# Decision Tree Learning

Dr. Xiaowei Huang

https://cgi.csc.liv.ac.uk/~xiaowei/

# Decision Tree up to now,

- Decision tree representation

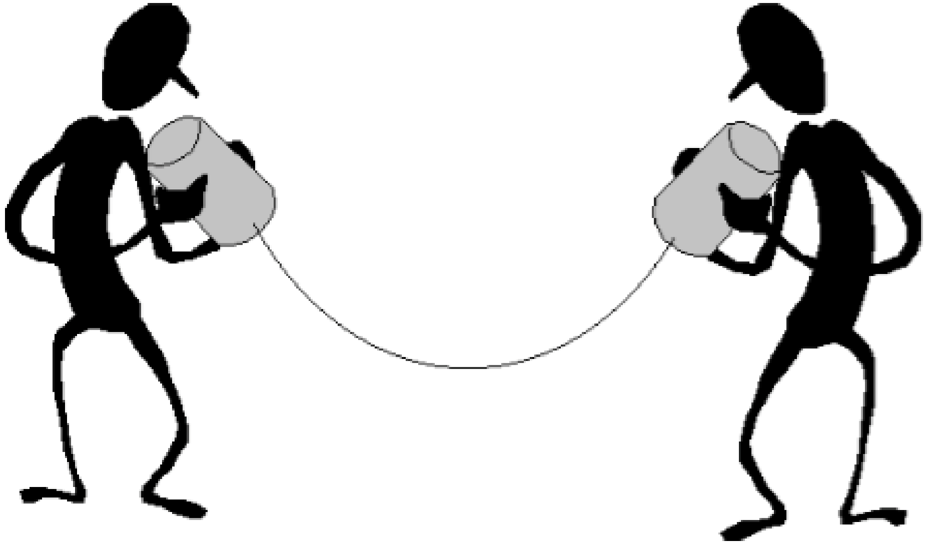
- A general top-down algorithm
- How to do splitting on numeric features
- Occam’s razor

# Today’s Topics

- Entropy and information gain
- Types of decision-tree splits
- Stopping criteria of decision trees
- Accuracy of decision trees
- Overfitting
- Variants of decision trees (extended material)

# 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 |

$$\frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 2$$

- 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

| 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 |

# Information theory background

| 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

$$0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75 < 2$$

# Information theory background

| 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 |

$$0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75 < 2$$

$$= 0.5 \times \log_2 0.5 + 0.25 \times \log_2 0.25 + 0.125 \times \log_2 0.125 + 0.125 \times \log_2 0.125$$

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

# Information theory background

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

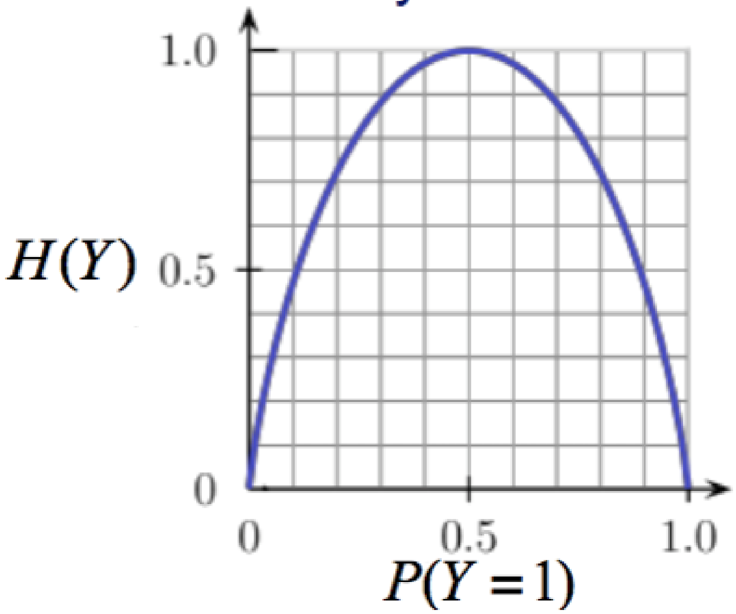
- optimal code uses $-\log_2 P(y)$ bits for event with probability $P(y)$

# 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

- **conditional entropy** (or equivocation) quantifies the amount of information needed to describe the outcome of a random variable given that the value of another random variable is known.
- What's the entropy of Y if we condition on some other variable X?

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

similar as the expected value?

- Where

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

Similar as entropy

# 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 |

# Example

- Let X = Outlook and Y = PlayTennis

- Can you compute H(Y|X)?

| | Yes | No |
|---|---|---|
| Sunny | 2/14 | 3/14 |
| Overcast | 4/14 | 0 |
| Rain | 3/14 | 2/14 |

$$\begin{aligned} H(Y|X) \quad = \quad & P(X = Sunny)H(Y|X = Sunny) && + \\ & P(X = Overcase)H(Y|X = Overcast) && + \\ & P(X = Rain)H(Y|X = Rain) \end{aligned}$$

# Example

- Let X = Outlook and Y = PlayTennis

- Can you compute H(Y|X)?

| | Yes | No |
|---|---|---|
| Sunny | 2/14 | 3/14 |
| Overcast | 4/14 | 0 |
| Rain | 3/14 | 2/14 |

$$
\begin{aligned}
H(Y|X = Sunny) &= -2/5 \log 2/5 - 3/5 \log 3/5 \\
H(Y|X = Overcast) &= 0 \\
H(Y|X = Rain) &= -3/5 \log 3/5 - 2/5 \log 2/5
\end{aligned}
$$

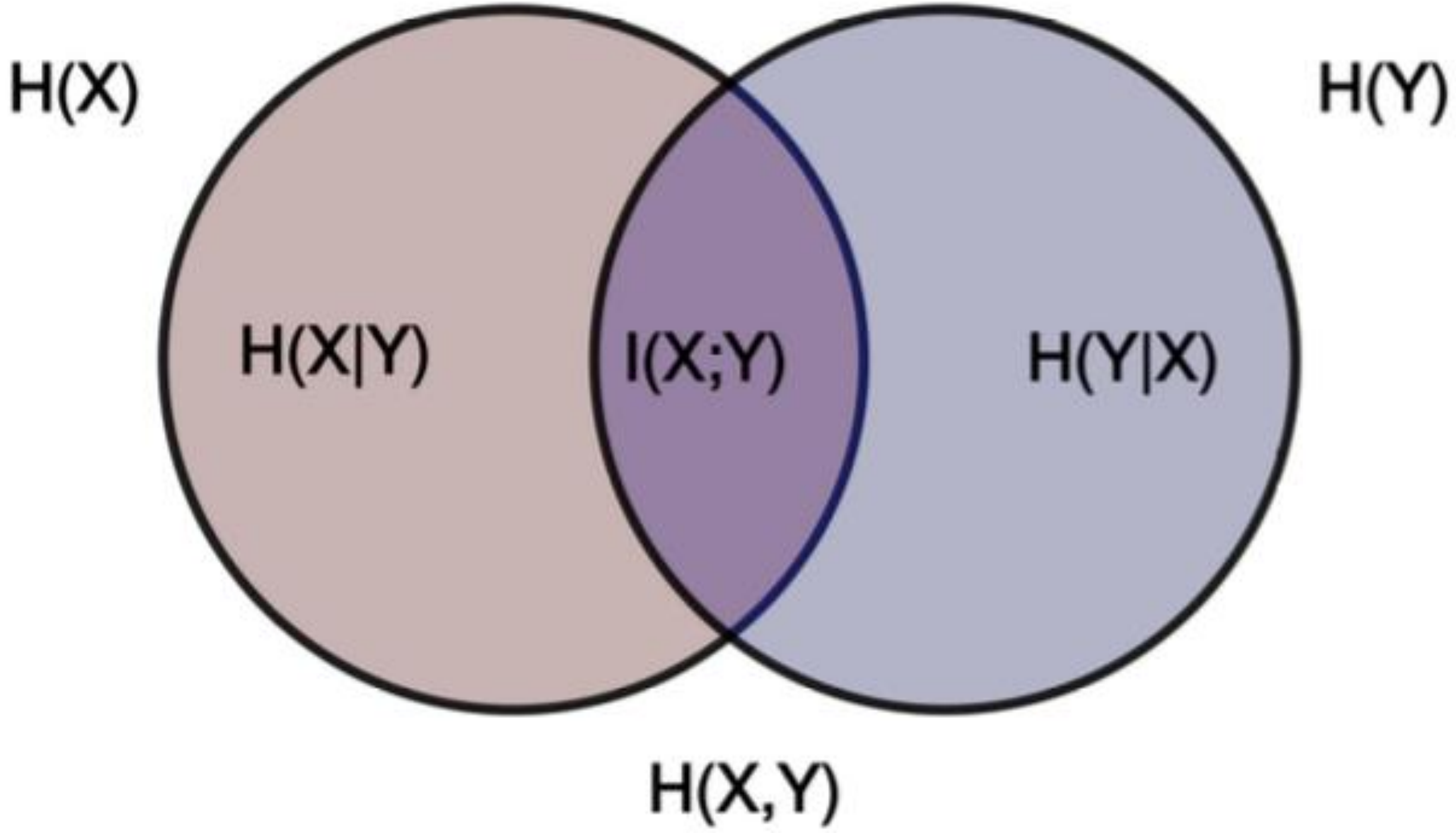
# Information gain (a.k.a. mutual information)

- 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




https://en.wikipedia.org/wiki/Mutual_information

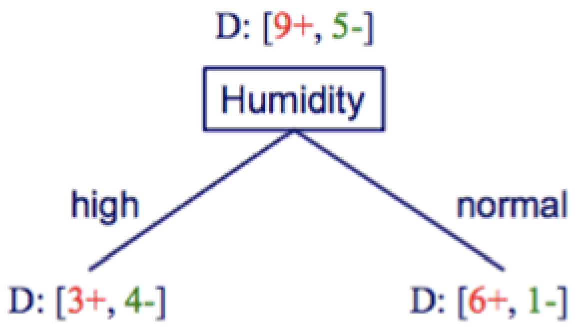
# Example

*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

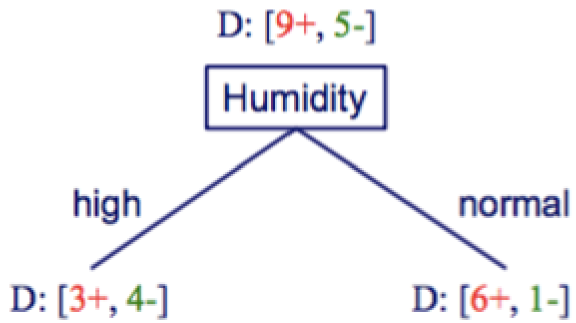
- What's the information gain of splitting on Humidity?




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

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

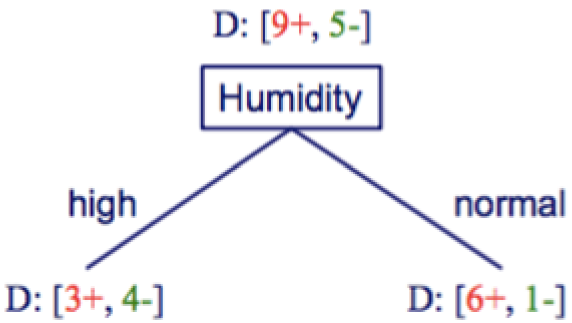
# Information gain example




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

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

# Information gain example




$H_D(Y \mid \text{Humidity})$ = P(Humidity=high)$H_D$(Y|Humidity=high) +
P(Humidity=normal)$H_D$(Y|Humidity=normal)

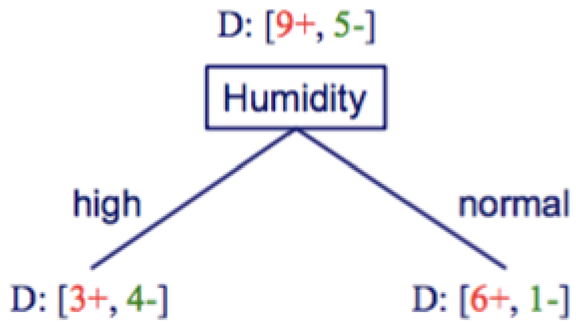
$$H(Y \mid X) = \sum_{x \in \text{values}(X)} P(X = x) H(Y \mid X = x)$$

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

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

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

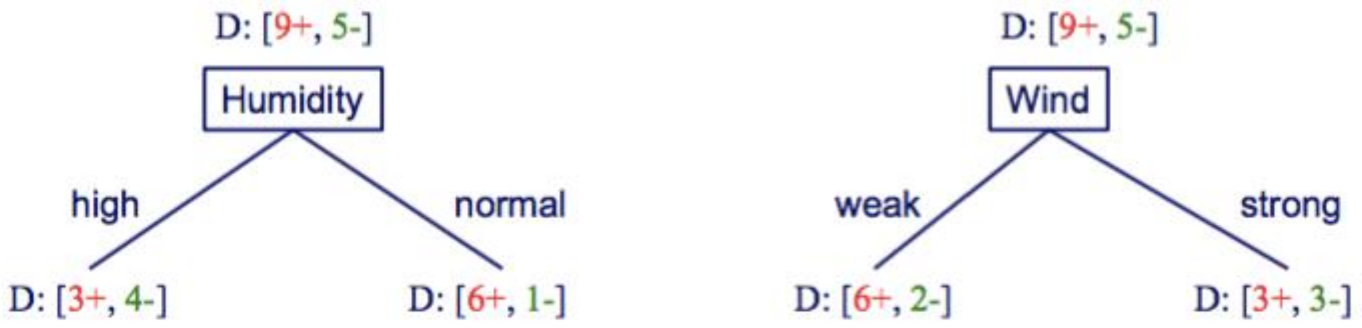
# Information gain example




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

# Information gain example

- Is it better to split on Humidity or Wind?




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

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

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

# 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

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

# Exercise

- Compute the following:

$$
\begin{aligned}
GainRatio(D, Humidity) &= \\
GainRatio(D, Wind) &= \\
GainRatio(D, Outlook) &=
\end{aligned}
$$

# Step (3): Stopping criteria

# 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

# Accuracy of Decision Tree

# Definition of Accuracy and Error

- Given a set D of samples and a trained model M, the accuracy is the percentage of correctly labeled samples. That is,

$$Accuracy(D, M) = \frac{|\{M(x) = l_x \mid x \in D\}|}{|D|}$$

Where $l_x$ is the true label of sample $x$ and $M(x)$ gives the predicted label of $x$ by $M$

- Error is a dual concept of accuracy.

But, what is D?

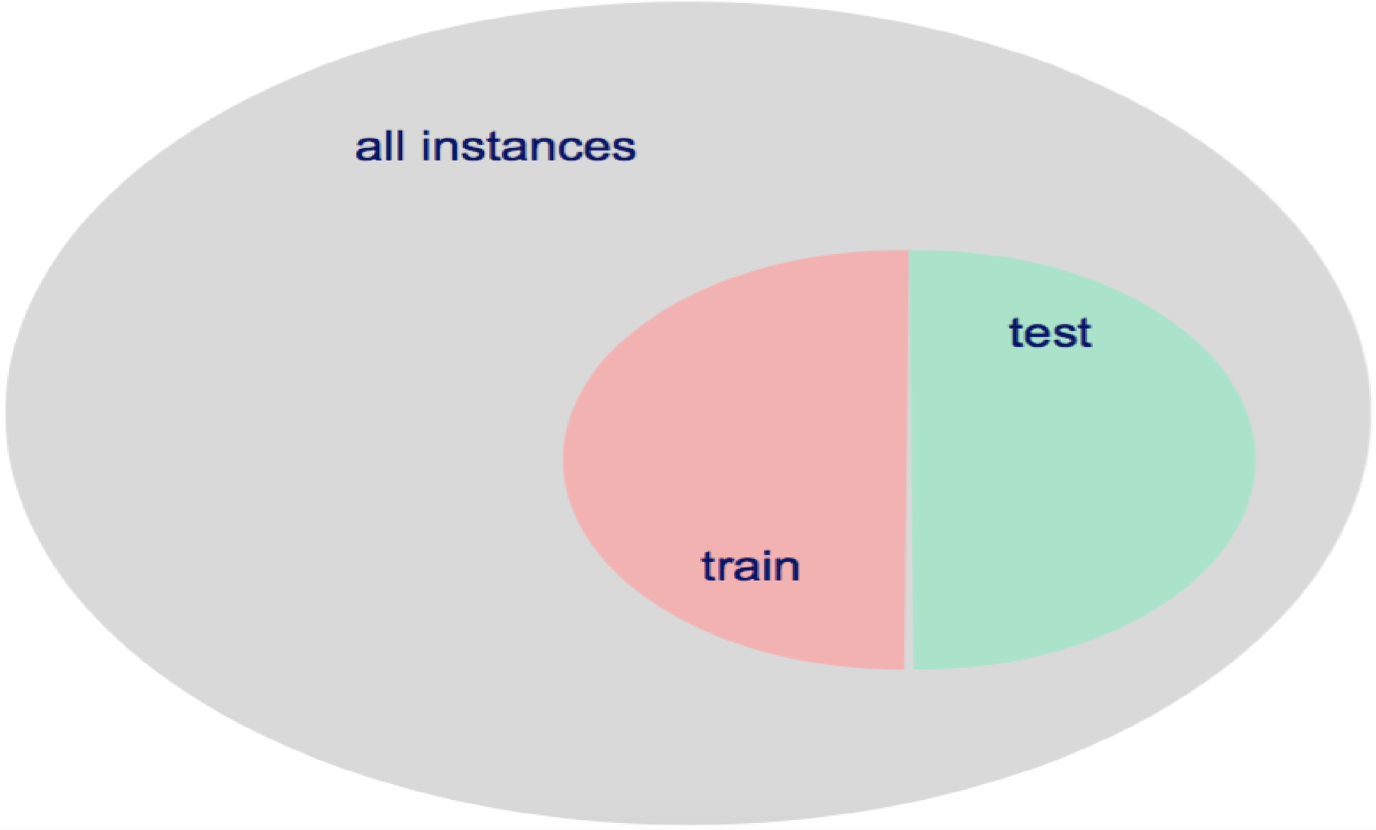
$$Error(D, M) = 1 - Accuracy(D, M)$$

# How can we 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?
    - Can never reach 1.0.
  - how accurate would it be on its training set?
    - Can be arbitrarily close to, or reach, 1.0 if model can be very large.

# 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 M over
  - training data: $Error(D_{training}, M)$
  - entire distribution of data: $Error(D_{true}, M)$
- model $M \in H$ *overfits* the training data if there is an alternative model $M' \in H$ such that

$$Error(D_{training}, M) < Error(D_{training}, M')$$

Perform better on training dataset

$$Error(D_{true}, M) > Error(D_{true}, M')$$

Perform worse on true distribution

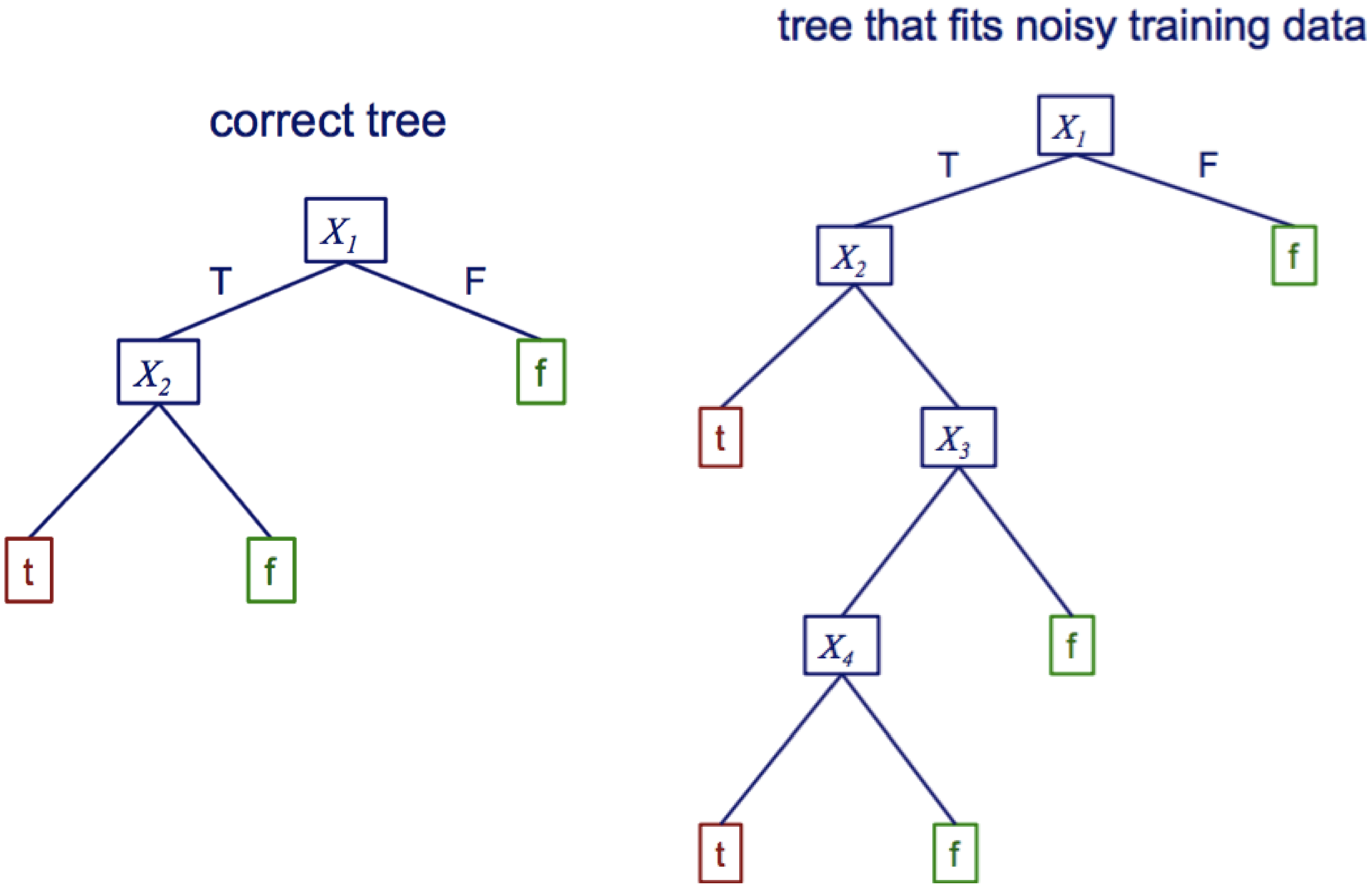
# Example 1: overfitting with noisy data

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

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | … | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | … | t |
| t | t | f | f | t | … | t |
| t | f | t | t | f | … | t |
| t | f | f | t | f | … | f |
| t | f | t | f | f | … | f |
| f | t | t | f | t | … | f |

noisy value

# Example 1: overfitting with noisy data




$$Y = X_1 \wedge X_2$$

A noisy data:

$X_1$ = t

$X_2$ = f

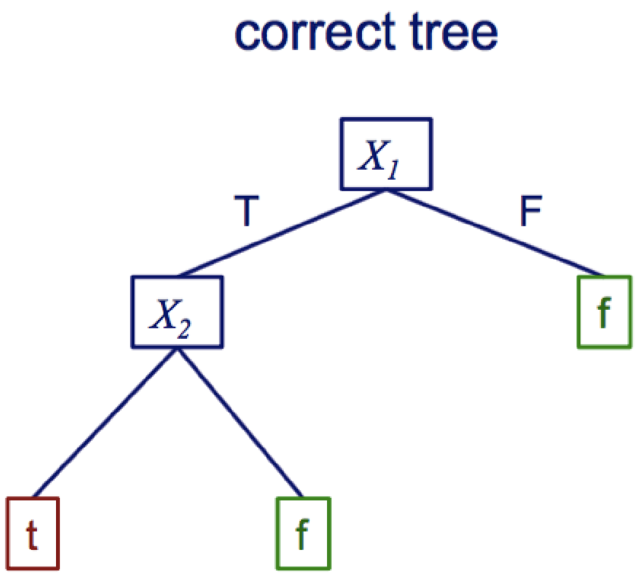
$X_3$ = t

$X_4$ = t

$X_5$ = f

Y = t

# Example 1: overfitting with noisy data



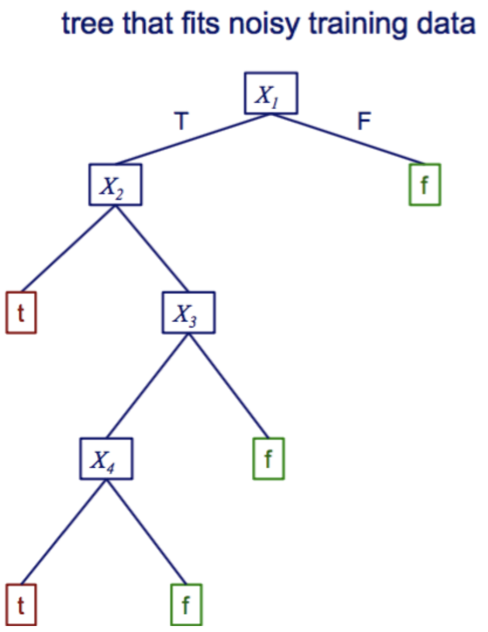

$$Y = X_1 \wedge X_2$$

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | … | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | … | t |
| t | t | f | f | t | … | t |
| t | f | t | t | f | … | t |
| t | f | f | t | f | … | f |
| t | f | t | f | f | … | f |
| f | t | t | f | t | … | f |

noisy value

- What is the accuracy?
  - Accuracy($D_{training}$,M) = 5/6
  - Accuracy($D_{true}$,M) = 100%

# Example 1: overfitting with noisy data



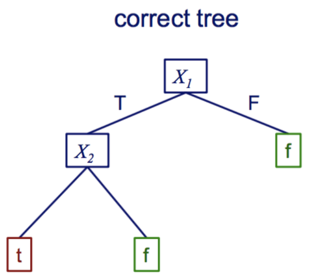
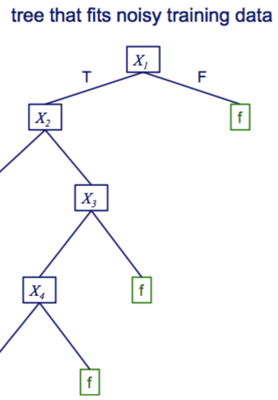

$$Y = X_1 \wedge X_2$$

- What is the accuracy?
  - Accuracy($D_{training}$,M) = 100%
  - Accuracy($D_{true}$,M) < 100%

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | … | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | … | t |
| t | t | f | f | t | … | t |
| t | f | t | t | f | … | t |
| t | f | f | t | f | … | f |
| t | f | t | f | f | … | f |
| f | t | t | f | t | … | f |

noisy value

# Example 1: overfitting with noisy data

| | | Training set accuracy | True accuracy |
|---|---|---|---|
| $M_1$ |  | 5/6 | 100% |
| $M_2$ |  | 100% | < 100 % |





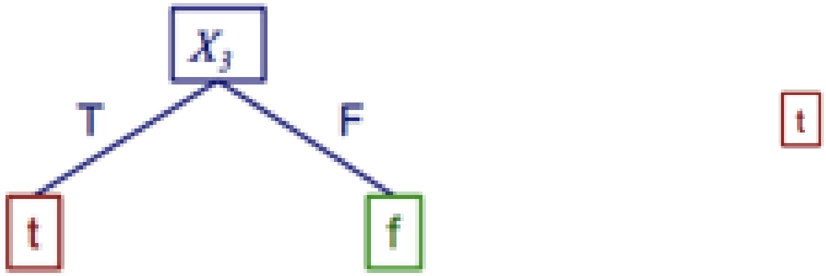
$M_2$ is overfitting!

# Example 2: overfitting with noise-free data

- suppose
  - the target concept is $Y = X_1 \wedge X_2$
  - $P(X_3 = t) = 0.5$ for both classes
  - $P(Y = t) = 0.66$
  - we're given the following training set

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | ... | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | ... | t |
| t | t | t | f | t | ... | t |
| t | t | t | t | f | ... | t |
| t | f | f | t | f | ... | f |
| f | t | f | f | t | ... | f |

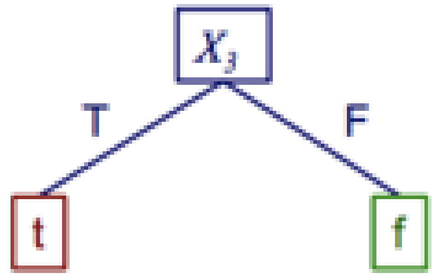
# Example 2: overfitting with noise-free data




$M_1$ $M_2$

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | ... | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | ... | t |
| t | t | t | f | t | ... | t |
| t | t | t | t | f | ... | t |
| t | f | f | t | f | ... | f |
| f | t | f | f | t | ... | f |

# Example 2: overfitting with noise-free data




$$Y = X_1 \wedge X_2$$

$P(X_3 = t) = 0.5$
$P(Y{=}t) = 0.66$

- What is the accuracy?
  - Accuracy($D_{training}$,M) = 100%
  - Accuracy($D_{true}$,M) = 50%

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | ... | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | ... | t |
| t | t | t | f | t | ... | t |
| t | t | t | t | f | ... | t |
| t | f | f | t | f | ... | f |
| f | t | f | f | t | ... | f |

# Example 2: overfitting with noise-free data

t

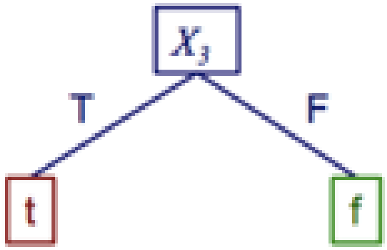
$$Y = X_1 \wedge X_2$$

$P(X_3 = t) = 0.5$
$P(Y{=}t) = 0.66$

- What is the accuracy?
  - $\text{Accuracy}(D_{training}, M) = 60\%$
  - $\text{Accuracy}(D_{true}, M) = 66\%$

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | ... | $Y$ |
|---|---|---|---|---|---|---|
| t | t | t | t | t | ... | t |
| t | t | t | f | t | ... | t |
| t | t | t | t | f | ... | t |
| t | f | f | t | f | ... | f |
| f | t | f | f | t | ... | f |

# Example 2: overfitting with noise-free data

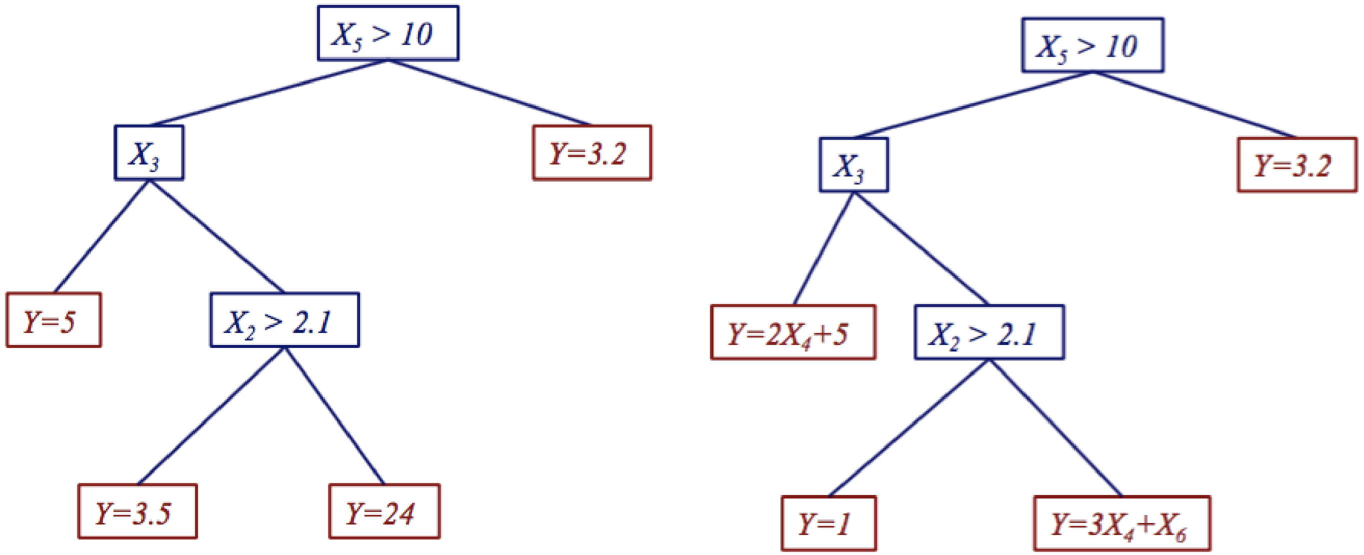
| | | Training set accuracy | True accuracy | |
|---|---|---|---|---|
| $M_1$ |  | 100% | 50% | $M_1$ is overfitting! |
| $M_2$ |  | 60% | 66% | |

- because the training set is a limited sample, there might be (combinations of) features that are correlated with the target concept by chance

# Variant: Regression Trees

# 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

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

target value for $i^{th}$ training instance

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

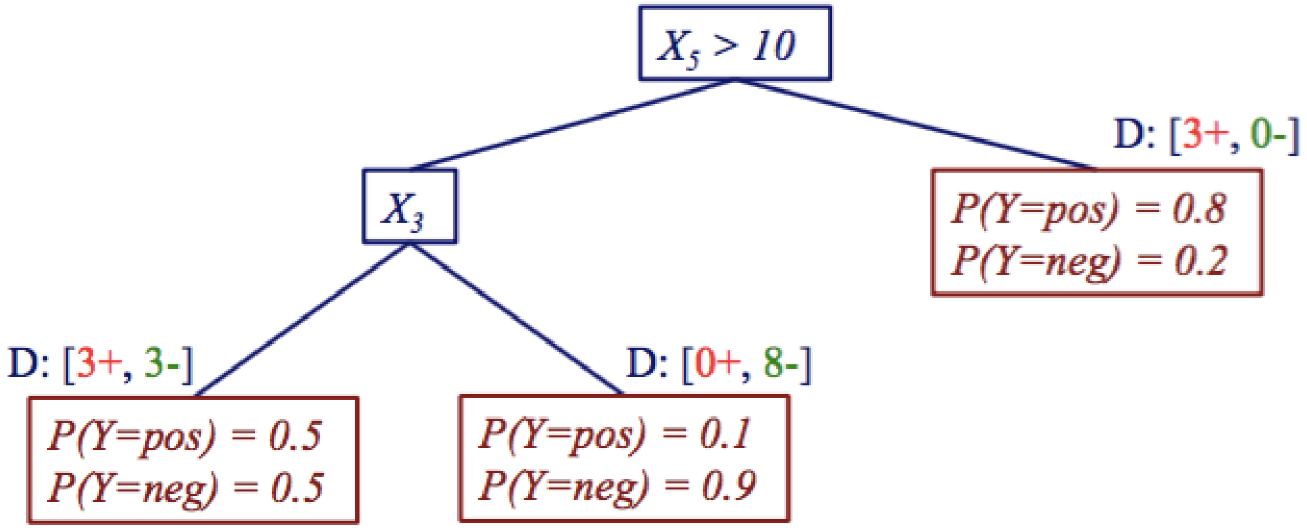
$$= \sum_{L \in \text{leaves}} \sum_{i \in L} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

- at each internal node, CART chooses the split that most reduces this quantity

# Variant: Probability estimation trees

# 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)

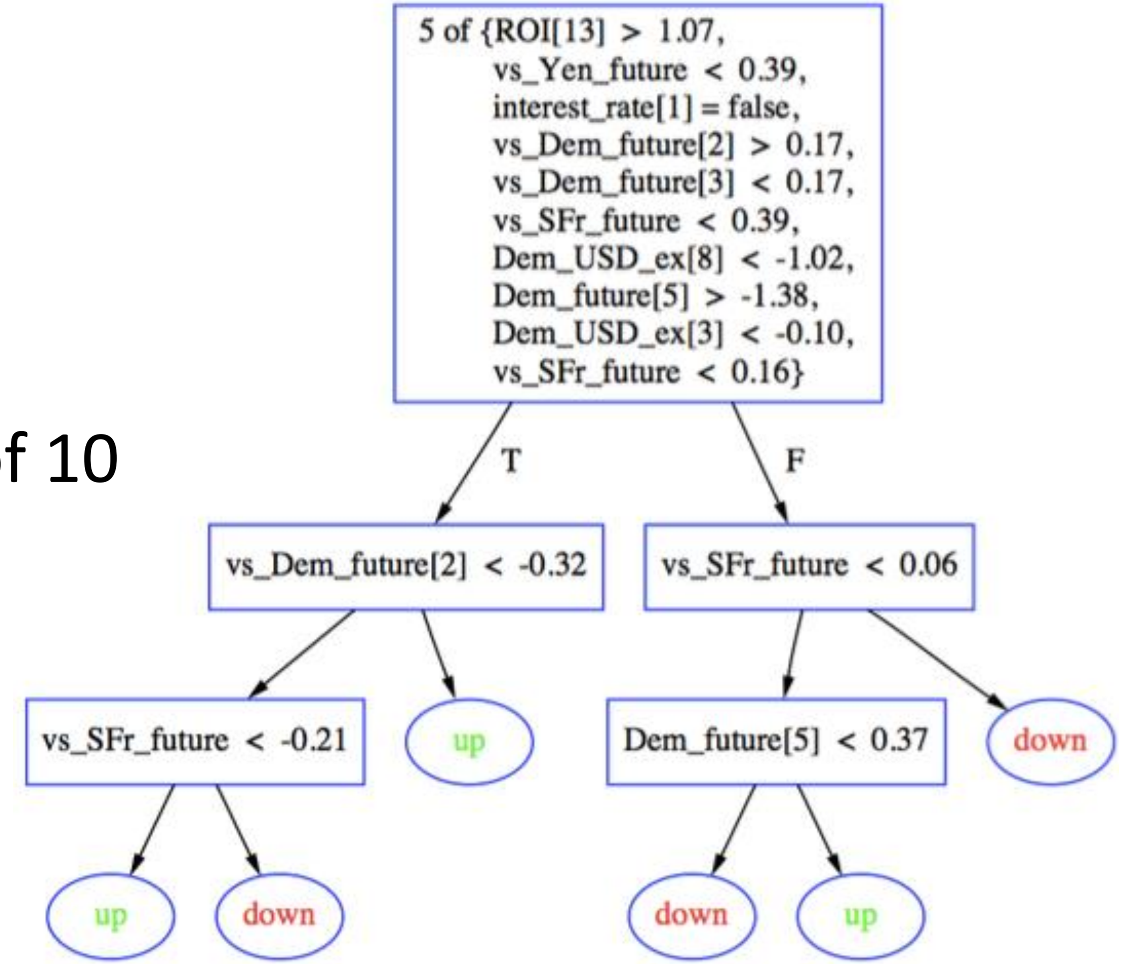


# Variant: m-of-n splits

# 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

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-of-n => m-of-(n+1)
  - 1 of { $X_1$=t, $X_3$=f } => 1 of { $X_1$=t, $X_3$=f, $X_7$=t }
  - m-of-n => (m+1)-of-(n+1)
  - 1 of { $X_1$=t, $X_3$=f } => 2 of { $X_1$=t, $X_3$=f, $X_7$=t }

# Variant: Lookahead

# 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

- OrdinaryFindBestSplit (set of training instances D, set of candidate splits C)

$maxgain = -\infty$

for each split $S$ in $C$

  $gain$ = InfoGain($D$, $S$)

  if $gain > maxgain$

    $maxgain = gain$

    $S_{best} = S$

return $S_{best}$

# 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 $C$

    $gain$ = EvaluateSplit($D, C, S$)

    if $gain > maxgain$

        $maxgain = gain$

        $S_{best} = S$

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 $k$

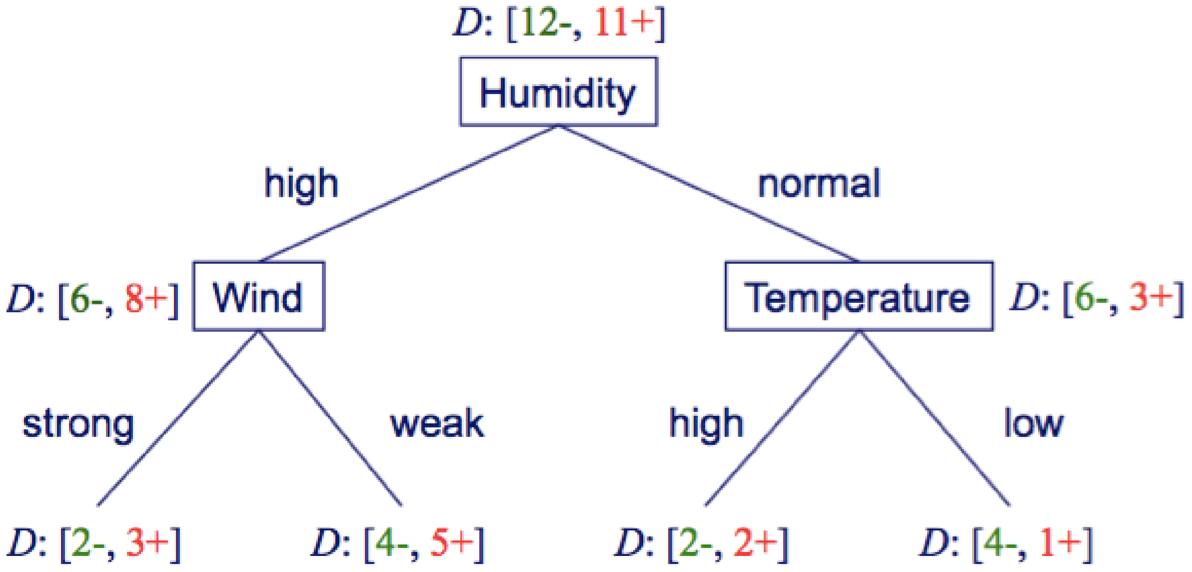
$S_k$ = OrdinaryFindBestSplit($D_k$, $C - S$)

// 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 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:

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

$$= \frac{5}{23}H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind} = \text{strong}) +$$

$$\frac{9}{23}H_D(Y \mid \text{Humidity} = \text{high}, \text{Wind} = \text{weak}) +$$

$$\frac{4}{23}H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature} = \text{high}) +$$

$$\frac{5}{23}H_D(Y \mid \text{Humidity} = \text{low}, \text{Temperature} = \text{low})$$

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

# 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