Non-Linear Classifiers 1: Decision Trees

Pattern Recognition and Image Analysis

Dr. Manal Helal – Fall 2014

Lecture 9

Overview

- Decision Trees (This Lecture – Week 11)
- Next Week – Midterm Exam (Week 12)
- Polynomial Classifier, RBF (Week 13)
- Nonlinear SVM (Week 13)
- Multi Layer Neural Networks (Week 14)
  - Two Layer Perceptron
  - Three Layer Perceptron
- Project Presentations (Week 15)
- Final Exam (Week 16)

Linearly Separable Data

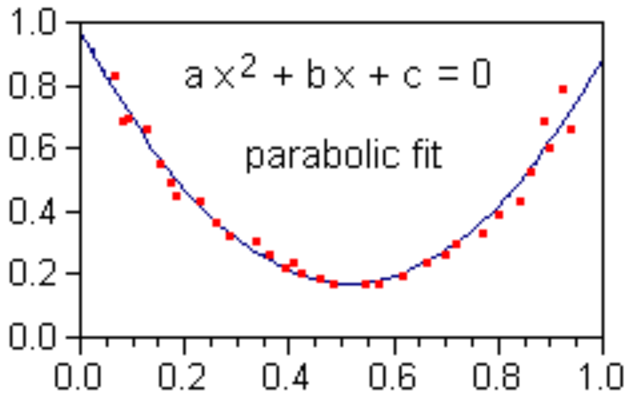
- A dataset is **linearly separable** iff ∃ a **separating hyperplane w**, such that:
  - $w_0 + \sum_i w_i x_i > 0$; if $\mathbf{x}=\{x_1,\ldots,x_n\}$ is a positive example
  - $w_0 + \sum_i w_i x_i < 0$; if $\mathbf{x}=\{x_1,\ldots,x_n\}$ is a negative example

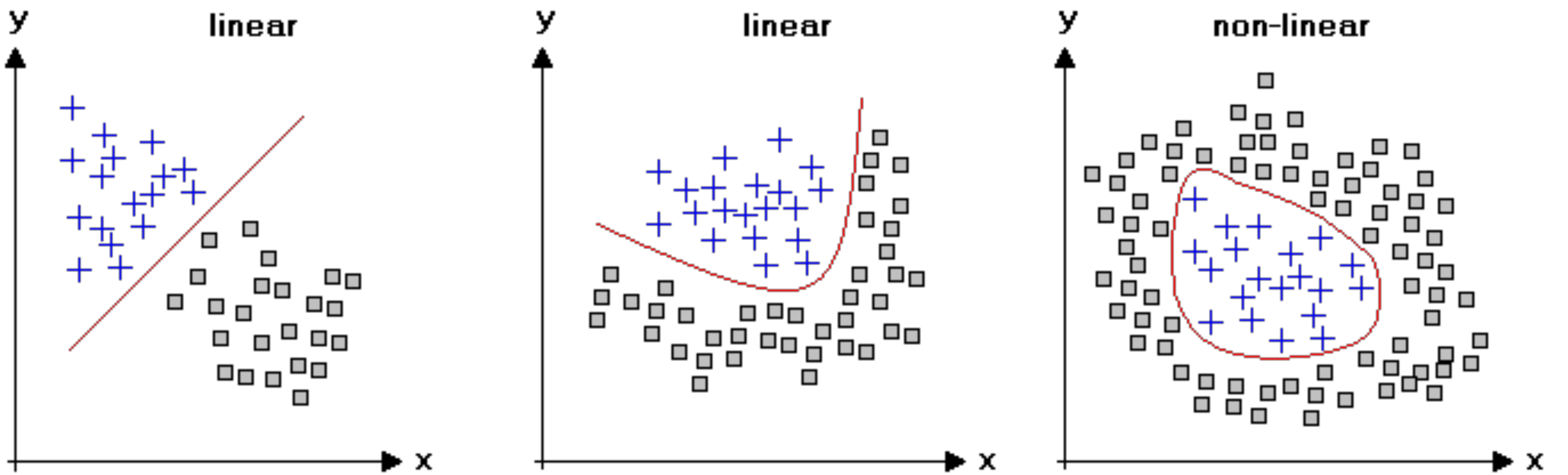
  - Typical linear features: $w_0 + \sum_i w_i x_i$

- 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 **w,** Data is linearly separable in higher dimensional spaces

non-linearly separable data

- Linear models are **linear in the parameters which have to be estimated**, but not ***necessarily*** in the independent variables.
- In the parabolic example, the parameters a, b, and c are linear.
- Multiple linear regression can be used to estimate the parameters of "curved" models.





Multiple Linear Regression

- Given

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n + \varepsilon$$

- Or

$$y = a_0 + \sum_{i=1}^{n} a_i x_i + \varepsilon$$

- Defining a hyper-plane in n dimensions, The parameter e defines the error, or the residual, with a mean of zero.

- MLR adjusts the parameters $a_1 \ldots a_n$, such that the sum of the squared errors is minimised to best fit the data.

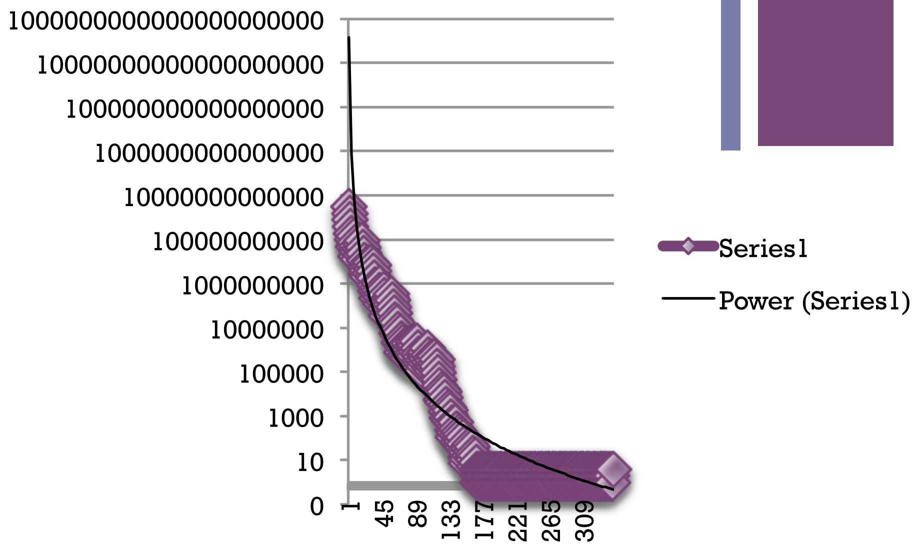
non-linearly separable data – non-linear classifier

- Choose a classifier $h_{\mathbf{w}}(\mathbf{x})$ that is non-linear in parameters **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)

1D Non-Linear Example

Starting from x = 998123456789, next x is computed using the non-linear mapping:

$$f(x) := \begin{cases} x/2 & \text{if } x \text{ is even} \\ 3x+1 & \text{if } x \text{ is odd} \end{cases}$$



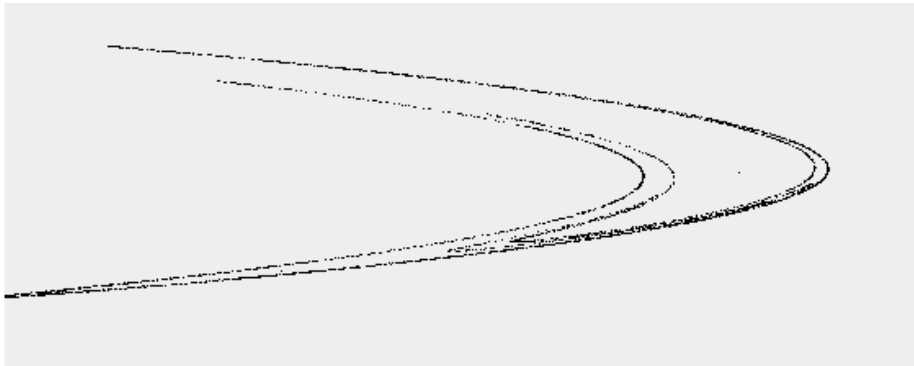
2D Non-Linear Example

The Henon map is the most studied two-dimensional map with chaotic behaviour.

f : R^2 —> R^2 which is

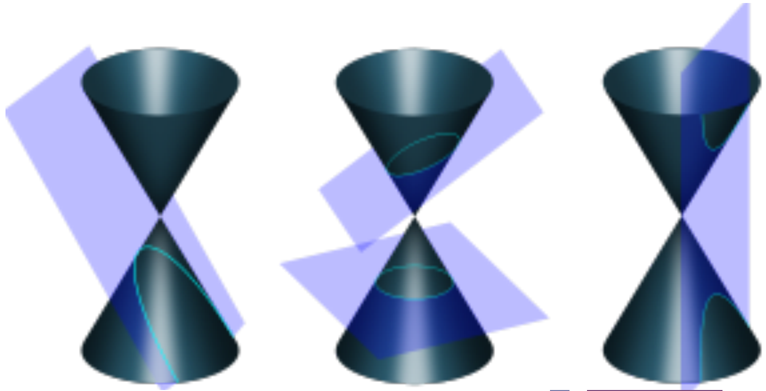
given by

$f(x, y) := (y + 1 - ax^2, bx)$



Conic Sections

Cutting Planes:



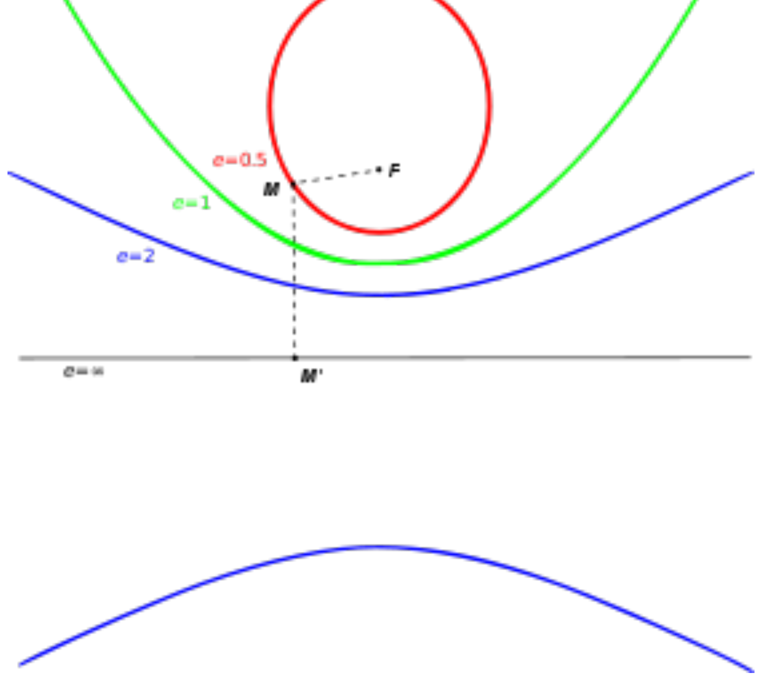
Ellipse: Closed curve.

Circle: closed and perpendicular to the symmetry axis

Parabola: parallel to exactly one generating line of the cone

Hyperbola: intersects both halves, producing two separate unbounded curves.

- F (the focus), L (the directrix Line) not containing F
- A nonnegative real number e (the eccentricity: a measure of how much the conic section deviates from being circular)
- The corresponding conic section consists of the locus of all points whose distance to F equals e times their distance to L.
  - For $e = 0$, we obtain a circle,
  - For $0 < e < 1$ we obtain an ellipse,
  - for $e = 1$ a parabola,
  - for $e > 1$ a hyperbola.

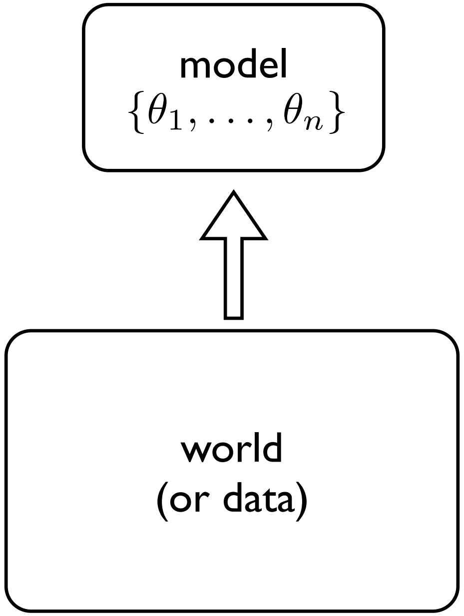


Learning and Decision Trees to learning

- What is learning?
  - more than just memorizing facts
  - learning the underlying *structure* of the problem or data
- A fundamental aspect of learning is *generalization:*
  - given a few examples, can you *generalize* to others?
- Learning is ubiquitous:
  - *medical diagnosis*: identify new disorders from observations
  - *loan applications*: predict risk of default
  - *prediction:* (climate, stocks, etc.) predict future from current and past data
  - *speech/object recognition*: from examples, generalize to others

Representation

- How do we model or represent the world?
- All learning requires some form of representation.
- Learning:
  - *adjust model parameters to match data*



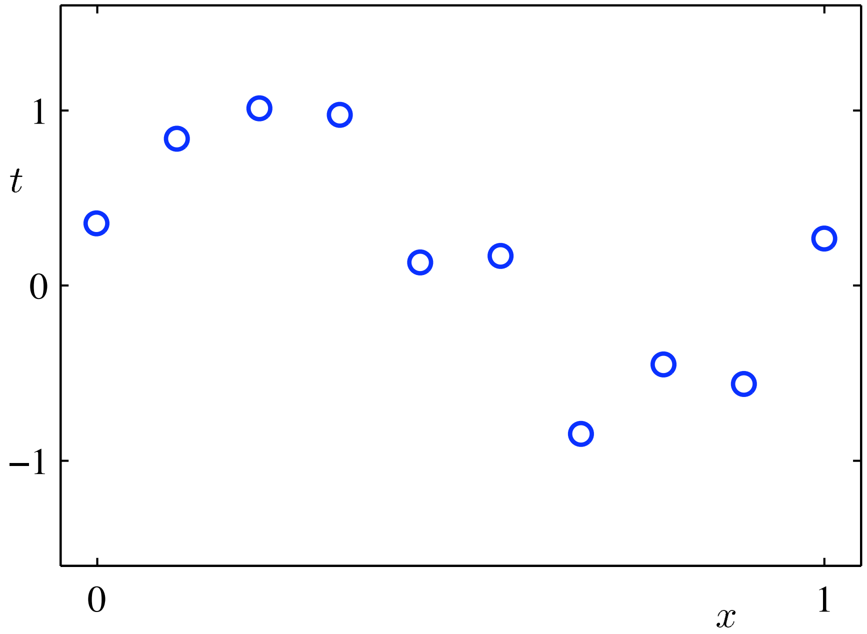
The complexity of learning

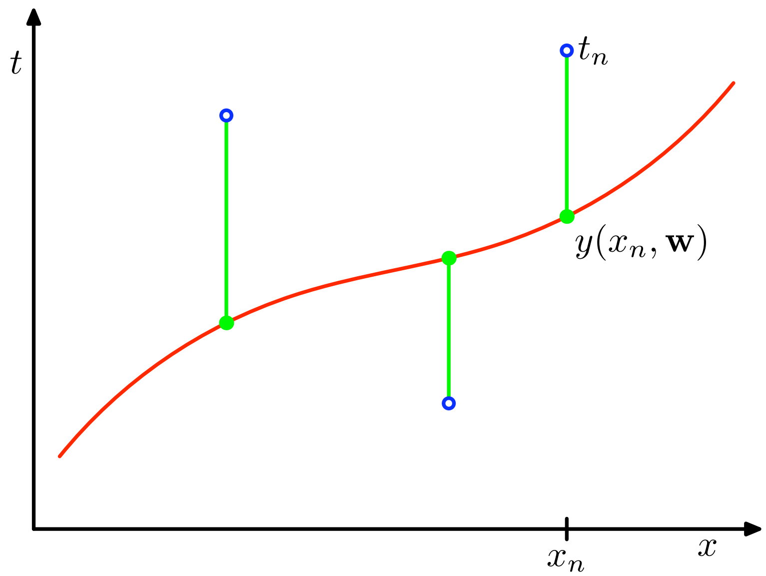
- Fundamental trade-off in learning:
  - *complexity of model vs. amount of data required to learn parameters*
- The more complex the model, the more it can describe, but the more data it requires to constrain the parameters.
- Consider a hypothesis space of N models:
  - How many bits would it take to identify which of the N models is 'correct'?
  - $\log_2(N)$ in the worst case
- Want simple models to explain examples and generalize to others
  - Ockham's (some say Occam) razor

Complex learning example: curve fitting

- How do we model the data?

$$t = \sin(2\pi x) + \text{noise}$$



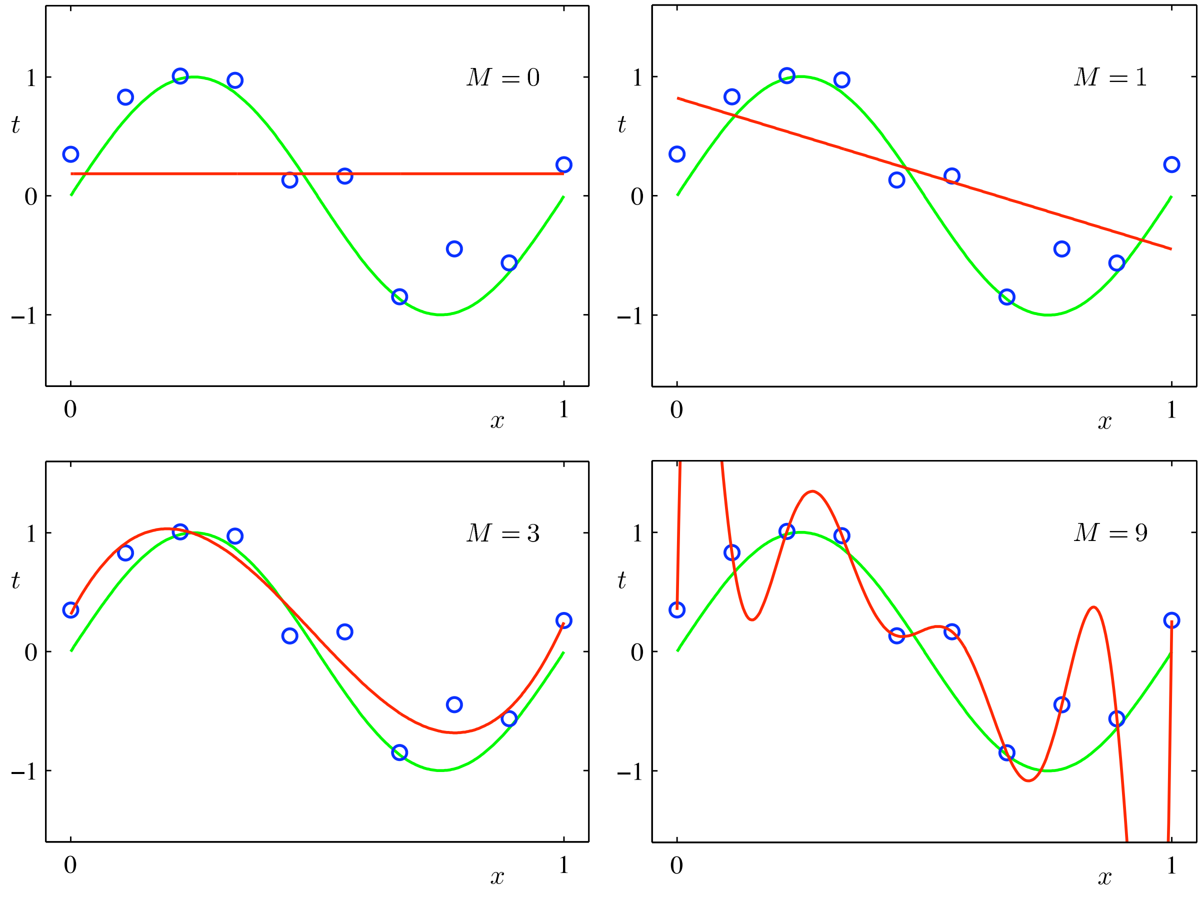


Polynomial curve fitting

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} [y(x_n, \mathbf{w}) - t_n]^2$$

$n=1$

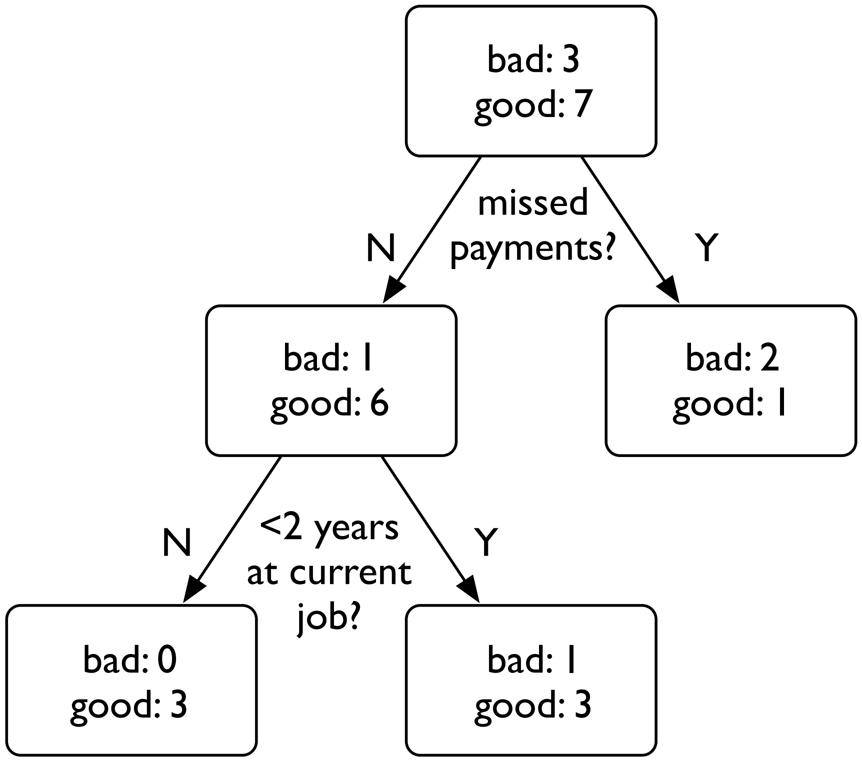


Decision trees: classifying from a set of attributes

- Each level splits the data according to different attributes
  - **goal**: achieve perfect classification with minimal number of decisions
  - not always possible due to noise or inconsistencies in the data

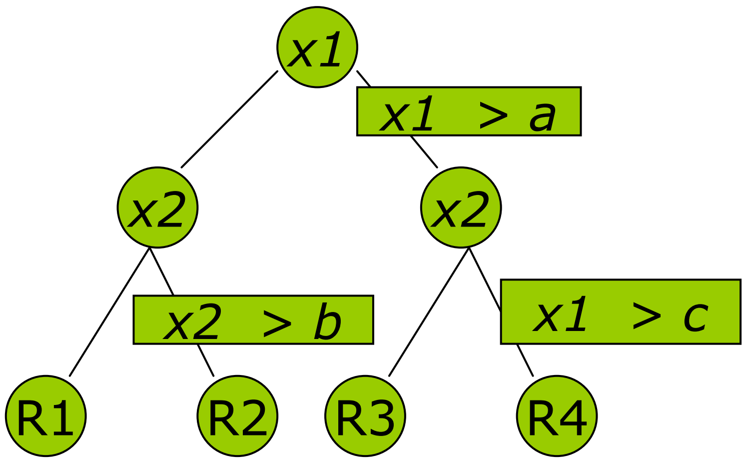
Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

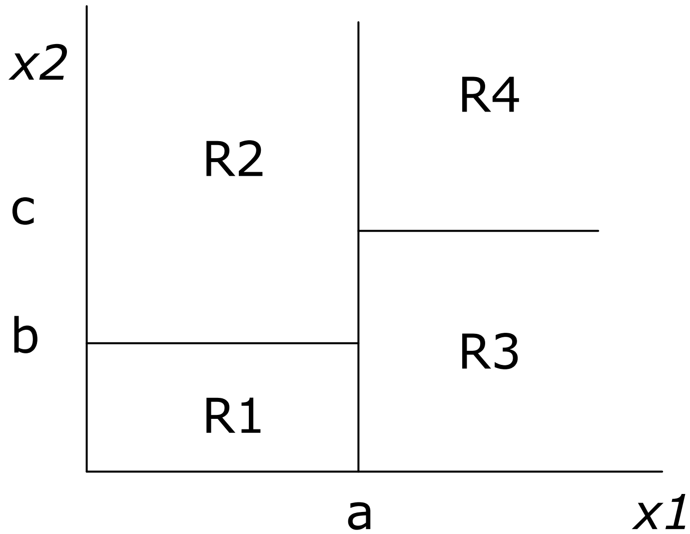


Decision Trees for Classification

- Input: Set of attribute-value pairs (same)
- Output: Set of classes (not a binary valued outcome of 'N' and 'P')
- Effectively dividing input space into decision regions
- Cuts in regions are parallel to input axes



Decision Tree



Decision Regions

Observations

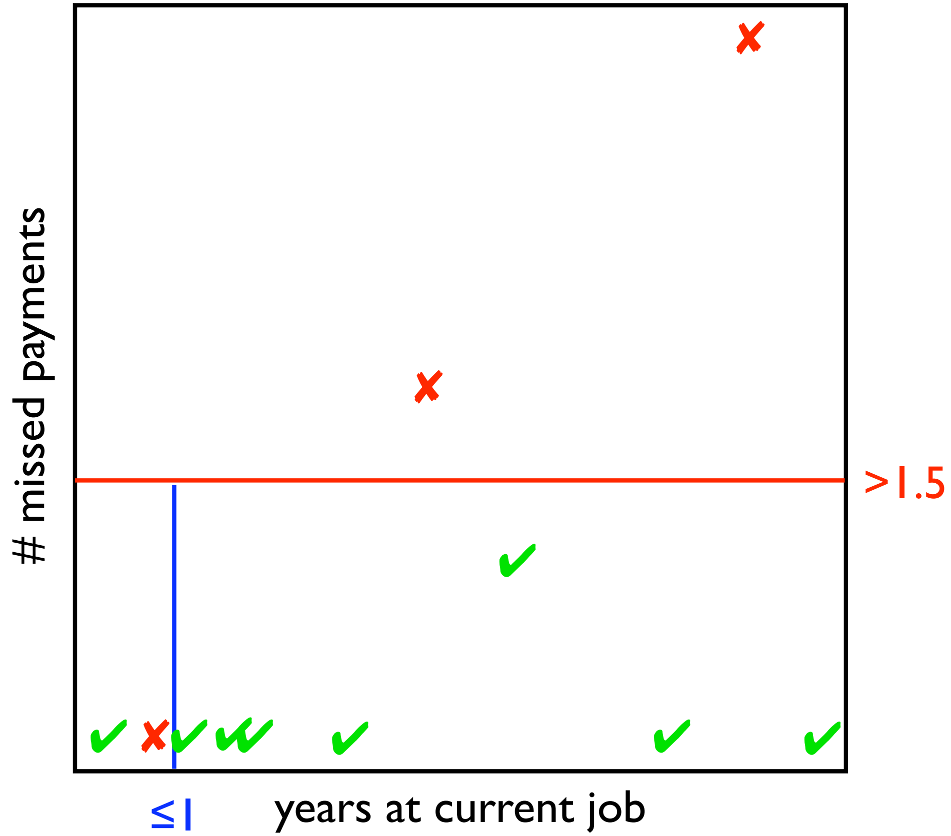
- Any boolean function can be represented by a decision tree.
- Not good for all functions, e.g.:
  - parity function: return 1 iff an even number of inputs are 1
  - majority function: return 1 if more than half inputs are 1
- best when a small number of attributes provide a lot of information
- Note: finding optimal tree for arbitrary data is NP-hard.

Decision trees with continuous values

- Now tree corresponds to order and placement of boundaries
- General case:
  - arbitrary number of attributes: binary, multi-valued, or continuous
  - output: binary, multi-valued (*decision or axis-aligned classification trees*), or continuous (*regression trees*)

Predicting credit risk

years at current job	# missed payments	defaulted?
7	0	N
0.75	0	Y
3	0	N
9	0	N
4	2	Y
0.25	0	N
5	1	N
8	4	Y
1.0	0	N
1.75	0	N

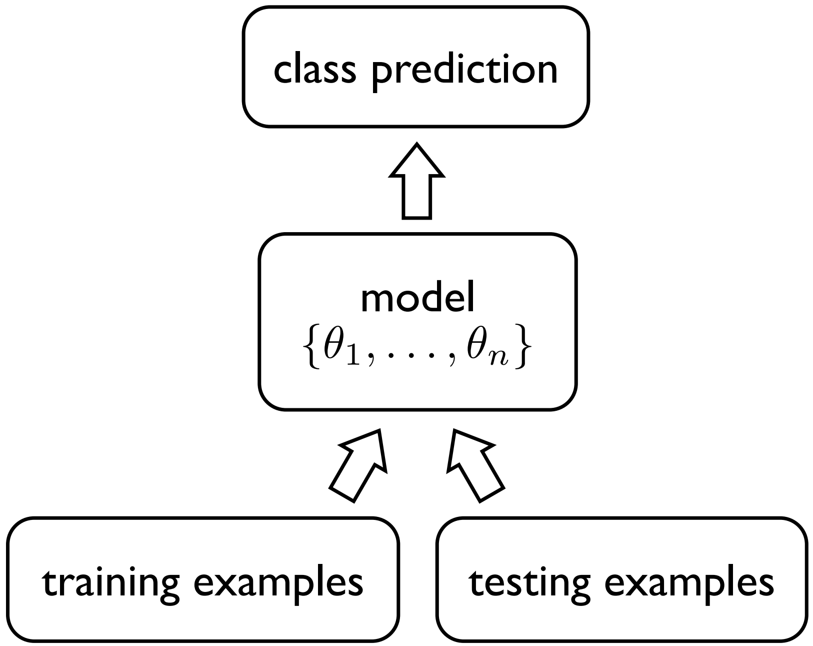


Examples

- loan applications
- medical diagnosis
- movie preferences (Netflix contest)
- spam filters
- security screening
- many real-word systems, and AI success
- In each case, we want
  - *accurate* classification, i.e. minimize error
  - *efficient* decision making, i.e. fewest # of decisions/tests
- decision sequence could be further complicated
  - want to minimize false negatives in medical diagnosis or minimize cost of test sequence
  - don't want to miss important email

Decision Trees

- Simple example of inductive learning
  1. *learn* decision tree from training examples
  2. *predict* classes for novel testing examples
- Generalization is how well we do on the testing examples.
- Only works if we can learn the underlying structure of the data.



Choosing the attributes

- How do we find a decision tree that agrees with the training data?
- Could just choose a tree that has one path to a leaf for each example
  - but this just memorizes the observations (assuming data are consistent)
  - we want it to *generalize* to new examples
- Ideally, best attribute would partition the data into positive and negative examples
- Strategy (greedy):
  - choose attributes that give the best partition first
- Want correct classification with fewest number of tests

Problems

- How do we choose which attribute or value to split on?
- When should we stop splitting?
- What do we do when we can't achieve perfect classification?
- What if tree is too large? Can we approximate with a smaller tree?

Basic algorithm for learning decision trees

1. starting with whole training data
2. select attribute or value along dimension that gives "best" split
3. create child nodes based on split
4. recurse on each child using child data until a stopping criterion is reached
   - all examples have same class
   - amount of data is too small
   - tree too large

- Central problem: How do we choose the "best" 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 is good (all true or all false)
  - Uniform distribution is bad

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

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

Measuring information

- A convenient measure to use is based on information theory.
  - How much "information" does an attribute give us about the class?
  - attributes that perfectly partition should given maximal information
  - unrelated attributes should give no information
- Information of symbol w:

$$I(w) \equiv -\log_2 P(w)$$

$$P(w) = 1/2$$
$$\Rightarrow I(w) = -\log_2 1/2 = 1 \text{ bit}$$

$$P(w) = 1/4$$
$$\Rightarrow I(w) = -\log_2 1/4 = 2 \text{ bits}$$

Information and Entropy

$$I(w) \equiv -\log_2 P(w)$$

- For a random variable *X* with probability *P(x)*, the entropy is the average (or expected) amount of information obtained by observing *x*:

$$H(X) = \sum_x P(x) I(x) = -\sum_x P(x) \log_2 P(x)$$

- Note: H(X) depends only on the probability, not the value.
- H(X) quantifies the uncertainty in the data in terms of bits
- H(X) gives a lower bound on cost (in bits) of coding (or describing) X

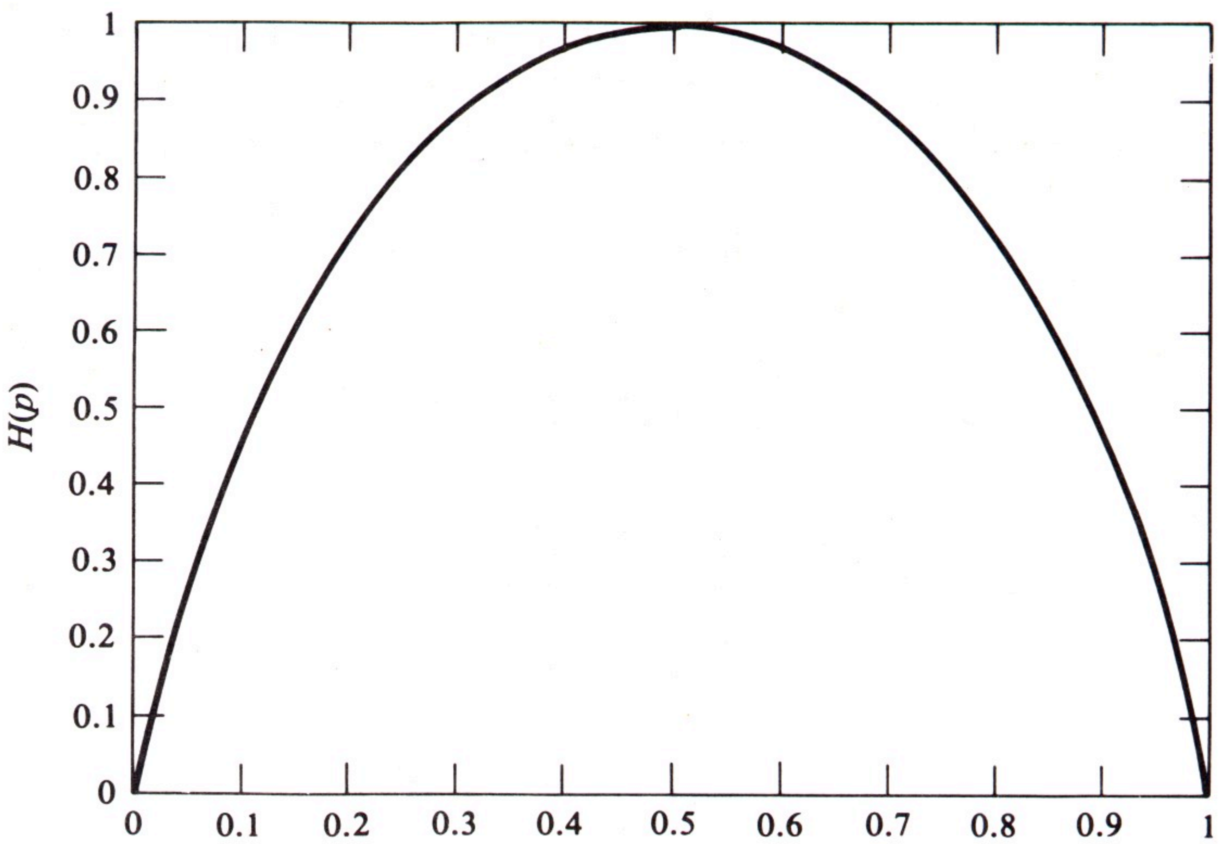
$$H(X) = -\sum_x P(x) \log_2 P(x)$$

$$P(\text{heads}) = 1/2 \Rightarrow -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1 \text{ bit}$$

$$P(\text{heads}) = 1/3 \Rightarrow -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.9183 \text{ bits}$$

Entropy of a binary random variable

- Entropy is maximum at p=0.5
- Entropy is zero at p= 0 or p=1



English character strings "A-Z" and space

The entropy *increases* as the data become less ordered.

$H_1 = 4.76$ bits/char

1. *Zero-order approximation.* (The symbols are independent and equiprobable.)

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

$H_2 = 4.03$ bits/char

2. *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI

   ALHENHTTPA OOBTTVA NAH BRL

•
•
•

$H_2 = 2.8$ bits/char

5. *Fourth-order approximation.* (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

   THE GENERATED JOB PROVIDUAL BETTER TRAND THE

   DISPLAYED CODE, ABOVERY UPONDULTS WELL THE

   CODERST IN THESTICAL IT DO HOCK BOTHE MERG.

Credit Risk Revisited

- How many bits does it take to specify the attribute of 'defaulted?'
  - P(defaulted = Y) = 3/10
  - P(defaulted = N) = 7/10

$$
\begin{aligned}
H(Y) &= -\sum_{i=\mathrm{Y,N}} P(Y = y_i)\log_2 P(Y = y_i) \\
&= -0.3\log_2 0.3 - 0.7\log_2 0.7 \\
&= 0.8813
\end{aligned}
$$

- How much can we *reduce* the entropy (or uncertainty) of 'defaulted' by knowing the other attributes?

- Ideally, we could reduce it to zero, in which case we classify perfectly.

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Conditional Entropy

- H(Y|X) is the remaining entropy of Y given X

or

- The expected (or average) entropy of P(y|x)

$$
\begin{aligned}
H(Y|X) &\equiv -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x) \\
&= -\sum_x P(x) \sum_y P(Y=y|X=x) \log_2 P(Y=y|X=x) \\
&= -\sum_x P(x) \quad H(Y|X=x)
\end{aligned}
$$

- H(Y|X=x) is the *specific conditional entropy*, i.e. the entropy of Y knowing the value of a specific attribute x.

Back to the credit risk example

$$
\begin{aligned}
H(Y|X) &\equiv -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x) \\
&= -\sum_x P(x) \sum_y P(Y=y|X=x) \log_2 P(Y=y|X=x) \\
&= -\sum_x P(x) \quad H(Y|X=x)
\end{aligned}
$$

$$
\begin{aligned}
H(\text{defaulted}|< \text{2years} = \text{N}) &= -\frac{4}{4+2}\log_2\frac{4}{4+2} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183 \\
H(\text{defaulted}|< \text{2years} = \text{Y}) &= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8133 \\
H(\text{defaulted}|< \text{2 years}) &= \frac{6}{10}0.9183 + \frac{4}{10}0.8133 = 0.8763
\end{aligned}
$$

$$
\begin{aligned}
H(\text{defaulted}|\text{missed} = \text{N}) &= -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5917 \\
H(\text{defaulted}|\text{missed} = \text{Y}) &= -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.9183 \\
H(\text{defaulted}|\text{missed}) &= \frac{7}{10}0.5917 + \frac{3}{10}0.9183 = 0.6897
\end{aligned}
$$

Mutual Information

- We now have the entropy - the minimal number of bits required to specify the target attribute:

$$H(Y) = \sum_y P(y) \log_2 P(y)$$

- The conditional entropy - the remaining entropy of Y knowing X

$$H(Y|X) = -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x)$$

- So we can now define the reduction of the entropy after learning Y.

- This is known as the *mutual information* between Y and X

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

Properties of Mutual Information

- Mutual information is symmetric

$$I(Y;X) = I(X;Y)$$

- In terms of probability distributions, it is written as

$$I(X;Y) = -\sum_{x,y} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- It is zero, if Y provides no information about X:

$$I(X;Y) = 0 \Leftrightarrow P(x) \text{ and } P(y) \text{ are independent}$$

- If Y = X then

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

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

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

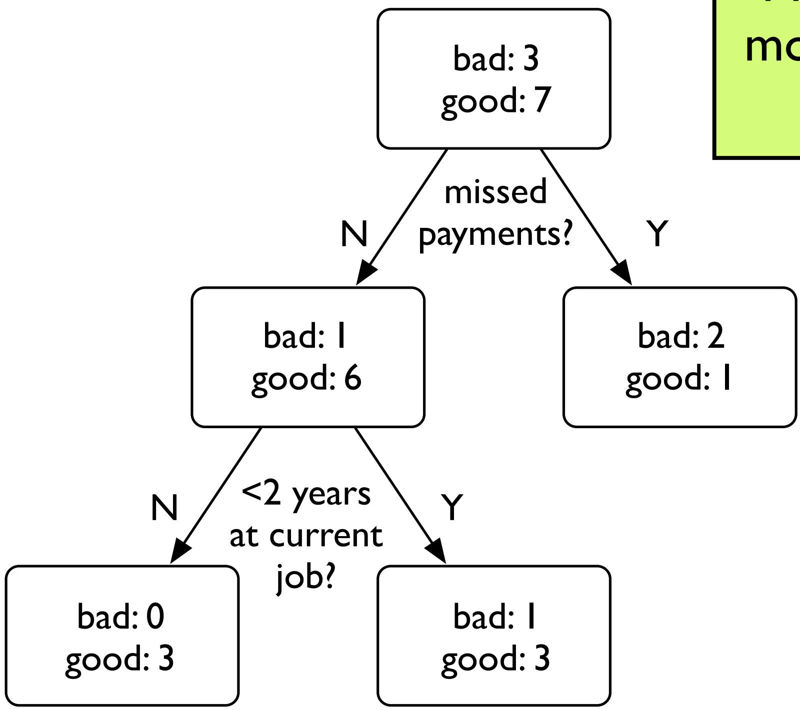
- Information gain is difference $IG(X) = H(Y) - H(Y \mid X)$

Information Gain

$$H(\text{defaulted}) - H(\text{defaulted}|< 2 \text{ years})$$
$$0.8813 - 0.8763 = 0.0050$$

$$H(\text{defaulted}) - H(\text{defaulted}|\text{missed})$$
$$0.8813 - 0.6897 = 0.1916$$

Missed payments are the most informative attribute about defaulting.



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

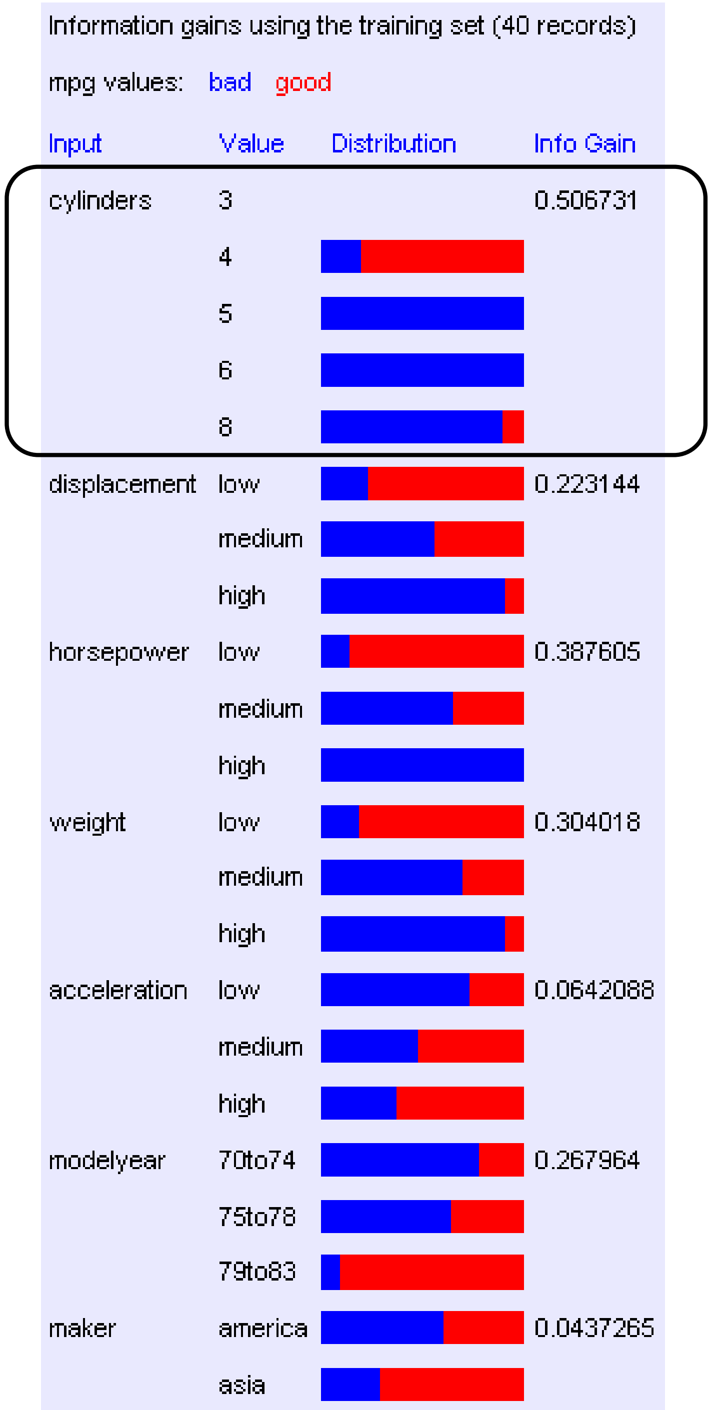
Example (from Andrew Moore): Predicting miles per gallon http://www.autonlab.org/tutorials/dtree.html

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

First step: calculate information gains

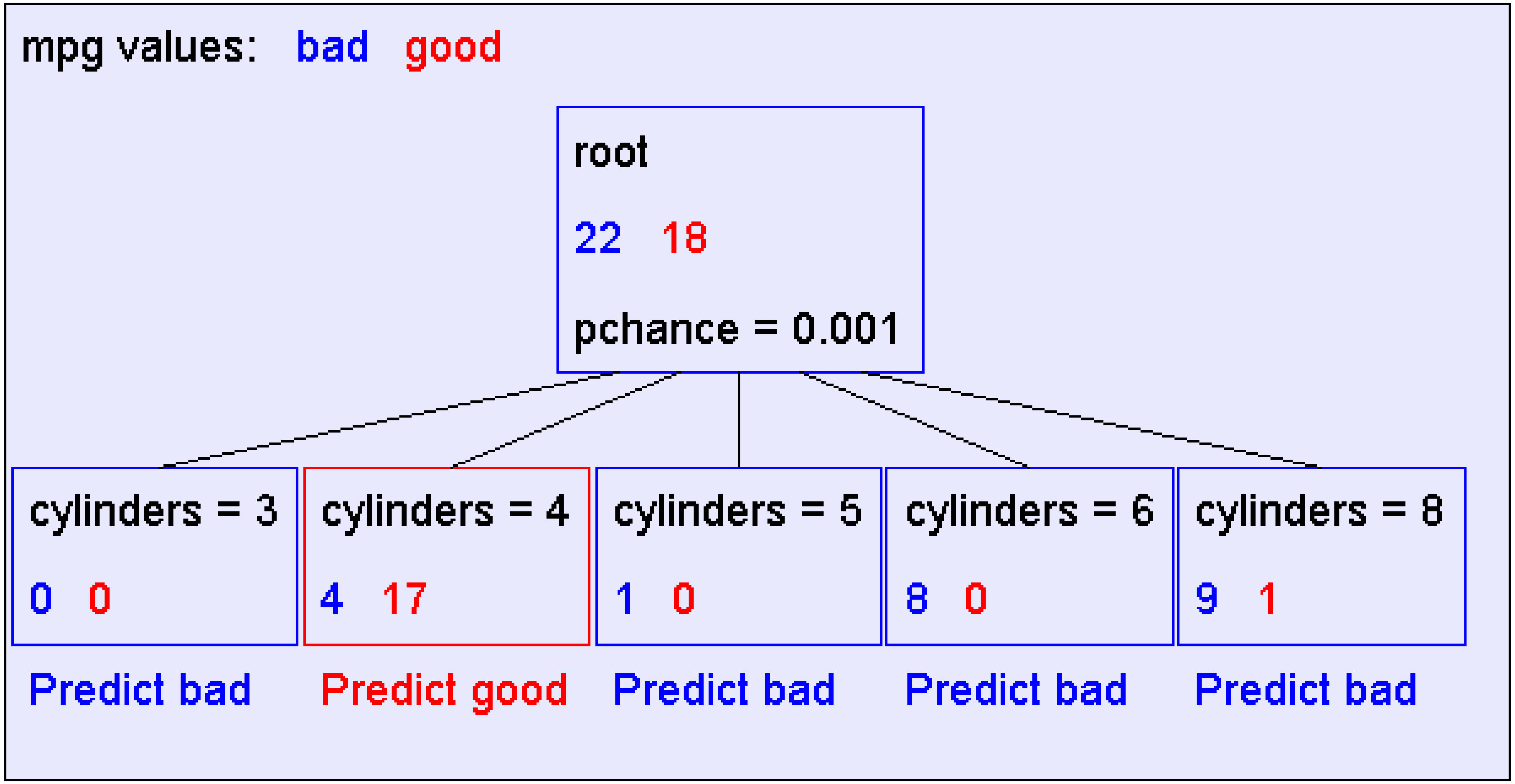
- Compute for information gain for each attribute

- In this case cylinders provide the most gain, because it nearly partitions the data.

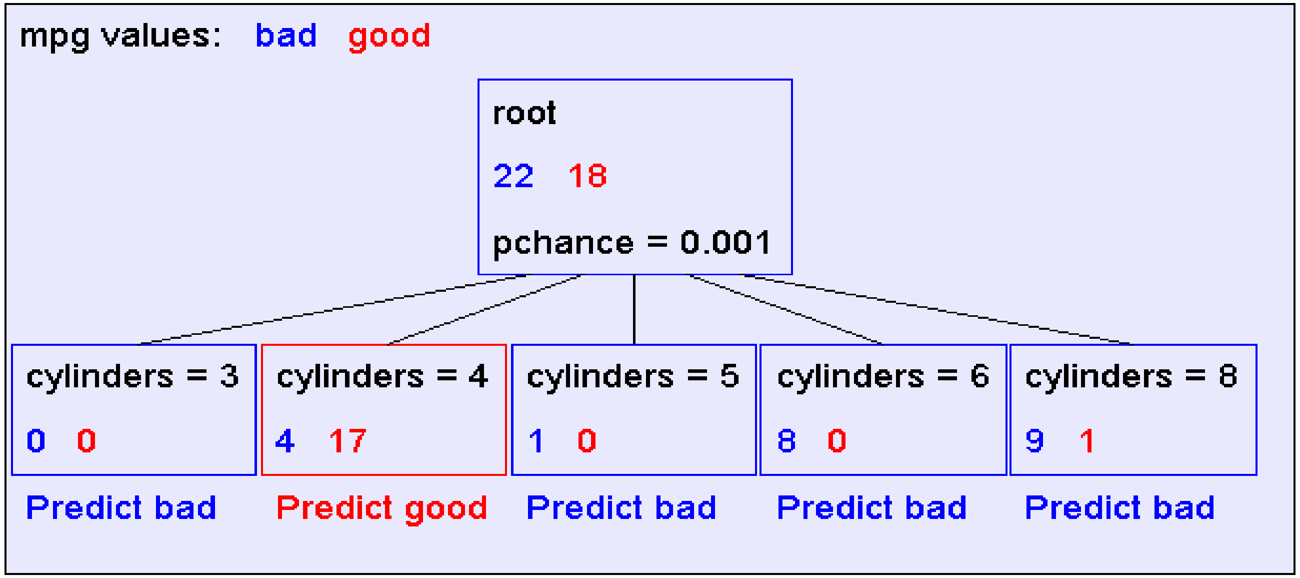


First decision: partition on cylinders

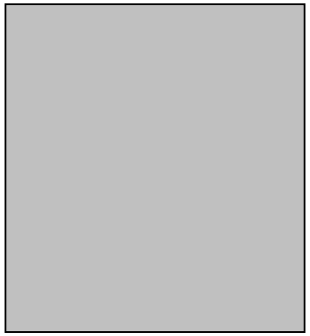
- Note the lopsided mpg class distribution.



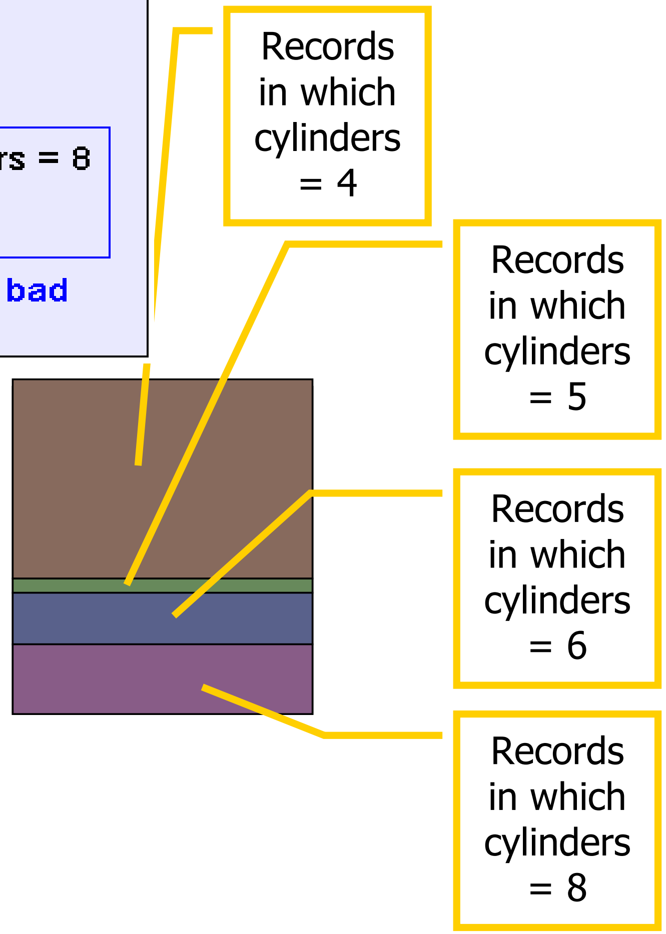
Recurse on child nodes to expand tree



Take the Original Dataset..

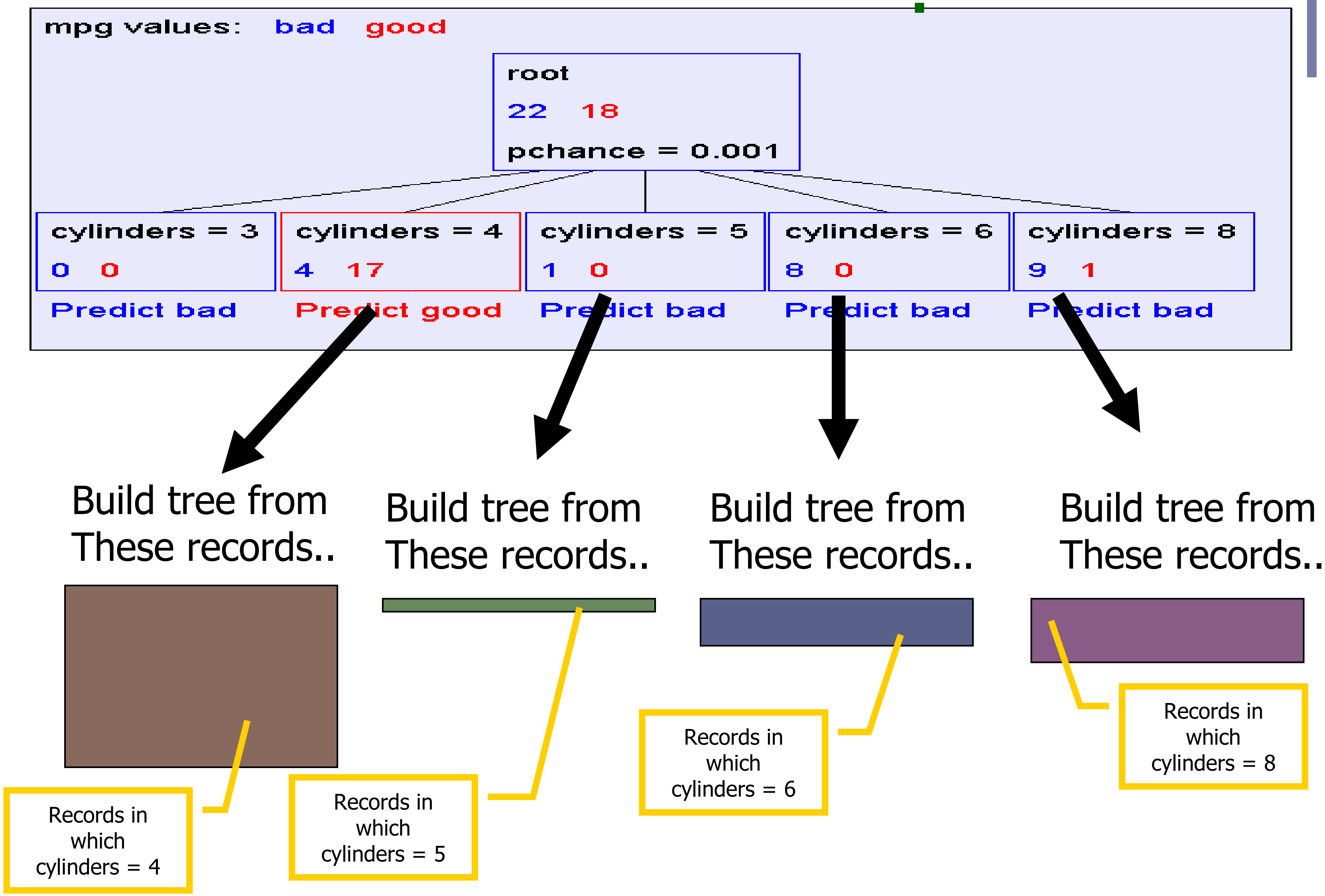


And partition it according to the value of the attribute we split on

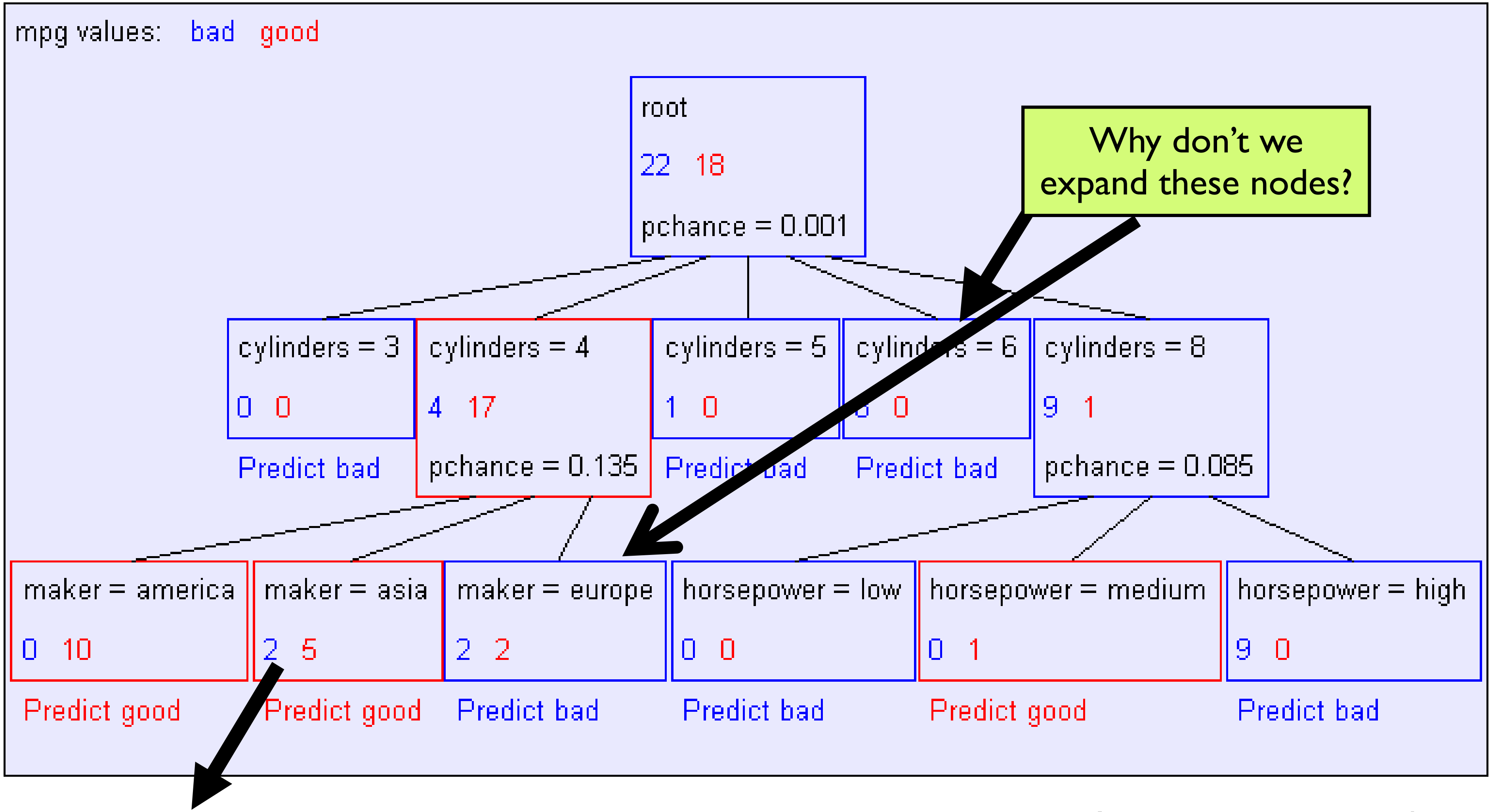


Expanding the tree: data is partitioned for each child

- Exactly the same, but with a smaller, conditioned datasets.



Second level of decisions



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)

- Base Case 1: Don't split a node if all matching records have the same output value

- Base Case 2: Don't split a node if none of the attributes can create multiple non-empty children
  - If all records have exactly the same set of input attributes then don't recurse

Is this a good idea?

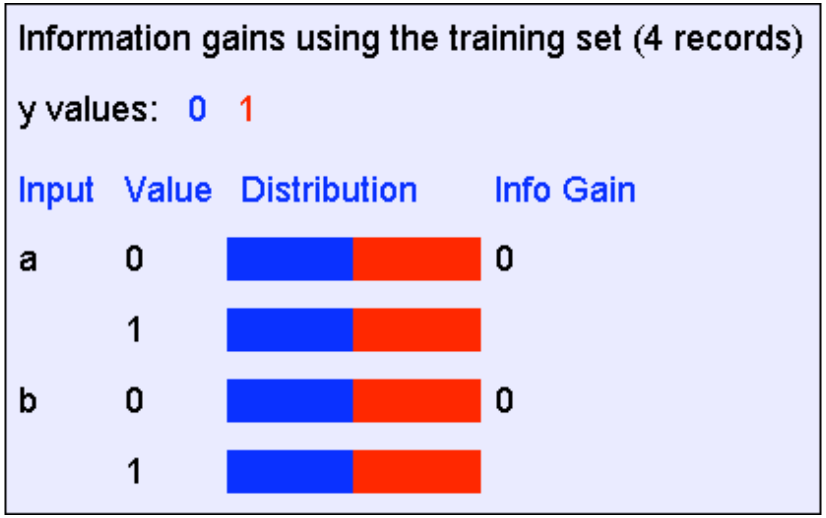
- Proposed Base Case 3:
  - If all attributes have zero information gain then don't recurse

The problem with Base Case 3

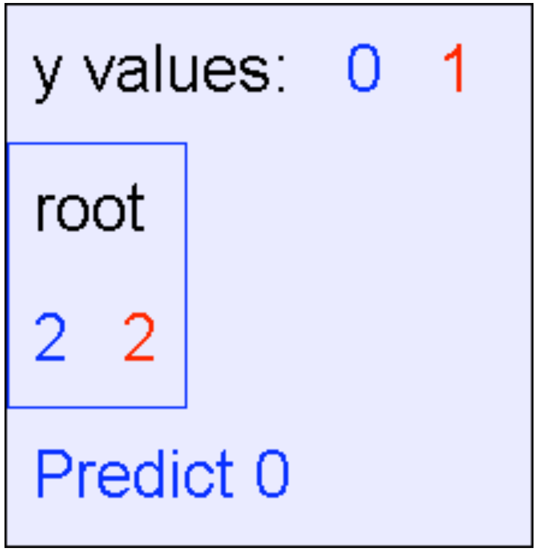
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



The resulting decision tree:

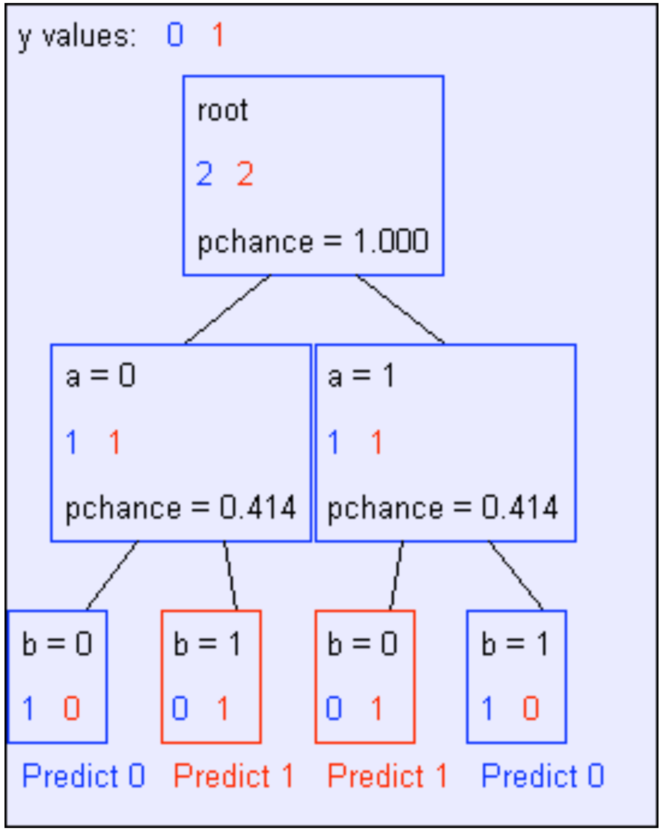


If we omit Base Case 3:

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

y = a XOR b

The resulting decision tree:



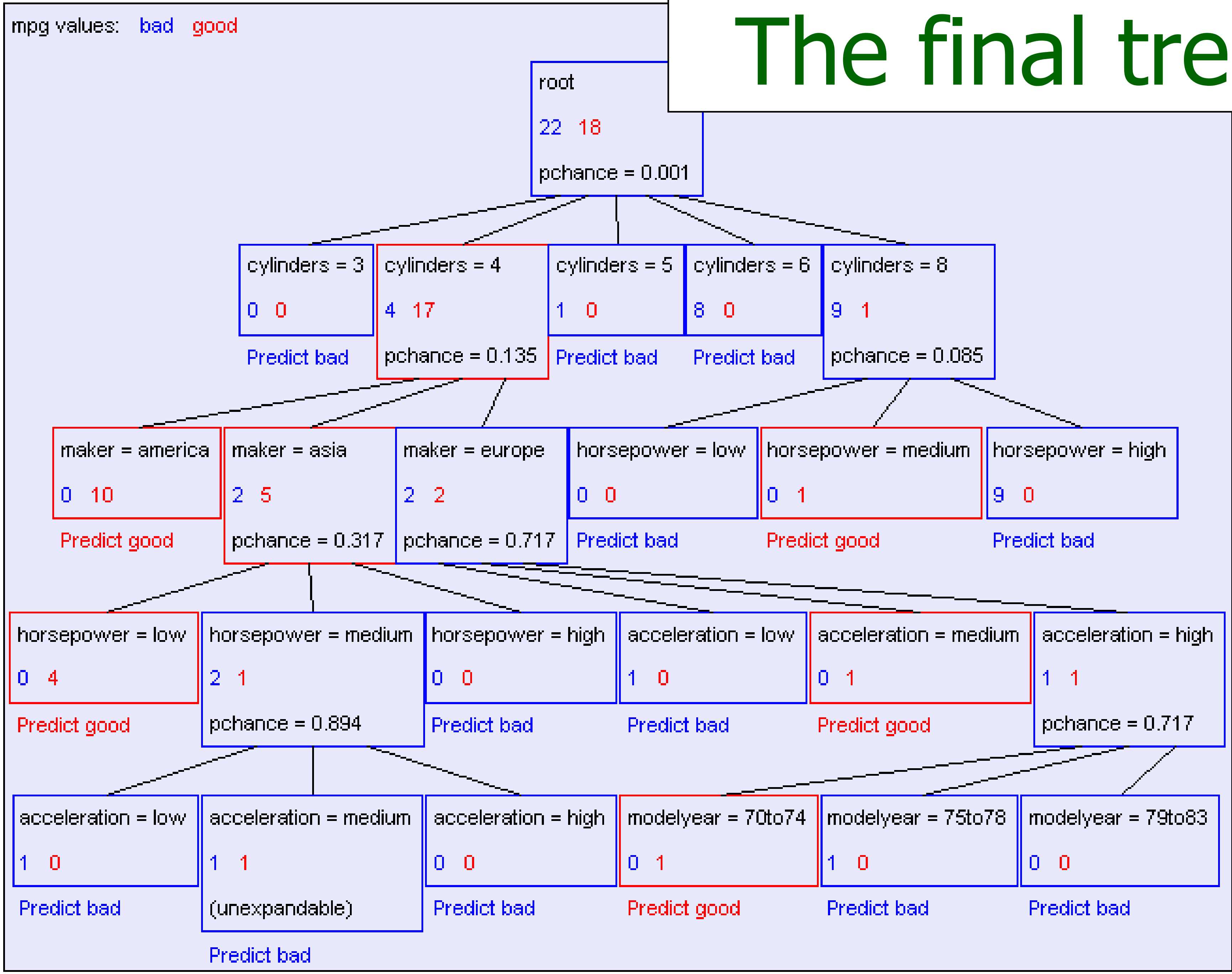
Basic Decision Tree Building Summarized

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.

The final tree



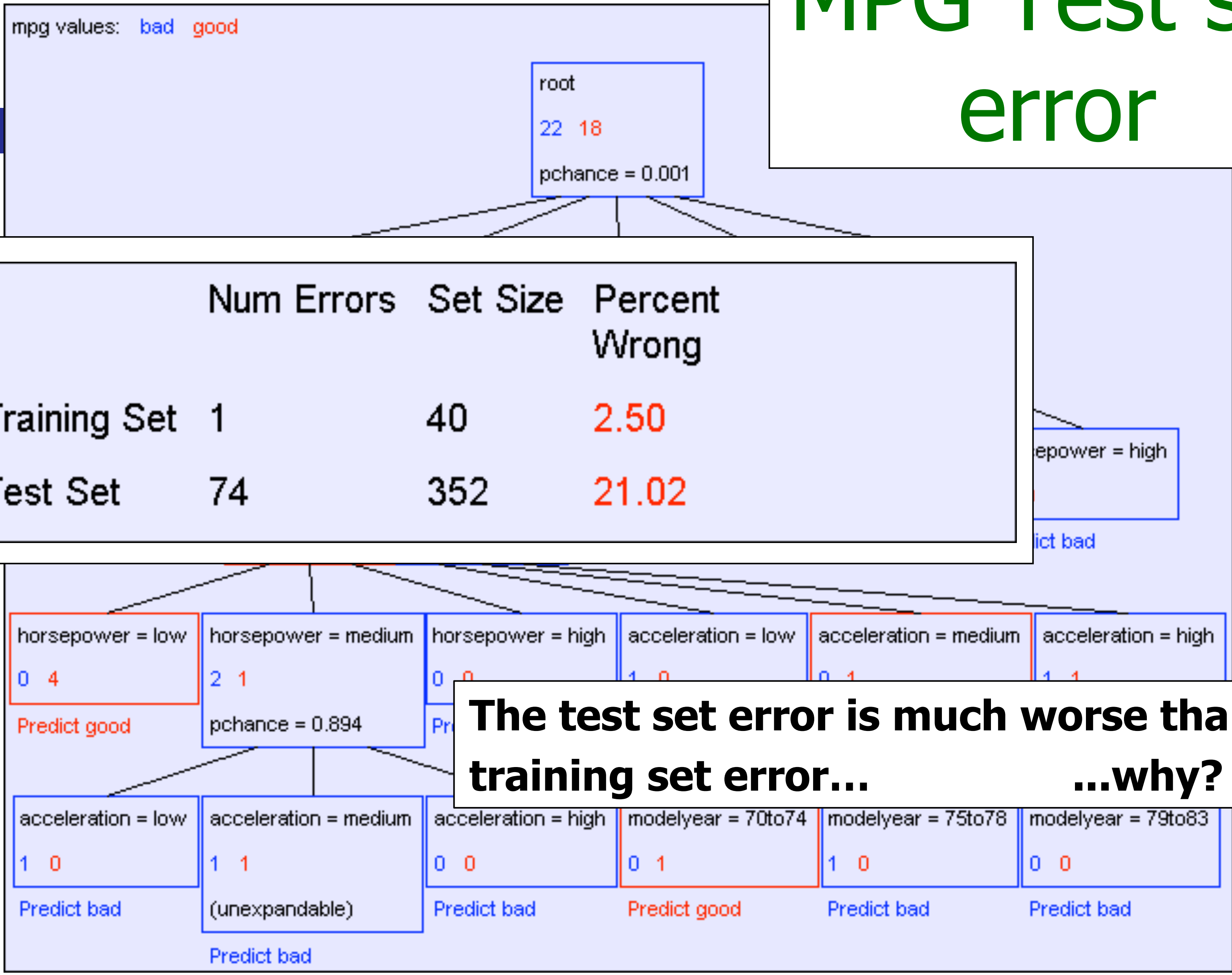
Decision trees & Learning Bias

- Decision trees will overfit
- Standard decision trees 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
  - Fixed number of leaves
  - Or something smarter…

Decision Trees for Classification

- To classify a new example – traverse tree and report leaf label
- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $\phi = A \wedge B \vee \neg A \wedge C$ ((A and B) or (not A and C))

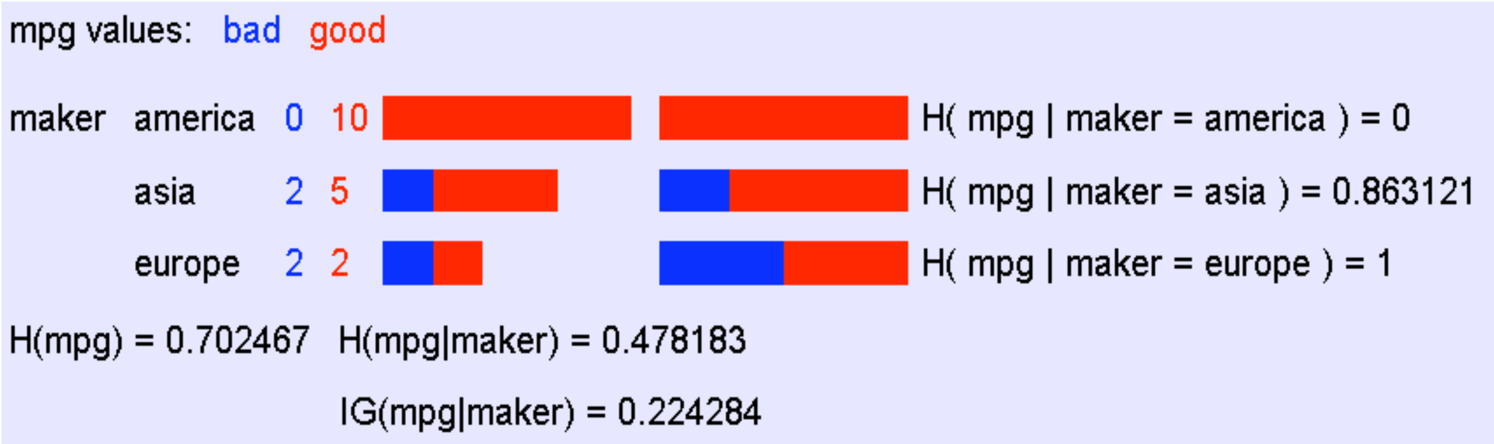
MPG Test set error



	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?

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 in your homework, you'll have fun! :))

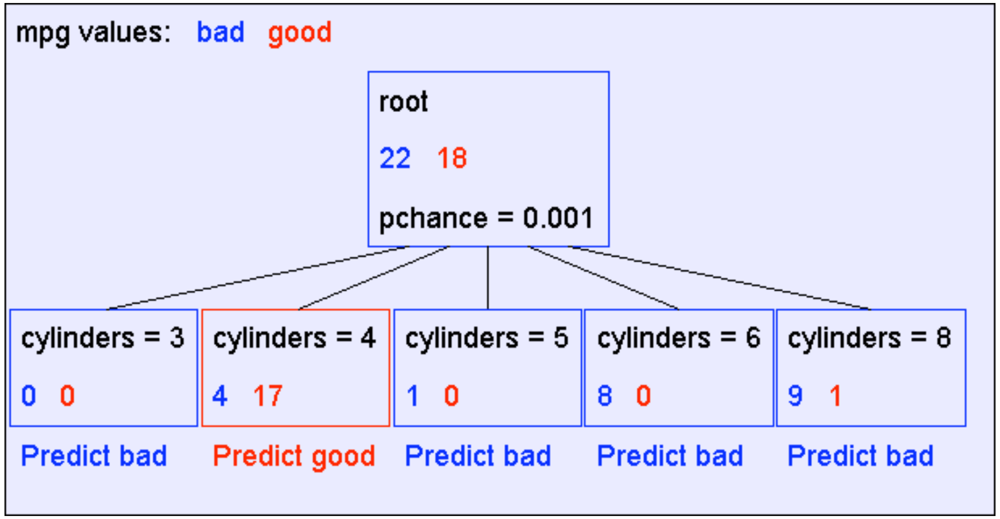
Using Chi-squared to avoid overfitting

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

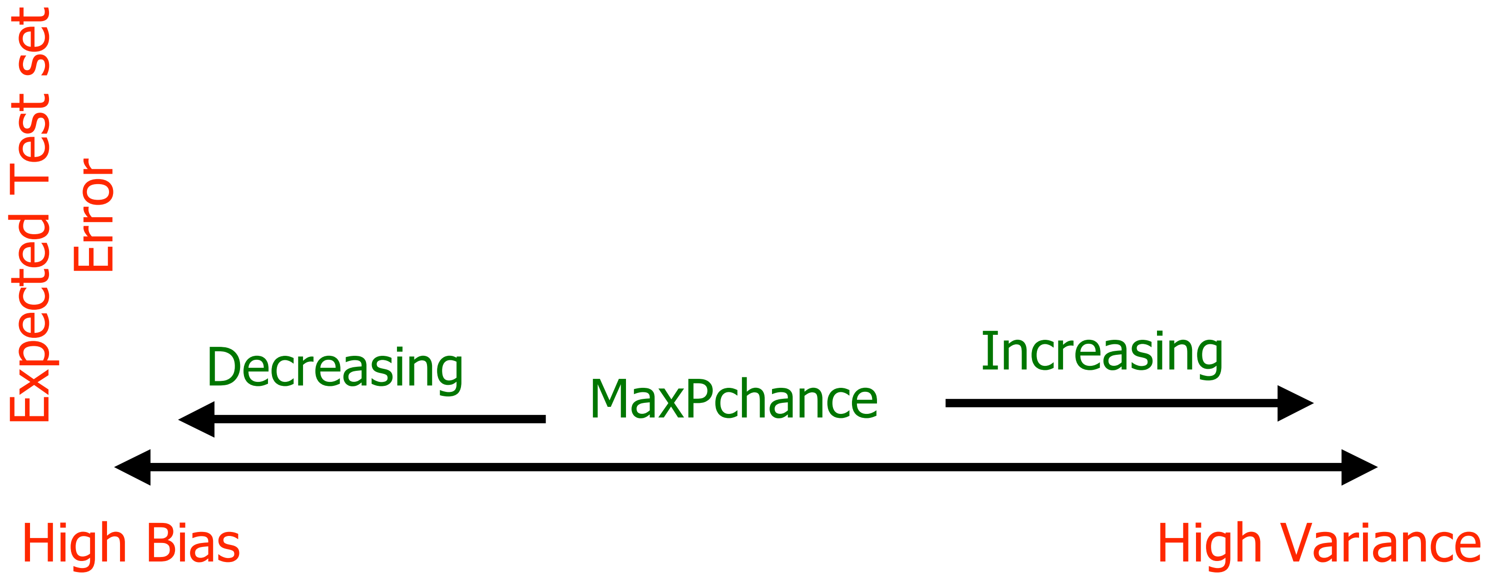


Note the improved test set accuracy compared with the unpruned tree

	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



Decision Trees for Regression

- Move from Discrete outcomes -> Continuous valued functions
- How do you measure the goodness of your classifier?
  - Loss = Number of misclassified inputs/data points
- How do you measure the goodness of your regression hypothesis?
  - Loss = Square Loss $L_D(f) = \mathbf{E}_{(x,y)\sim D}(f(x) - y)^2$
  - Loss = Absolute Loss $\ell_D(f) = \mathbf{E}_{(x,y)\sim D}|f(x) - y|$
- There are greedy heuristic based algorithms that build regression trees iteratively

Decision Trees in Practice

- Deal with Overfitting : Pruning away low information gain, or statistically insignificant attributes
- k-fold cross-validation: To deal with overfitting
- Advantages:
  - Human readability White box classifier
- Disadvantages:
  - Parallel splits in input space - as opposed to Diagonal splits ($x_i < x_j$) make some problems harder to learn
  - Splits are very sensitive to training data

Assignment 1 (5 marks)

- Implement the decision tree building algorithm presented in this lecture, and submit the code and the calculation results for each node IG to explain the final tree
- Update the algorithm to avoid overfitting using chi-squared method, and submit the code and the calculation results for each node IG to explain the final tree
- Due date: 27 December, 2014, 11 p.m.