Decision Trees

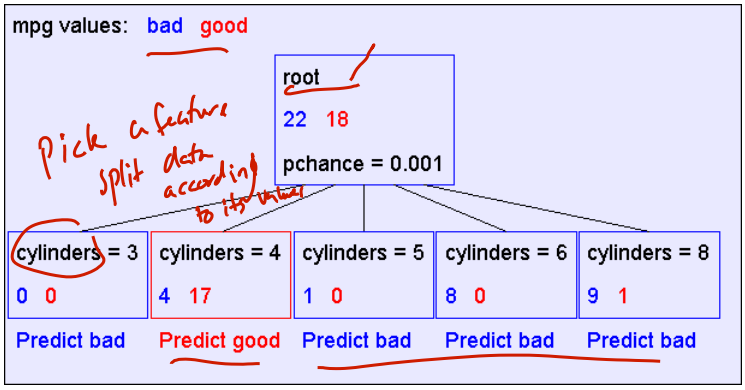
Machine Learning – CSE446

Carlos Guestrin

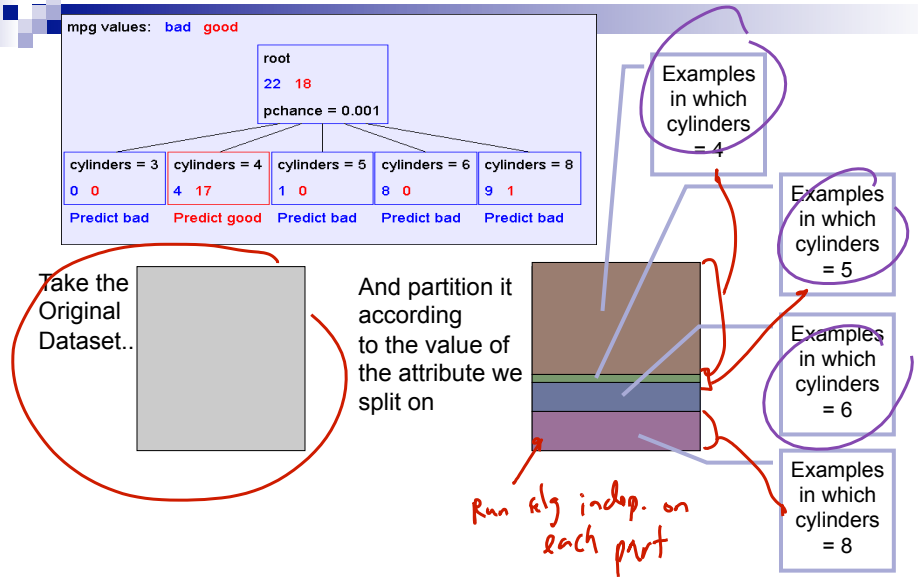
University of Washington

April 22, 2013

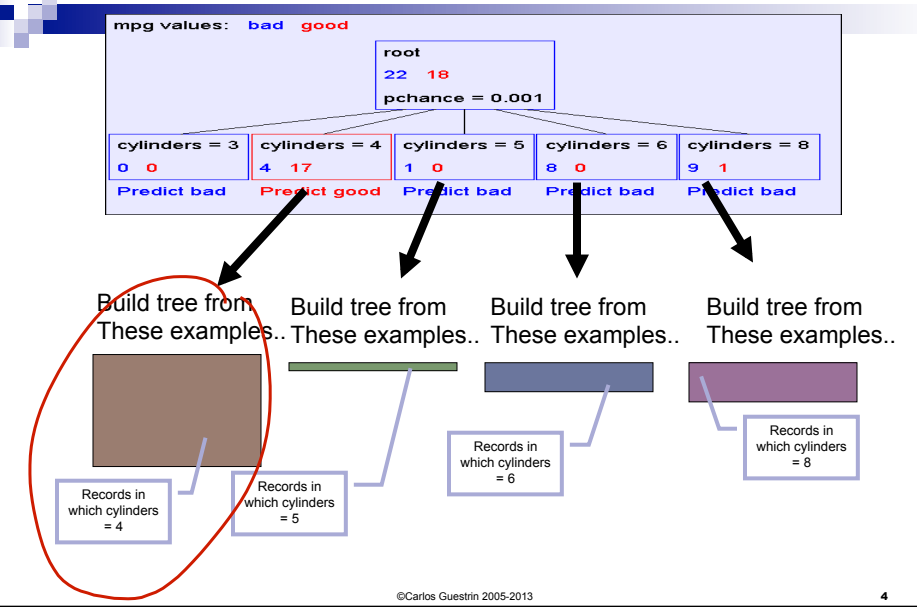
A Decision Stump



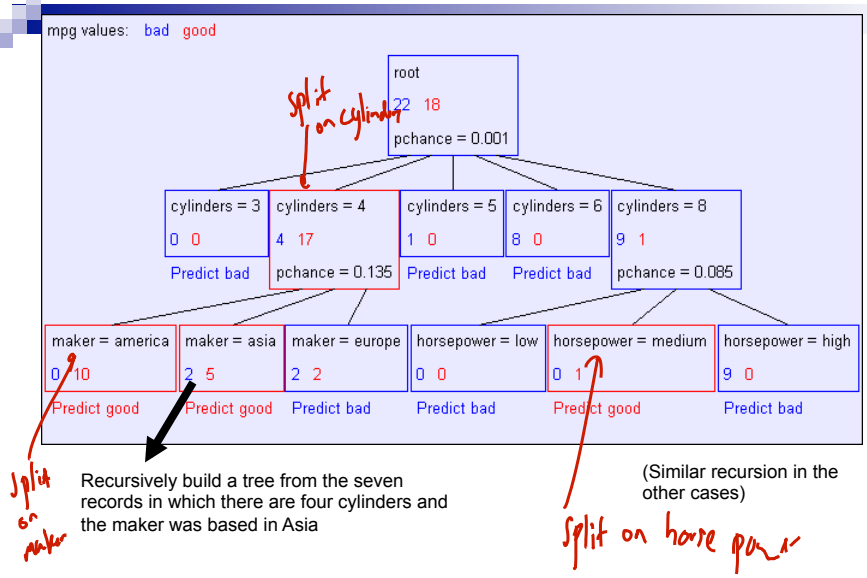
Recursion Step



Recursion Step

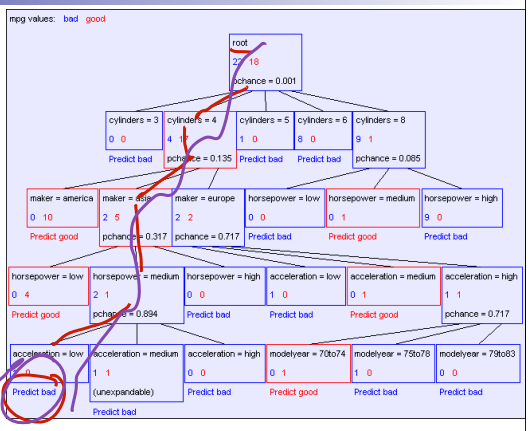


Second level of tree



Classification of a new example

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

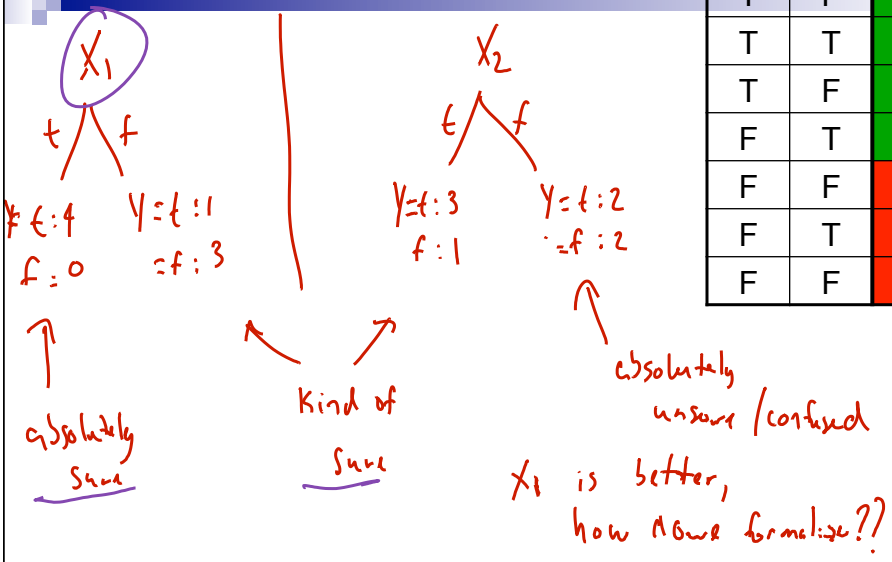


Learning decision trees is hard!!!

- 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



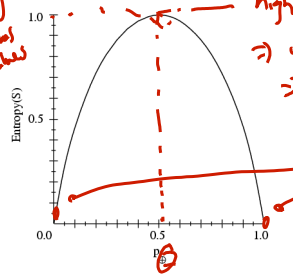
Entropy

Entropy $H(Y)$ 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)



Information gain

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before you split
  - Entropy after split
    - Weight by probability of following each branch, i.e., normalized number of records

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

- Information gain is difference $IG(X) = H(Y) - H(Y \mid X)$

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

Learning decision trees

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

predict most common class

on each branch of tree

When do we stop?

1. When all data is from 1 class?
2. Entropy is below threshold ???
3. Nothing to split on — split on vars no vars give a diff split

Suppose we want to predict MPG

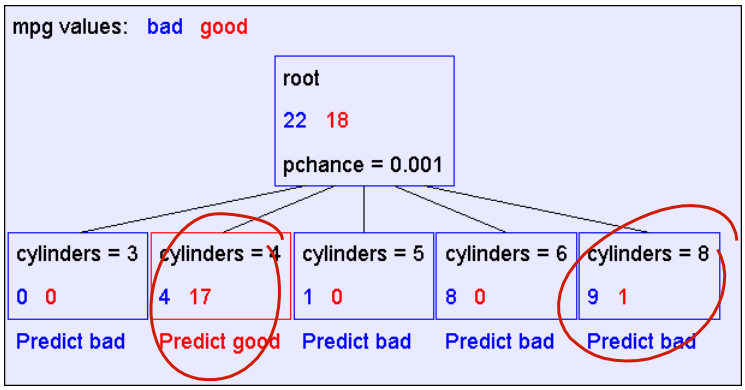
Look at all the information gains…

Information gains using the training set (40 records)

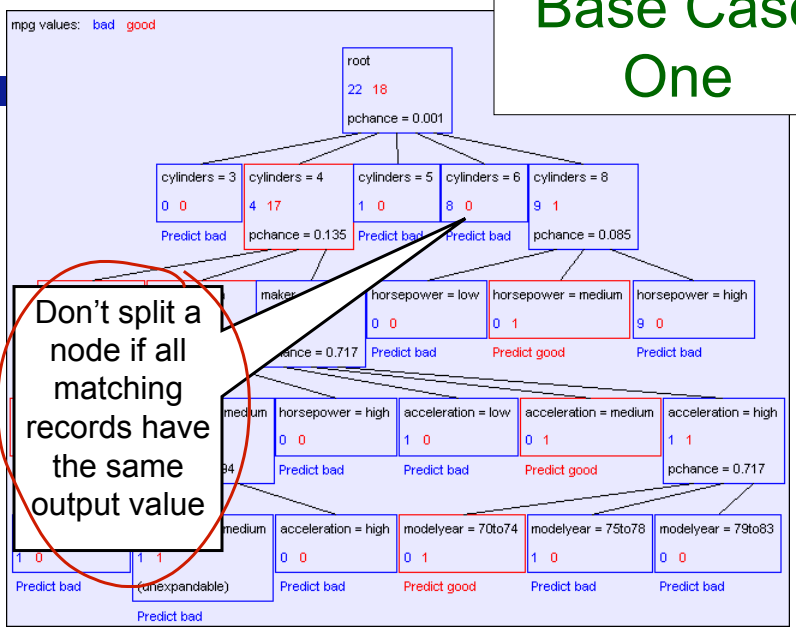
mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	3		0.506731
	4		
	5		
	6		
	8		
displacement	low		0.223144
	medium		
	high		
horsepower	low		0.387605
	medium		
	high		
weight	low		0.304018
	medium		
	high		
acceleration	low		0.0642088
	medium		
	high		
modelyear	70to74		0.267964
	75to78		
	79to83		
maker	america		0.0437265
	asia		

A Decision Stump

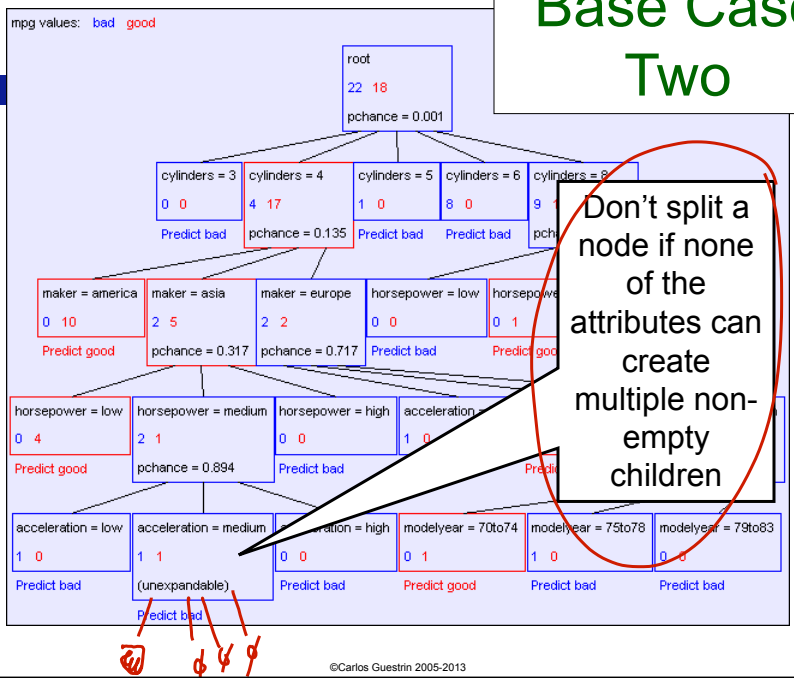


Base Case One

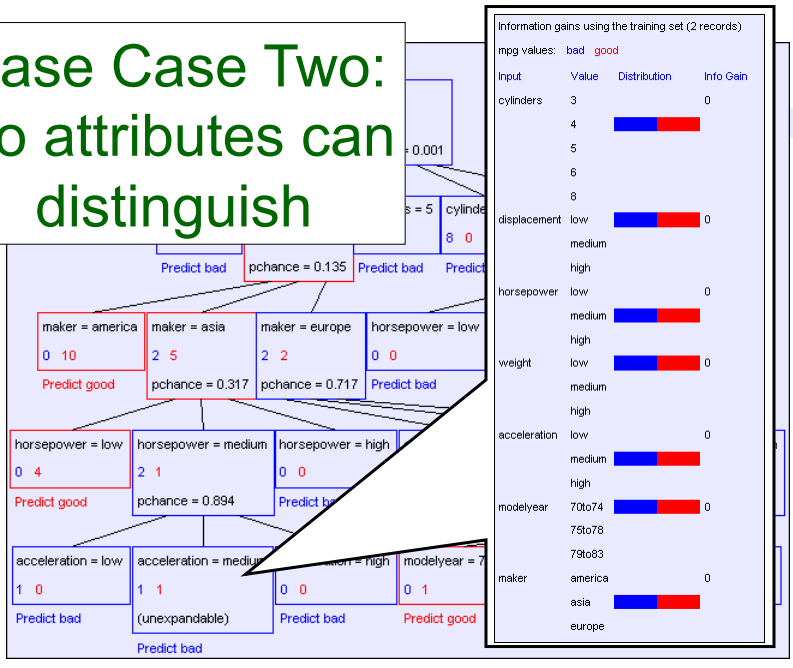


Don't split a node if all matching records have the same output value

Base Case Two



Base Case Two: No attributes can distinguish

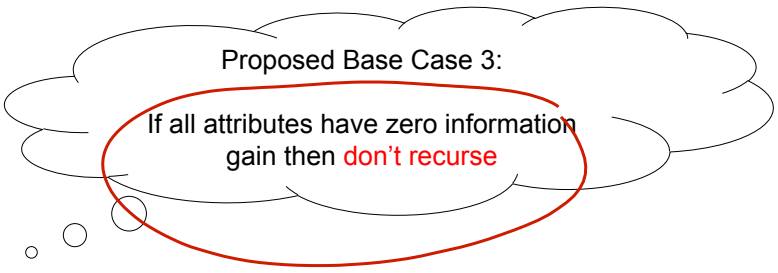


Base Cases

- 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



- *Is this a good idea?*

The problem with Base Case 3

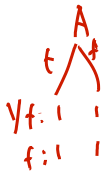
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Y = A XOR B

if I tell you value of A learn nothing about y ⇒ IG(A) = 0

if I tell you both A & B know everything about y

The information gains:



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		

The resulting bad decision tree:

y values: 0 1

root

2 2

Predict 0

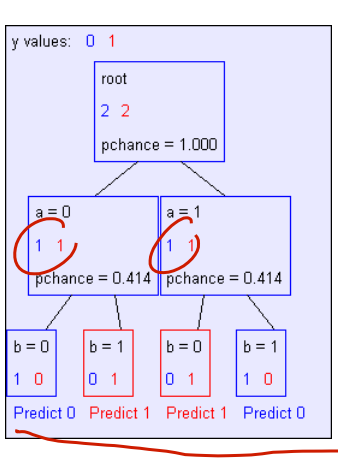
If we omit Base Case 3:

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

y = a XOR b

The resulting decision tree:

perfect prediction if split on both



Basic Decision Tree Building Summarized

BuildTree(*DataSet,Output*)

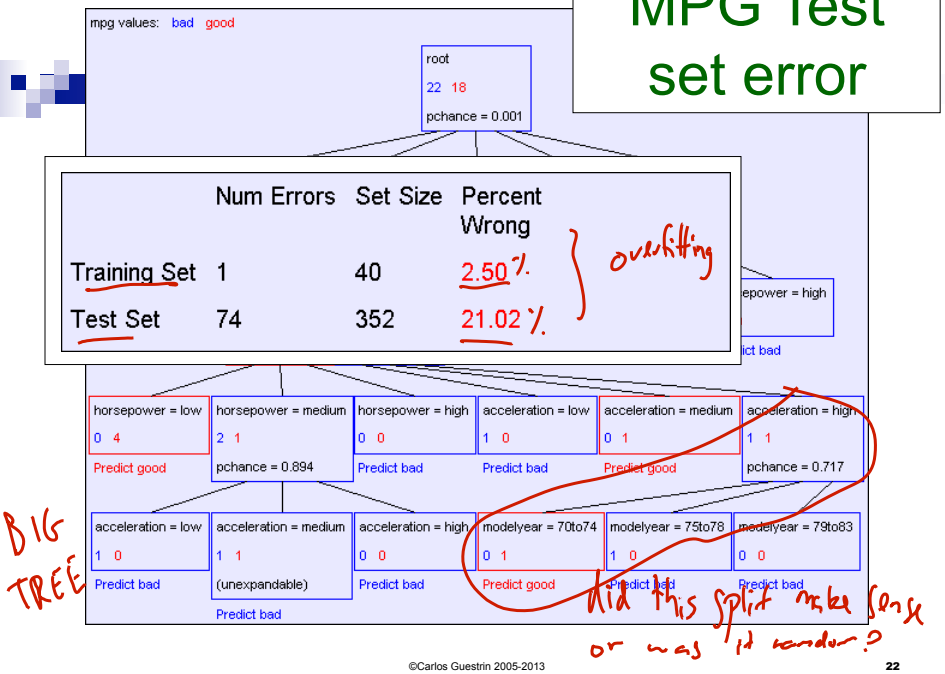
- If all output values are the same in *DataSet*, return a leaf node that says "predict this unique output"
- If all input values are the same, return a leaf node that says "predict the majority output"
- Else find attribute *X* with highest Info Gain
- Suppose *X* has n_X distinct values (i.e. X has 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*)

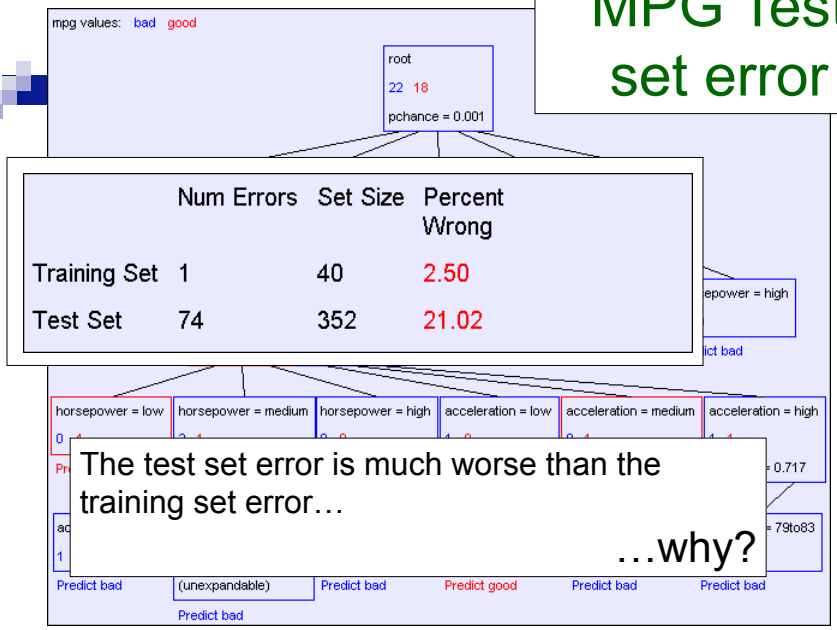
    Where DS_i built consists of all those records in DataSet for which X = *i*th distinct value of X.

Recurse non-stop

MPG Test set error



MPG Test set error



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

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

…why?

Decision trees & Learning Bias

Suppose no "label noise", i.e., no 2 data points with same X that have different Y

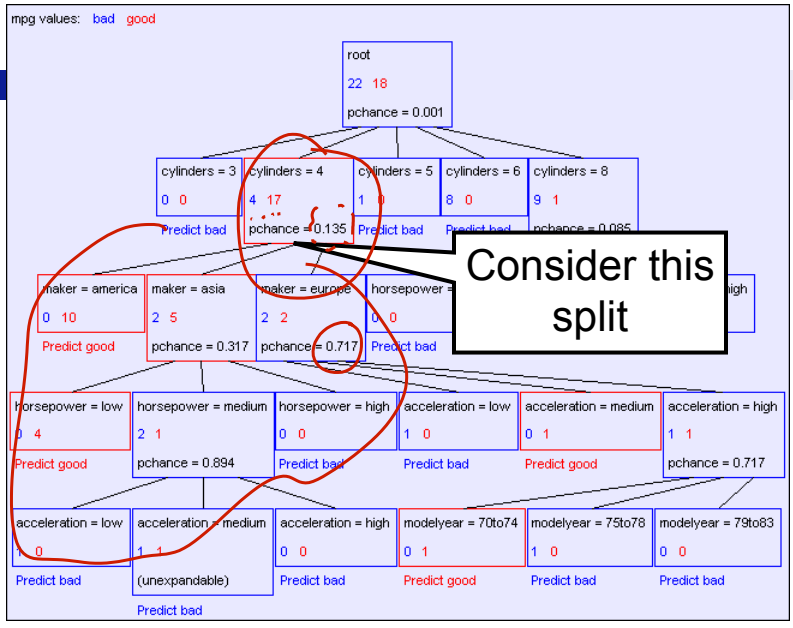
⇒ if keep splitting decision trees: $error_{train}(Tree) = 0$

⇒ "bias" = 0, but variance is high

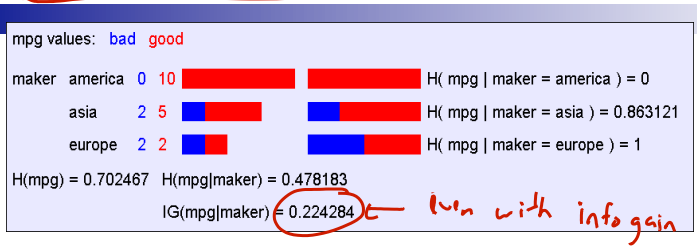
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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

Decision trees will overfit

- Standard decision trees are have no learning bias
  - Training set error is always zero!
    - (If there is no label noise)
  - Lots of variance
  - Will definitely overfit!!!
  - Must bias towards simpler trees
- Many strategies for picking simpler trees:
  - Fixed depth : 1, 2 or 3
  - Fixed number of leaves
  - Or something smarter…

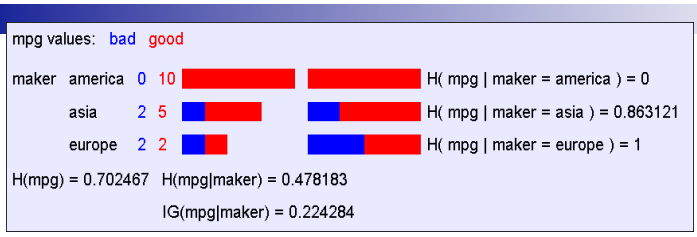


A chi-square test



- 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



- 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 7.2%

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

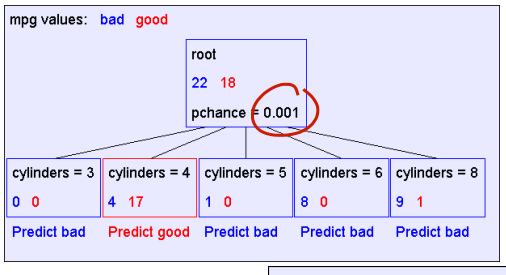
Using Chi-squared to avoid overfitting

- 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 you 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 MaxPchance = 0.1, you will see the following MPG decision tree:

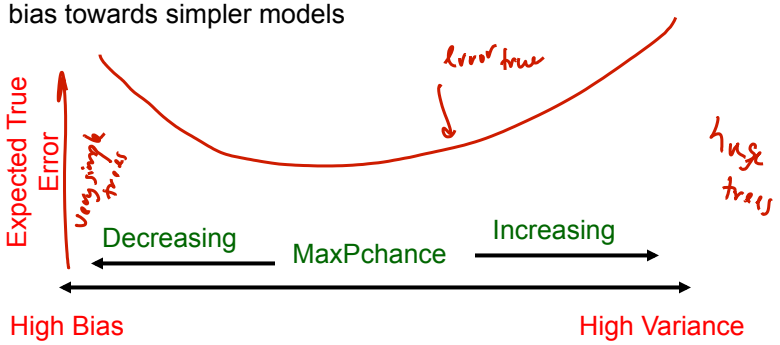


Note the improved test set accuracy compared with the unpruned tree

	Num Errors	Set Size	Percent Wrong
Training Set	5	40	12.50
Test Set	56	352	15.91

MaxPchance

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



Real-Valued inputs

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

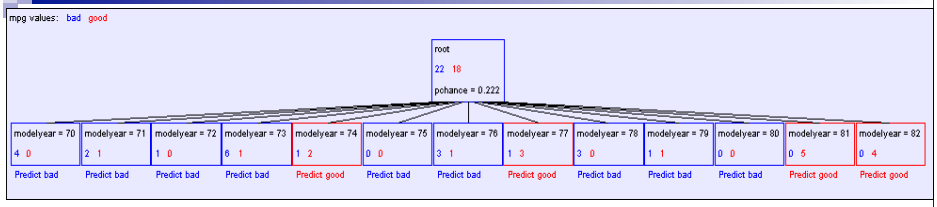
mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	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	europe
bad	5	131	103	2830	15.9	78	europe

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

Threshold splits

- Binary tree, split on attribute X
  - One branch: X < t
  - Other branch: X ≥ t

Choosing threshold split

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

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 >= t) P(X >= 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
- Note, may split on an attribute multiple times, with different thresholds

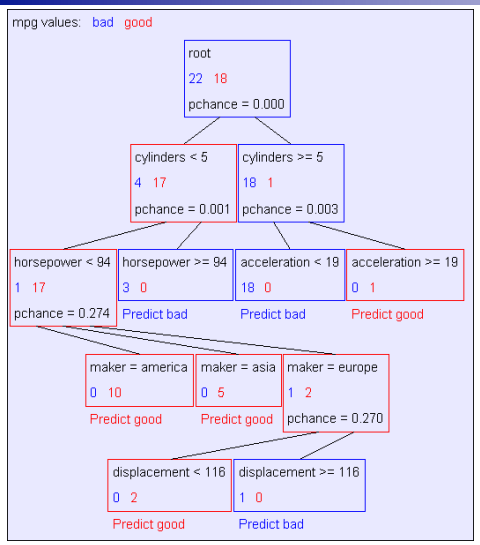
Example with MPG

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		
	europe		

Example tree using reals



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

Random forests: "mixtures of decision trees" is extremely popular, very useful

Acknowledgements

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

Boosting

Machine Learning – CSE446
Carlos Guestrin
University of Washington

April 22, 2013

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 too badly
- **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…**

Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!

$h_t : X \rightarrow Y : \{-1, +1\}$

α_t is strength of vote

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

weak classifier

weight of vote

- **But how do you ???**
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers? α_t

Boosting [Schapire, 1989]

decision stump, DT, LR, Naive Bayes ... your choice

$h_t : X \rightarrow \{-1, +1\}$

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

$h_t(x) \rightarrow \{-1, +1\}$, $\forall \{-1, +1\}$

$y^j h_t(x^j) > 0 \Rightarrow$ correct classification

$y^j h_t(x^j) < 0 \Rightarrow$ incorrect classification

- On each iteration t:
  - weight each training example by how incorrectly it was classified so far
  - Learn a hypothesis – h_t ← focus on "difficult parts of the space" ⇒ increase weight
  - A strength for this hypothesis – α_t
- Final classifier:

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

- **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(j) – weight of *j*th training example ($\mathbf{x}^j$,y^j)
  - Interpretations:
    - *j*th training example counts as D(j) examples
    - If I were to "resample" data, I would get more samples of "heavier" data points
- **Now, in all calculations, whenever used, *j*th training example counts as D(j) "examples"**