Naïve Bayes & Logistic Regression,
See class website:
Mitchell's Chapter (required)
Ng & Jordan '02 (optional)
Gradient ascent and extensions:
Koller & Friedman Chapter 1.4
Decision Trees: many possible refs., e.g.,
Mitchell, Chapter 3

Logistic Regression (Continued)
Generative v. Discriminative
Decision Trees

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

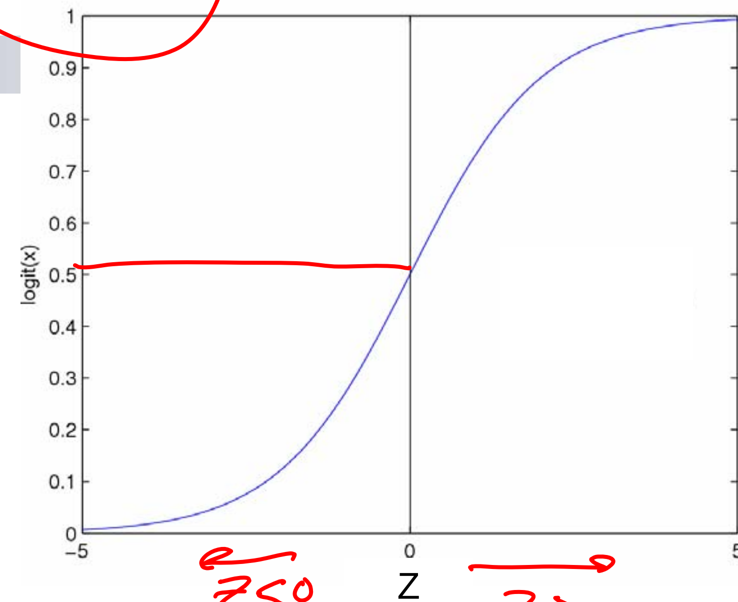
February 1st, 2006

Announcements

- Recitations stay on Thursdays
  - 5-6:30pm in Wean 5409
  - This week: Naïve Bayes & Logistic Regression
- **Extension** for the first homework:
  - **Due Wed. Feb 8th** beginning of class
  - Mitchell's chapter is most useful reading

Logistic Regression

Logistic function (or Sigmoid): $g(z) = \dfrac{1}{1 + exp(-z))}$



$z<0$ $z>0$

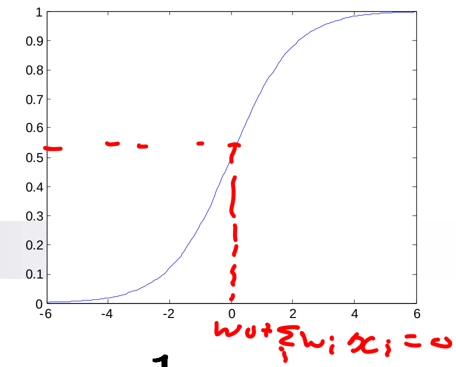
- Learn P(Y|**X**) directly!
  - Assume a particular functional form
  - Sigmoid applied to a linear function of the data:

regression part

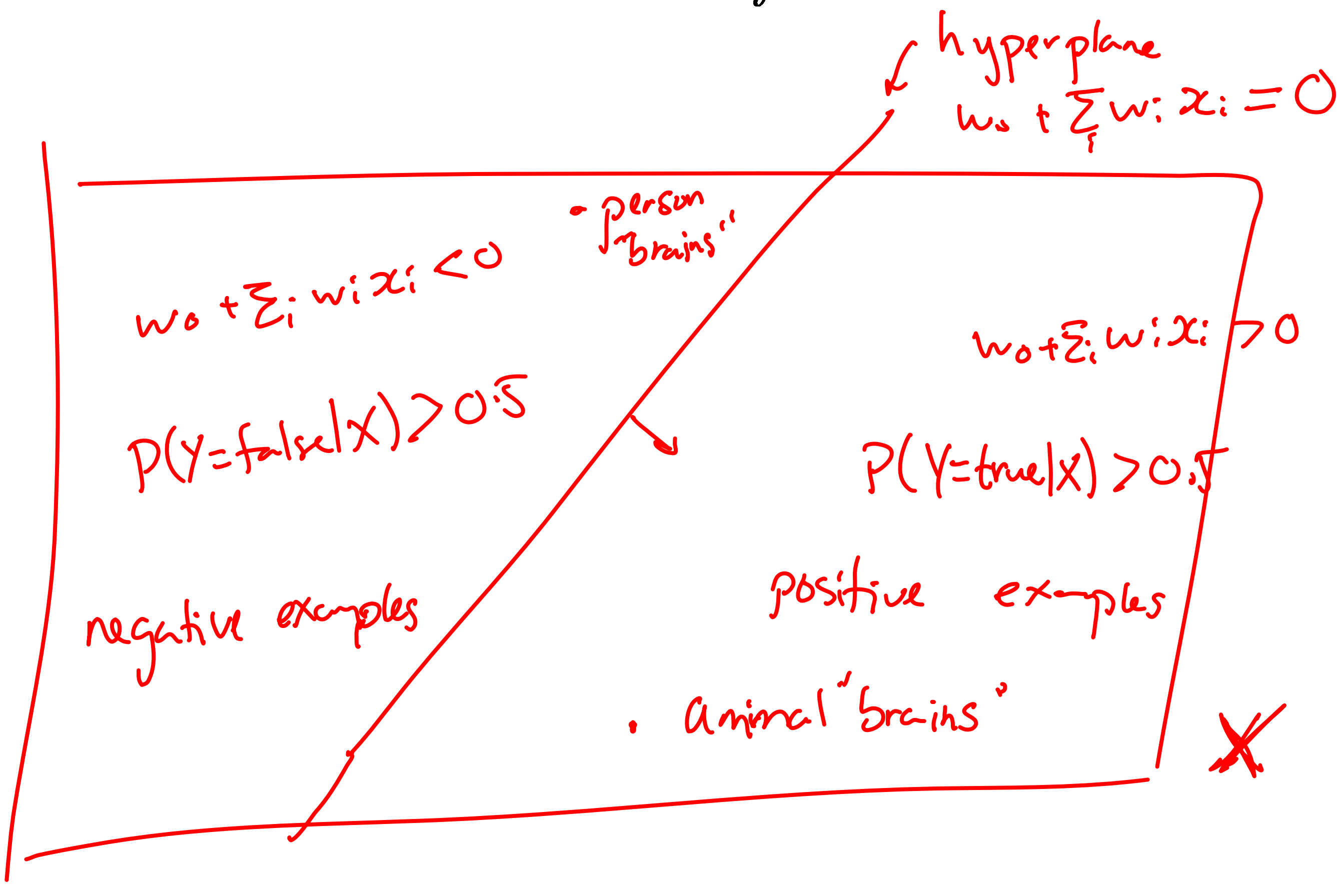
$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

$$P(Y=0|X) = 1 - P(Y=1|X)$$

Logistic Regression – a Linear classifier



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$



Logistic regression v. Naïve Bayes

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, < X1 … Xn >
  - Y is boolean
- Could use a Gaussian Naïve Bayes classifier
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli(θ,1-θ)
- What does that imply about the form of P(Y|X)?

$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

Cool!!!!

Gaussian Naïve Bayes v. Logistic Regression

a choice of LR parameters

Set of Gaussian Naïve Bayes parameters

Set of Logistic Regression parameters

a set w give you at least 1 set of μ, σ

- Representation equivalence
  - **But only in a special case!!!** (GNB with class-independent variances)
- But what's the difference???
- **LR makes no assumptions about** P(**X**|Y) **in learning**!!!
- **Loss function!!!**
  - Optimize different functions $\rightarrow$ Obtain different solutions

Logistic regression more generally

$w_i^T X = w_{0i} + \sum_j w_{ji} X_j$

- Logistic regression in more general case, where $Y \in \{Y_1 \ldots Y_R\}$: learn R-1 sets of weights

$\sum_i P(y_i|X) = 1 = \sum_{i=1}^{R-1} e^{w_i^T X} + 1$

$1 + \sum_{j=1}^{R-1} e^{w_j^T X}$

for $k<R$

$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^n w_{ki} X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji} X_i)}$$

for $k=R$ (normalization, so no weights for this class)

$$P(Y = y_R|X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji} X_i)}$$

categorical to numbers, e.g., X_i: true → +1, false → -1

Features can be discrete or continuous!

Logistic regression with more than 2 classes – an example

- $Y \in \{Y_1 \ldots Y_R\}$: learn *R-1* sets of weights

$\sum_{y_i} P(Y=y_i|X)=1$

for *k<R*

$$P(Y=y_k|X)=\frac{\exp(w_{k0}+\sum_{i=1}^{n} w_{ki}X_i)}{1+\sum_{j=1}^{R-1}\exp(w_{j0}+\sum_{i=1}^{n} w_{ji}X_i)}$$

for *k=R* (normalization,

so no weights for this class)

$$P(Y=y_R|X)=\frac{1}{1+\sum_{j=1}^{R-1}\exp(w_{j0}+\sum_{i=1}^{n} w_{ji}X_i)}$$

Z

Features can be discrete or continuous!

$Y=\{S,P,F\}$

$P(Y=S|X) \propto \frac{e^{w_{0S}+\sum_i w_{iS}X_i}}{Z}$

$P(Y=P|X) \propto \frac{e^{w_{0P}+\sum_i w_{iP}X_i}}{Z}$

$P(Y=F|X) \propto \frac{1}{Z}$ (normalization

$P(Y=S|X)+P(Y=P|X)+P(Y=F|X)=1$

$\frac{e^{w_{0S}+\sum_i w_{iS}X_i}}{Z}+\frac{e^{w_{0P}+\sum_i w_{iP}X_i}}{Z}+\frac{1}{Z}=1$

$Z=1+e^{w_{0S}+\sum_i w_{iS}X_i}+e^{w_{0P}\sum_i w_{iP}X_i}$

Loss functions: Likelihood v. Conditional Likelihood

chain rule of prob.:

$p(x, y \mid w) = P(y \mid x, w) \cdot P(x \mid w)$

- Generative (Naïve Bayes) Loss function:

  Data likelihood

$$\ln P(\mathcal{D} \mid \mathbf{w}) = \sum_{j=1}^{N} \ln P(\mathbf{x}^j, y^j \mid \mathbf{w})$$

$$= \sum_{j=1}^{N} \underbrace{\ln P(y^j \mid \mathbf{x}^j, \mathbf{w})}_{P(y \mid x)} + \sum_{j=1}^{N} \underbrace{\ln P(\mathbf{x}^j \mid \mathbf{w})}_{P(x)}$$

- Discriminative models cannot compute P($\mathbf{x}^j$|**w**)!
- But, discriminative (logistic regression) loss function:

  Conditional Data Likelihood

$$\ln P(\mathcal{D}_Y \mid \mathcal{D}_{\mathbf{X}}, \mathbf{w}) = \sum_{j=1}^{N} \ln P(y^j \mid \mathbf{x}^j, \mathbf{w})$$

  - Doesn't waste effort learning P(X) – focuses on P(Y|**X**) all that matters for classification

Expressing Conditional Log Likelihood

j^{th} example

$$l(\mathbf{w}) \equiv \sum_j \ln P(y^j|\mathbf{x}^j,\mathbf{w})$$

$y^j = \begin{cases} 0 \\ 1 \end{cases}$

$$P(Y=0|\mathbf{X},\mathbf{w}) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|\mathbf{X},\mathbf{w}) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$l(\mathbf{w}) = \sum_j \overbrace{y^j \ln P(y^j=1|\mathbf{x}^j,\mathbf{w})}^{\text{if } y_j=1} + \overbrace{(1-y^j)\ln P(y^j=0|\mathbf{x}^j,\mathbf{w})}^{\text{if } y_j=0}$$

$\rightarrow \ln P(y_j=1|x^j,w)$

$= w_0 + \sum_i w_i x_i - \ln\left(1+e^{w_0+\sum_i w_i x_i}\right)$

$\rightarrow \ln P(y_j=0|x^j w) = \cancel{\ln 1}^{0} - \ln\left(1+e^{w_0+\sum_i w_i x_i}\right)$

Maximizing Conditional Log Likelihood

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$\begin{aligned} l(\mathbf{w}) &\equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w}) \\ &= \sum_j y^j(w_0+\sum_i^n w_i x_i^j) - \ln(1+exp(w_0+\sum_i^n w_i x_i^j)) \end{aligned}$$

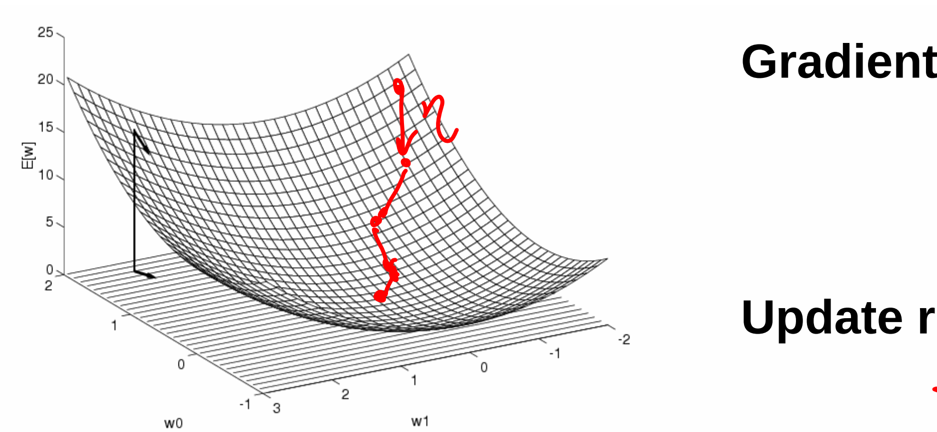
Good news: $l(\mathbf{w})$ is concave function of $\mathbf{w}$ $\rightarrow$ no locally optimal solutions

Bad news: no closed-form solution to maximize $l(\mathbf{w})$

Good news: concave functions easy to optimize

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave
  $\rightarrow$ Find optimum with gradient ascent



Gradient: $$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$$

Learning rate, η>0

Update rule: $$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

local minima:

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent much better (see reading) with line search.

Maximize Conditional Log Likelihood: Gradient ascent

$\frac{\partial}{\partial w} e^{f(w)} = f'(w) e^{f(w)}$

$\frac{\partial}{\partial w} \ln f(w) = \frac{f'(w)}{f(w)}$

derivative $\frac{\partial l}{\partial w_0}$ is slightly different !!

$$l(\mathbf{w}) = \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + exp(w_0 + \sum_{i=1}^n w_i x_i^j))$$

$i = 1:n$

$$\frac{\partial l}{\partial w_i} = \sum_j y^j x_i^j - \frac{x_i^j e^{w_0 + \sum_i w_i x_i^j}}{1 + e^{w_0 + \sum_i w_i x_i^j}}$$

$$= \sum_j x_i^j \left(y^j - \frac{e^{w_0 + \sum_i w_i x_i^j}}{1 + e^{w_0 + \sum_i w_i x_i^j}} \right)$$

$$= \sum_j x_i^j \left[y^j - P(Y=1 \mid x^j, w) \right]$$

$$\frac{\partial l}{\partial w_0} = \sum_j \left[y^j - p(y=1 \mid x^j, w) \right]$$

Gradient ascent algorithm: iterate until change < ε

For all i, $i = 1:n$
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \widehat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

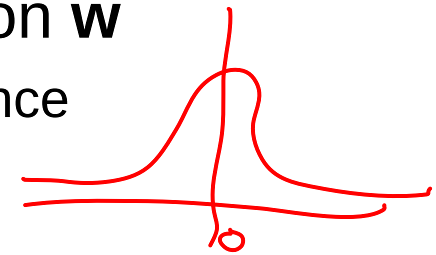
repeat

$i = 0$ slightly different

That's all M(C)LE. How about MAP?

$$p(\mathbf{w} \mid Y, \mathbf{X}) \;\propto\; P(Y \mid \mathbf{X}, \mathbf{w})\, p(\mathbf{w})$$

- One common approach is to define priors on **w**
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero



- Corresponds to ***Regularization***
  - Helps avoid very large weights and overfitting
  - Explore this in your homework
  - More on this later in the semester

- MAP estimate

prior

likelihood

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

M(C)AP as Regularization

w_i indep w_j

$w_i \sim N(0, \kappa^2)$

$$\ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$p(\mathbf{w}) = \prod_i \frac{1}{\kappa\sqrt{2\pi}} \; e^{\frac{-w_i^2}{2\kappa^2}}$$

$$= \ln p(w) + \underbrace{\ln \prod_{j=1}^{N} P(y^j \mid x^j, w)}$$

$$= \sum_i \ln \frac{1}{\kappa\sqrt{2\pi}} - \frac{w_i^2}{2\kappa^2} + \ell(w)$$

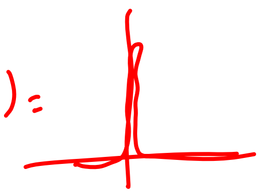
pick κ large

can overfit $p(w)$:

pick κ small

very biased

$p(w)$:



$$\max_w \; \ell(w) - \underbrace{\sum_i \frac{w_i^2}{2\kappa^2}}_{\text{small parameters}}$$

$\ell(w)$ → fit data

Penalizes high weights, also applicable in linear regression (see homework)

Gradient of M(C)AP

$$\frac{\partial}{\partial w_i} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$p(\mathbf{w}) = \prod_i \frac{1}{\kappa\sqrt{2\pi}} \; e^{\frac{-w_i^2}{2\kappa^2}}$$

$$= \frac{\partial}{\partial w_i} \ln p(w) + \frac{\partial}{\partial w_i} \underbrace{\ln \prod_{j=1}^{N} p(y^j \mid x^j, w)}_{\ell(w)}$$

$$\frac{\partial}{\partial w_i}\left(? - \frac{w_i^2}{2\kappa^2} \right)$$

$$- \underbrace{\frac{w_i}{\kappa^2}} + \frac{\partial}{\partial w_i} \ell(w)$$

$P(Y=1|X) = \frac{e^{w_0+\sum_i w_i x_i}}{1+e^{\cdots}}$

$P(Y=0|X) = \frac{1}{1+e^{\cdots}}$

MLE vs MAP

- Maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln\left[\prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w})\right]$$

$$\boxed{w_i \leftarrow w_i + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]}$$ standard

- Maximum conditional a posteriori estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln\left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w})\right]$$

$\lambda = \frac{1}{\kappa^2}$

$$\boxed{w_i \leftarrow w_i + \eta \left\{-\lambda w_i + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]\right\}}$$

Naïve Bayes vs Logistic Regression

Consider Y boolean, X_i continuous, $X=<X_1 \ldots X_n>$

Number of parameters:

- NB: 4n +1
- LR: n+1

Estimation method:

- NB parameter estimates are uncoupled
- LR parameter estimates are coupled

G. Naïve Bayes vs. Logistic Regression 1

[Ng & Jordan, 2002]

- Generative and Discriminative classifiers

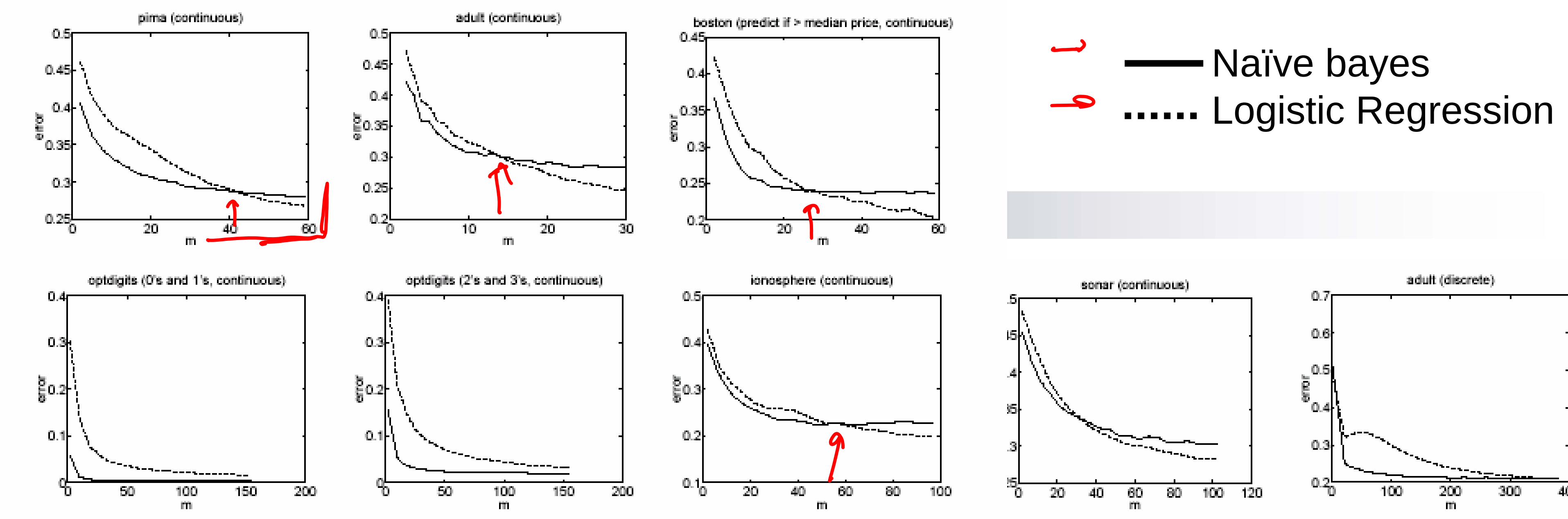
- Asymptotic comparison (# training examples → infinity)
  - when model correct indep assumptions
    - GNB, LR produce identical classifiers
  - when model incorrect
    - LR is less biased – does not assume conditional independence
      - **therefore LR expected to outperform GNB**

G. Naïve Bayes vs. Logistic Regression 2

[Ng & Jordan, 2002]

- Generative and Discriminative classifiers

- Non-asymptotic analysis
  - convergence rate of parameter estimates, n = # of attributes in X
    - Size of training data to get close to infinite data solution
    - GNB needs $O(\log n)$ samples
    - LR needs $O(n)$ samples
  - **GNB converges more quickly to its (perhaps less helpful) asymptotic estimates**



Some experiments from UCI data sets

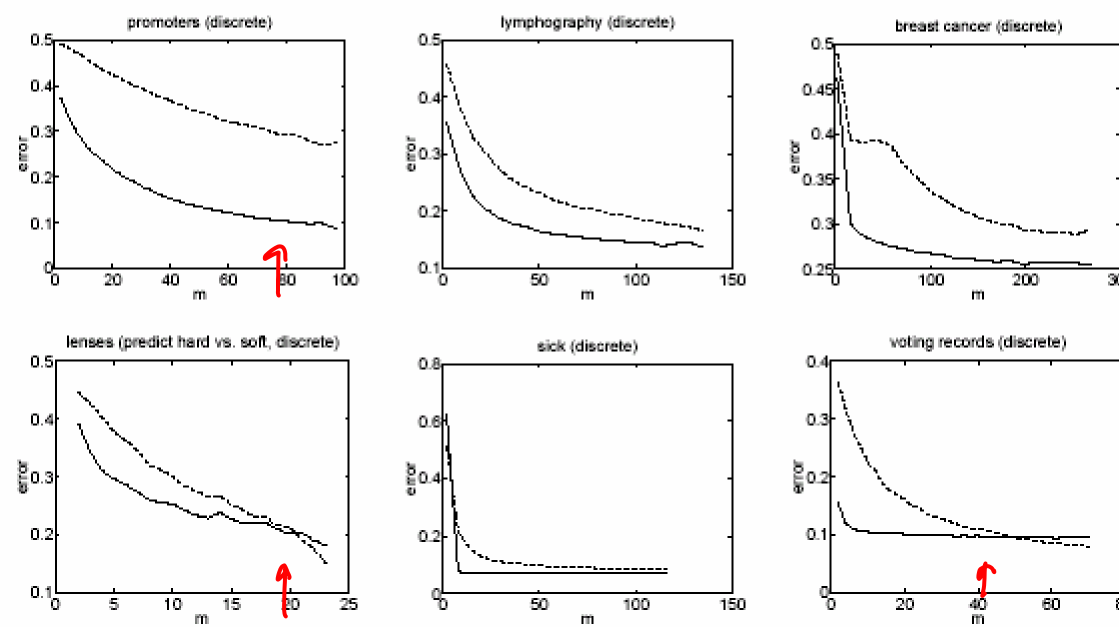


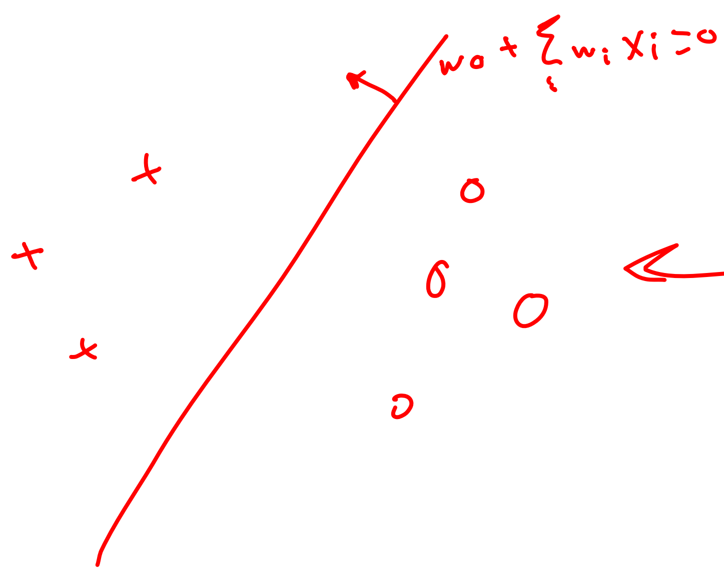
Figure 1: Results of 15 experiments on datasets from the UCI Machine Learnin repository. Plots are of generalization error vs. m (averaged over 1000 randon train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

What you should know about Logistic Regression (LR)

- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class $\rightarrow$ assumption on P(**X**|Y)
  - LR: Functional form of P(Y|**X**), no assumption on P(**X**|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave $\rightarrow$ global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization
- Convergence rates
  - GNB (usually) needs more data
  - LR (usually) gets to better solutions in the limit

Linear separability

- 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

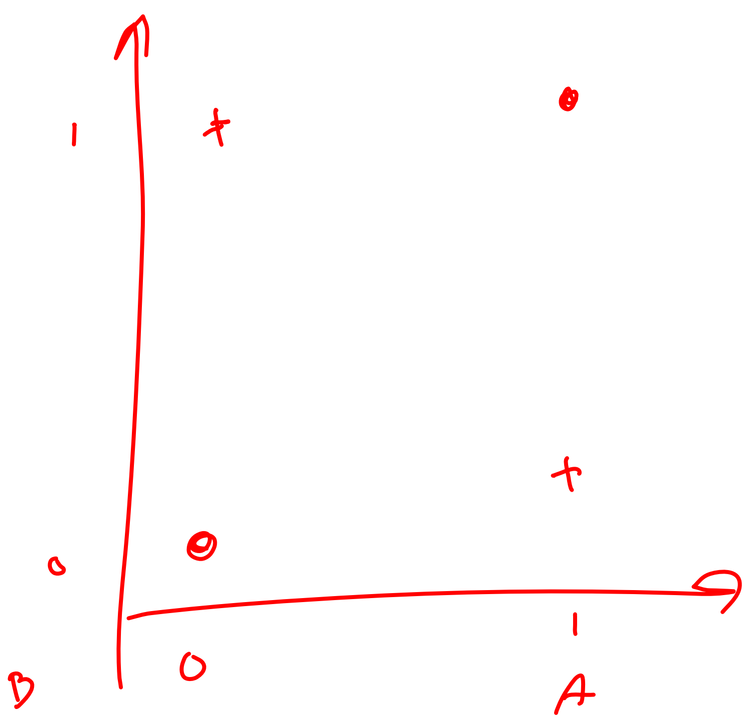


Not linearly separable data

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

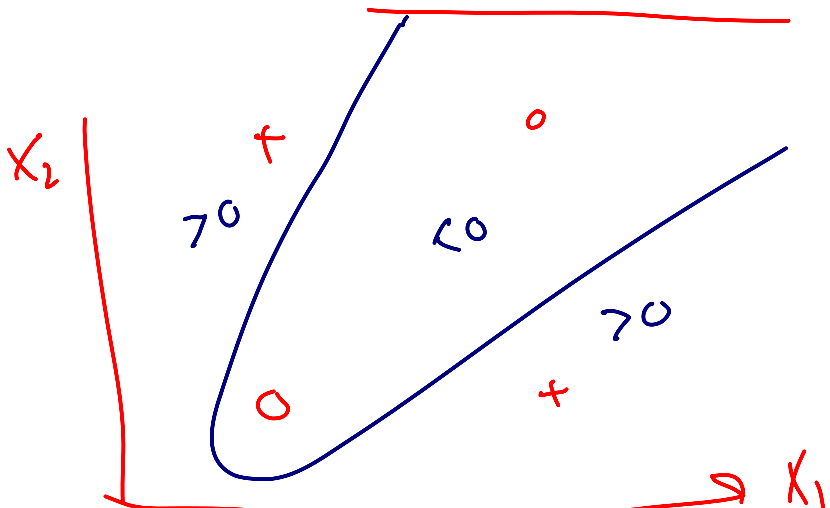
A xor B

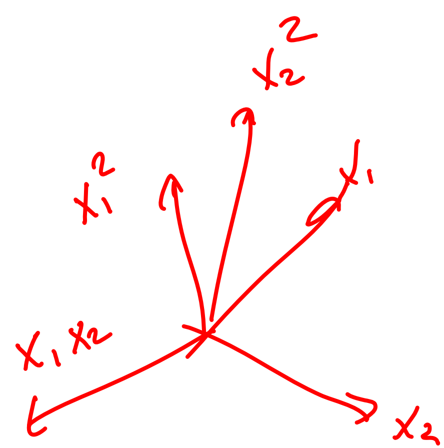
no hyperplane!



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 $h_{\mathbf{w}}(\mathbf{x})$ 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 **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

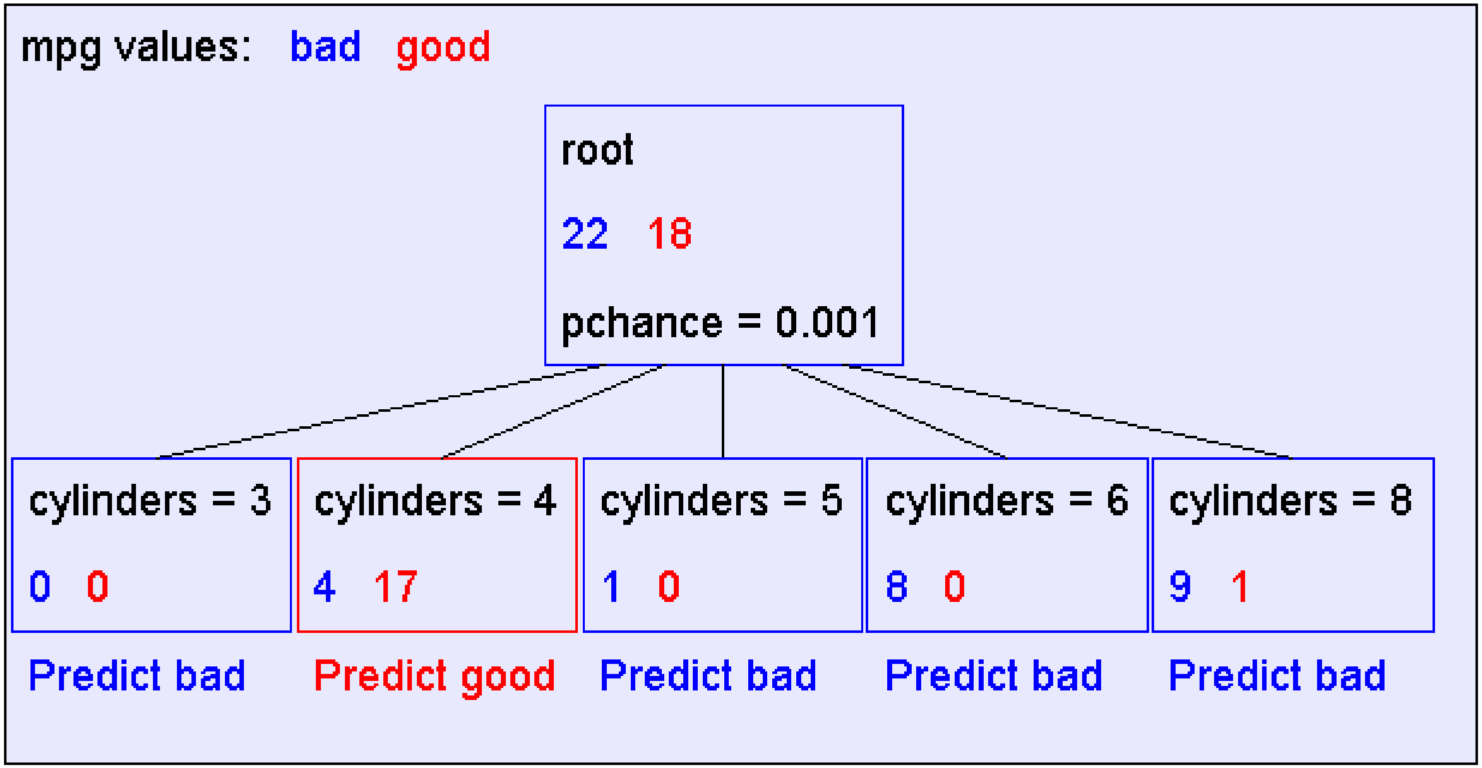
Suppose we want to predict MPG

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	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

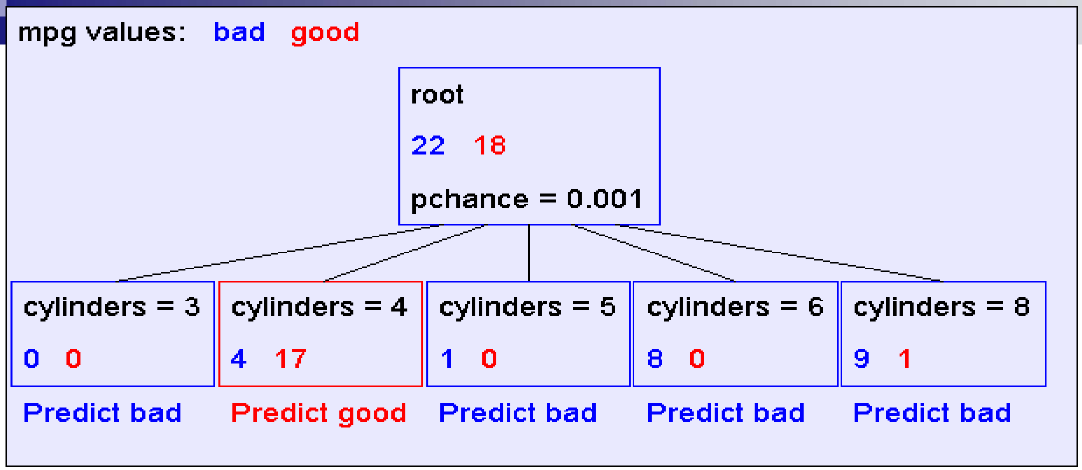
40 Records

From the UCI repository (thanks to Ross Quinlan)

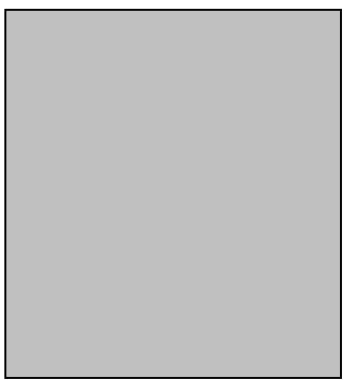
A Decision Stump



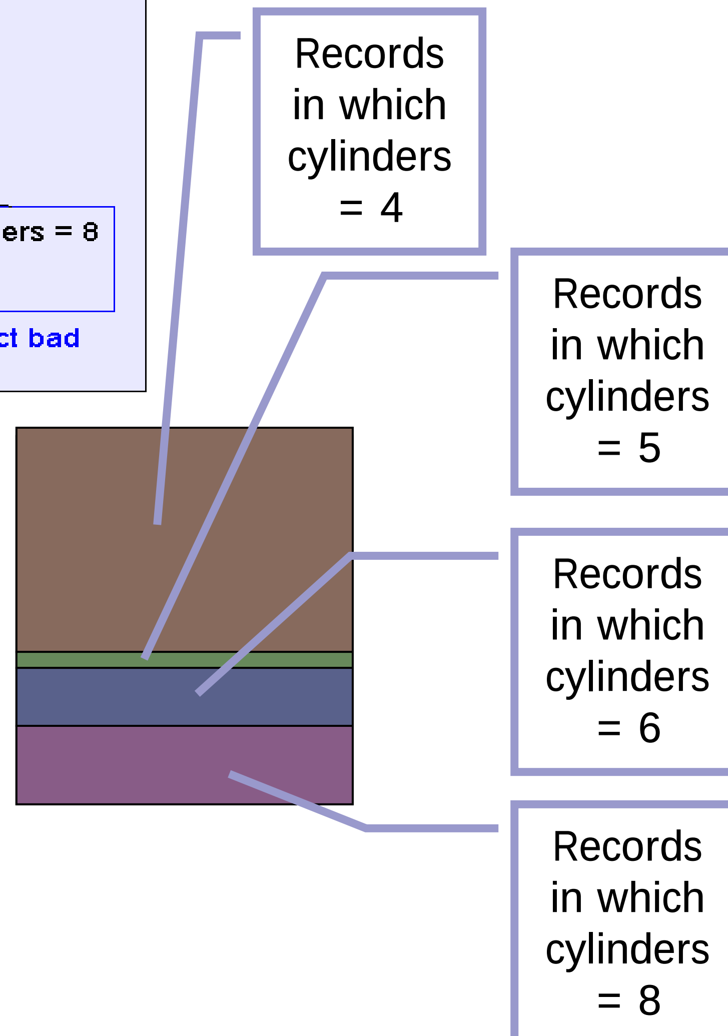
Recursion Step



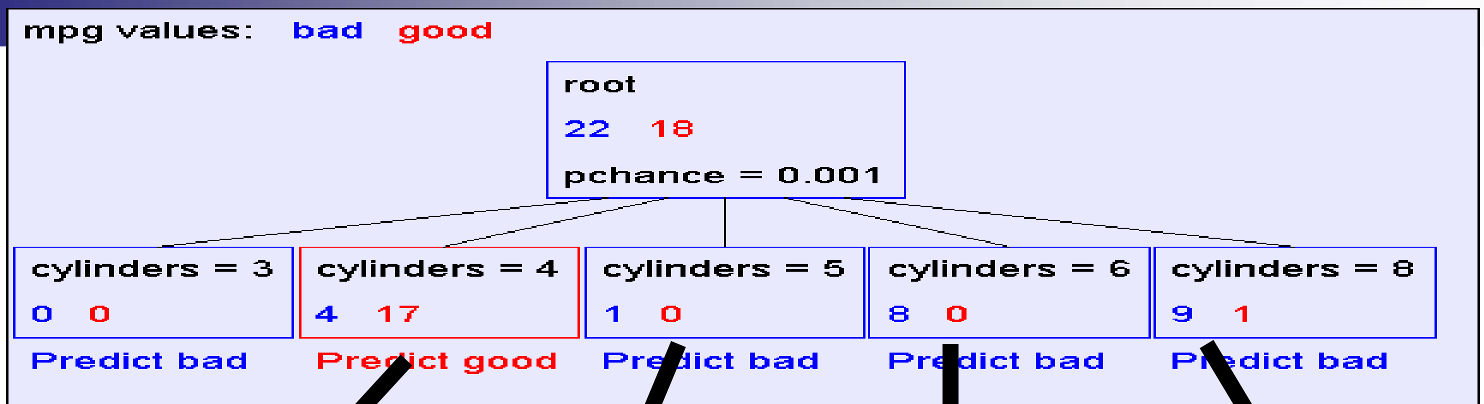
Take the Original Dataset..



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



Recursion Step



Build tree from These records..

Build tree from These records..

Build tree from These records..

Build tree from These records..

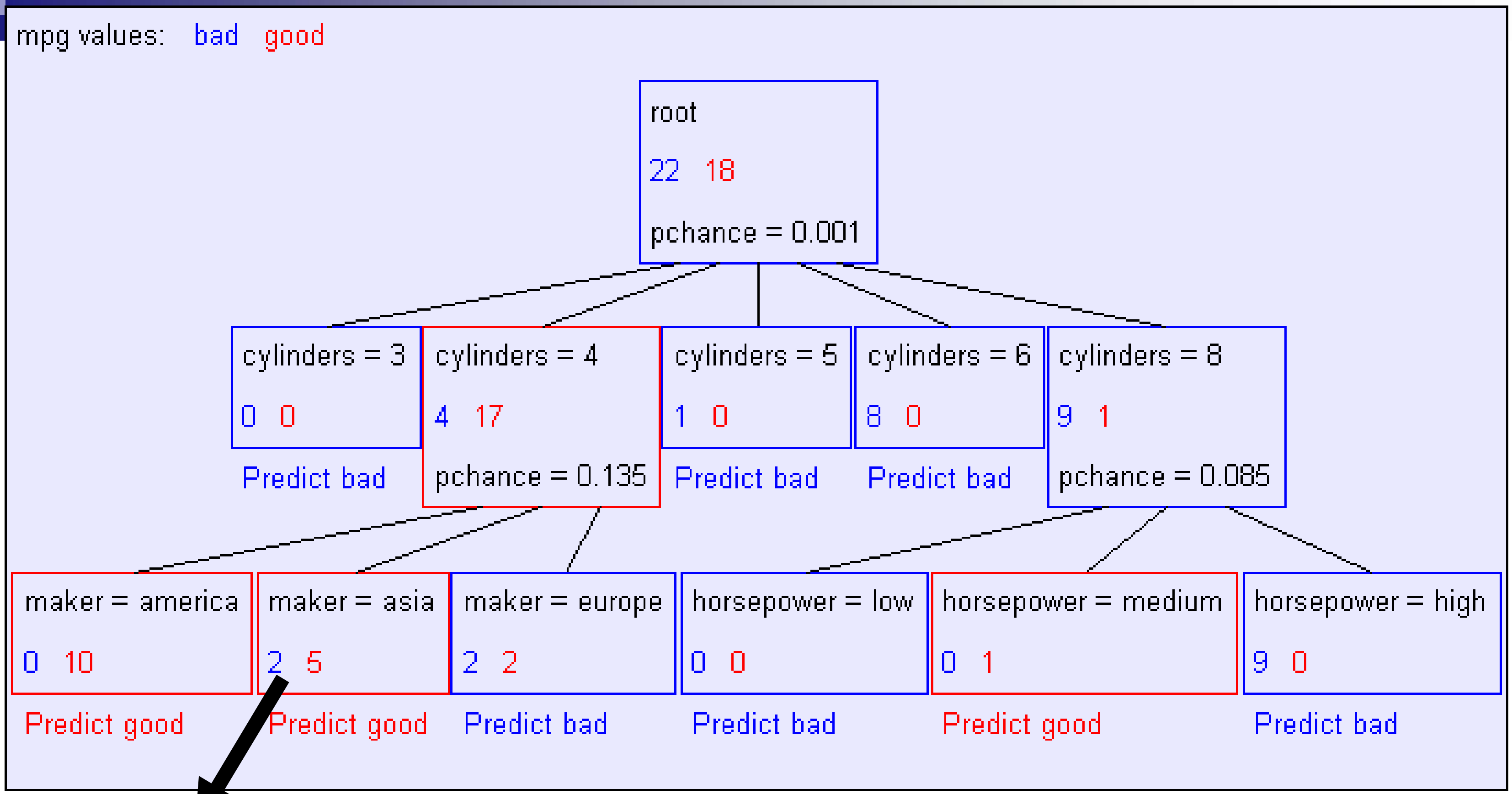
Records in which cylinders = 4

Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

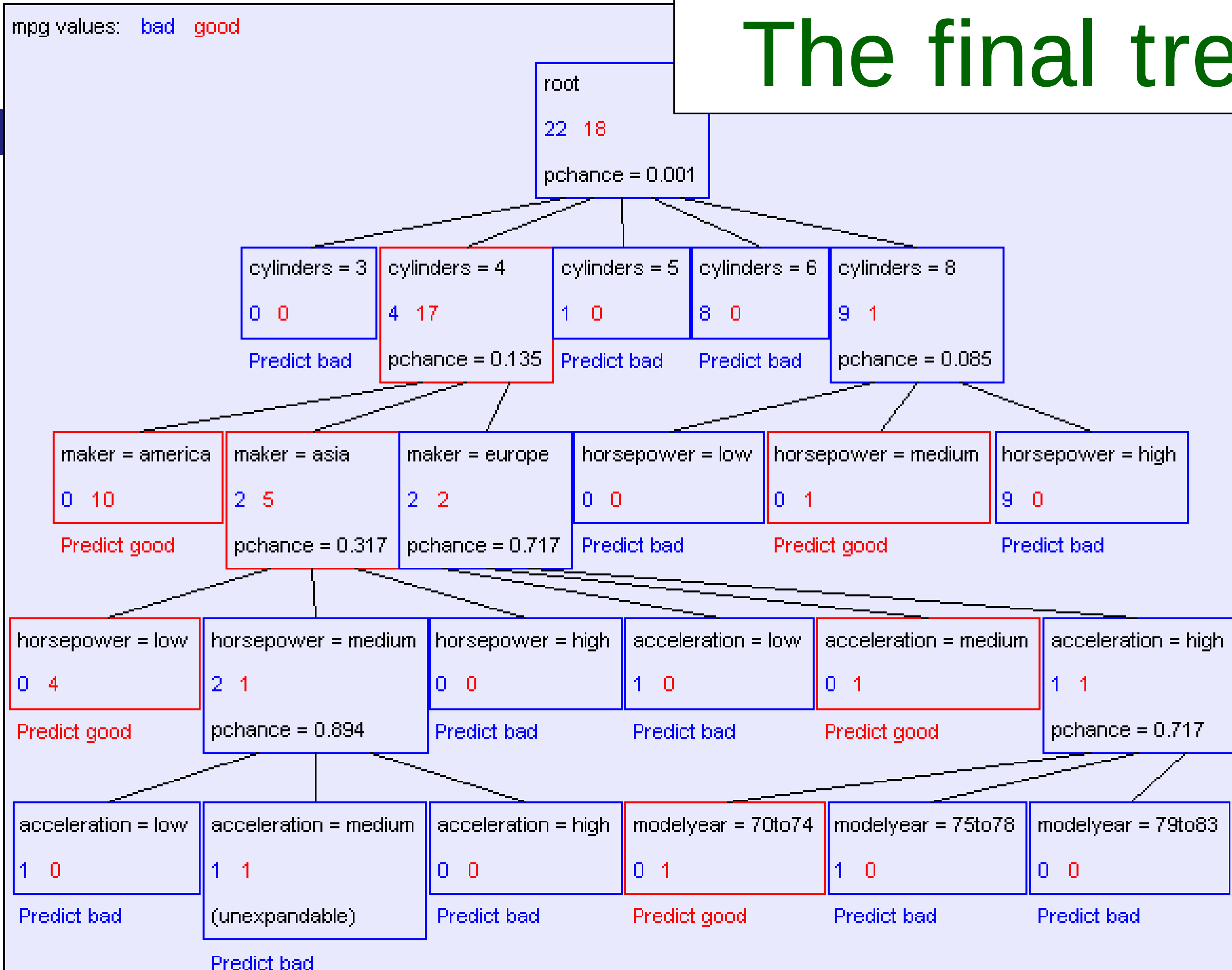
Second level of tree



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

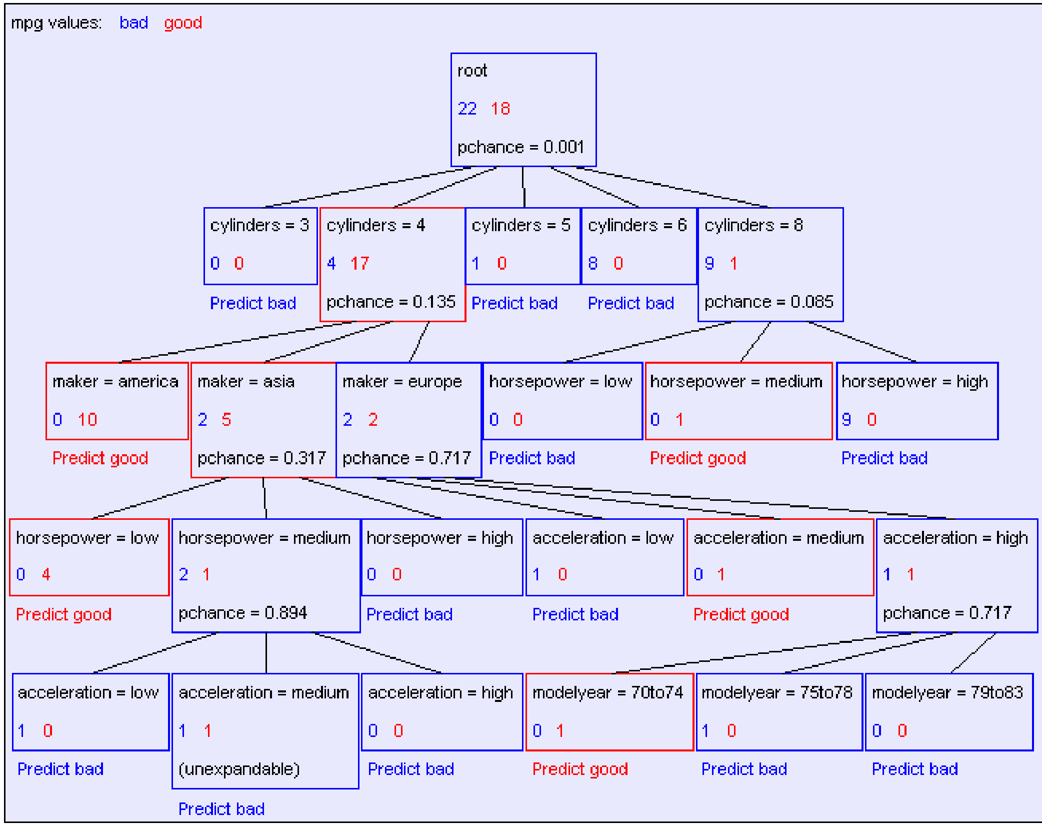
(Similar recursion in the other cases)

The final tree



Classification of a new example

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



Are all decision trees equal?

- 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))

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

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

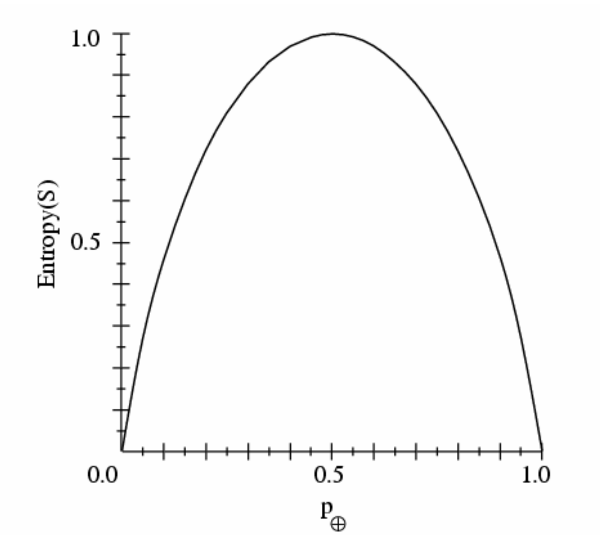
Entropy

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)



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
  - 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 $IG(X) = H(Y) - H(Y \mid X)$

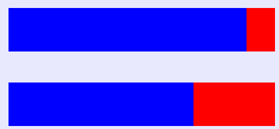
Learning decision trees

- 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

Information Gain Example

wealth values: poor rich

gender	Female	14423	1769	H(wealth | gender = Female) = 0.497654
	Male	22732	9918	H(wealth | gender = Male) = 0.885847

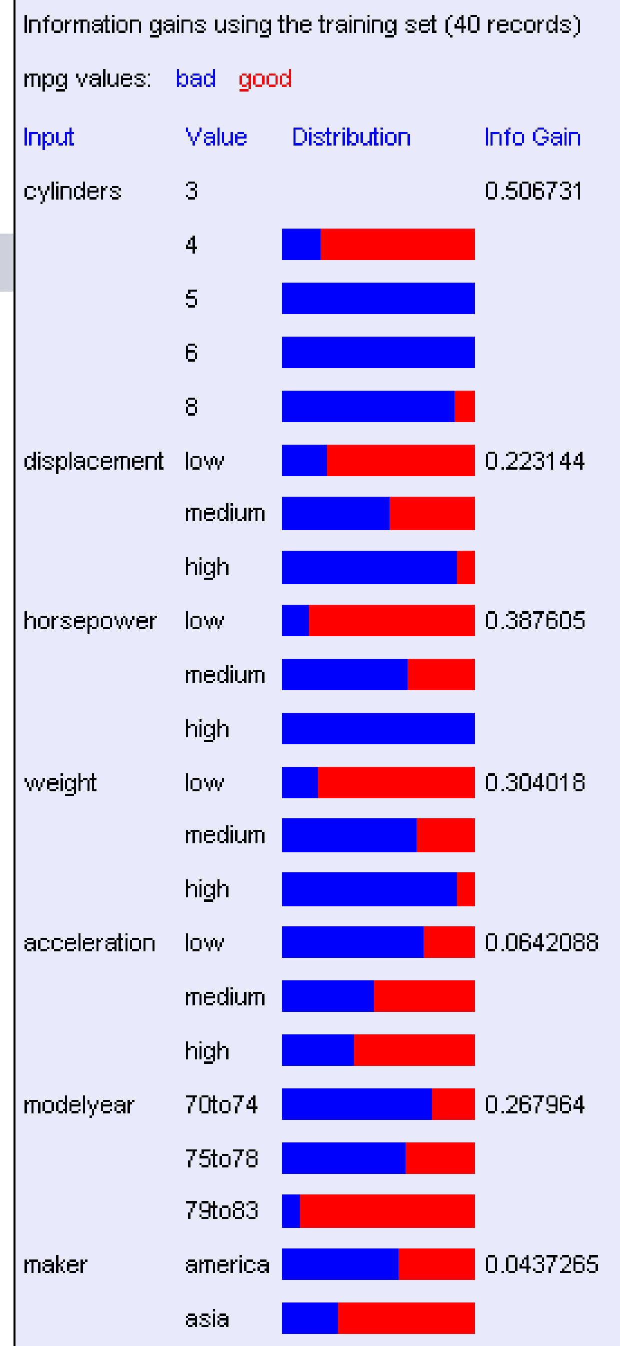


H(wealth) = 0.793844 H(wealth|gender) = 0.757154

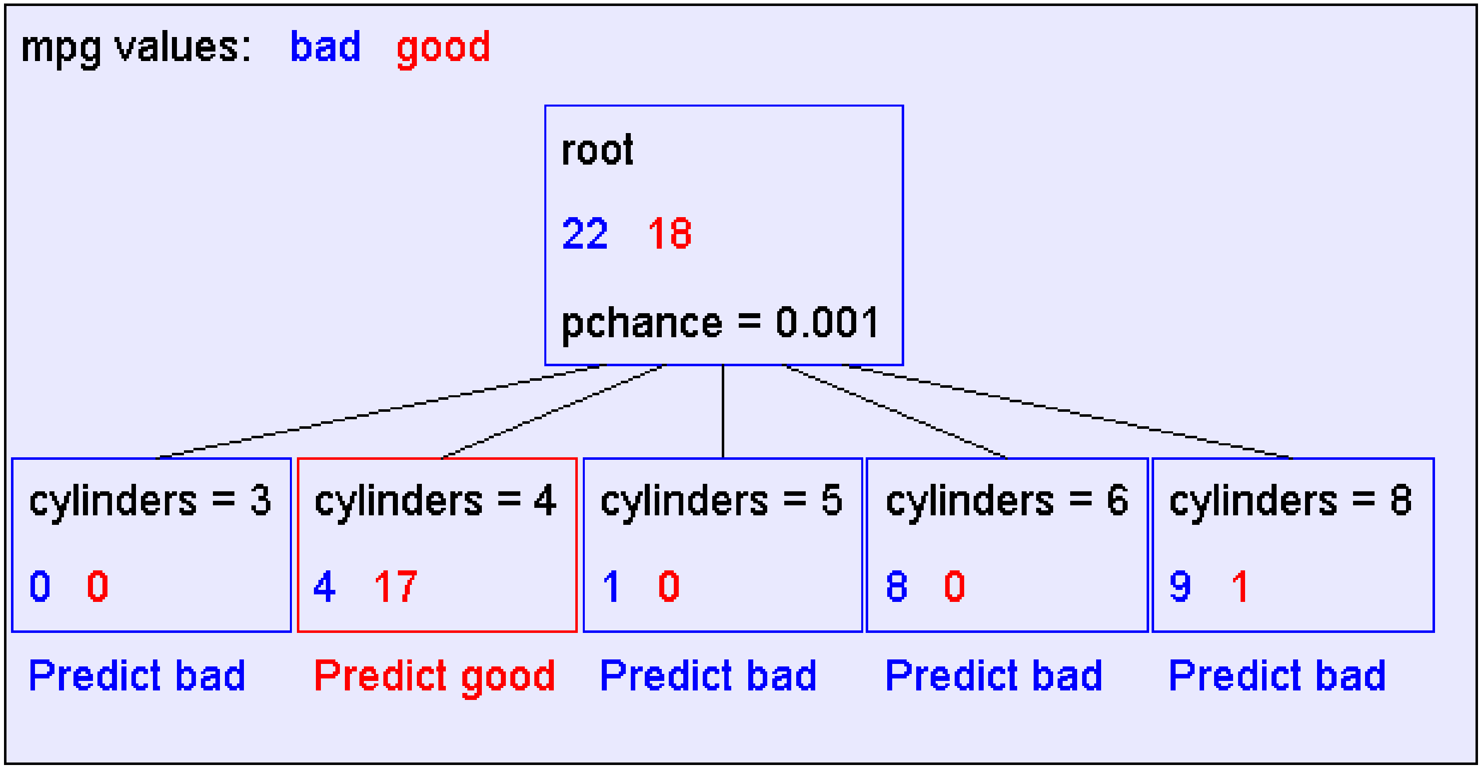
IG(wealth|gender) = 0.0366896

Suppose we want to predict MPG

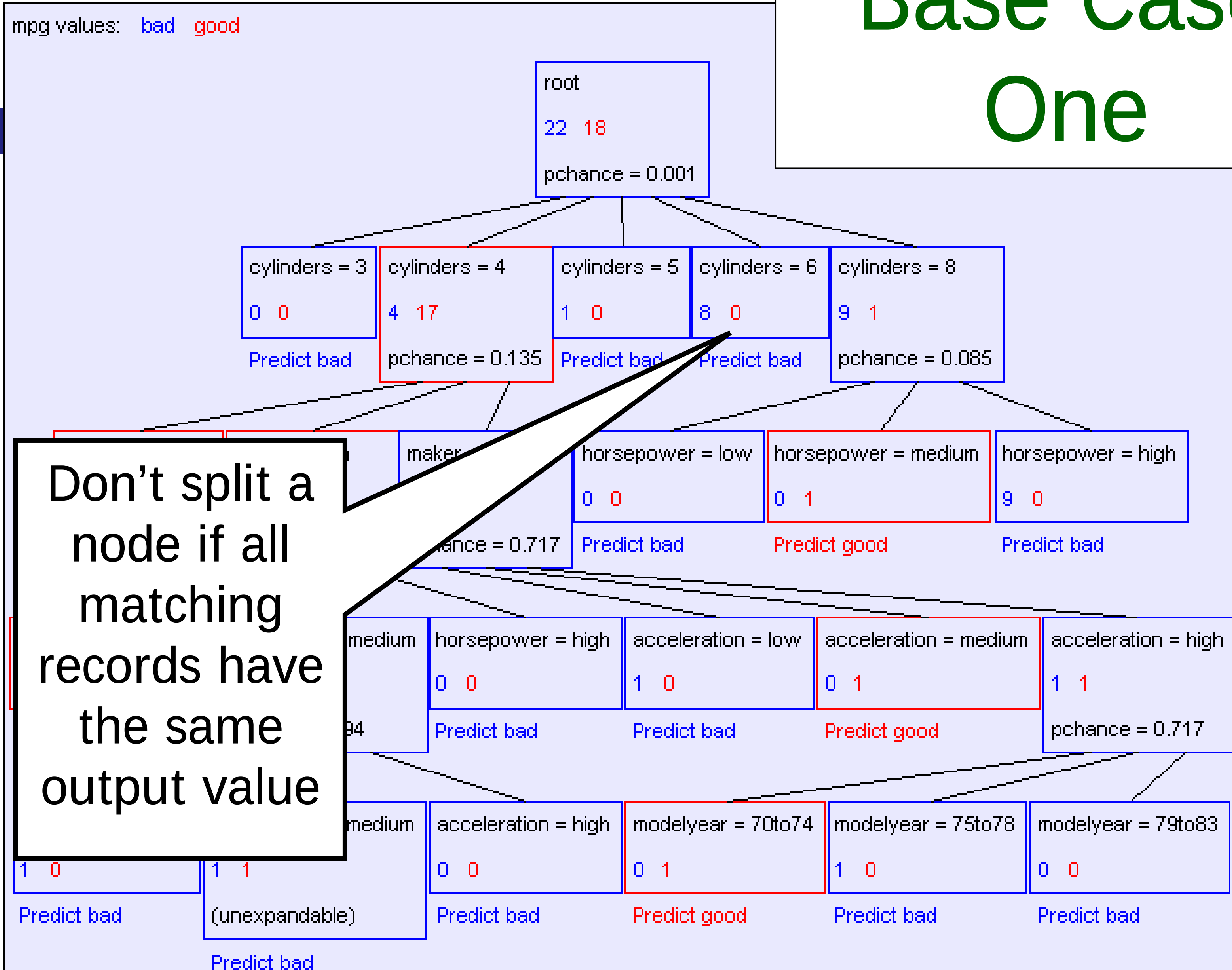
Look at all the information gains…



A Decision Stump

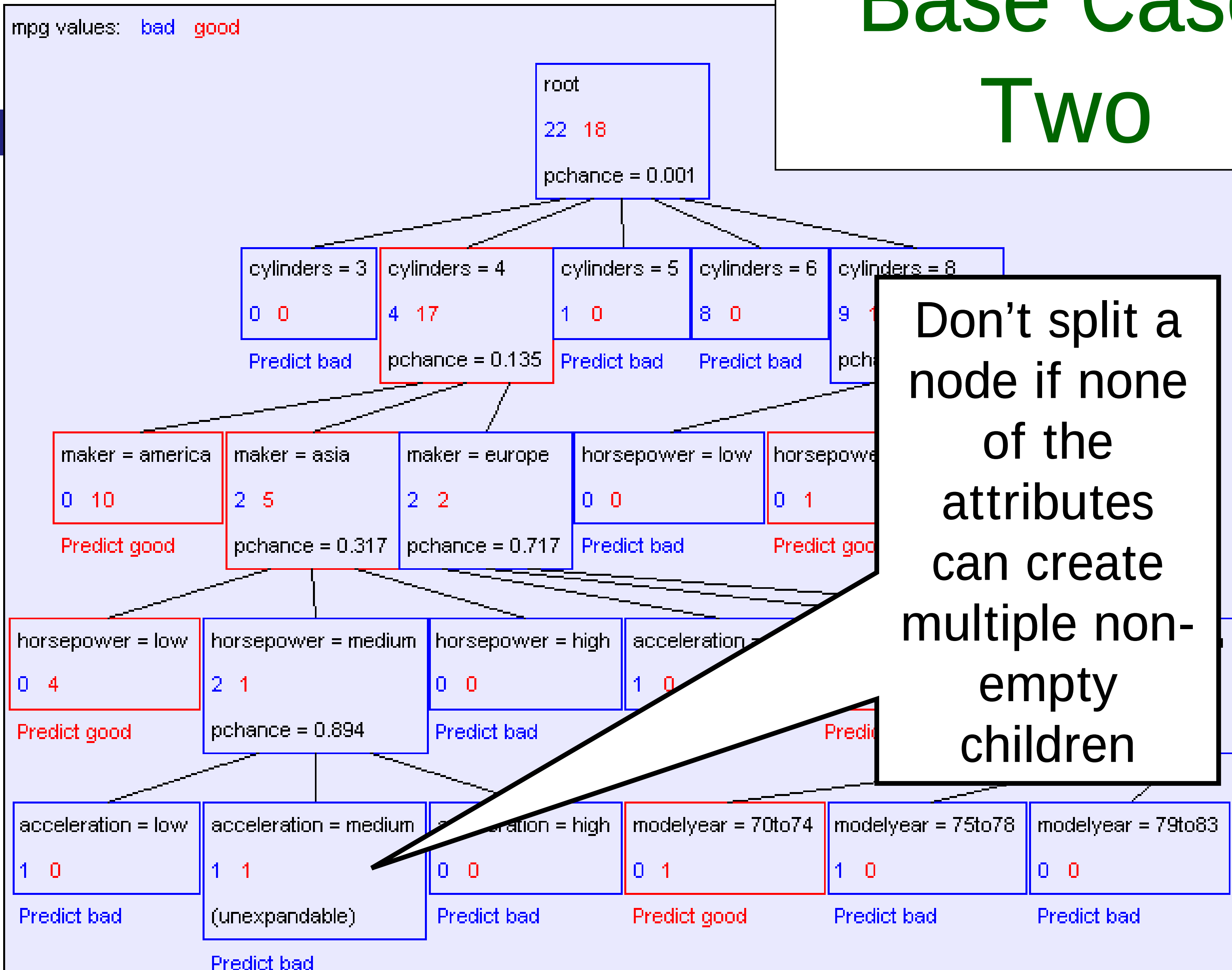


Base Case One

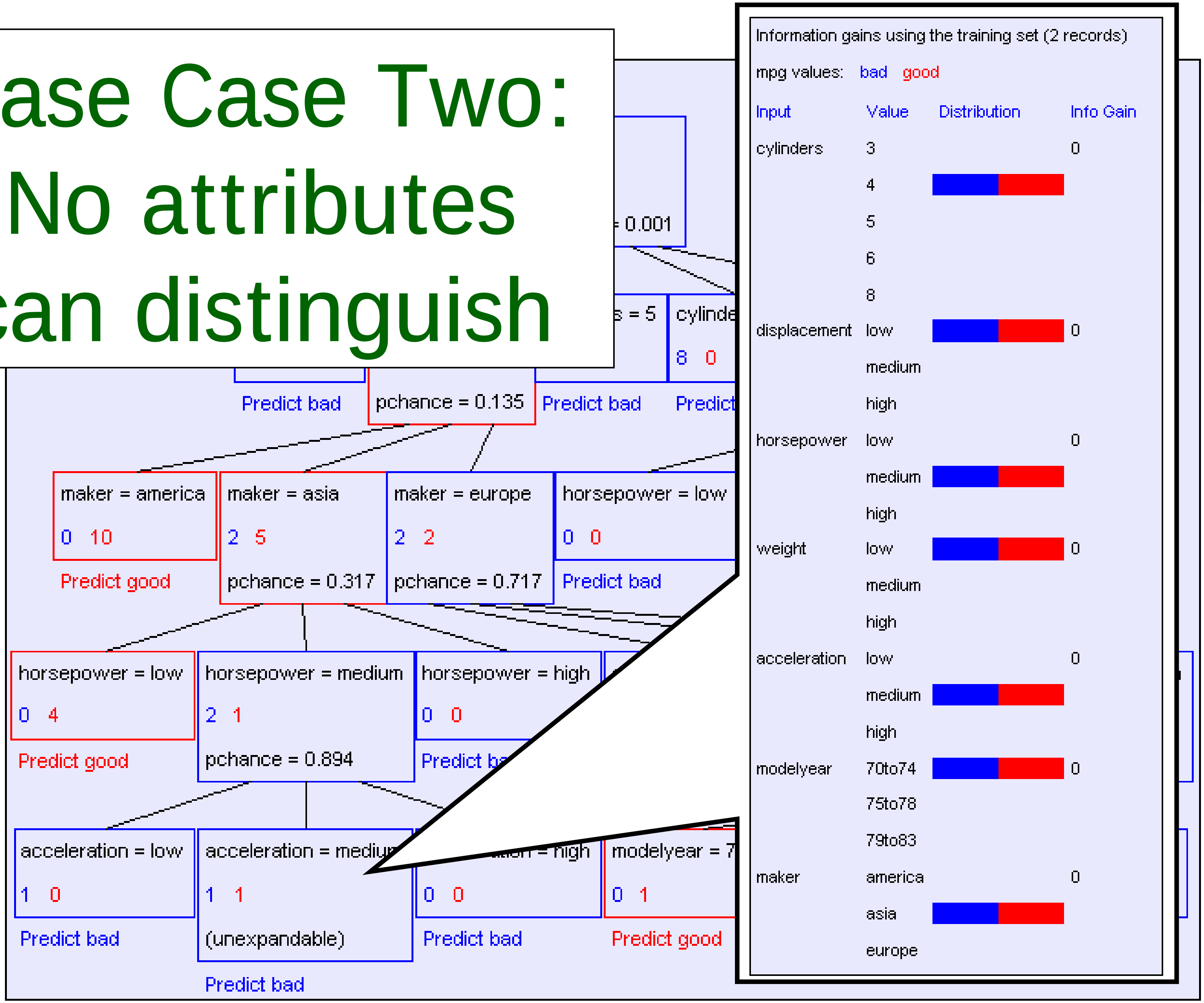


Don't split a node if all matching records have the same output value

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

Proposed Base Case 3:

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

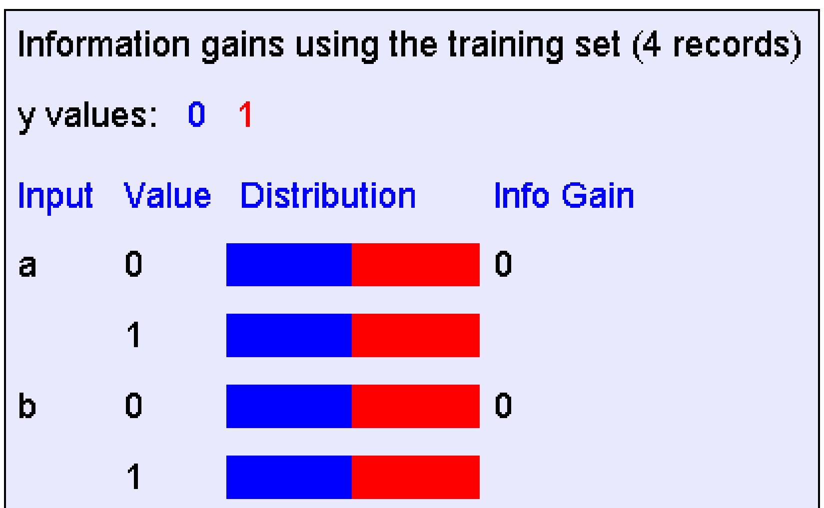
- *Is this a good idea?*

The problem with Base Case 3

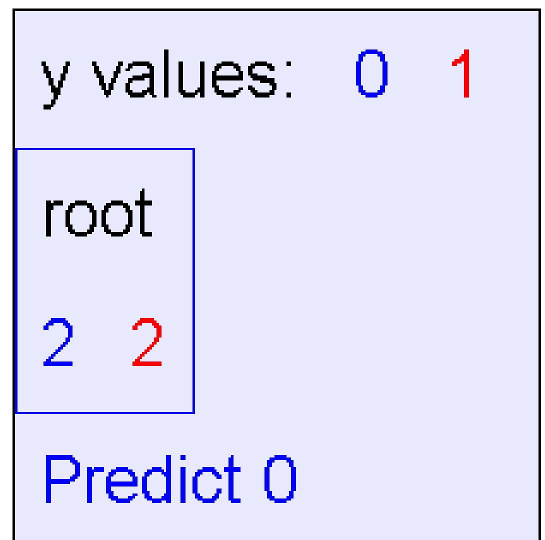
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



The resulting decision tree:

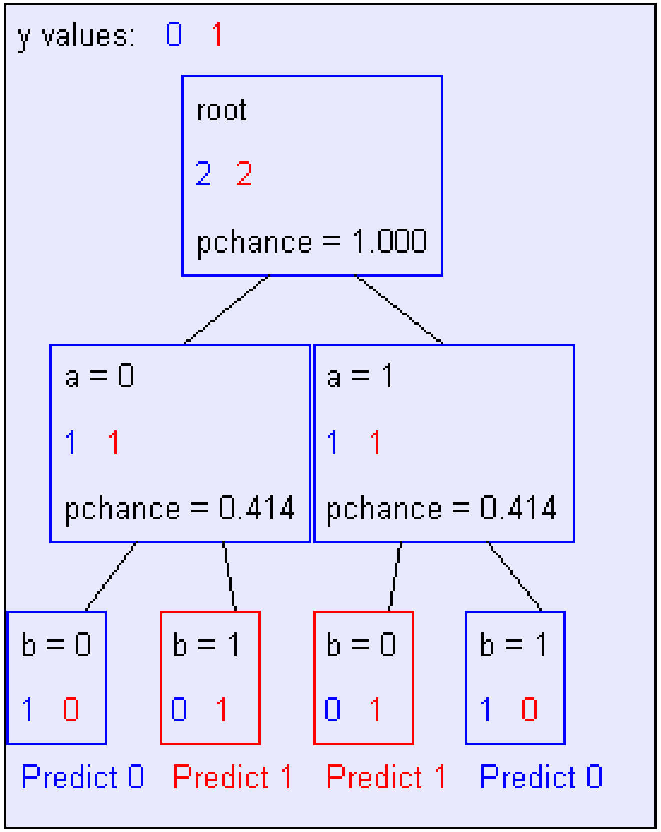


If we omit Base Case 3:

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.

Real-Valued inputs

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

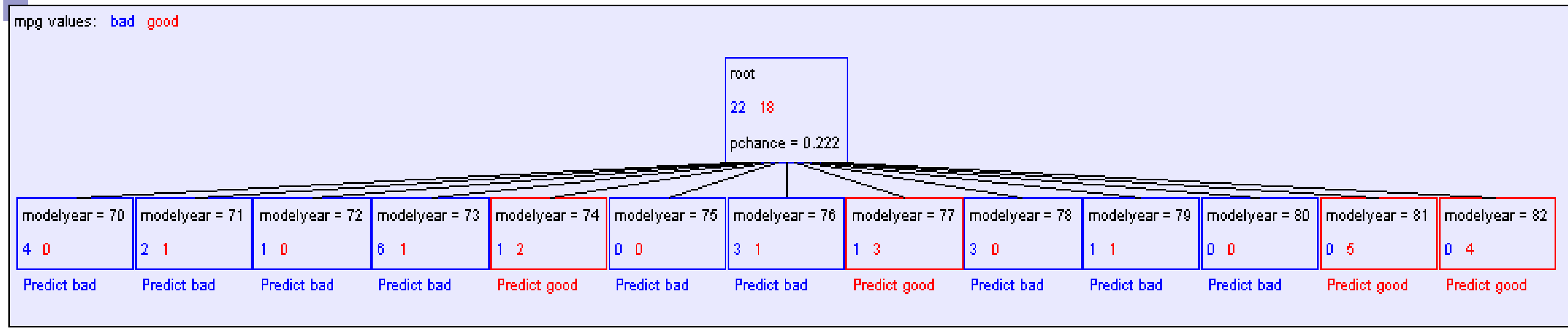
mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	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: $X \geq t$

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 $\{x_1,\ldots,x_m\}$
  - 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

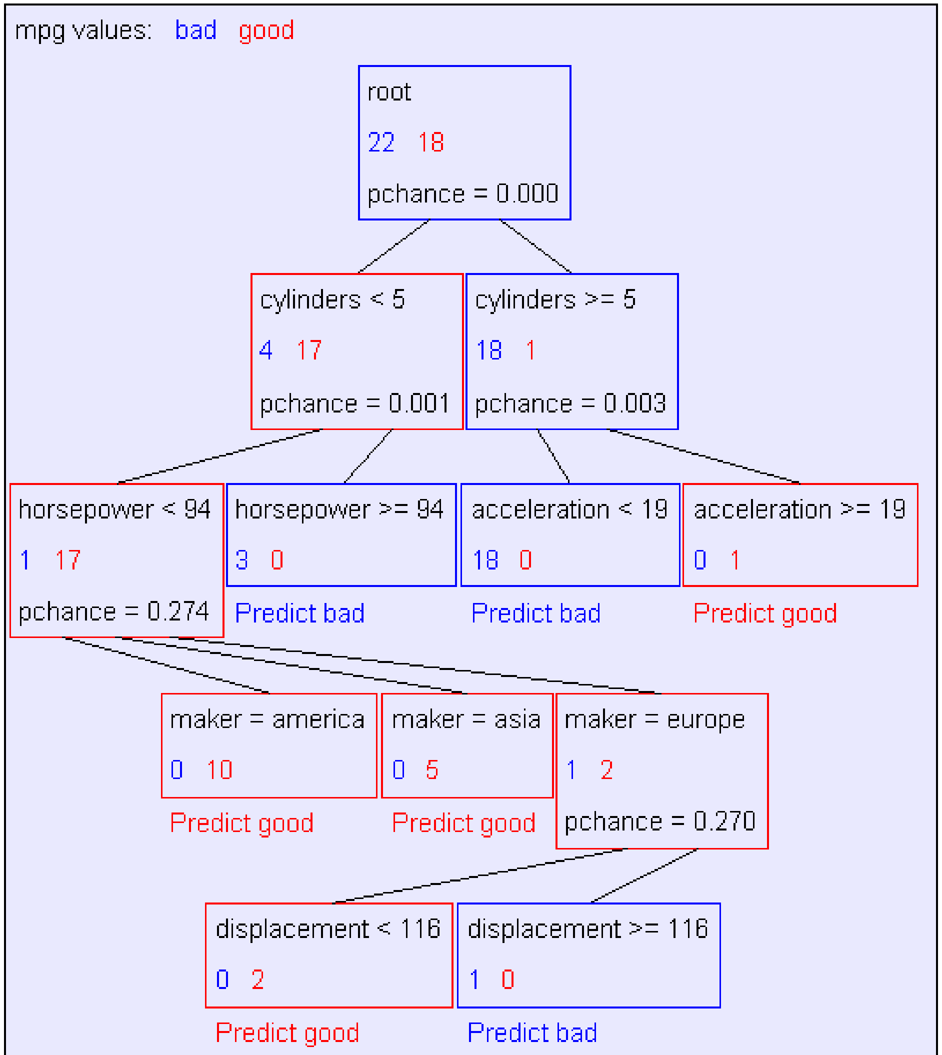
Example with MPG

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
- It's possible to get in trouble with overfitting (more next lecture)

Acknowledgements

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