Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Decision Trees

Andrew W. Moore

Associate Professor

School of Computer Science

Carnegie Mellon University

www.cs.cmu.edu/~awm

awm@cs.cmu.edu

412-268-7599

 July 30, 2001

Here is a dataset

age	employme	education	edur	marital	...	job	relation	race	gender	hour	country	wealth
					...							
39	State_gov	Bachelors	13	Never_mar	...	Adm_cleric	Not_in_far	White	Male	40	United_Sta	poor
51	Self_emp_	Bachelors	13	Married	...	Exec_man	Husband	White	Male	13	United_Sta	poor
39	Private	HS_grad	9	Divorced	...	Handlers_c	Not_in_far	White	Male	40	United_Sta	poor
54	Private	11th	7	Married	...	Handlers_c	Husband	Black	Male	40	United_Sta	poor
28	Private	Bachelors	13	Married	...	Prof_speci	Wife	Black	Female	40	Cuba	poor
38	Private	Masters	14	Married	...	Exec_man	Wife	White	Female	40	United_Sta	poor
50	Private	9th	5	Married_sp	...	Other_serv	Not_in_far	Black	Female	16	Jamaica	poor
52	Self_emp_	HS_grad	9	Married	...	Exec_man	Husband	White	Male	45	United_Sta	rich
31	Private	Masters	14	Never_mar	...	Prof_speci	Not_in_far	White	Female	50	United_Sta	rich
42	Private	Bachelors	13	Married	...	Exec_man	Husband	White	Male	40	United_Sta	rich
37	Private	Some_coll	10	Married	...	Exec_man	Husband	Black	Male	80	United_Sta	rich
30	State_gov	Bachelors	13	Married	...	Prof_speci	Husband	Asian	Male	40	India	rich
24	Private	Bachelors	13	Never_mar	...	Adm_cleric	Own_child	White	Female	30	United_Sta	poor
33	Private	Assoc_acc	12	Never_mar	...	Sales	Not_in_far	Black	Male	50	United_Sta	poor
41	Private	Assoc_voc	11	Married	...	Craft_repai	Husband	Asian	Male	40	*MissingVa	rich
34	Private	7th_8th	4	Married	...	Transport_	Husband	Amer_India	Male	45	Mexico	poor
26	Self_emp_	HS_grad	9	Never_mar	...	Farming_fis	Own_child	White	Male	35	United_Sta	poor
33	Private	HS_grad	9	Never_mar	...	Machine_c	Unmarried	White	Male	40	United_Sta	poor
38	Private	11th	7	Married	...	Sales	Husband	White	Male	50	United_Sta	poor
44	Self_emp_	Masters	14	Divorced	...	Exec_man	Unmarried	White	Female	45	United_Sta	rich
41	Private	Doctorate	16	Married	...	Prof_speci	Husband	White	Male	60	United_Sta	rich
:	:	:	:	:	:	:	:	:	:	:	:	:

48,000 records, 16 attributes [Kohavi 1995]

Classification

- A Major Data Mining Operation
- Give one attribute (e.g wealth), try to predict the value of new people's wealths by means of some of the other available attributes.
- Applies to categorical outputs

  - Categorical attribute: an attribute which takes on two or more discrete values. Also known as a symbolic attribute.
  - Real attribute: a column of real numbers

Today's lecture

- Information Gain for measuring association between inputs and outputs
- Learning a decision tree classifier from data

About this dataset

- It is a tiny subset of the 1990 US Census.
- It is publicly available online from the UCI Machine Learning Datasets repository

Used Attributes

age	edunum	race	hours_worked
employment	marital	gender	country
taxweighting	job	capitalgain	wealth
education	relation	capitalloss	agegroup

This color = Real-valued   This color = Symbol-valued

Successfully loaded a new dataset from the file \adult.fds. It has 16 attributes and 48842 records.

What can you do with a dataset?

- Well, you can look at histograms…

Value	Frequency
Female	16192
Male	32650

Gender

Value	Frequency
Divorced	6633
Married_AF_spouse	37
Married	22379
Married_spouse_absent	628
Never_married	16117
Separated	1530
Widowed	1518

Marital Status

Contingency Tables

- A better name for a histogram:

  A One-dimensional Contingency Table

- Recipe for making a k-dimensional contingency table:
  1. Pick *k* attributes from your dataset. Call them $a_1, a_2, \ldots a_k$.
  2. For every possible combination of values, $a_1{=}x_1, a_2{=}x_2, \ldots a_k{=}x_k$, record how frequently that combination occurs

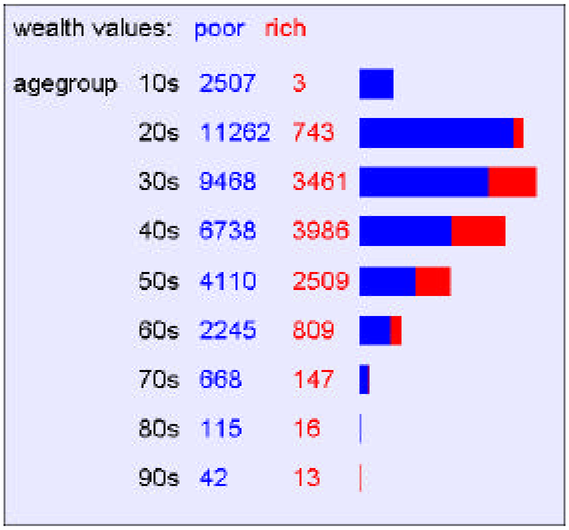
Fun fact: A database person would call this a "k-dimensional datacube"

A 2-d Contingency Table

wealth values:		poor	rich
agegroup	10s	2507	3
	20s	11262	743
	30s	9468	3461
	40s	6738	3986
	50s	4110	2509
	60s	2245	809
	70s	668	147
	80s	115	16
	90s	42	13

- For each pair of values for attributes (agegroup,wealth) we can see how many records match.

A 2-d Contingency Table

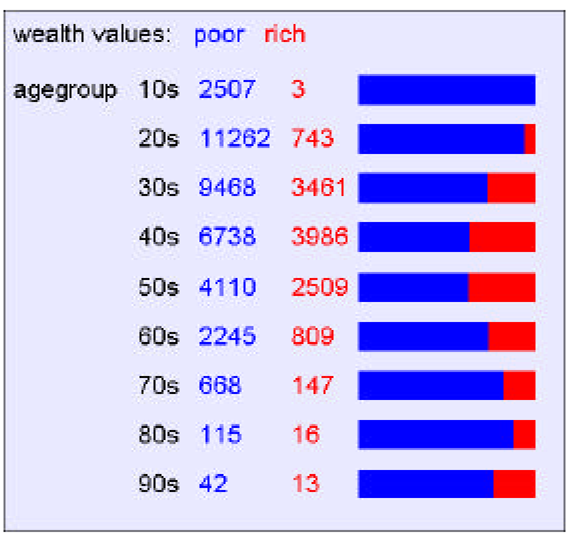


wealth values: poor rich

		poor	rich
agegroup	10s	2507	3
	20s	11262	743
	30s	9468	3461
	40s	6738	3986
	50s	4110	2509
	60s	2245	809
	70s	668	147
	80s	115	16
	90s	42	13

- Easier to appreciate graphically

A 2-d Contingency Table



wealth values: poor rich

		poor	rich
agegroup	10s	2507	3
	20s	11262	743
	30s	9468	3461
	40s	6738	3986
	50s	4110	2509
	60s	2245	809
	70s	668	147
	80s	115	16
	90s	42	13

- Easier to see “interesting” things if we stretch out the histogram bars

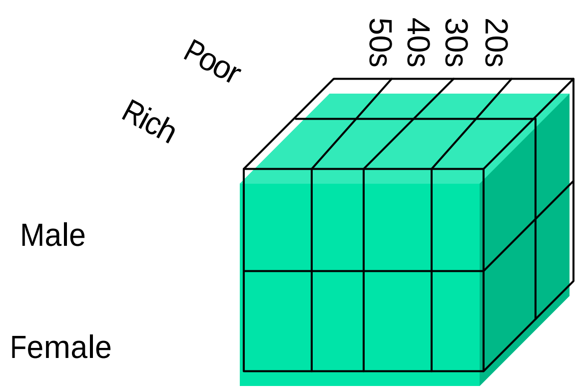
A bigger 2-d contingency table

job values: *MissingValue* Adm_clerical Armed_Forces Craft_repair Exec_managerial Farming_fishing Handlers_cleaners Machine_op_inspct Other_service Priv_house_serv Prof_specialty Protective_serv Sales Tech_support Transport_moving

marital	Divorced	270	1192	0	679	890	90	197	434	762	46	795	121	664	239	254
	Married_AF_spouse	5	6	0	4	3	1	1	1	5	0	4	1	5	0	1
	Married	928	1495	7	3818	3600	869	724	1469	1088	27	3182	583	2491	609	1489
	Married_spouse_absent	45	84	0	77	52	35	32	37	92	9	64	7	55	9	30
	Never_married	1242	2360	8	1301	1260	434	1029	872	2442	99	1849	237	1992	506	486
	Separated	97	224	0	160	126	23	63	123	275	21	145	23	146	48	56
	Widowed	222	250	0	73	155	38	26	86	259	40	133	11	151	35	39

3-d contingency tables

- These are harder to look at!



On-Line Analytical Processing (OLAP)

- Software packages and database add-ons to do this are known as OLAP tools
- They usually include point and click navigation to view slices and aggregates of contingency tables
- They usually include nice histogram visualization

Time to stop and think

- Why would people want to look at contingency tables?

Let's continue to think

- With 16 attributes, how many 1-d contingency tables are there?
- How many 2-d contingency tables?
- How many 3-d tables?
- With 100 attributes how many 3-d tables are there?

Let's continue to think

- With 16 attributes, how many 1-d contingency tables are there? 16
- How many 2-d contingency tables? 16-choose-2 = 16 * 15 / 2 = 120
- How many 3-d tables? 560
- With 100 attributes how many 3-d tables are there? 161,700

Manually looking at contingency tables

- Looking at one contingency table: *can be as much fun as reading an interesting book*
- Looking at ten tables: *as much fun as watching CNN*
- Looking at 100 tables: *as much fun as watching an infomercial*
- Looking at 100,000 tables: *as much fun as a three-week November vacation in Duluth with a dying weasel.*

Data Mining

- Data Mining is all about automating the process of searching for patterns in the data.

Which patterns are interesting?

Which might be mere illusions?

And how can they be exploited?

Data Mining

- Data Mining is all about automating the process of searching for patterns in the data.

Which patterns are interesting?

That's what we'll look at right now.

And the answer will turn out to be the engine that drives decision tree learning.

Which might be mere illusions?

And how can they be exploited?

Deciding whether a pattern is interesting

- We will use information theory
- A very large topic, originally used for compressing signals
- But more recently used for data mining…

Deciding whether a pattern is interesting

- We will use information theory
- A very large topic, originally used for compressing signals
- But more recently used for data mining…

(The topic of Information Gain will now be discussed, but you will find it in a separate Andrew Handout)

Searching for High Info Gains

- Given something (e.g. wealth) you are trying to predict, it is easy to ask the computer to find which attribute has highest information gain for it.

wealth values: poor rich

		poor	rich	
relation	Husband	10870	8846	H(wealth | relation = Husband) = 0.992385
	Not_in_family	11307	1276	H(wealth | relation = Not_in_family) = 0.473439
	Other_relative	1454	52	H(wealth | relation = Other_relative) = 0.216617
	Own_child	7470	111	H(wealth | relation = Own_child) = 0.110192
	Unmarried	4816	309	H(wealth | relation = Unmarried) = 0.328606
	Wife	1238	1093	H(wealth | relation = Wife) = 0.997207

H(wealth) = 0.793844 H(wealth|relation) = 0.628421

IG(wealth|relation) = 0.165423

Learning Decision Trees

- A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output.
- To decide which attribute should be tested first, simply find the one with the highest information gain.
- Then recurse…

A small dataset: Miles Per Gallon

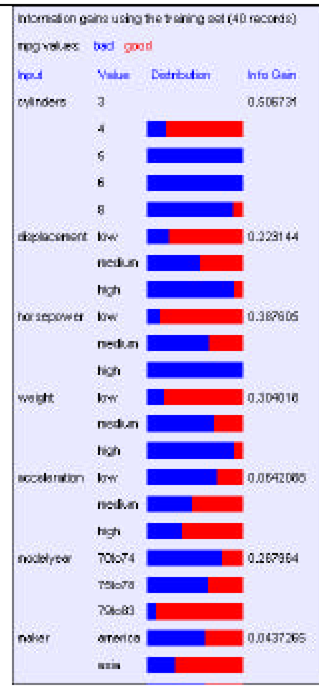
40
Records

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

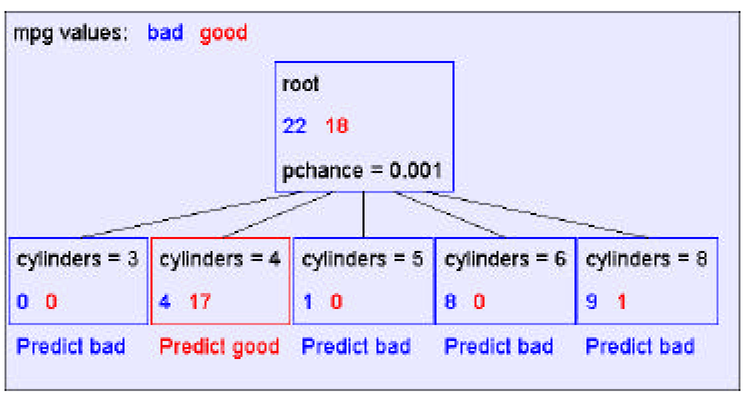
From the UCI repository (thanks to Ross Quinlan)

Suppose we want to predict MPG.

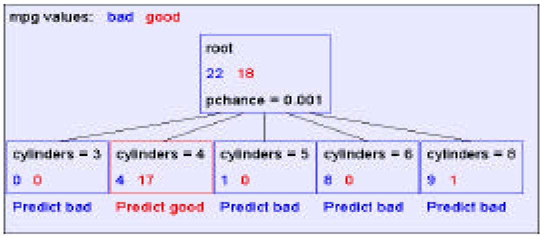
Look at all the information gains...



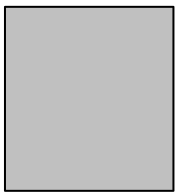
A Decision Stump



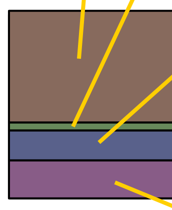
Recursion Step



Take the Original Dataset..



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



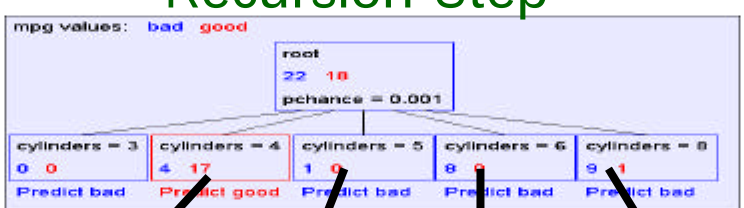
Records in which cylinders = 4

Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

Recursion Step



Build tree from These records..

Build tree from These records..

Build tree from These records..

Build tree from These records..

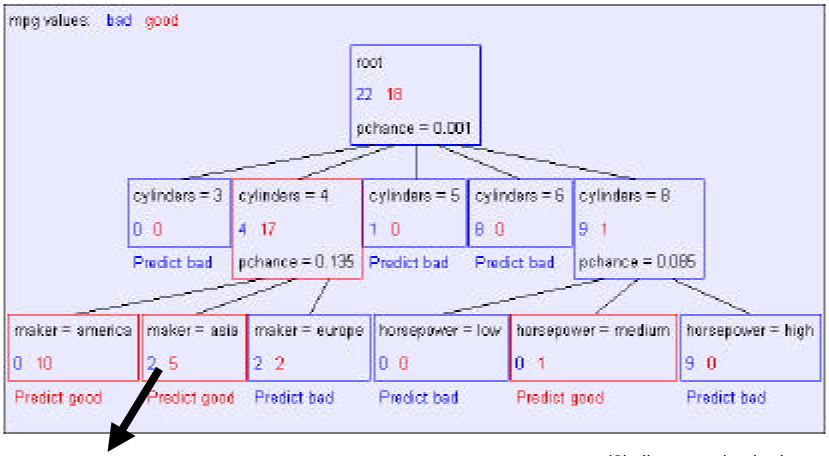
Records in which cylinders = 4

Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

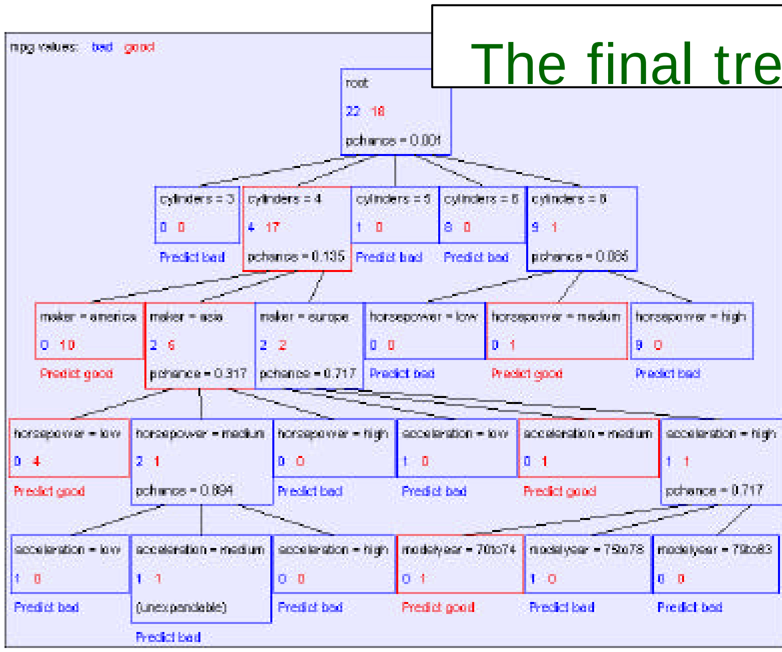
Second level of tree

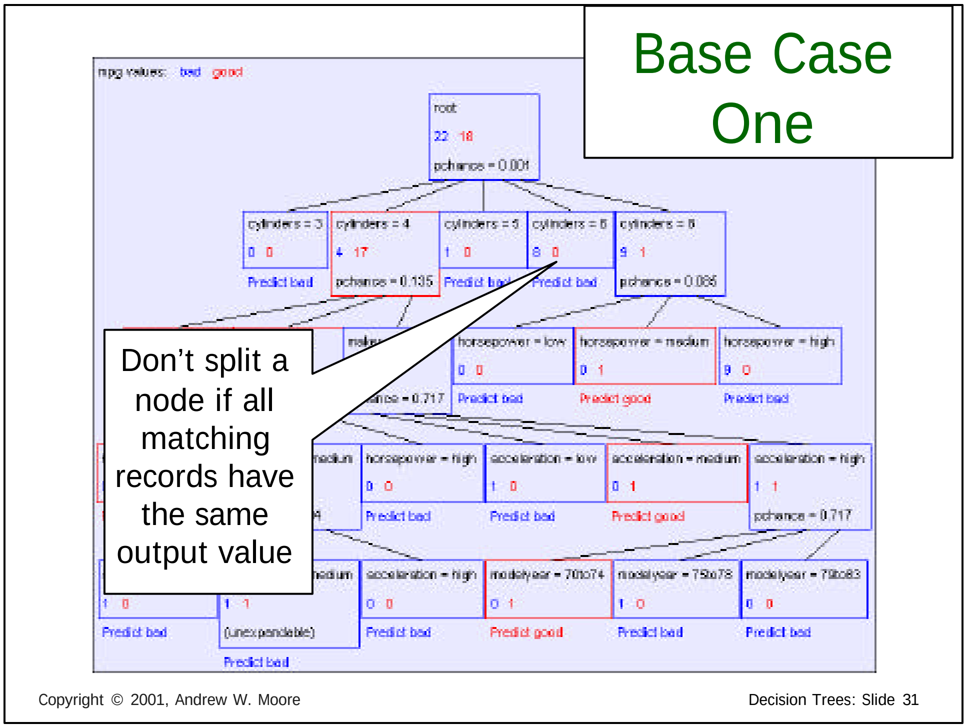


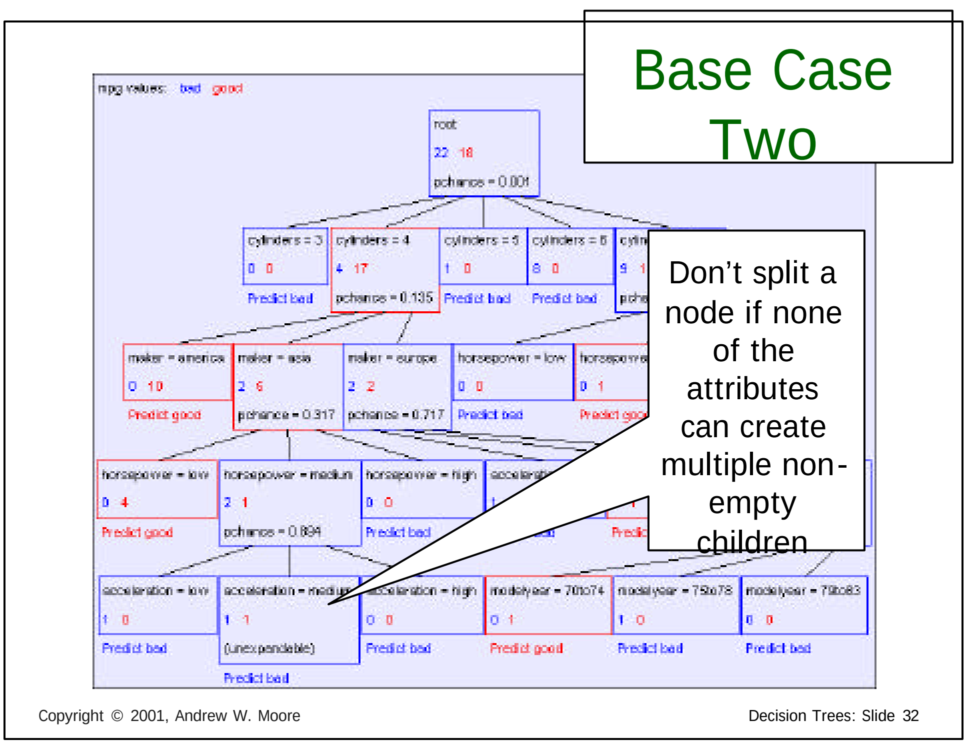
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

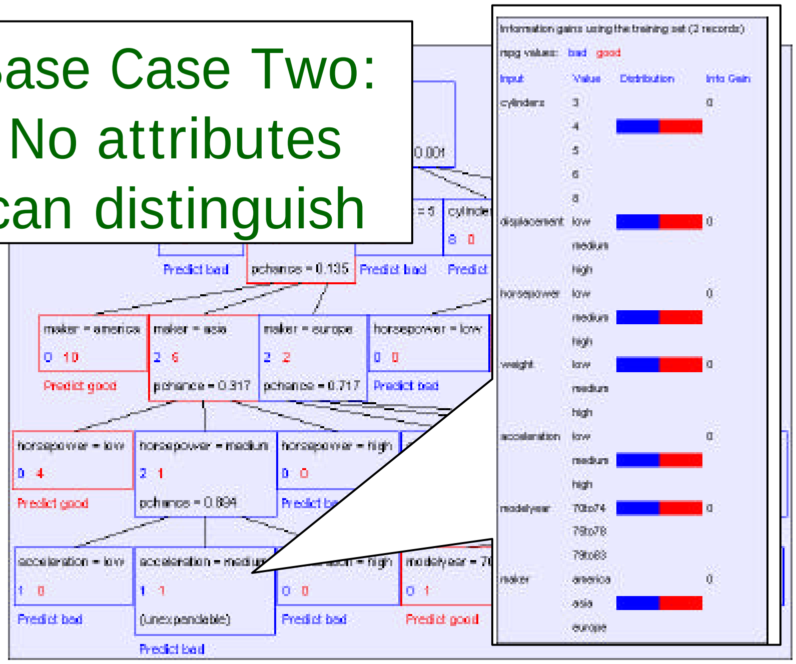
The final tree




Base Case One
Don't split a node if all matching records have the same output value
Copyright © 2001, Andrew W. Moore
Decision Trees: Slide 31


Base Case Two
Don't split a node if none of the attributes can create multiple non-empty children
Copyright © 2001, Andrew W. Moore
Decision Trees: Slide 32

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

Proposed Base Case 3:

If all attributes have zero information gain then don’t recurse

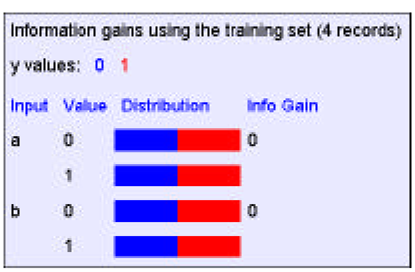
- *Is this a good idea?*

The problem with Base Case 3

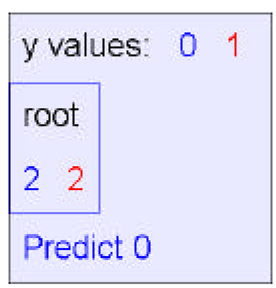
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



The resulting decision tree:

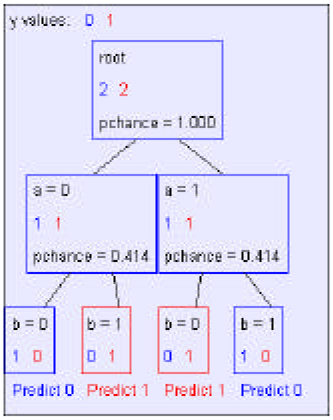


If we omit Base Case 3:

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

y = a XOR b

The resulting decision tree:



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.

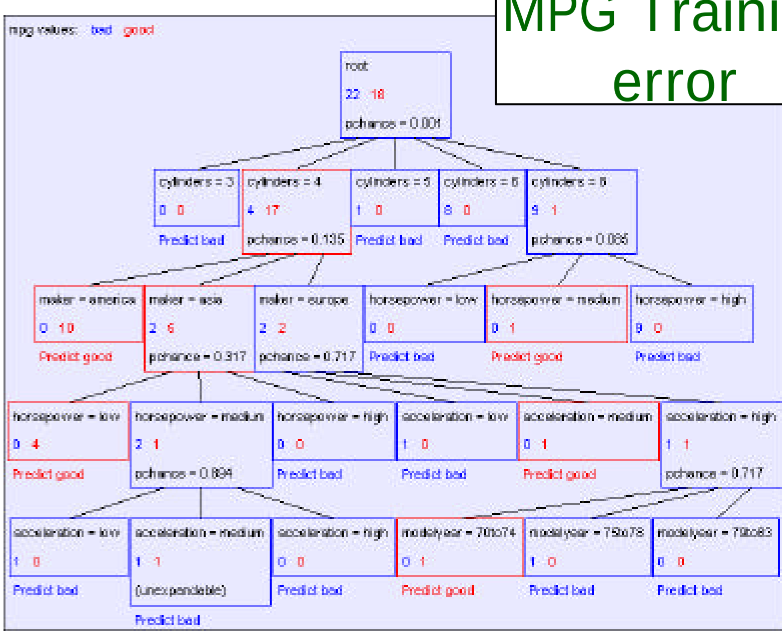
Training Set Error

- For each record, follow the decision tree to see what it would predict

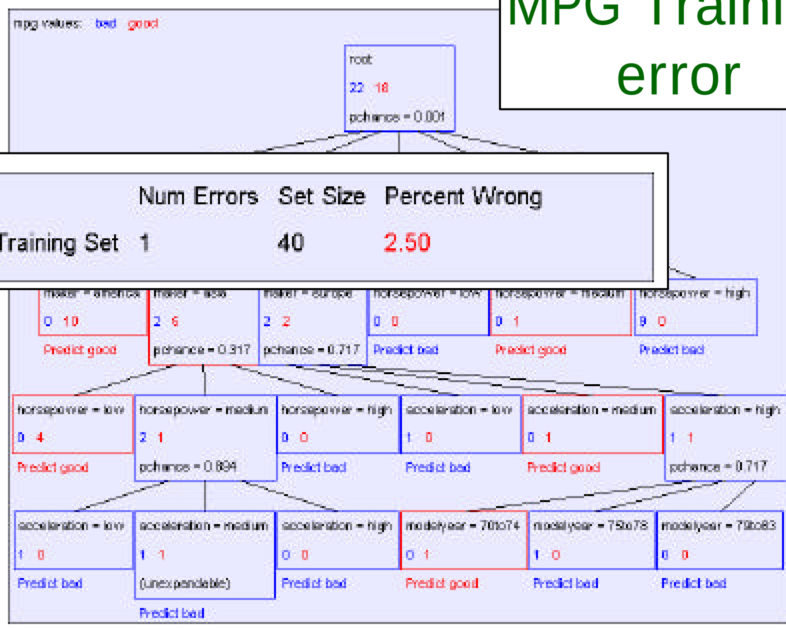
  For what number of records does the decision tree's prediction disagree with the true value in the database?

- This quantity is called the *training set error*. The smaller the better.

MPG Training error

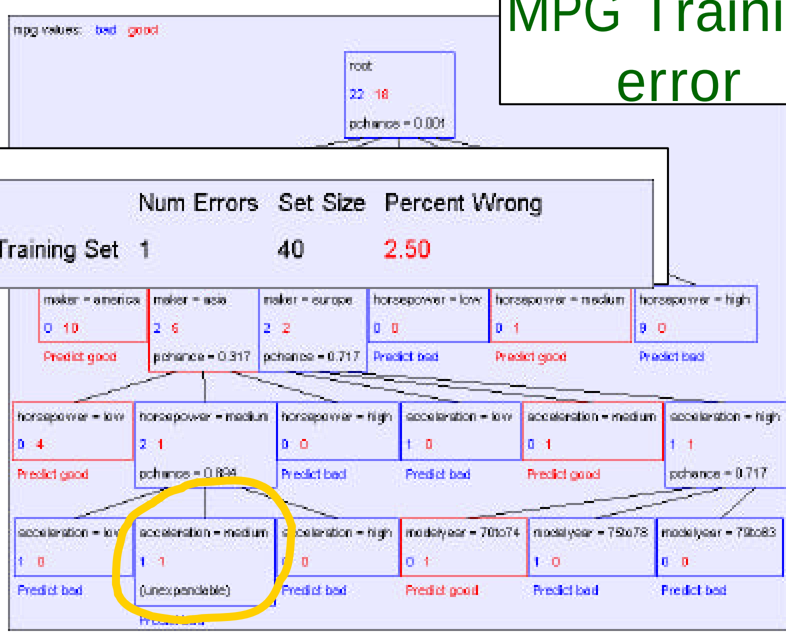


MPG Training error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50

MPG Training error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50

Stop and reflect: Why are we doing this learning anyway?

- It is not usually in order to predict the training data's output on data we have already seen.

Stop and reflect: Why are we doing this learning anyway?

- It is not usually in order to predict the training data's output on data we have already seen.
- It is more commonly in order to predict the output value for future data we have not yet seen.

Stop and reflect: Why are we doing this learning anyway?

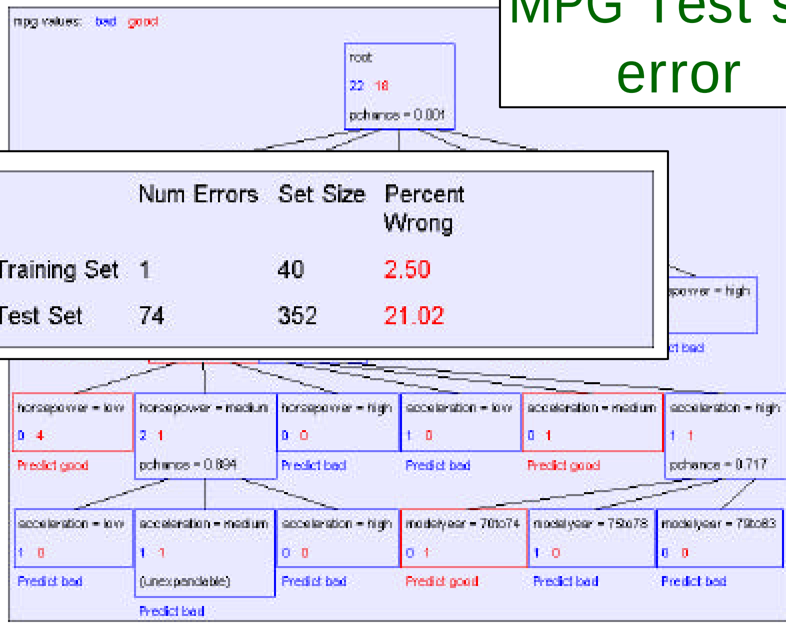
- It is not usually in order to predict the training data's output on data we have already seen.
- It is more commonly in order to predict the output value for future data we have not yet seen.

Warning: A common data mining misperception is that the above two bullets are the only possible reasons for learning. There are at least a dozen others.

Test Set Error

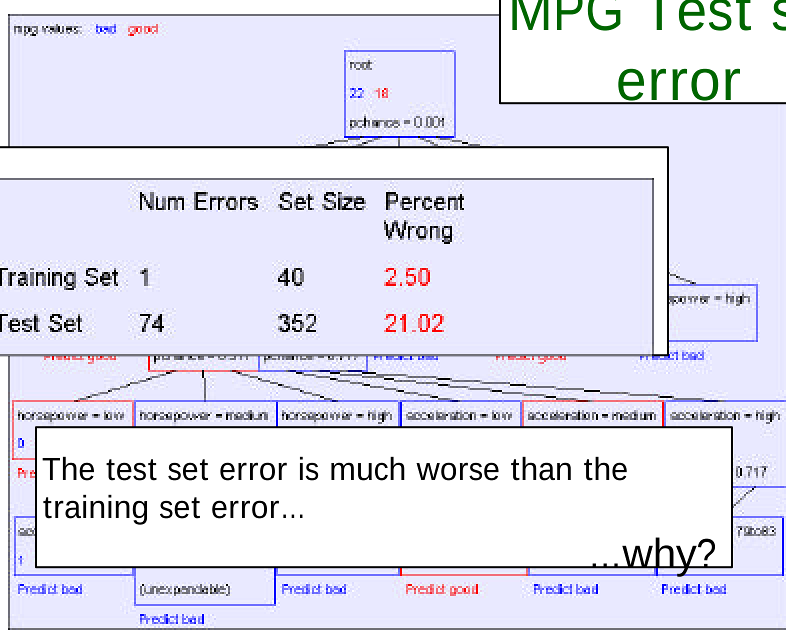
- Suppose we are forward thinking.
- We hide some data away when we learn the decision tree.
- But once learned, we see how well the tree predicts that data.
- This is a good simulation of what happens when we try to predict future data.
- And it is called Test Set Error.

MPG Test set error



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

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?

An artificial example

- We'll create a training dataset

Five inputs, all bits, are generated in all 32 possible combinations

Output y = copy of e, Except a random 25% of the records have y set to the opposite of e

32 records

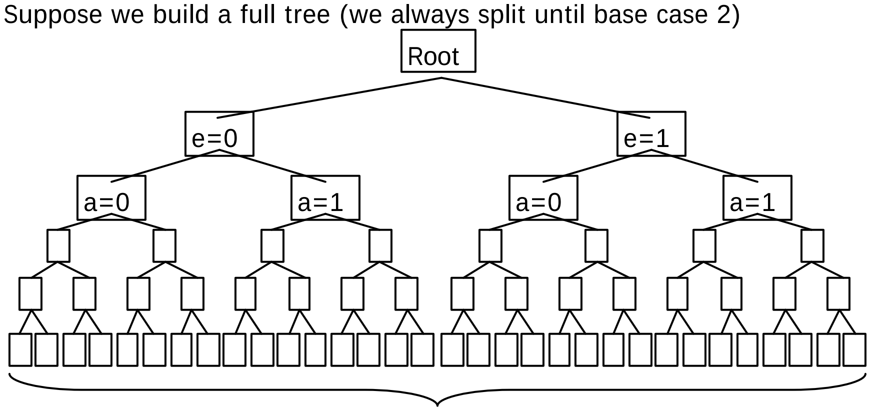
a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

In our artificial example

- Suppose someone generates a test set according to the same method.
- The test set is identical, except that some of the y's will be different.
- Some y's that were corrupted in the training set will be uncorrupted in the testing set.
- Some y's that were uncorrupted in the training set will be corrupted in the test set.

Building a tree with the artificial training set

- Suppose we build a full tree (we always split until base case 2)



25% of these leaf node labels will be corrupted

Training set error for our artificial tree

All the leaf nodes contain exactly one record and so…

- We would have a training set error of zero

Testing the tree with the test set

	1/4 of the tree nodes are corrupted	3/4 are fine
1/4 of the test set records are corrupted	1/16 of the test set will be correctly predicted for the wrong reasons	3/16 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	3/16 of the test predictions will be wrong because the tree node is corrupted	9/16 of the test predictions will be fine

In total, we expect to be wrong on 3/8 of the test set predictions

What's this example shown us?

- This explains the discrepancy between training and test set error
- But more importantly… …it indicates there's something we should do about it if we want to predict well on future data.

Suppose we had less data

- Let’s not look at the irrelevant bits

Output y = copy of e, except a random 25% of the records have y set to the opposite of e

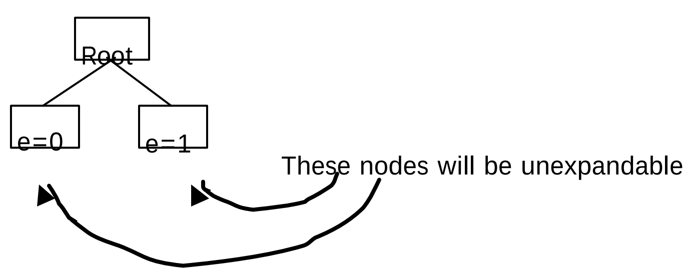
These bits are hidden

32 records

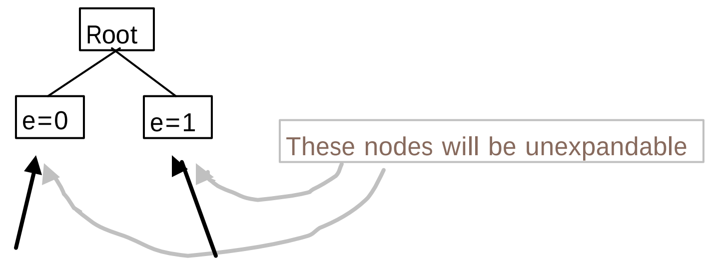
a	b	c	d	e	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

What decision tree would we learn now?

Without access to the irrelevant bits…



Without access to the irrelevant bits...



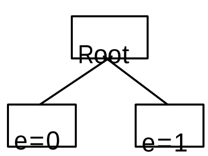
In about 12 of the 16 records in this node the output will be 0

So this will almost certainly predict 0

In about 12 of the 16 records in this node the output will be 1

So this will almost certainly predict 1

Without access to the irrelevant bits...



	almost certainly none of the tree nodes are corrupted	almost certainly all are fine
1/4 of the test set records are corrupted	n/a	1/4 of the test set will be wrongly predicted because the test record is corrupted
3/4 are fine	n/a	3/4 of the test predictions will be fine

In total, we expect to be wrong on only 1/4 of the test set predictions

Overfitting

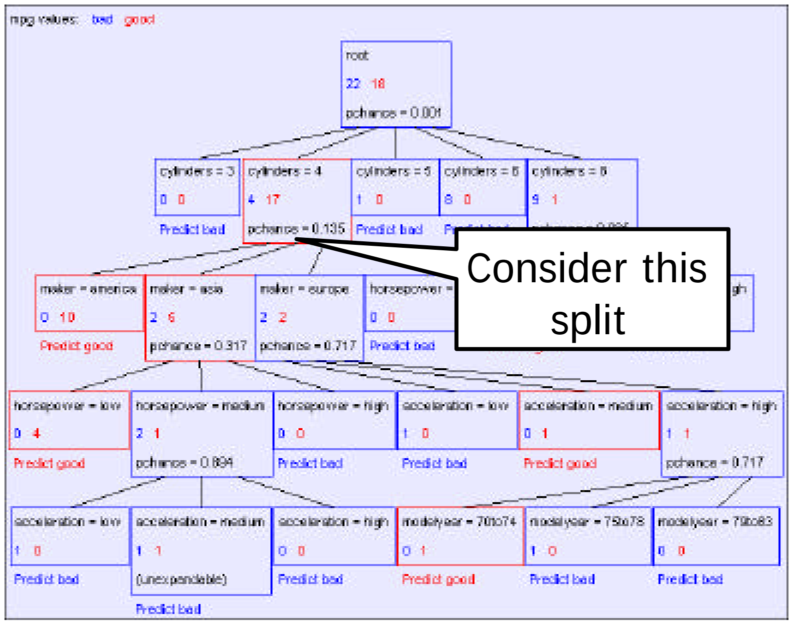
- Definition: If your machine learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is overfitting.
- Fact (theoretical and empirical): If your machine learning algorithm is overfitting then it may perform less well on test set data.

Avoiding overfitting

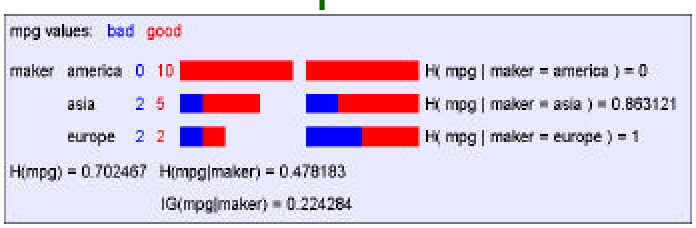
- Usually we do not know in advance which are the irrelevant variables
- ...and it may depend on the context

  For example, if y = a AND b then b is an irrelevant variable only in the portion of the tree in which a=0

But we can use simple statistics to warn us that we might be overfitting.

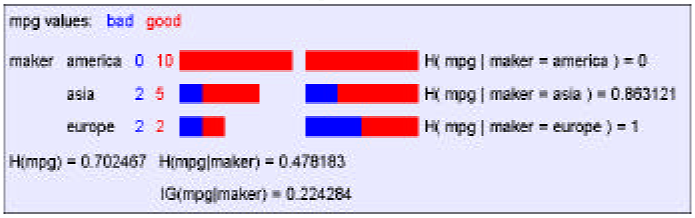


A chi-squared test



- 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-squared 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-squared test, the answer is 13.5%.

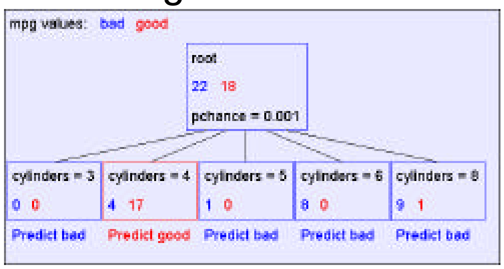
Using Chi-squared to avoid overfitting

- 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

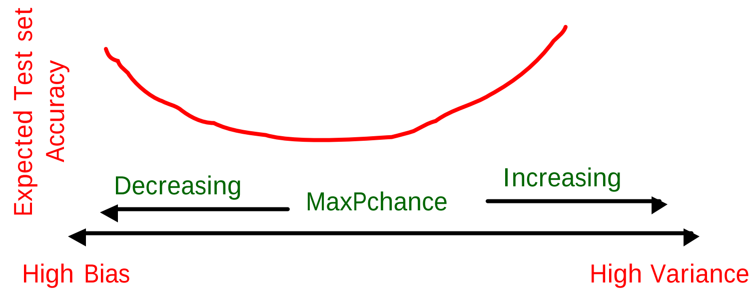
	Num Errors	Set Size	Percent Wrong
Training Set	5	40	12.50
Test Set	56	352	15.91

MaxPchance

- Good news: The decision tree can automatically adjust its pruning decisions according to the amount of apparent noise and data.
- Bad news: The user must come up with a good value of MaxPchance. (Note, Andrew usually uses 0.05, which is his favorite value for any magic parameter).
- Good news: But with extra work, the best MaxPchance value can be estimated automatically by a technique called cross-validation.

MaxPchance

- Technical note (dealt with in other lectures): MaxPchance is a regularization parameter.



The simplest tree

- Note that this pruning is heuristically trying to find

The simplest tree structure for which all within-leaf-node disagreements can be explained by chance

- This is not the same as saying “the simplest classification scheme for which…”
- Decision trees are biased to prefer classifiers that can be expressed as trees.

Expressiveness of Decision Trees

- Assume all inputs are Boolean and all outputs are Boolean.
- What is the class of Boolean functions that are possible to represent by decision trees?
- Answer: All Boolean functions.

Simple proof:

1. Take any Boolean function
2. Convert it into a truth table
3. Construct a decision tree in which each row of the truth table corresponds to one path through the decision tree.

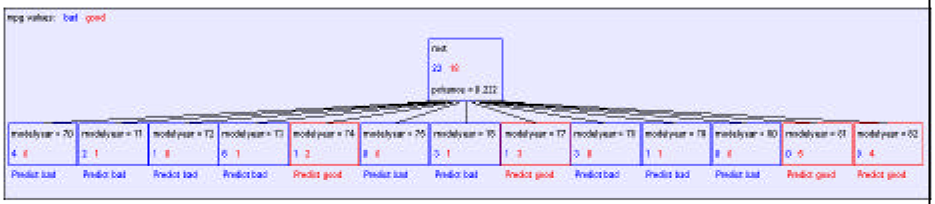
Real-Valued inputs

- 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	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

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 over fit

Note pchance is 0.222 in the above...if MaxPchance was 0.05 that would end up pruning away to a single root node.

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

Computational Issues

- You can compute IG*(Y|X) in time

$$R \log R + 2 R n_y$$

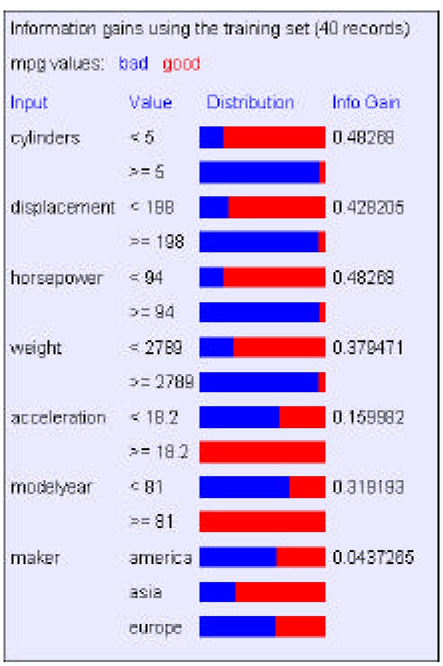
- Where

R is the number of records in the node under consideration

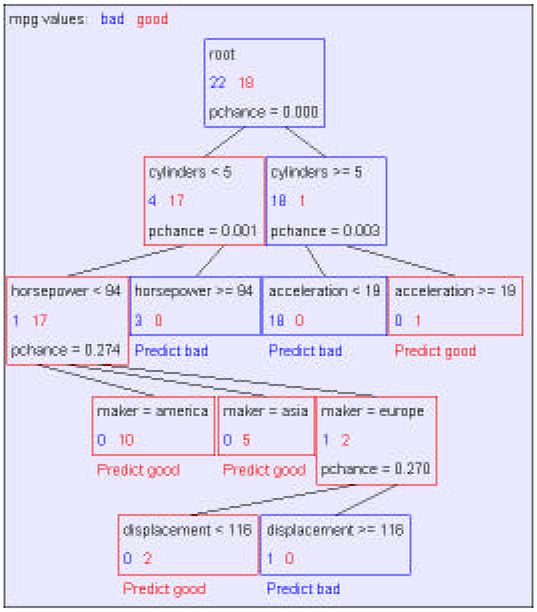
n_y is the arity (number of distinct values of) Y

How?

Sort records according to increasing values of X. Then create a 2xn_y contingency table corresponding to computation of IG(Y|X:x_{min}). Then iterate through the records, testing for each threshold between adjacent values of X, incrementally updating the contingency table as you go. For a minor additional speedup, only test between values of Y that differ.

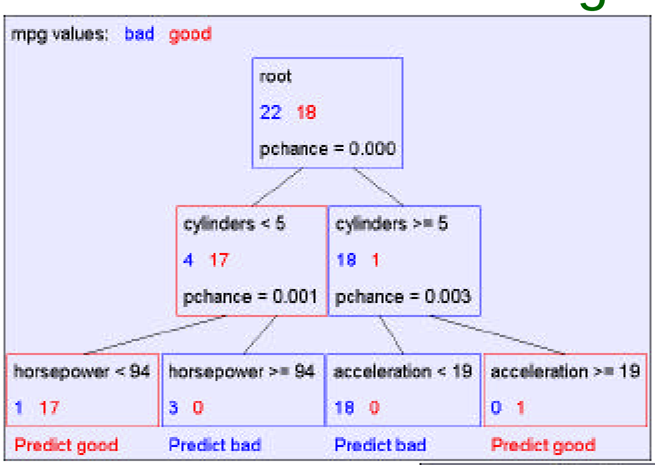


Example with MPG



Unpruned tree using reals

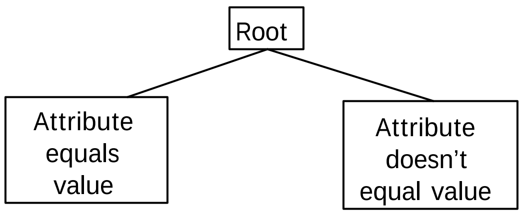
Pruned tree using reals



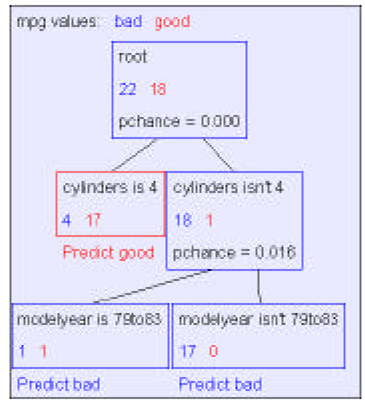
	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	53	352	15.06

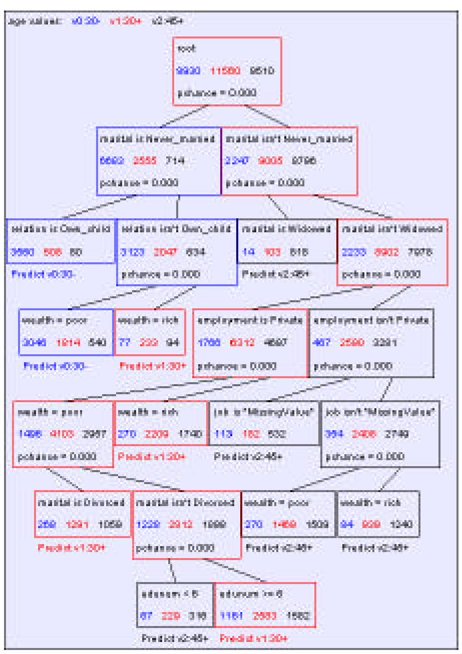
Binary categorical splits

- One of Andrew's favorite tricks
- Allow splits of the following form

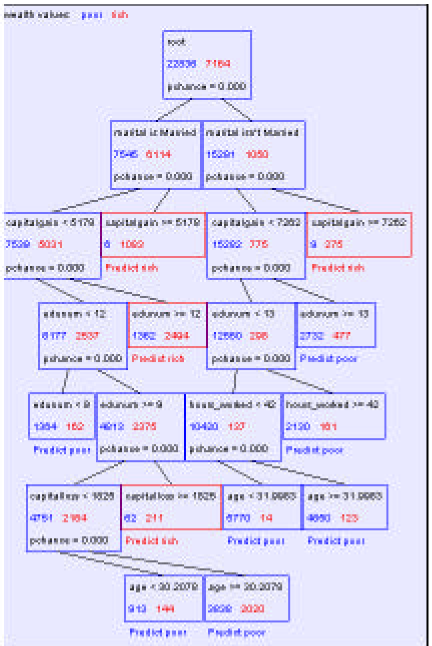


Example:



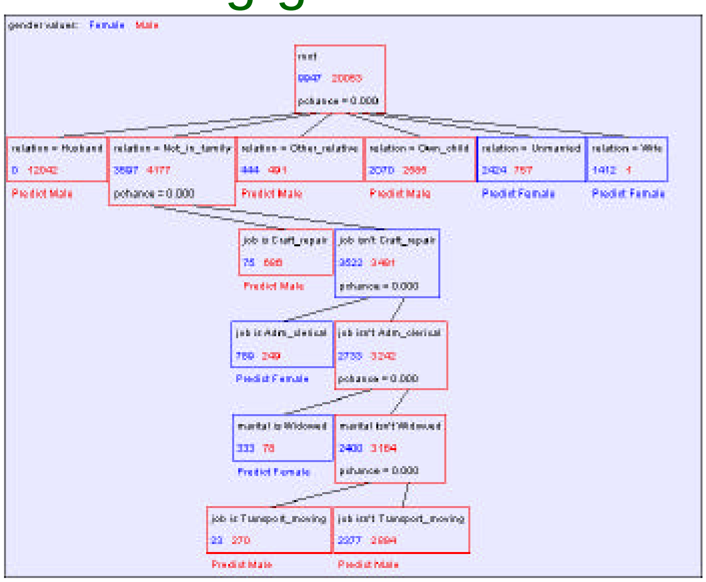


Predicting age from census



Predicting wealth from census

Predicting gender from census



Conclusions

- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- It's possible to get in trouble with overfitting
- They do classification: predict a categorical output from categorical and/or real inputs

What you should know

- What's a contingency table?
- What's information gain, and why we use it
- The recursive algorithm for building an unpruned decision tree
- What are training and test set errors
- Why test set errors can be bigger than training set
- Why pruning can reduce test set error
- How to exploit real-valued inputs

What we haven't discussed

- It's easy to have real-valued outputs too---these are called Regression Trees*
- Bayesian Decision Trees can take a different approach to preventing overfitting
- Computational complexity (straightforward and cheap) *
- Alternatives to Information Gain for splitting nodes
- How to choose MaxPchance automatically *
- The details of Chi-Squared testing *
- Boosting---a simple way to improve accuracy *

* = discussed in other Andrew lectures

For more information

- Two nice books
  - L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.
  - C4.5 : Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan
- Dozens of nice papers, including
  - Learning Classification Trees, Wray Buntine, Statistics and Computation (1992), Vol 2, pages 63-73
  - Kearns and Mansour, On the Boosting Ability of Top-Down Decision Tree Learning Algorithms, STOC: ACM Symposium on Theory of Computing, 1996"
- Dozens of software implementations available on the web for free and commercially for prices ranging between $50 - $300,000

Discussion

- Instead of using information gain, why not choose the splitting attribute to be the one with the highest prediction accuracy?
- Instead of greedily, heuristically, building the tree, why not do a combinatorial search for the optimal tree?
- If you build a decision tree to predict wealth, and marital status, age and gender are chosen as attributes near the top of the tree, is it reasonable to conclude that those three inputs are the major causes of wealth?
- ..would it be reasonable to assume that attributes not mentioned in the tree are not causes of wealth?
- ..would it be reasonable to assume that attributes not mentioned in the tree are not correlated with wealth?