

***Decision Trees: many possible refs., e.g.,
Mitchell, Chapter 3
Boosting: (Linked from class website)
Schapire '01***

Decision Trees Boosting

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

February 6th, 2006

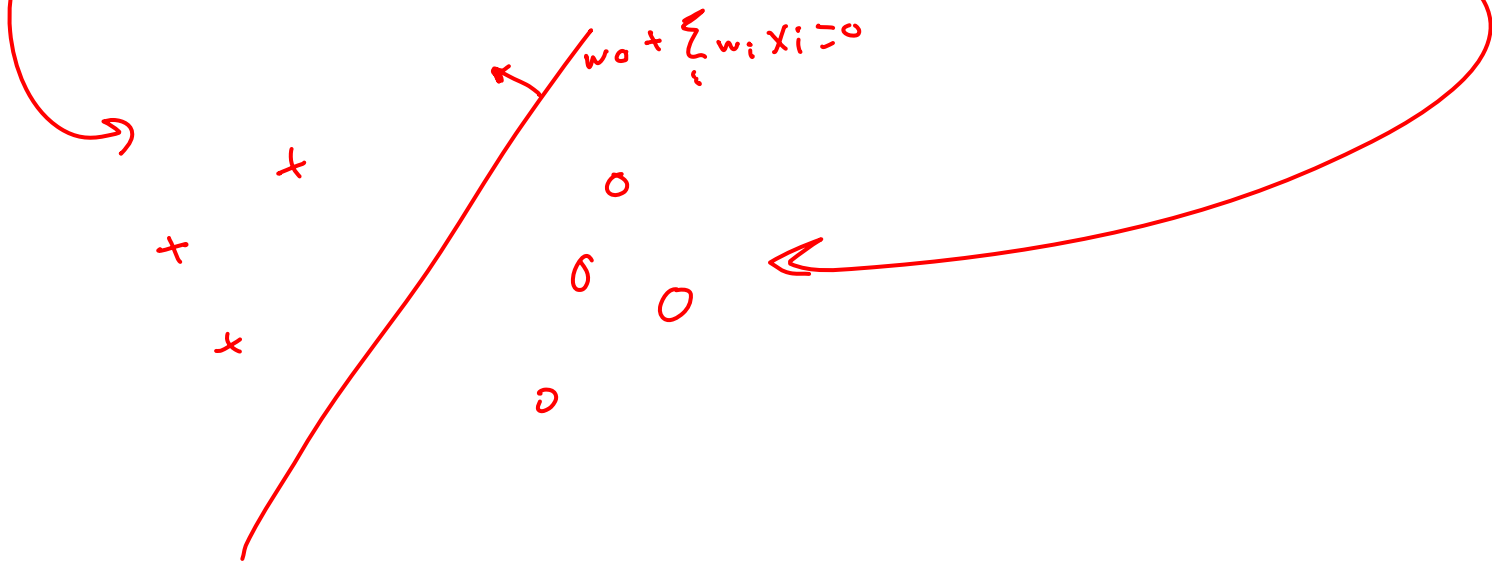
Announcements



- Recitations stay on Thursdays
 - 5-6:30pm in Wean 5409
 - This week: Decision Trees and Boosting
- **Pittsburgh won the Super Bowl !!**

Linear separability

- A dataset is **linearly separable** iff \exists a **separating hyperplane**:
 - $\exists \mathbf{w}$, such that:
 - $w_0 + \sum_i w_i x_i > 0$; if $\mathbf{x}=\{x_1, \dots, x_n\}$ is a positive example
 - $w_0 + \sum_i w_i x_i < 0$; if $\mathbf{x}=\{x_1, \dots, x_n\}$ is a negative example



Not linearly separable data

- Some datasets are **not linearly separable**!



*A XOR B
no hyper-plane!*

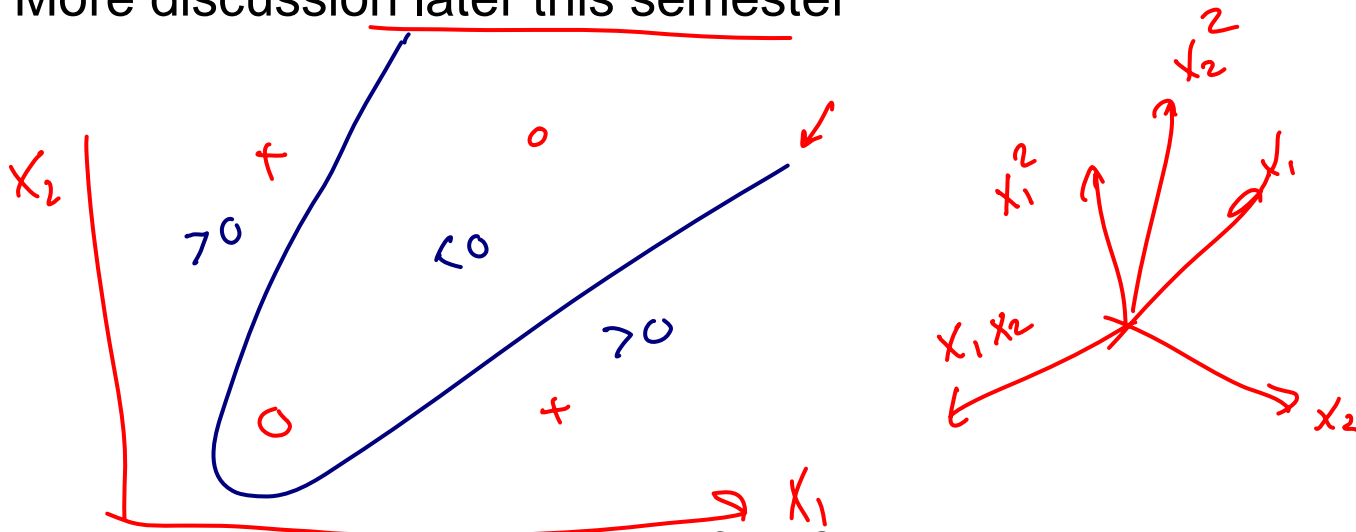
Addressing non-linearly separable data – Option 1, non-linear features

■ Choose non-linear features, e.g.,

- Typical linear features: $w_0 + \sum_i w_i x_i$
- Example of non-linear features:
 - Degree 2 polynomials, $w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$

■ Classifier $h_w(\mathbf{x})$ still linear in parameters \mathbf{w}

- Usually easy to learn (closed-form or convex/concave optimization)
- Data is linearly separable in higher dimensional spaces
- More discussion later this semester

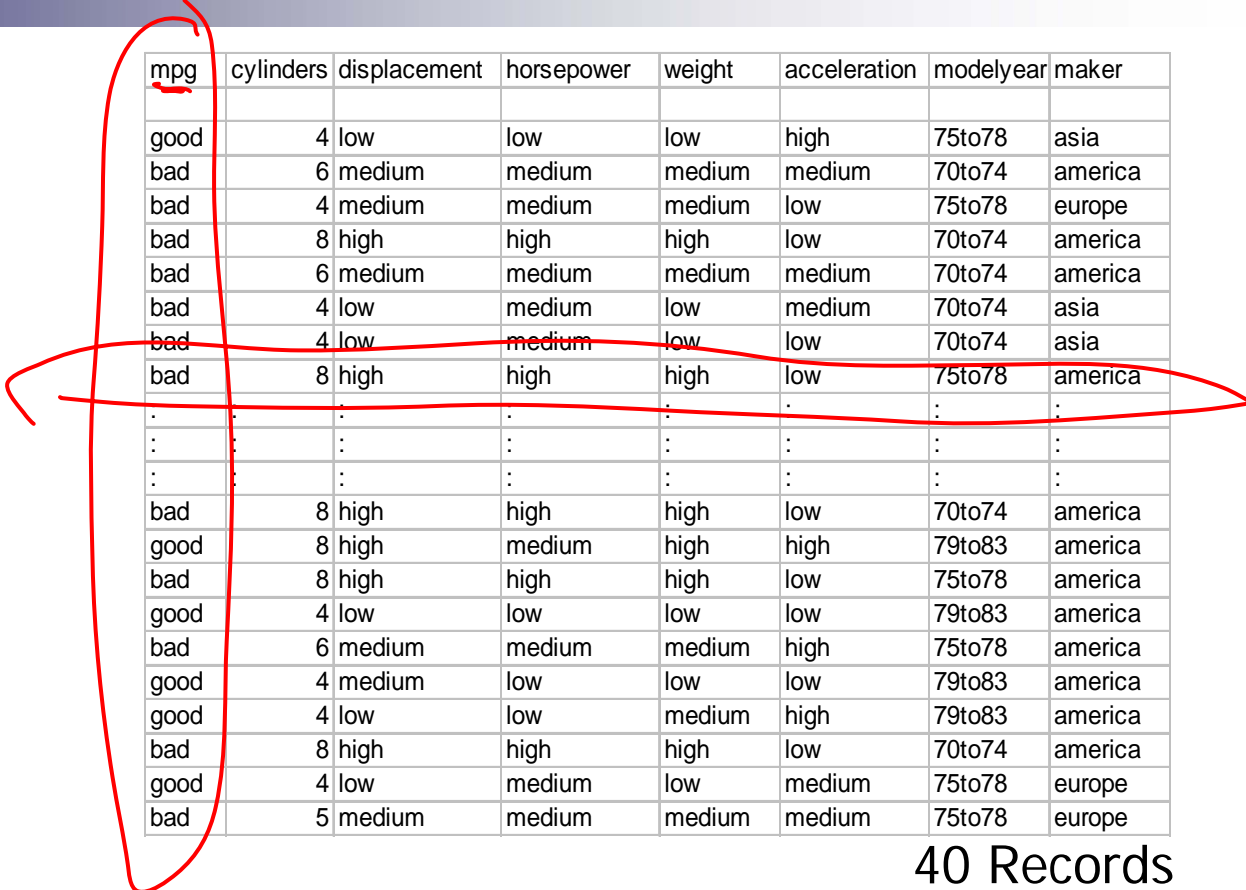


Addressing non-linearly separable data – Option 2, non-linear classifier

- Choose a classifier $h_{\mathbf{w}}(\mathbf{x})$ that is non-linear in parameters \mathbf{w} , e.g.,
 - Decision trees, neural networks, nearest neighbor,...
- More general than linear classifiers
- But, can often be harder to learn (non-convex/concave optimization required)
- But, but, often very useful
- (BTW. Later this semester, we'll see that these options are not that different)

A small dataset: Miles Per Gallon

Suppose we want to predict MPG



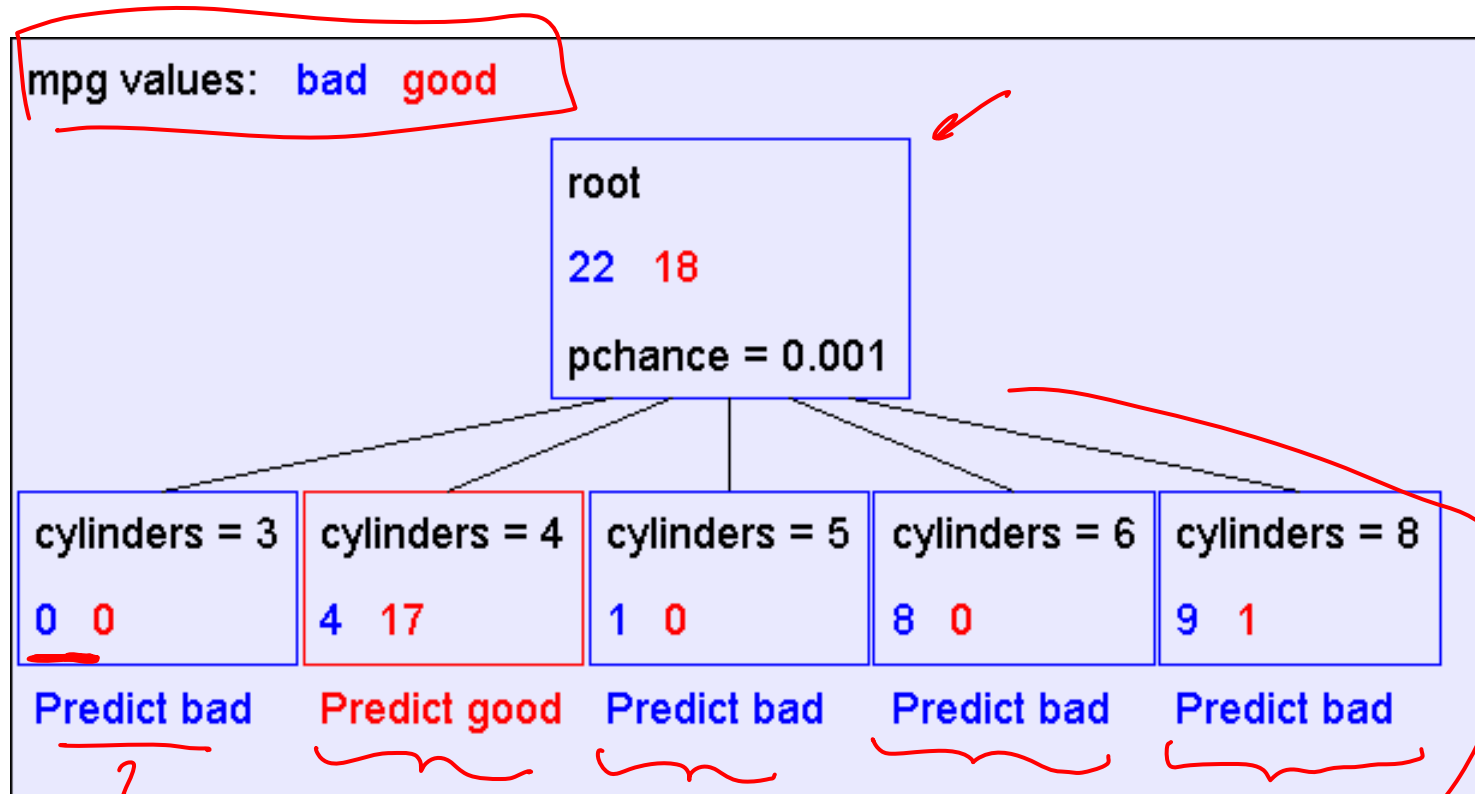
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europa
bad	5	medium	medium	medium	medium	75to78	europa

40 Records

From the UCI repository (thanks to Ross Quinlan)

A Decision Stump

1 split



bad gas mileage

Recursion Step

mpg values: bad good

root
22 18
pchance = 0.001

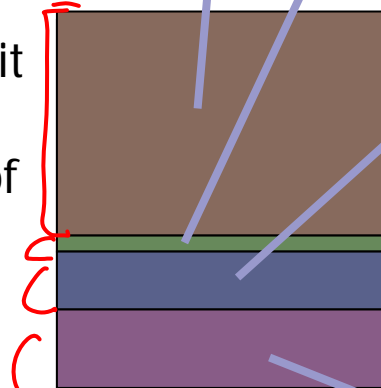
pick split

cylinders = 3	cylinders = 4	cylinders = 5	cylinders = 6	cylinders = 8
0 0	4 17	1 0	8 0	9 1
Predict bad	Predict good	Predict bad	Predict bad	Predict bad

Take the
Original
Dataset..



And partition it
according
to the value of
the attribute
we split on



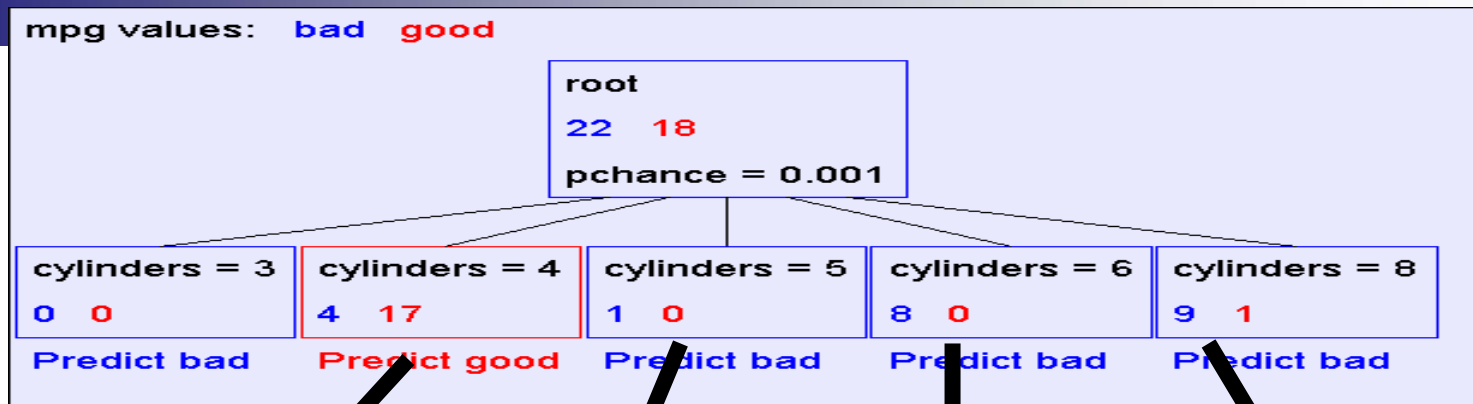
Records
in which
cylinders
= 4

Records
in which
cylinders
= 5

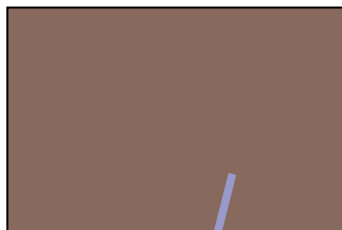
Records
in which
cylinders
= 6

Records
in which
cylinders
= 8

Recursion Step



Build tree from
These records..



Records in
which
cylinders = 4

Build tree from
These records..



Records in
which
cylinders = 5

Build tree from
These records..



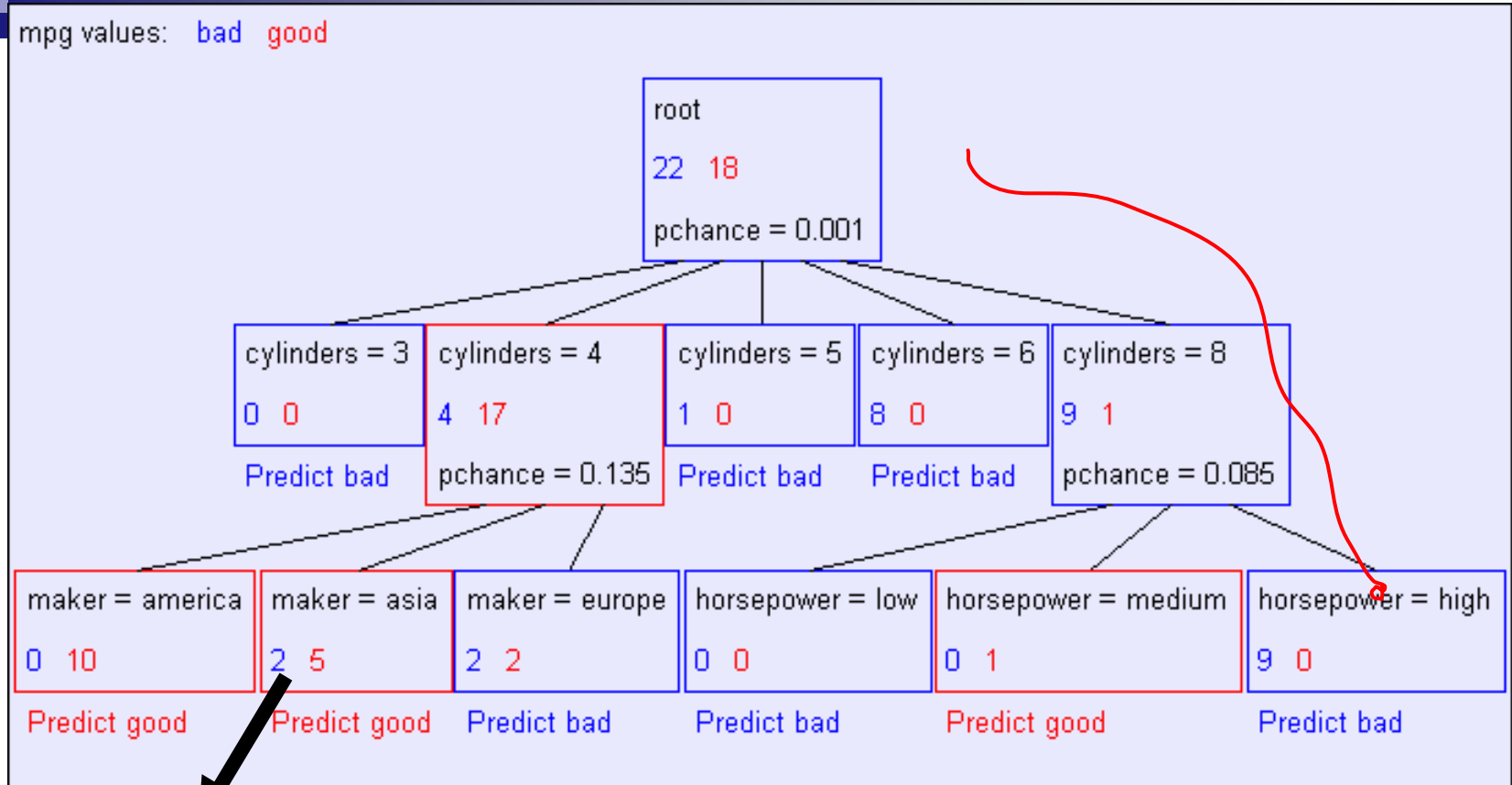
Records in
which
cylinders = 6

Build tree from
These records..



Records in
which
cylinders = 8

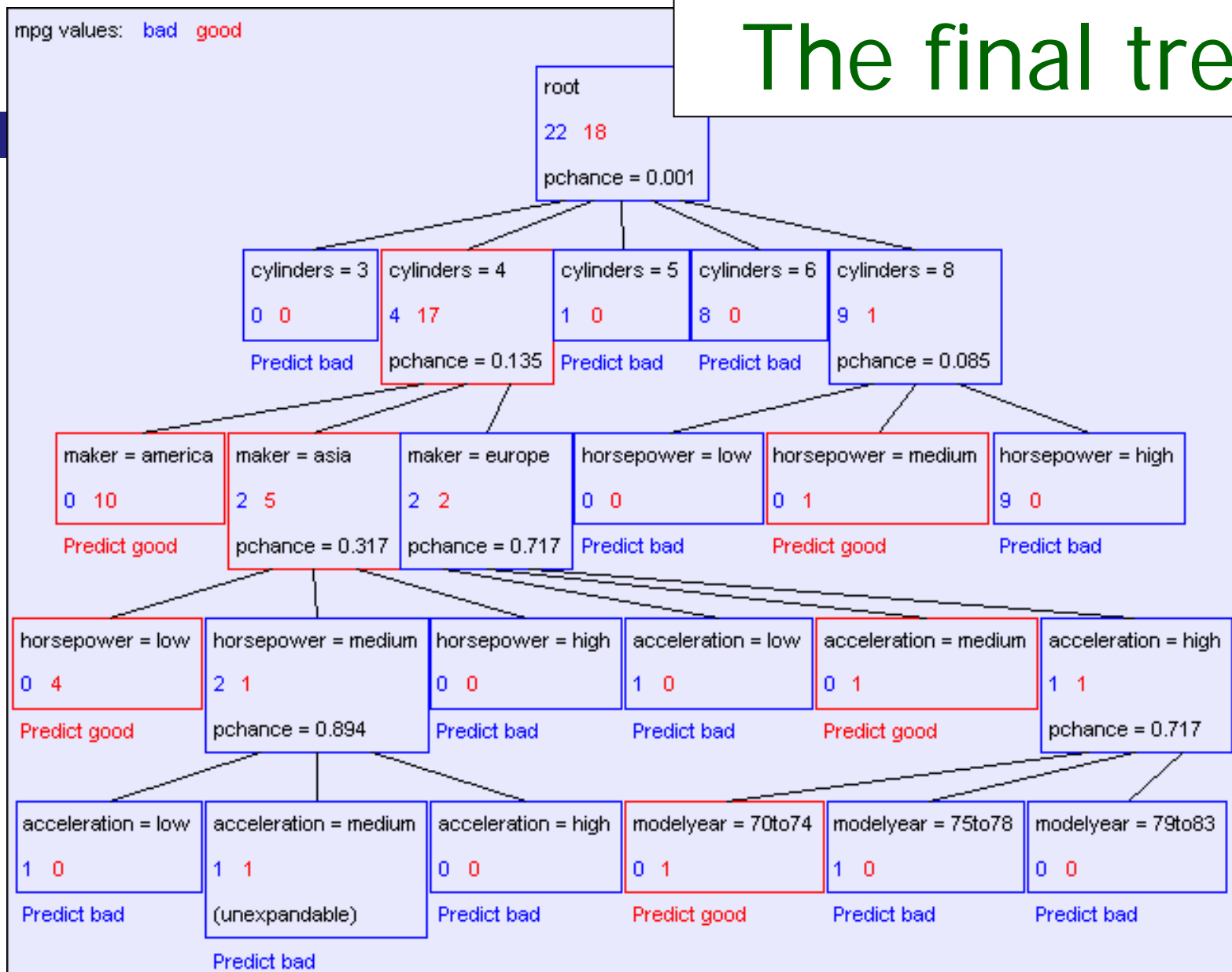
Second level of tree



Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

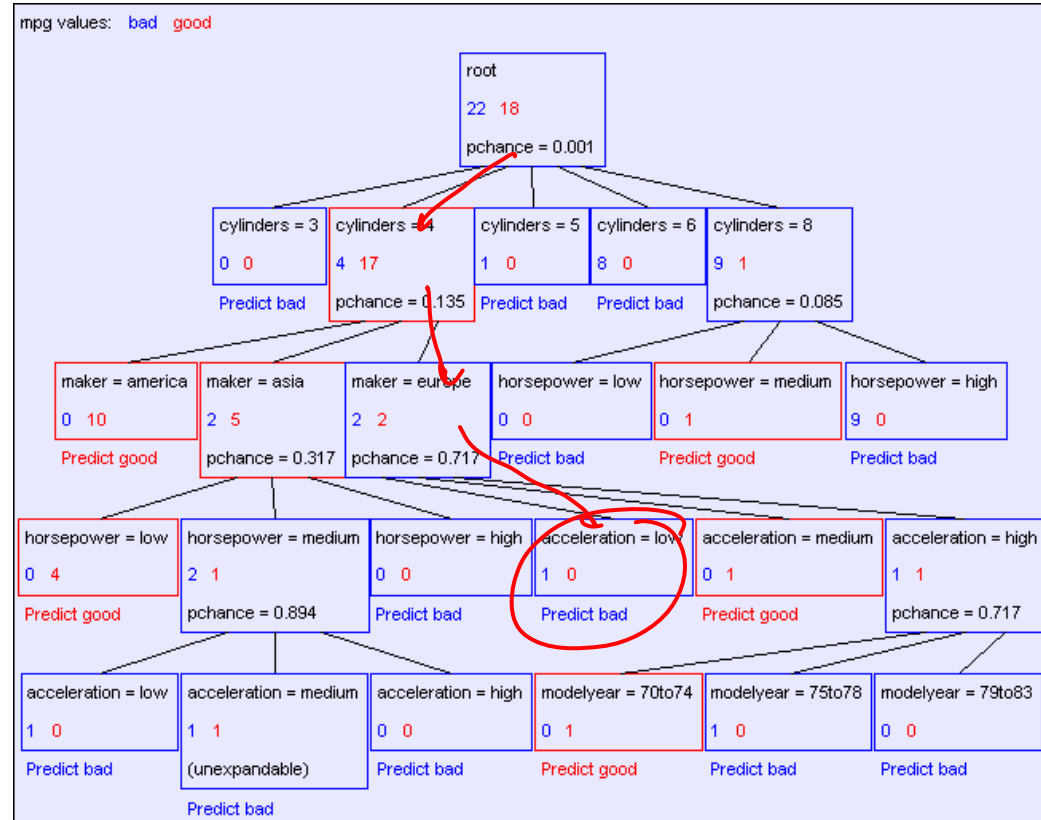
The final tree



Classification of a new example

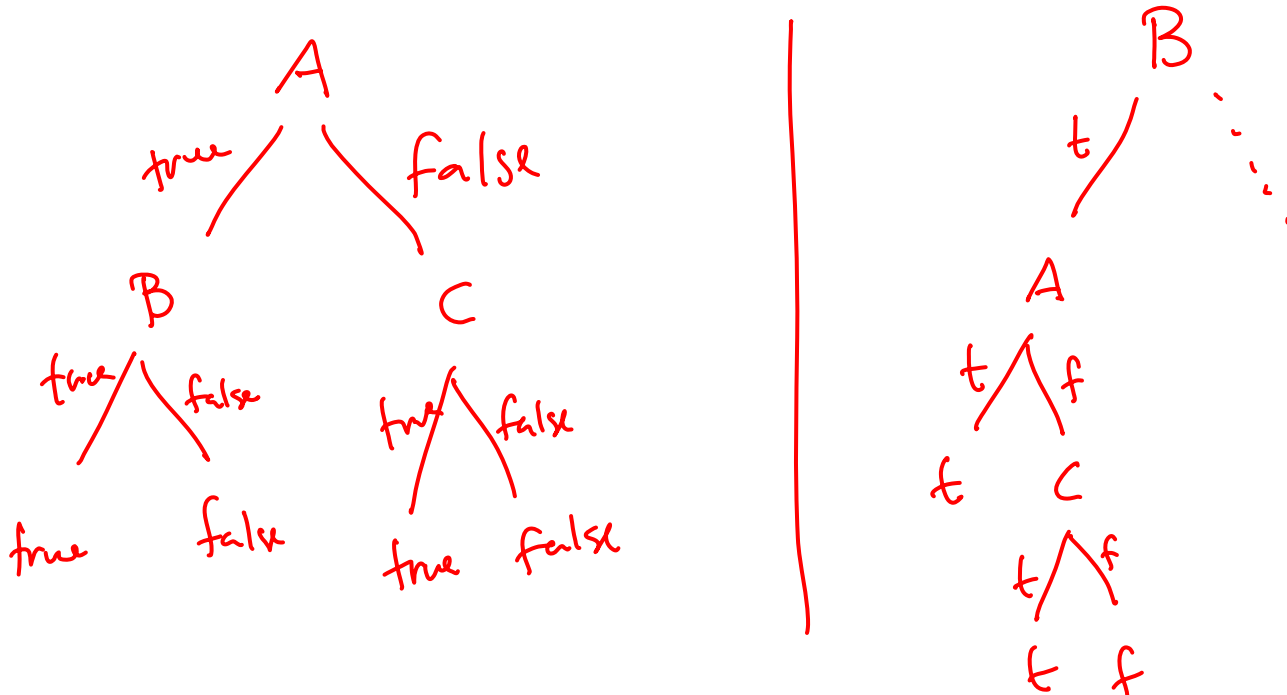
■ Classifying a test example – traverse tree and report leaf label

< C_y = 4, Europe, acc = low >
⇒ predict bad



Are all decision trees equal?

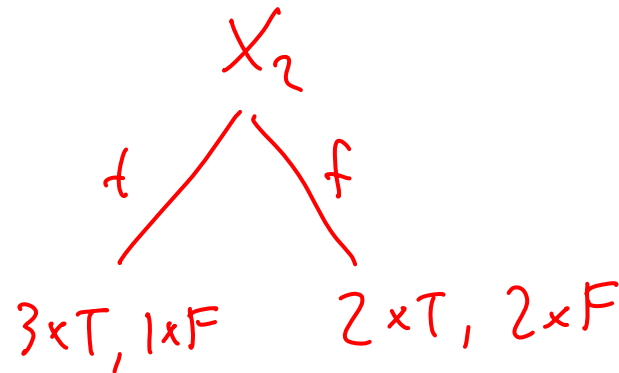
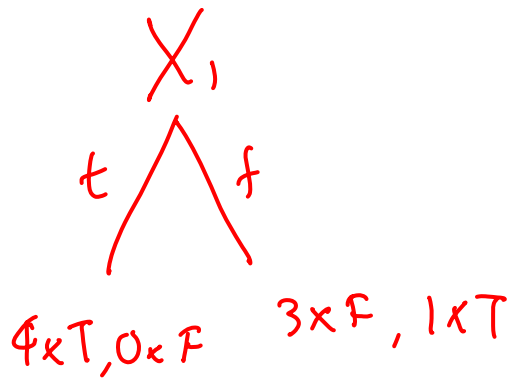
- Many trees can represent the same concept
- But, not all trees will have the same size!
 - e.g., $\phi = \underline{A \wedge B \vee \neg A \wedge C}$ ((A and B) or (not A and C))



Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]
- Resort to a greedy heuristic:
 - Start from empty decision tree
 - Split on next best attribute (feature)
 - Recurse

Choosing a good attribute



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Measuring uncertainty

- Good split if we are more certain about classification after split
 - Deterministic good (all true or all false)
 - Uniform distribution bad

$P(Y=A) = 1/2$	$P(Y=B) = 1/4$	$P(Y=C) = 1/8$	$P(Y=D) = 1/8$
----------------	----------------	----------------	----------------

better
less uncertainty

$P(Y=A) = 1/4$	$P(Y=B) = 1/4$	$P(Y=C) = 1/4$	$P(Y=D) = 1/4$
----------------	----------------	----------------	----------------

worse
more uncertainty

Entropy

def: $0 \log 0 = 0$

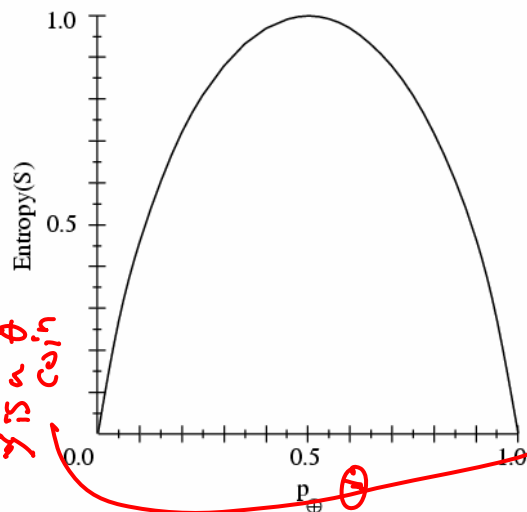
$\lim_{x \rightarrow 0^+} x \ln x = 0$

Entropy $H(X)$ of a random variable Y

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



coin flip with param. θ

$P(H) = \theta$
 $P(T) = 1 - \theta$

Handwritten notes on the left side of the graph:
 $H(Y) = -\theta \log \theta - (1-\theta) \log (1-\theta)$
 Y is a coin

Andrew Moore's Entropy in a nutshell



Low Entropy



High Entropy

Andrew Moore's Entropy in a nutshell



Low Entropy

..the values (locations of soup) sampled entirely from within the soup bowl



High Entropy

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room

Information gain

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

■ Advantage of attribute – decrease in uncertainty

□ Entropy of Y before you split $H(Y)$

□ Entropy after split *on X* : $H(Y|X)$

- Weight by probability of following each branch, i.e., normalized number of records

$$H(Y | X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

*weigh by
prob. follow a branch*

$$H(Y) = - \frac{5}{6} \log \frac{5}{6} - \frac{1}{6} \log \frac{1}{6}$$

$$H(Y|X_1) = \frac{2}{3} H(Y|X=T) + \frac{1}{3} H(Y|X=F)$$

■ Information gain is difference

$$IG(X) = H(Y) - H(Y | X)$$

$$H(Y) - H(Y|X_1) = H(Y) - \frac{1}{3}$$

BTW. $H(Y) - H(Y|X) \geq 0$ (information never hurts)

$$H(Y|X_1=T) = 0 ; H(Y|X_1=F) = 1$$

$$H(Y|X_1) = \frac{2}{3} \cdot 0 + \frac{1}{3} \cdot 1 = \frac{1}{3}$$

Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
 - Use, for example, information gain to select attribute
 - Split on $\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y | X_i)$
- Recurse

there are
other
heuristics

Information Gain Example

wealth values: poor rich

gender Female 14423 1769  $H(\text{wealth} \mid \text{gender} = \text{Female}) = 0.497654$

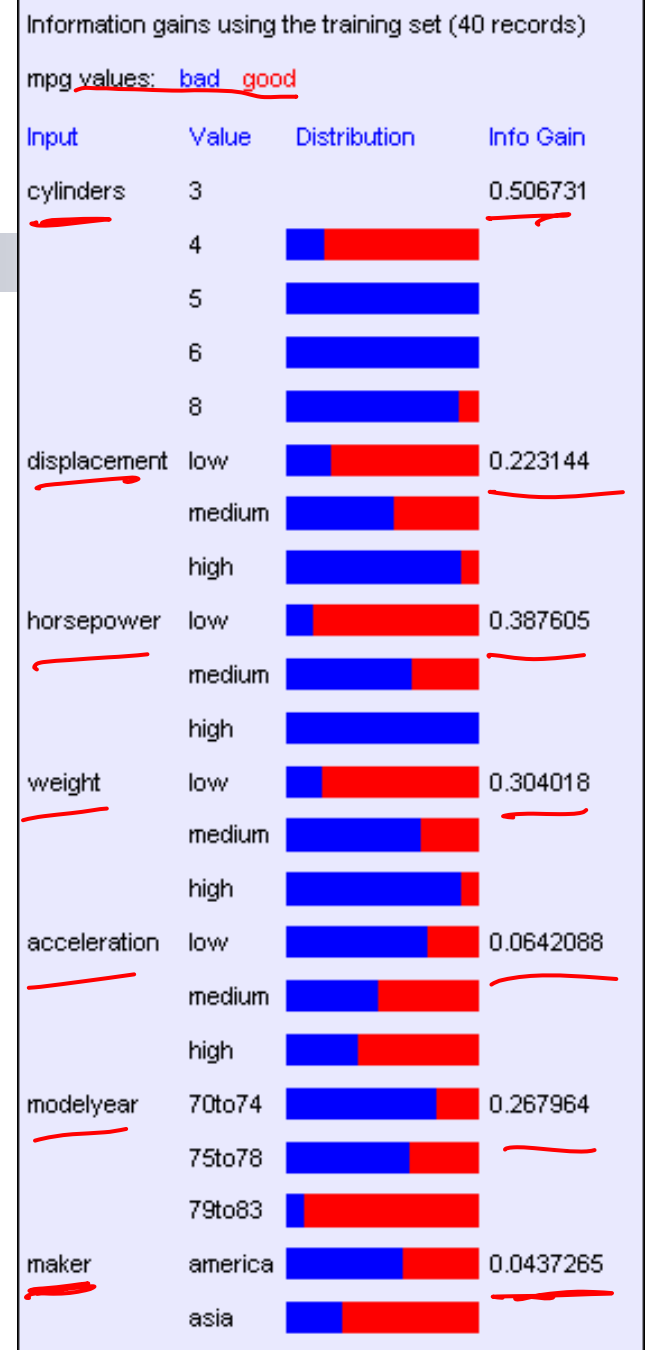
Male 22732 9918  $H(\text{wealth} \mid \text{gender} = \text{Male}) = 0.885847$

$H(\text{wealth}) = 0.793844$ $H(\text{wealth} \mid \text{gender}) = 0.757154$

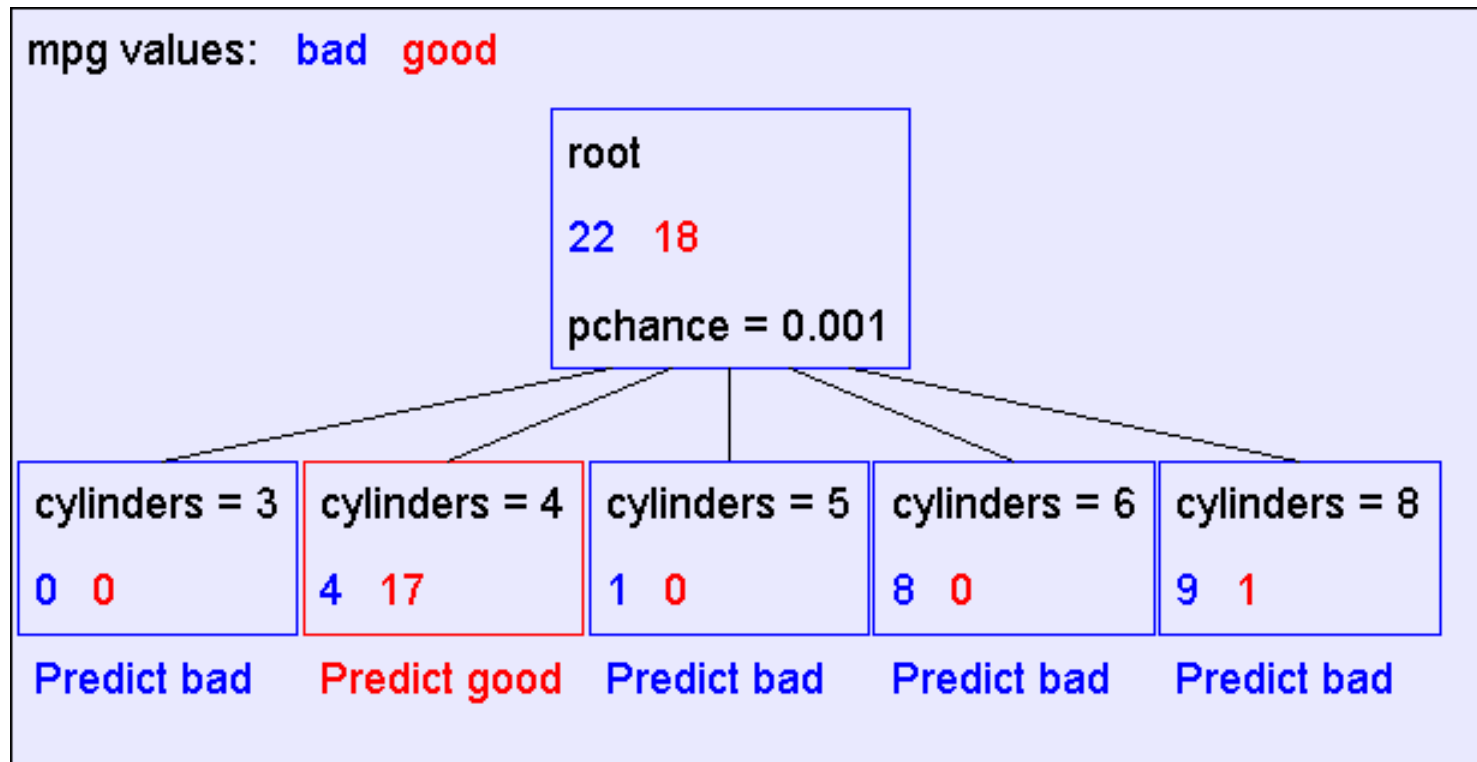
$IG(\text{wealth} \mid \text{gender}) = 0.0366896$

Suppose we want to
predict MPG

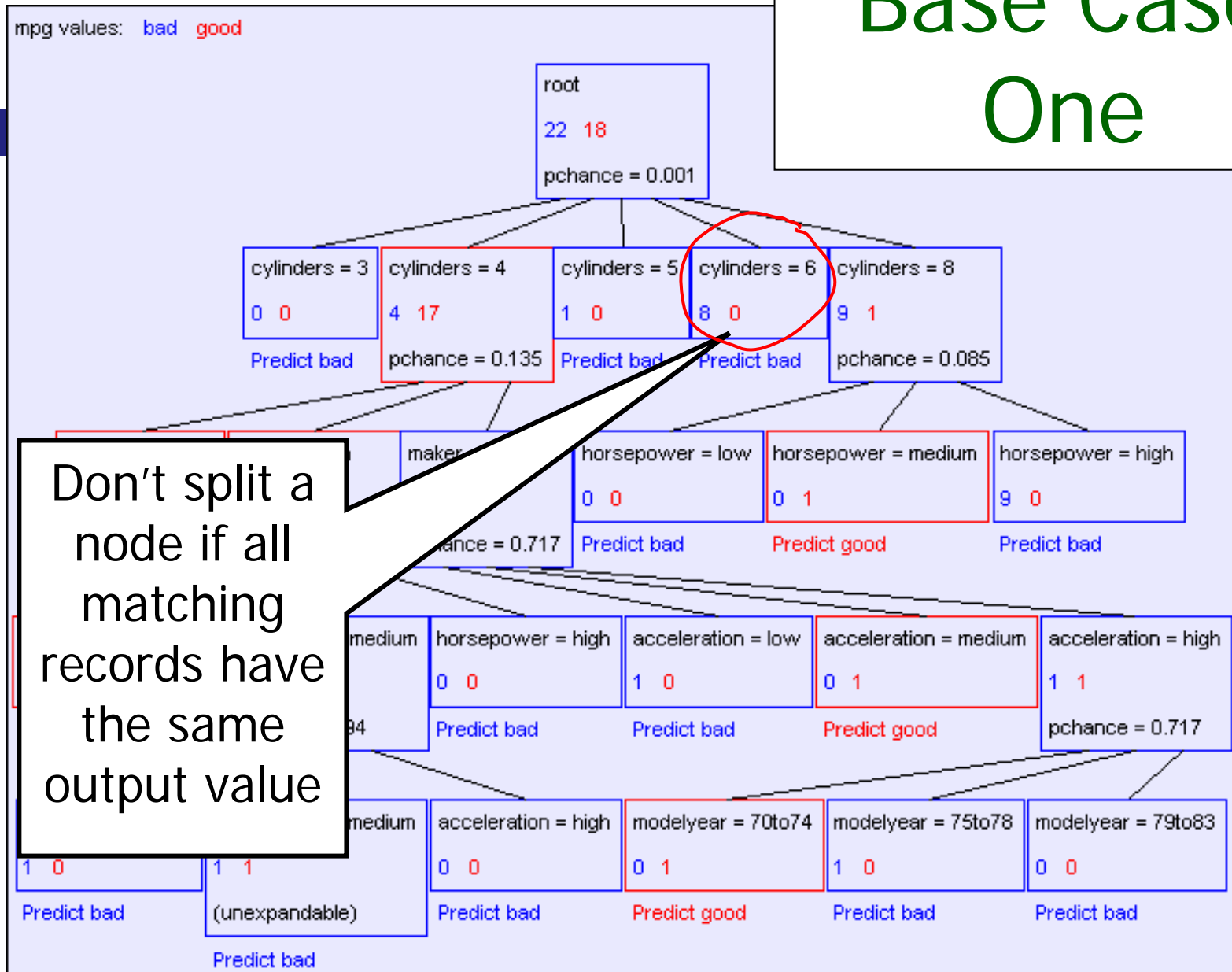
Look at all the
information
gains...



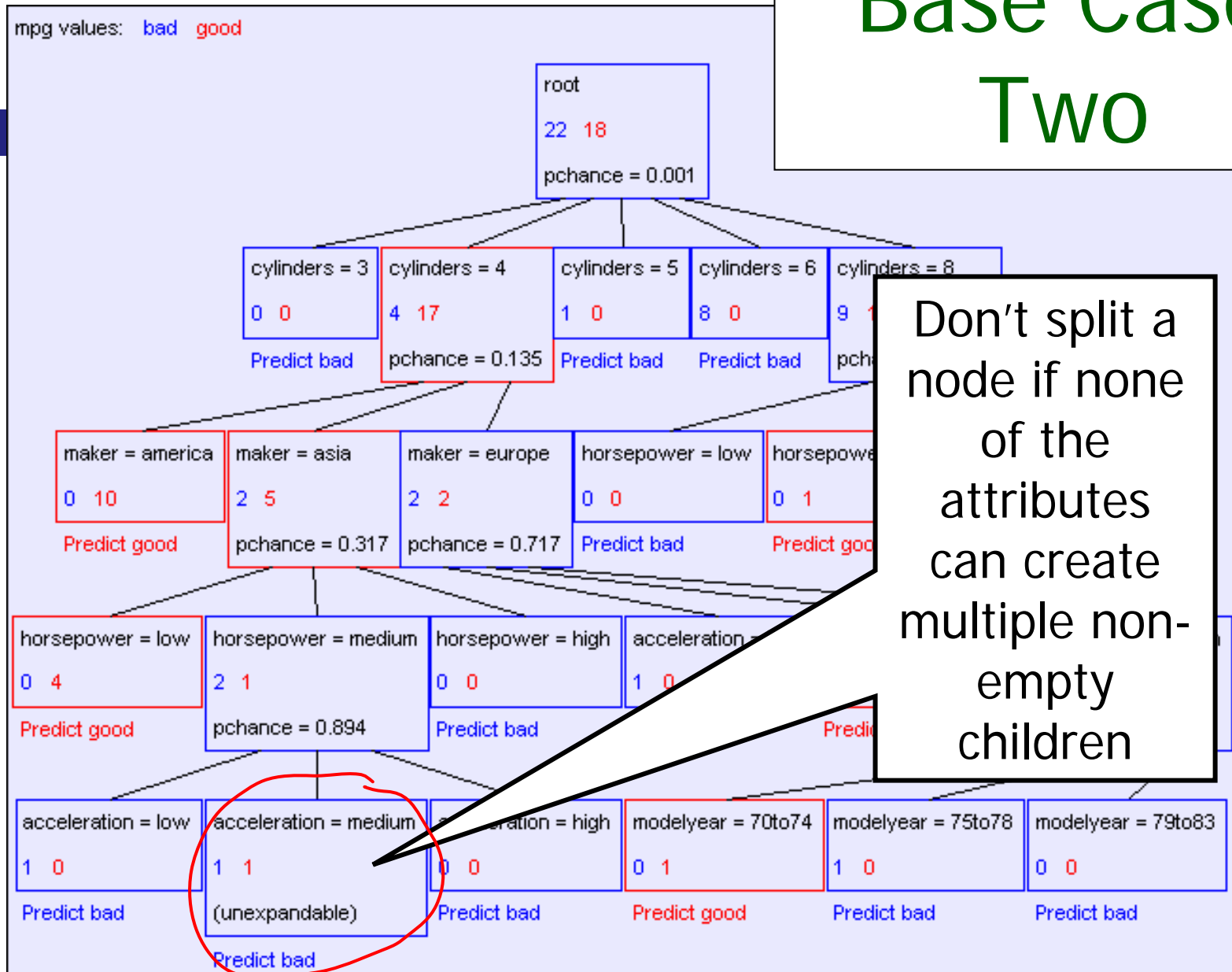
A Decision Stump



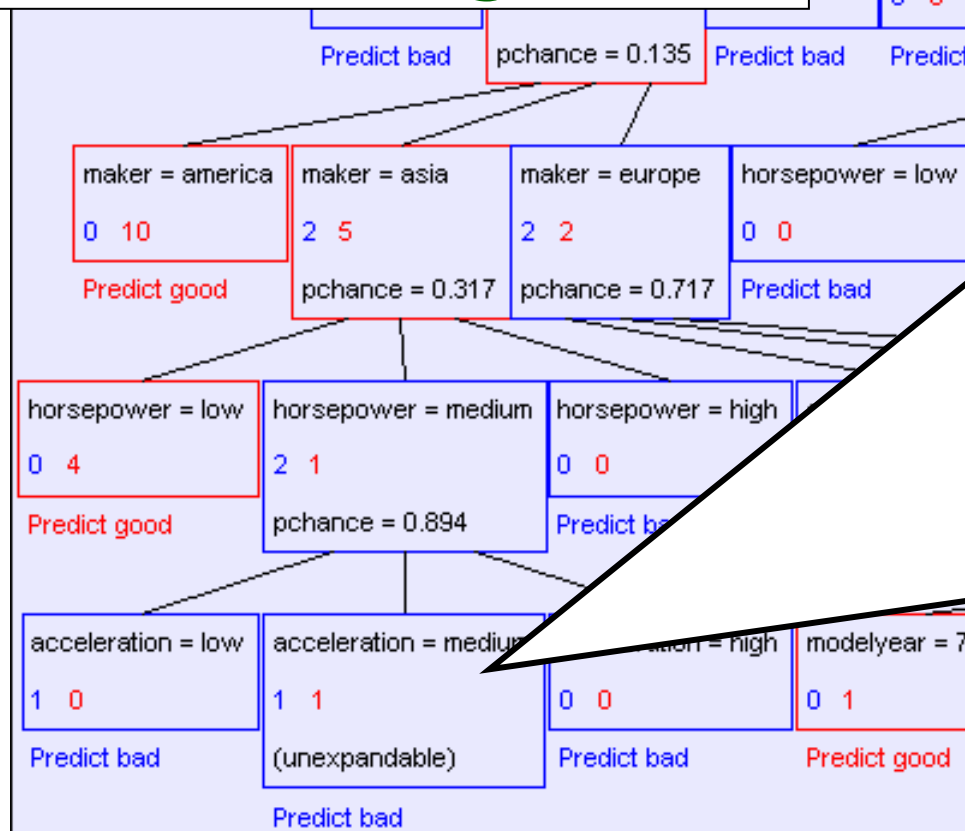
Base Case One



Base Case Two



Base Case Two: No attributes can distinguish



Information gains using the training set (2 records)

mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	3		0
	4		
	5		
	6		
	8		
displacement	low		0
	medium		
	high		
horsepower	low		0
	medium		
	high		
weight	low		0
	medium		
	high		
acceleration	low		0
	medium		
	high		
modelyear	70to74		0
	75to78		
	79to83		
maker	america		0
	asia		
	europa		

Base Cases



- Base Case One: If all records in current data subset have the same output then **don't recurse**
- Base Case Two: If all records have exactly the same set of input attributes then **don't recurse**

Base Cases: An idea

- Base Case One: If all records in current data subset have the same output then **don't recurse**
- Base Case Two: If all records have exactly the same set of input attributes then **don't recurse**

Proposed Base Case 3:

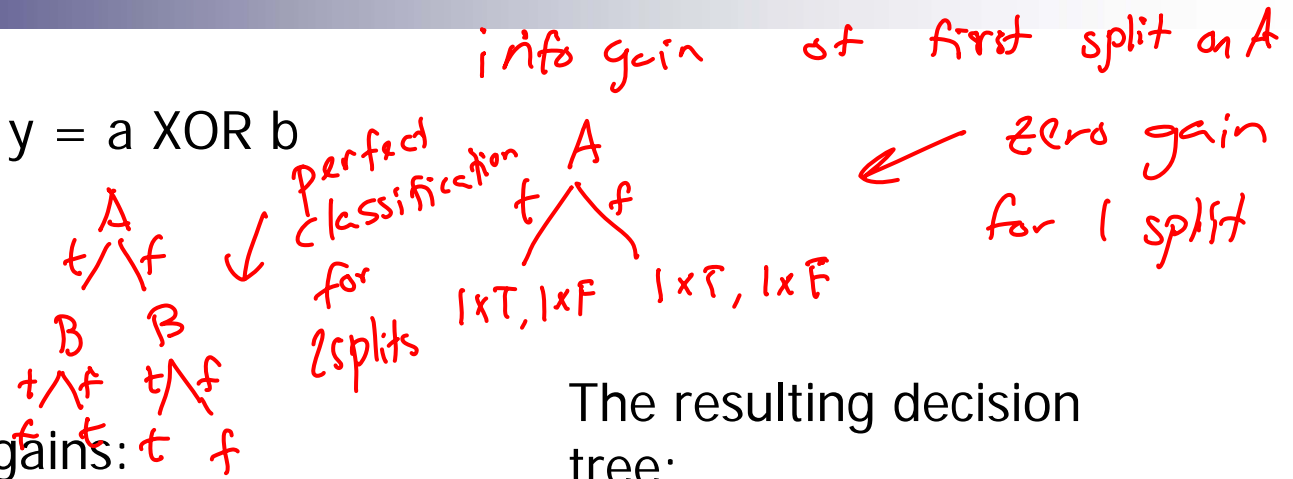
If all attributes have zero information
gain then **don't recurse**

• *Is this a good idea?*

The problem with Base Case 3

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ XOR } b$$



The information gains:

The resulting decision tree:

Information gains using the training set (4 records)

y values: 0 1

Input Value Distribution Info Gain

a 0  0

1 

b 0  0

1 

y values: 0 1

root

2 2

Predict 0

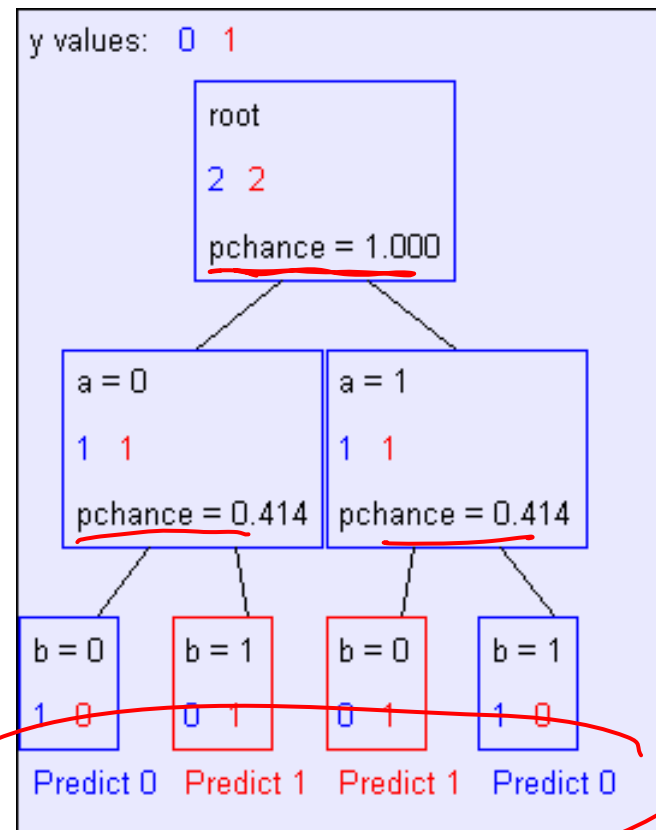
If we omit Base Case 3:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ XOR } b$$

The resulting decision tree:

*perfect
classif.*



Basic Decision Tree Building

Summarized

`BuildTree(DataSet, Output)`

- If all output values are the same in *DataSet*, return a leaf node that says “predict this unique output”
- If all input values are the same, return a leaf node that says “predict the majority output”
- Else find attribute X with highest Info Gain
- Suppose X has n_X distinct values (i.e. X has arity n_X).
 - Create and return a non-leaf node with n_X children.
 - The i th child should be built by calling

`BuildTree(DS_i , Output)`

Where DS_i built consists of all those records in *DataSet* for which $X = i$ th distinct value of X .

Real-Valued inputs

- What should we do if some of the inputs are real-valued?

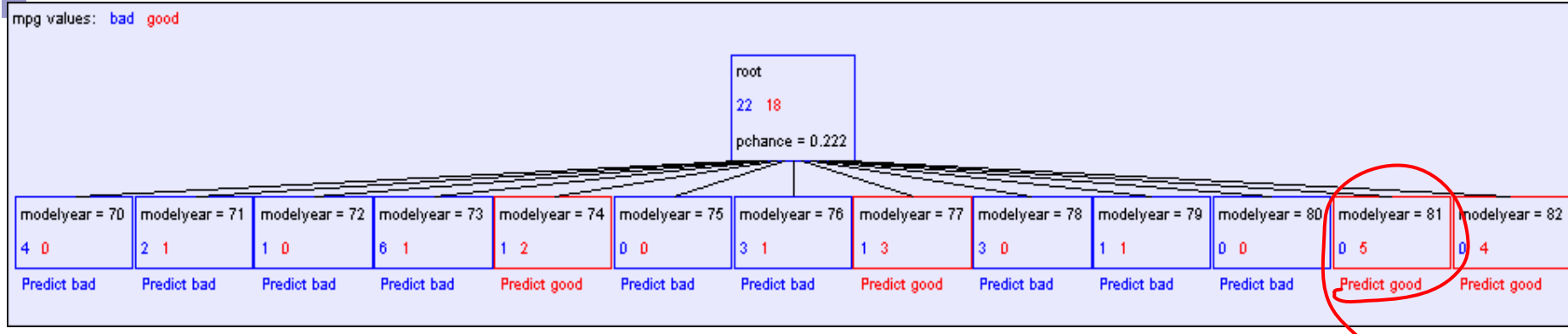
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europa
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	america
bad	8	455	225	4425	10	70	america
good	4	107	86	2464	15.5	76	europa
bad	5	131	103	2830	15.9	78	europa

Infinite number of possible split values!!!

Finite dataset, only finite number of relevant splits!

Idea One: Branch on each possible real value

“One branch for each numeric value” idea:

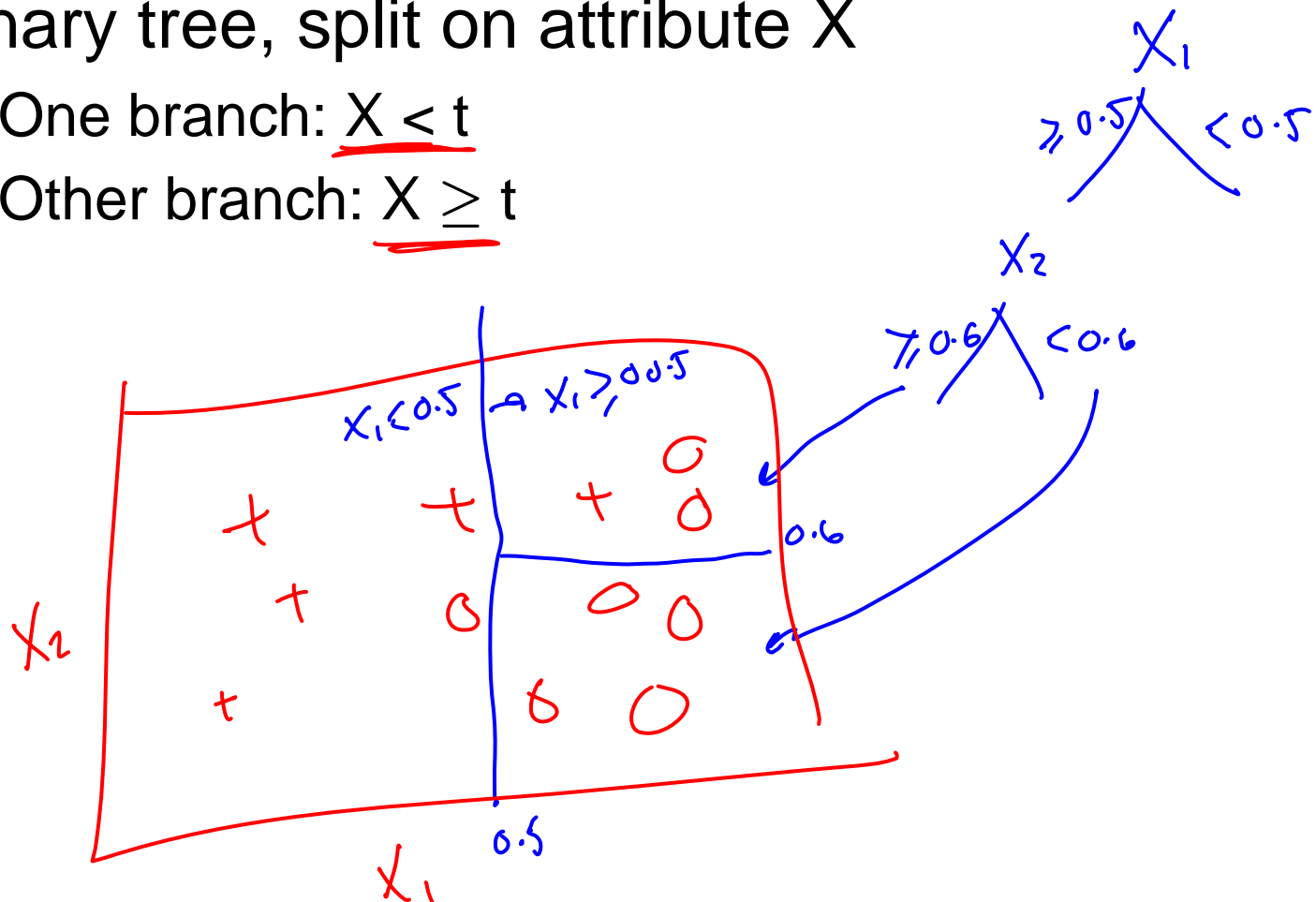


Hopeless: with such high branching factor will shatter the dataset and overfit *bad idea!*

Threshold splits

- Binary tree, split on attribute X

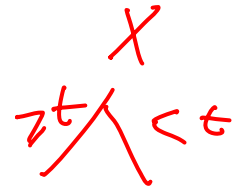
- One branch: $X < t$
- Other branch: $X \geq t$



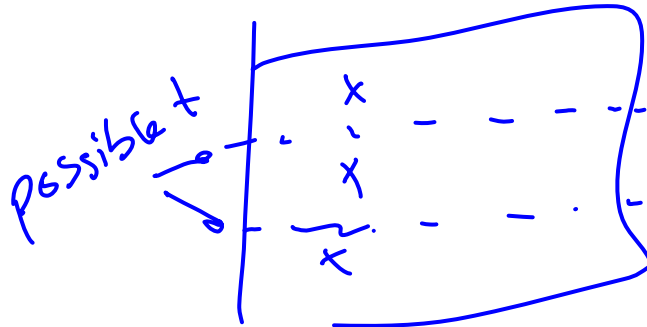
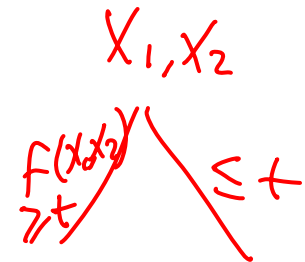
Choosing threshold split

- Binary tree, split on attribute X
 - One branch: $X < t$
 - Other branch: $X \geq t$
- Search through possible values of t
 - Seems hard!!!
- But only finite number of t 's are important
 - Sort data according to X into $\{x_1, \dots, x_m\}$
 - Consider split points of the form $x_i + (x_{i+1} - x_i)/2$

axis aligned split



non axis aligned


















A better idea: thresholded splits

- Suppose X is real valued
- Define $IG(Y|X:t)$ as $H(Y) - H(Y|X:t)$
- Define $H(Y|X:t) =$
$$H(Y|X < t) P(X < t) + H(Y|X \geq t) P(X \geq t)$$
 - $IG(Y|X:t)$ is the information gain for predicting Y if all you know is whether X is greater than or less than t
- Then define $IG^*(Y|X) = \max_t IG(Y|X:t)$
- For each real-valued attribute, use $IG^*(Y|X)$ for assessing its suitability as a split

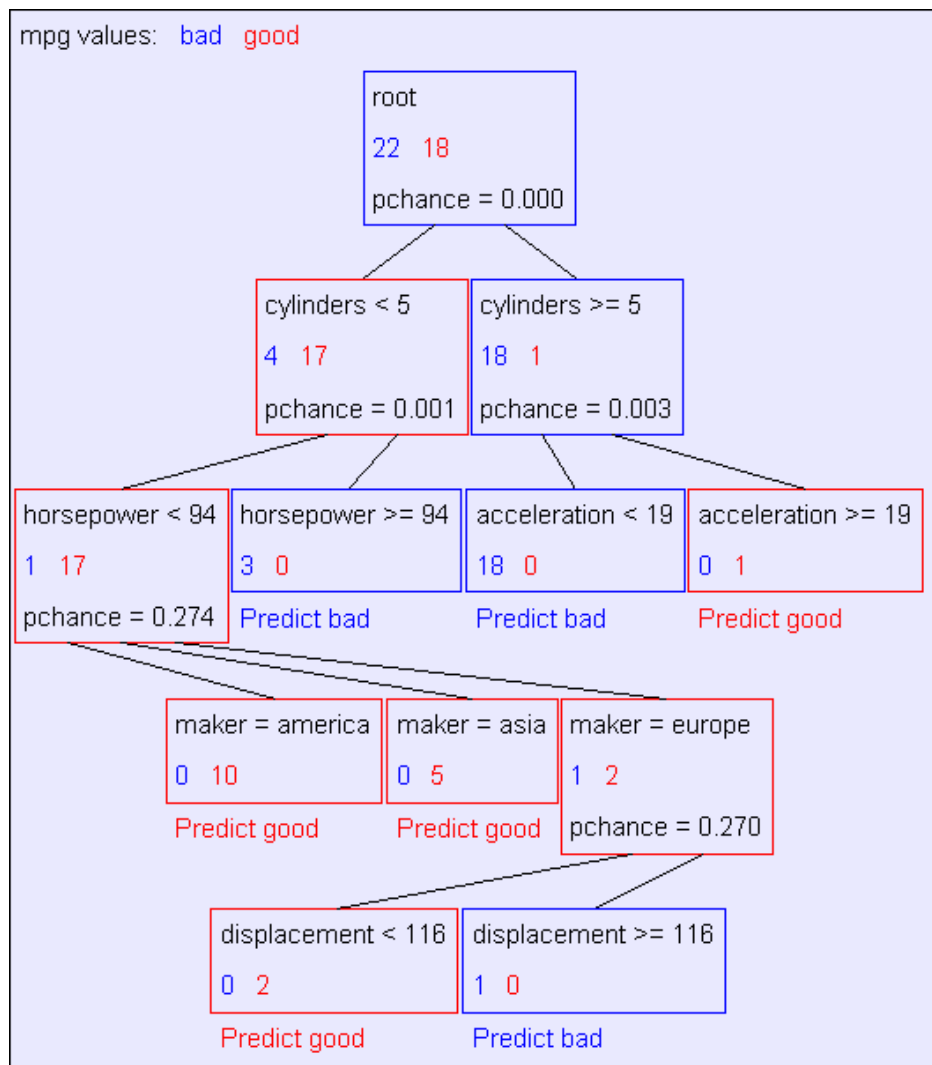
Example with MPG

Information gains using the training set (40 records)

mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	< 5		0.48268
	>= 5		
displacement	< 198		0.428205
	>= 198		
horsepower	< 94		0.48268
	>= 94		
weight	< 2789		0.379471
	>= 2789		
acceleration	< 18.2		0.159982
	>= 18.2		
modelyear	< 81		0.319193
	>= 81		
maker	america		0.0437265
	asia		
	europa		

Example tree using reals



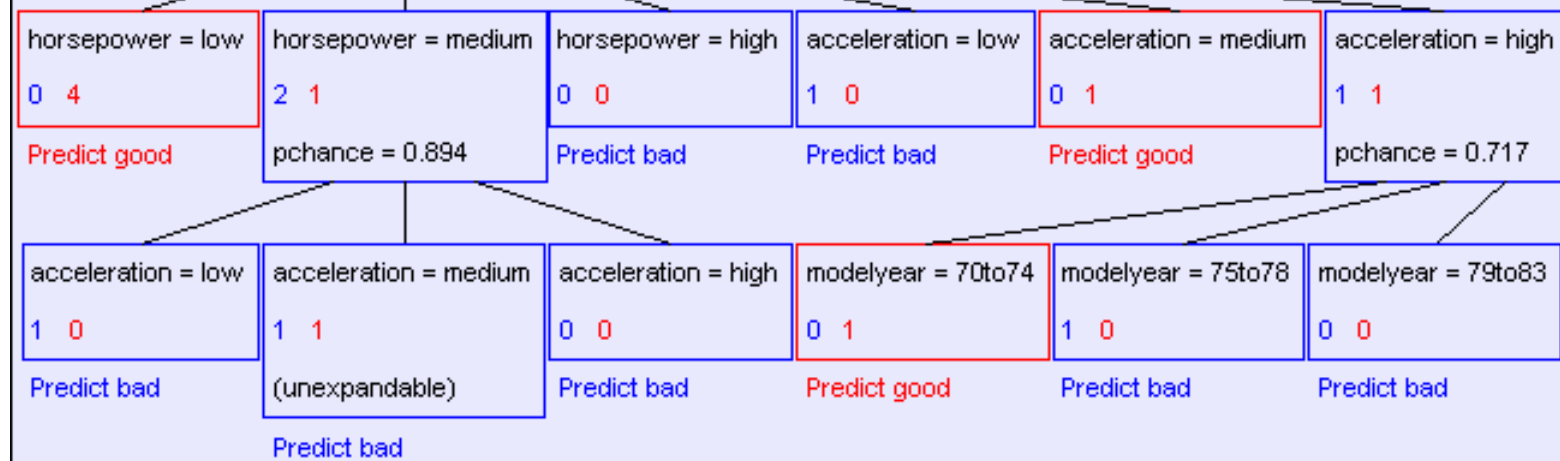
MPG Test set error

mpg values: bad good

root
22 18
pchance = 0.001

	Num Errors	Set Size	<u>Percent Wrong</u>
<u>Training Set</u>	1	40	<u>2.50</u>
<u>Test Set</u>	74	352	<u>21.02</u>

why?
overfitting!



MPG Test set error

mpg values: bad good

root
22 18
pchance = 0.001

	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

epower = high

ict bad

horsepower = low

horsepower = medium

horsepower = high

acceleration = low

acceleration = medium

acceleration = high

The test set error is much worse than the training set error...

...why?

Predict bad

(unexpandable)

Predict bad

Predict good

Predict bad

Predict bad

Predict bad

Decision trees & Learning Bias

Any "separable" data

→ no examples with
same attrib. values
have diff. labels

Can be classified exactly with
a d. tree

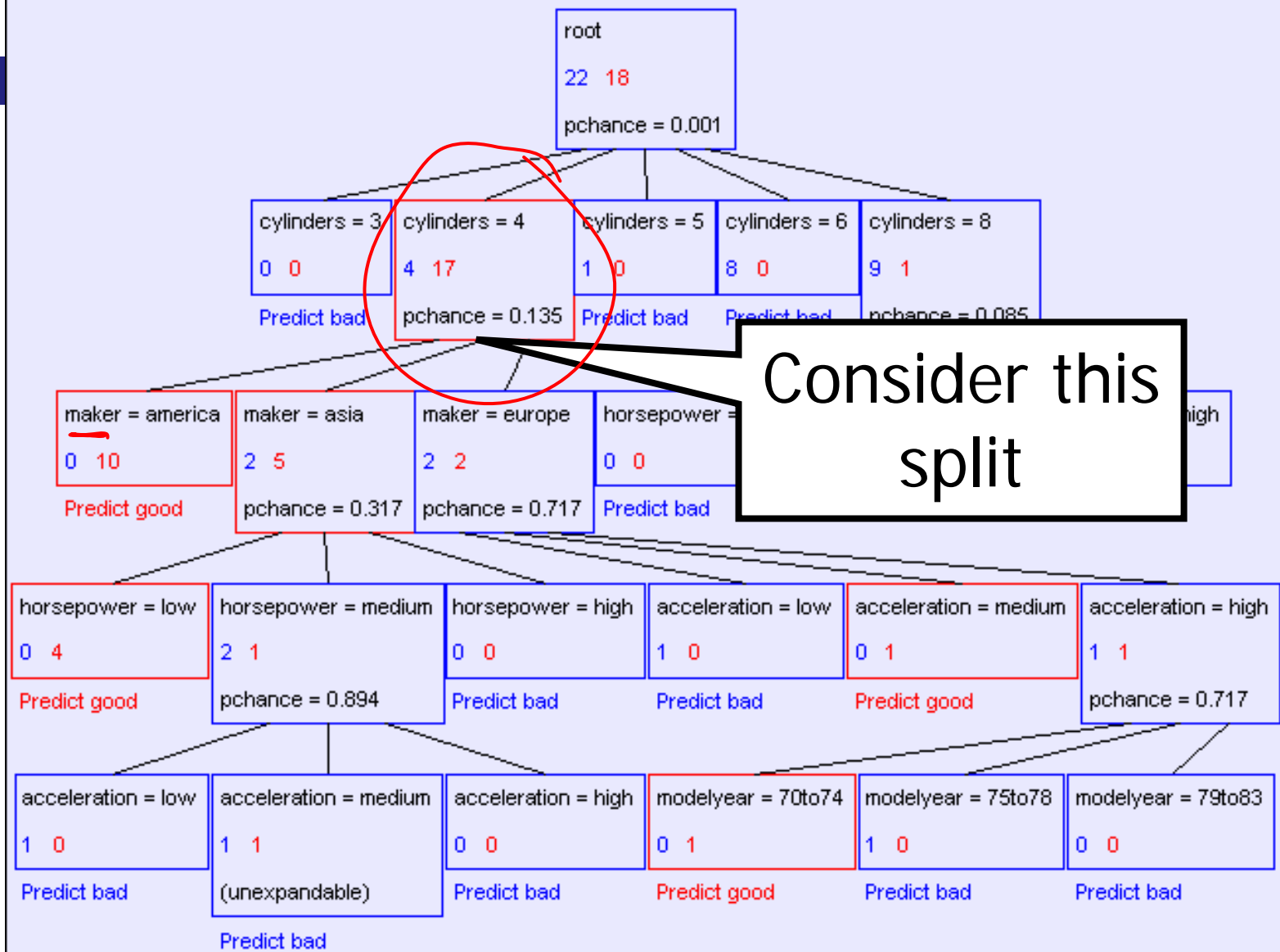
→ (almost) zero bias!

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europa
bad	5	medium	medium	medium	medium	75to78	europa

Decision trees will overfit

- Standard decision trees are have no learning biased
 - Training set error is always zero!
 - Lots of variance
 - Will definitely overfit!!!
 - Must bias towards simpler trees
- Many strategies for picking simpler trees:
 - Fixed depth *5 d. trees*
 - Fixed number of leaves *1000 leaves*
 - Or something smarter...

mpg values: bad good



A chi-square test

mpg values: bad good

maker	america	<u>0</u>	<u>10</u>			$H(\text{mpg} \mid \text{maker} = \text{america}) = \underline{0}$
	asia	<u>2</u>	<u>5</u>			$H(\text{mpg} \mid \text{maker} = \text{asia}) = \underline{0.863121}$
	europa	<u>2</u>	<u>2</u>			$H(\text{mpg} \mid \text{maker} = \text{europa}) = \underline{1}$






$H(\text{mpg}) = 0.702467$ $H(\text{mpg} \mid \text{maker}) = 0.478183$

$IG(\text{mpg} \mid \text{maker}) = 0.224284$

- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

A chi-square test

mpg values: bad good

maker	america	0	10			$H(\text{mpg} \mid \text{maker} = \text{america}) = 0$
	asia	2	5			$H(\text{mpg} \mid \text{maker} = \text{asia}) = 0.863121$
	europa	2	2			$H(\text{mpg} \mid \text{maker} = \text{europa}) = 1$

$H(\text{mpg}) = 0.702467$ $H(\text{mpg} \mid \text{maker}) = 0.478183$
 $IG(\text{mpg} \mid \text{maker}) = 0.224284$

- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

By using a particular kind of chi-square test, the answer is 13.5%

(Such simple hypothesis tests are very easy to compute, unfortunately, not enough time to cover in the lecture)

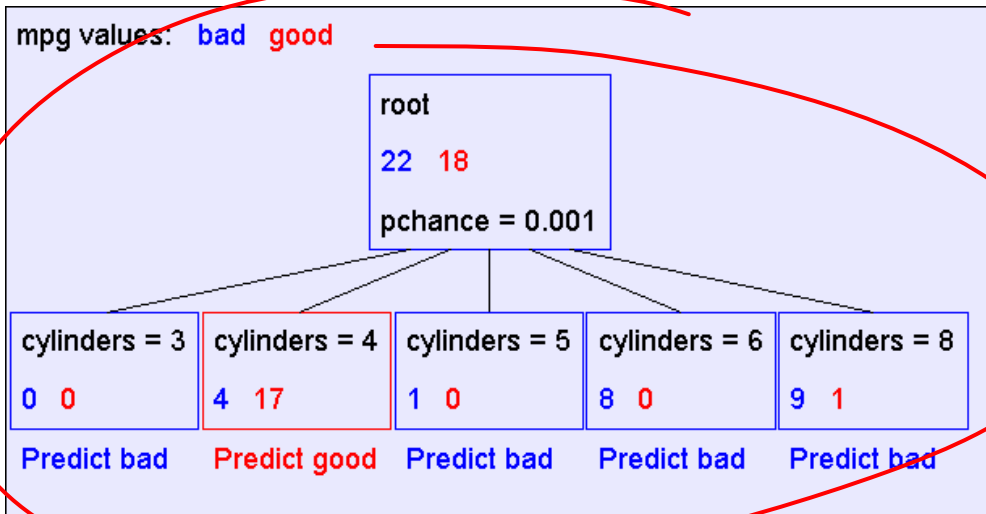
Using Chi-squared to avoid overfitting

- Build the full decision tree as before
- But when you can grow it no more, start to prune:
 - Beginning at the bottom of the tree, delete splits in which $p_{chance} > MaxPchance$
 - Continue working your way up until there are no more prunable nodes

MaxPchance is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise

Pruning example

- With $\text{MaxPchance} = 0.1$, you will see the following MPG decision tree:



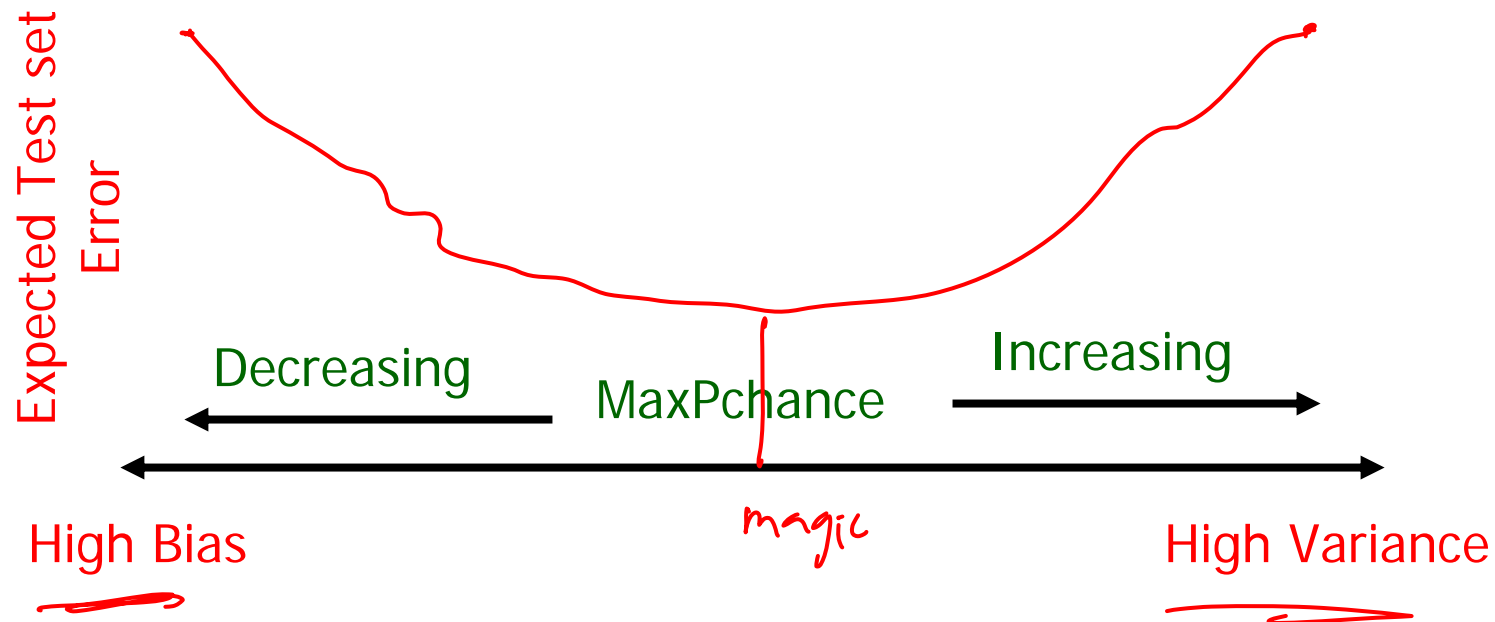
get a decision stump

Note the improved test set accuracy compared with the unpruned tree

	Num Errors	Set Size	Percent Wrong
Training Set	5	40	<u>12.50</u> higher
Test Set	56	352	<u>15.91</u> lower

MaxPchance

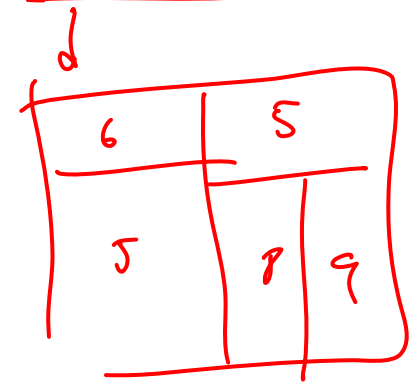
- Technical note MaxPchance is a regularization parameter that helps us bias towards simpler models



- We'll learn to choose the value of these magic parameters soon!

What you need to know about decision trees

- Decision trees are one of the most popular data mining tools
 - Easy to understand
 - Easy to implement
 - Easy to use
 - Computationally cheap (to solve heuristically)
- Information gain to select attributes (ID3, C4.5,...)
- Presented for classification, can be used for regression and density estimation too
- Decision trees will overfit!!!
 - Zero bias classifier → Lots of variance
 - Must use tricks to find “simple trees”, e.g.,
 - Fixed depth/Early stopping
 - Pruning
 - Hypothesis testing



Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
 - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
 - Low variance, don't usually overfit
- **Simple (a.k.a. weak) learners are bad**
 - High bias, can't solve hard learning problems
- Can we make weak learners always good???
 - **No!!!**
 - **But often yes...**

Boosting [Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t :
 - weight each training example by how incorrectly it was classified
 - Learn a hypothesis – h_t
 - A strength for this hypothesis – α_t
- Final classifier:
- **Practically useful**
- **Theoretically interesting**

Learning from weighted data

- Sometimes not all data points are equal
 - Some data points are more equal than others
- Consider a weighted dataset
 - $D(i)$ – weight of i th training example (\mathbf{x}^i, y^i)
- Now, in all calculations, whenever used, i th training example counts as $D(i)$ “examples”
 - e.g., MLE for Naïve Bayes, redefine $Count(Y=y)$ to be weighted count

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train base learner using distribution D_t .
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right).$$

Figure 1: The boosting algorithm AdaBoost.

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train base learner using distribution D_t .
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\epsilon_t = P_{i \sim D_t}[\mathbf{x}^i \neq y^i]$$

$$\epsilon_t = \frac{1}{\sum_{i=1}^m D_t(i)} \sum_{i=1}^m D_t(i) \delta(h_t(x_i) \neq y_i)$$

What α_t to choose for hypothesis h_t ?


Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i))$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

What α_t to choose for hypothesis h_t ?

Training error of final classifier is bounded by:


$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

What α_t to choose for hypothesis h_t ?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

If we minimize $\prod_t Z_t$, we minimize our training error

We can tighten this bound by choosing α_t and h_t on each iteration to minimize Z_t .

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

What α_t to choose for hypothesis h_t ?

We can minimize this bound by choosing α_t on each iteration to minimize Z_t .

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Strong, weak classifiers

- If each classifier is (at least slightly) better than random
 - $\epsilon_t < 0.5$
- AdaBoost will achieve zero *training error* (exponentially fast):

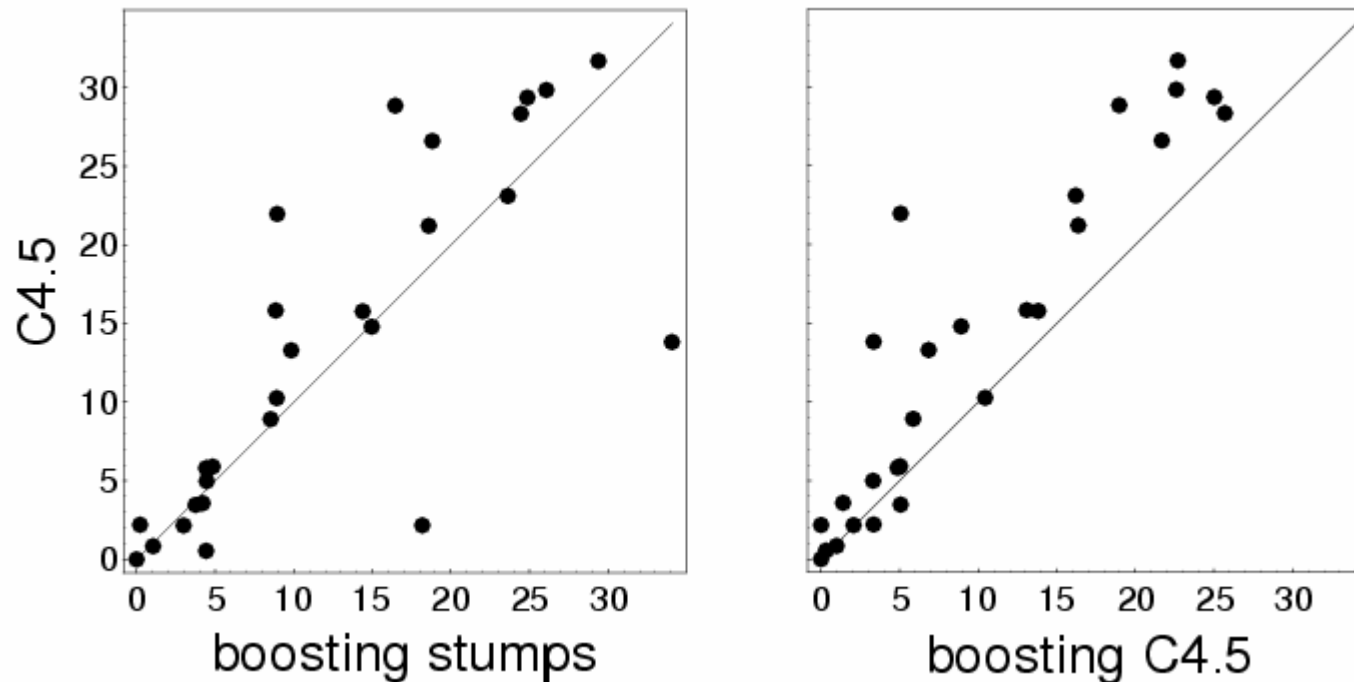
$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \prod_t Z_t \leq \exp \left(-2 \sum_{t=1}^T (1/2 - \epsilon_t)^2 \right)$$

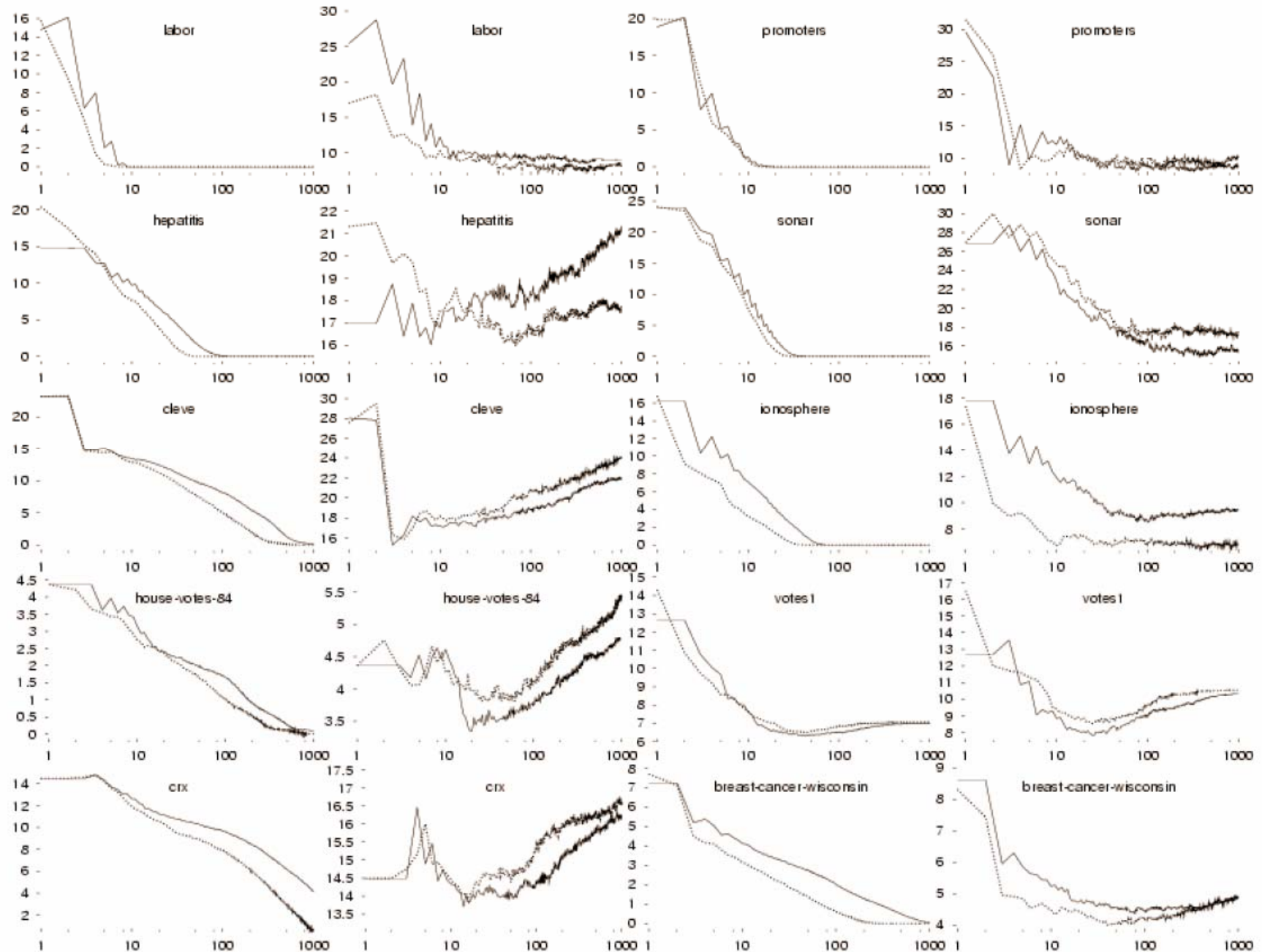
- Is it hard to achieve better than random training error?

Boosting: Experimental Results

[Freund & Schapire, 1996]

Comparison of C4.5, Boosting C4.5, Boosting decision stumps (depth 1 trees), 27 benchmark datasets





Boosting and Logistic Regression

Logistic regression assumes:

$$P(Y = 1|X) = \frac{1}{1 + \exp(f(x))}$$

And tries to maximize data likelihood:

$$P(data|H) = \prod_{i=1}^m \frac{1}{1 + \exp(-y_i f(x_i))}$$

Equivalent to minimizing log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

Boosting minimizes similar loss function!!

$$\frac{1}{m} \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$$

Both smooth approximations of 0/1 loss!

Logistic regression and Boosting

Logistic regression:

- Minimize loss fn

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where x_j predefined

Boosting:

- Minimize loss fn

$$\sum_{i=1}^m \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where $h(x_i)$ defined
dynamically to fit data

- Weights α_j learned
incrementally

What you need to know about Boosting

- Combine weak classifiers to obtain very strong classifier
 - Weak classifier – slightly better than random on training data
 - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
 - Similar loss functions
 - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
 - Boosted decision stumps!
 - Very simple to implement, very effective classifier

Acknowledgements



- Much of the decision trees material in the presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:
 - <http://www.cs.cmu.edu/~awm/tutorials>
- Much of the boosting material in the presentation is courtesy of Tom Mitchell