$$

$$H(Y|X) = 0.5 \times H(Y|X = black) + 0.5 \times H(Y|white) = 1.5$$

Information gain (a.k.a. mutual information)

- Mutual information between two random variables:

$$I(Y;X) = H(Y) - H(Y|X)$$

- Measures how much uncertainty of Y that X can reduce

Y=Type/X=Color	Black	White
Trek	0.25	0.25
Specialized	0.125	0.125
Cervelo	0.125	0
Serrota	0	0.125

$$I(Y;X) = H(Y) - H(Y|X) = 1.75 - 1.5 = 0.25$$

Relations between the concepts

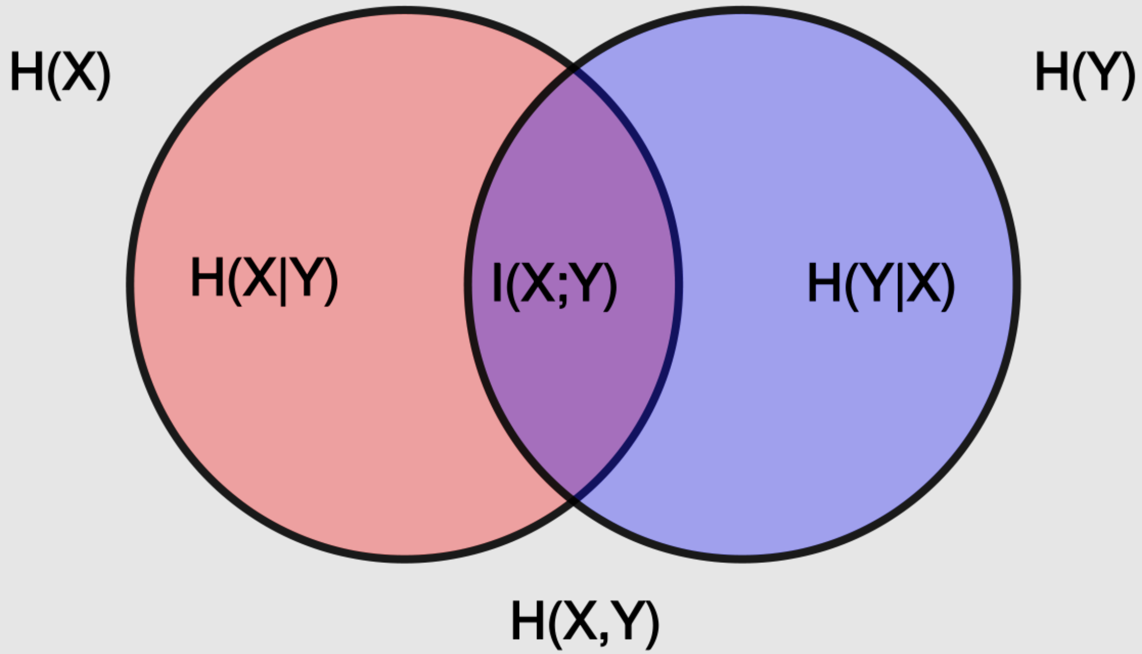


Figure from wikipedia.org

Information gain for choosing splits

- 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

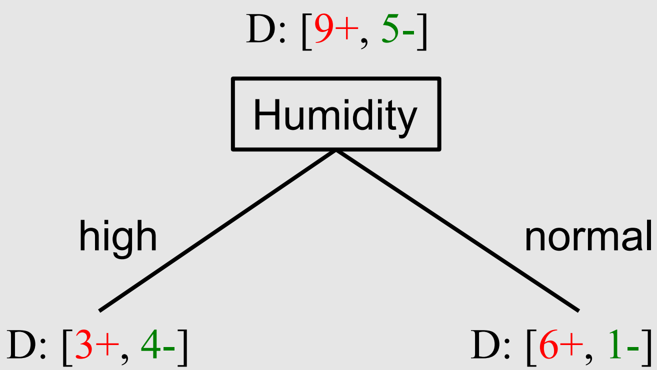
Information gain example

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?



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

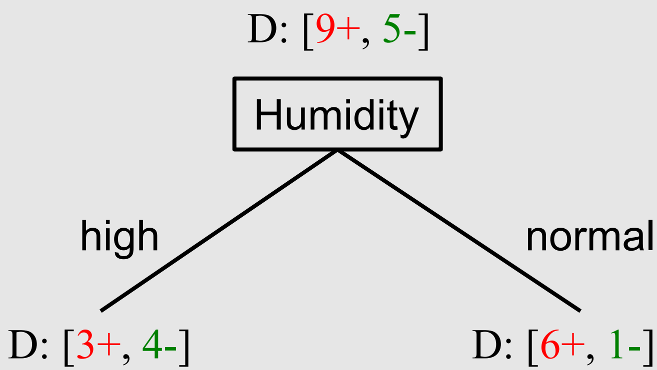
$$H_D(Y \mid \text{high}) = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$
$$= 0.985$$

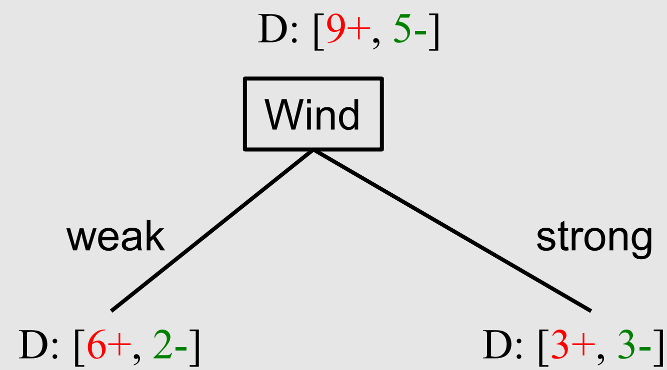
$$H_D(Y \mid \text{normal}) = -\frac{6}{7}\log_2\left(\frac{6}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)$$
$$= 0.592$$

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

Information gain example

- Is it better to split on Humidity or Wind?





$$H_D(Y \mid \textsf{weak}) = 0.811 \qquad H_D(Y \mid \textsf{strong}) = 1.0$$

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

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

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

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

THANK YOU

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.