Machine Learning as Search & as Continuous Optimization

CSE 573

DANIEL WELD

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

Improved by

- Carlos Guestrin, Luke Zettlemoyer, Dan Weld

Logistics

PS1 due Thurs 1/19

PS2 due Thurs 1/26

Machine Learning

Study of algorithms that
improve their <u>performance</u>
at some <u>task</u>
with <u>experience</u>

Space of ML Problems

Type of Supervision

(eg, Experience, Feedback)

What is Being Learned?

	Labeled Examples	**Reward**	**Examples w/o labels**
Discrete Function	Classification		Clustering
Continuous Function	Regression		
Policy	Apprenticeship Learning	Reinforcement Learning	

Classification

from data to discrete classes

Task: Predicting class membership (eg spam or not?)

Output = F: messages → T/F

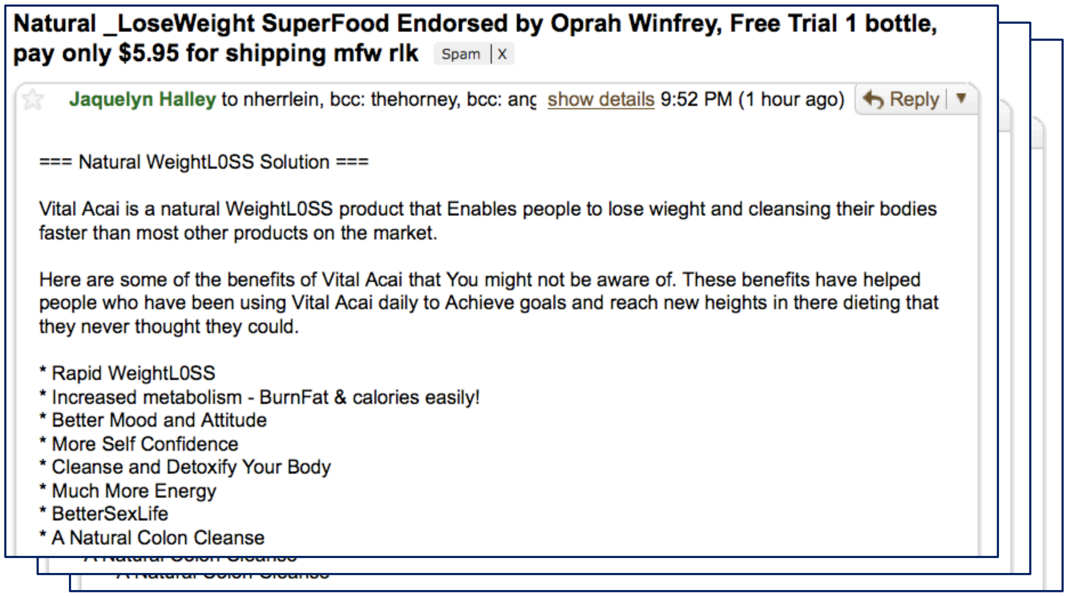
Performance: Accuracy of prediction

Learning as Function Approximation

Experience: Labeled examples

{ … <message$_i$, T>… }

Training Data for Spam Filtering



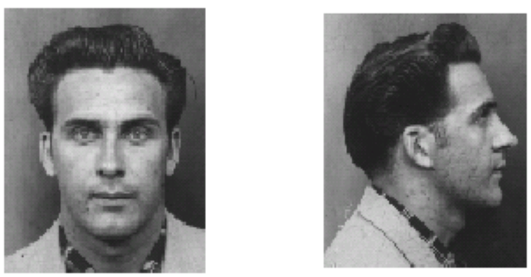
"Features"

a	…	homework	…	viagra	…	label
5		0		2		T
7		1		0		F

Weather prediction

Object detection

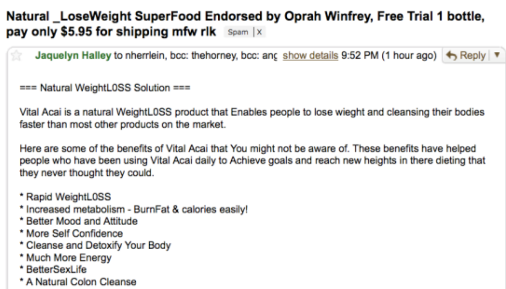
(Prof. H. Schneiderman)



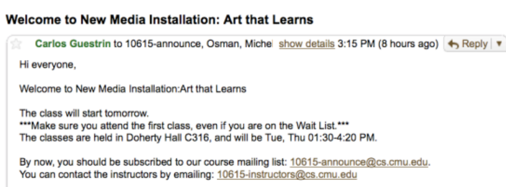
Example training images
for each orientation

The classification pipeline

Training

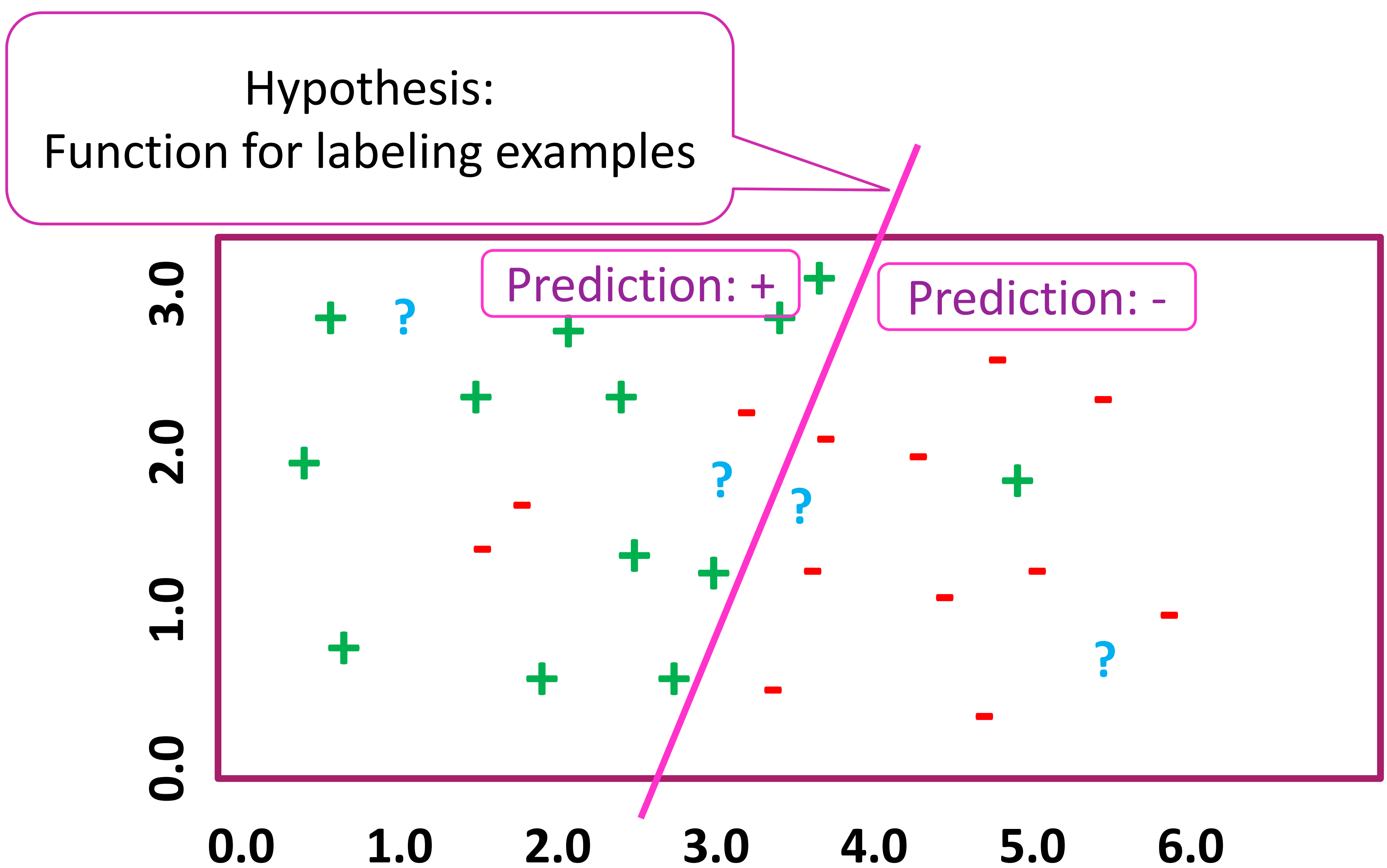


Testing



Measure accuracy here

Classifier


Hypothesis:
Function for labeling examples
Prediction: +
Prediction: -
3.0
2.0
1.0
0.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0

Key Concepts

Generalization

Hypotheses must ***generalize*** to correctly classify instances not in the training data.

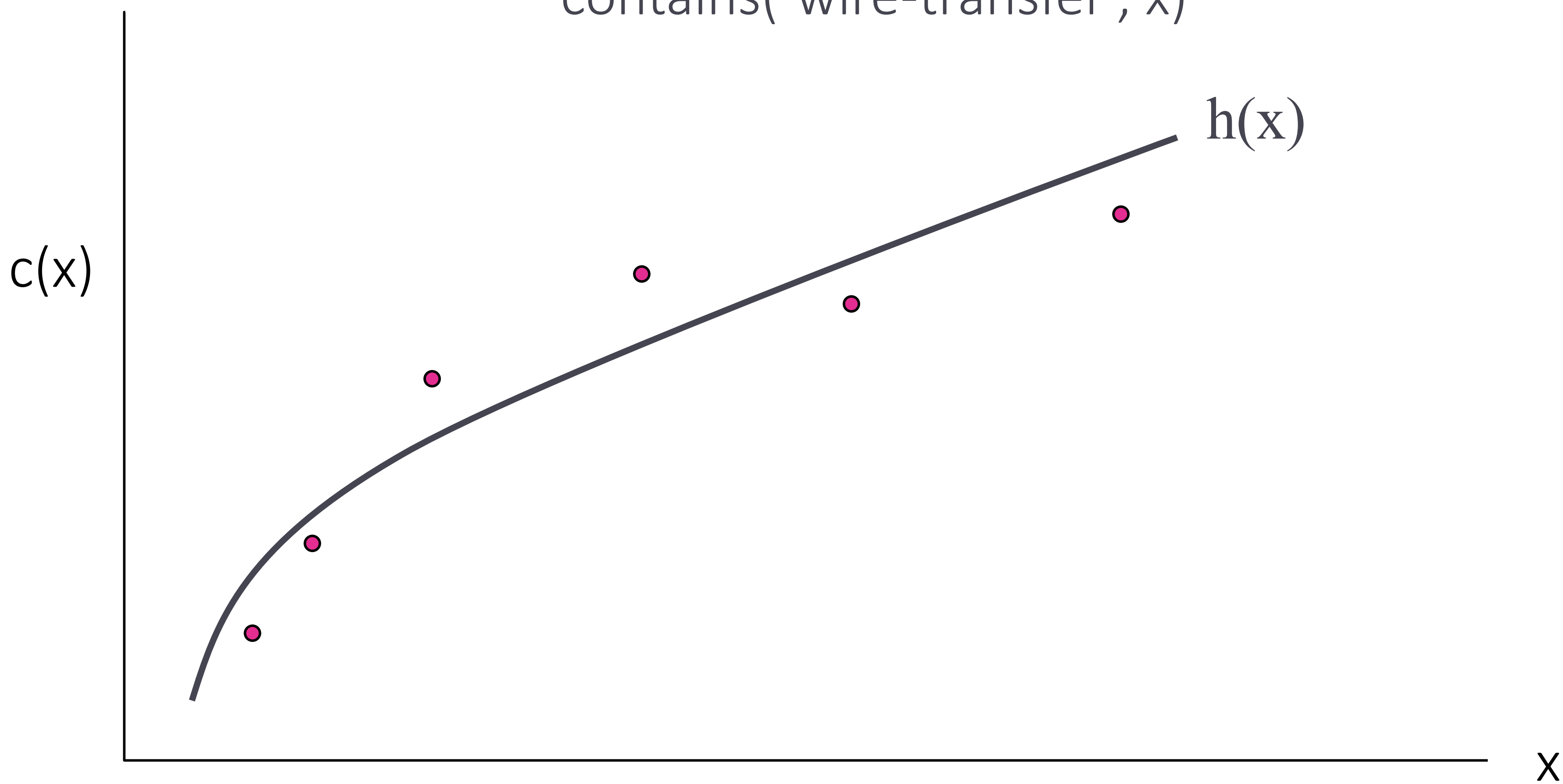
Simply memorizing training examples is a consistent hypothesis ***that does not generalize***.

ML = Function Approximation

May not be any perfect fit

Classification ~ discrete functions

h(x) = contains(`nigeria', x) $\wedge$
contains(`wire-transfer', x)



Why is Learning Possible?

Experience alone never justifies any conclusion about any unseen instance.

Learning occurs when

PREJUDICE meets DATA!

Frobnitz?

Bias

The nice word for prejudice is "bias".

- Different from "Bias" in statistics

What kind of hypotheses will you ***consider***?

- What is allowable ***range*** of functions you use when approximating?
- E.g., pure conjunctions, linear separators, …

What kind of hypotheses do you ***prefer***?

- E.g., simple with few parameters

"It is needless to do more when less will suffice"
– William of Occam,
died 1349 of the Black plague

ML as Optimization

Specify Preference Bias

- aka “Loss Function”

Solve using optimization

- Combinatorial
- Convex
- Linear
- Nasty

Overfitting

Hypothesis H is *overfit* when $\exists$ H' and

- H has ***smaller*** error on training examples, but
- H has ***bigger*** error on test examples

Overfitting

Hypothesis H is *overfit* when ∃ H' and

- H has ***smaller*** error on training examples, but
- H has ***bigger*** error on test examples

Causes of overfitting

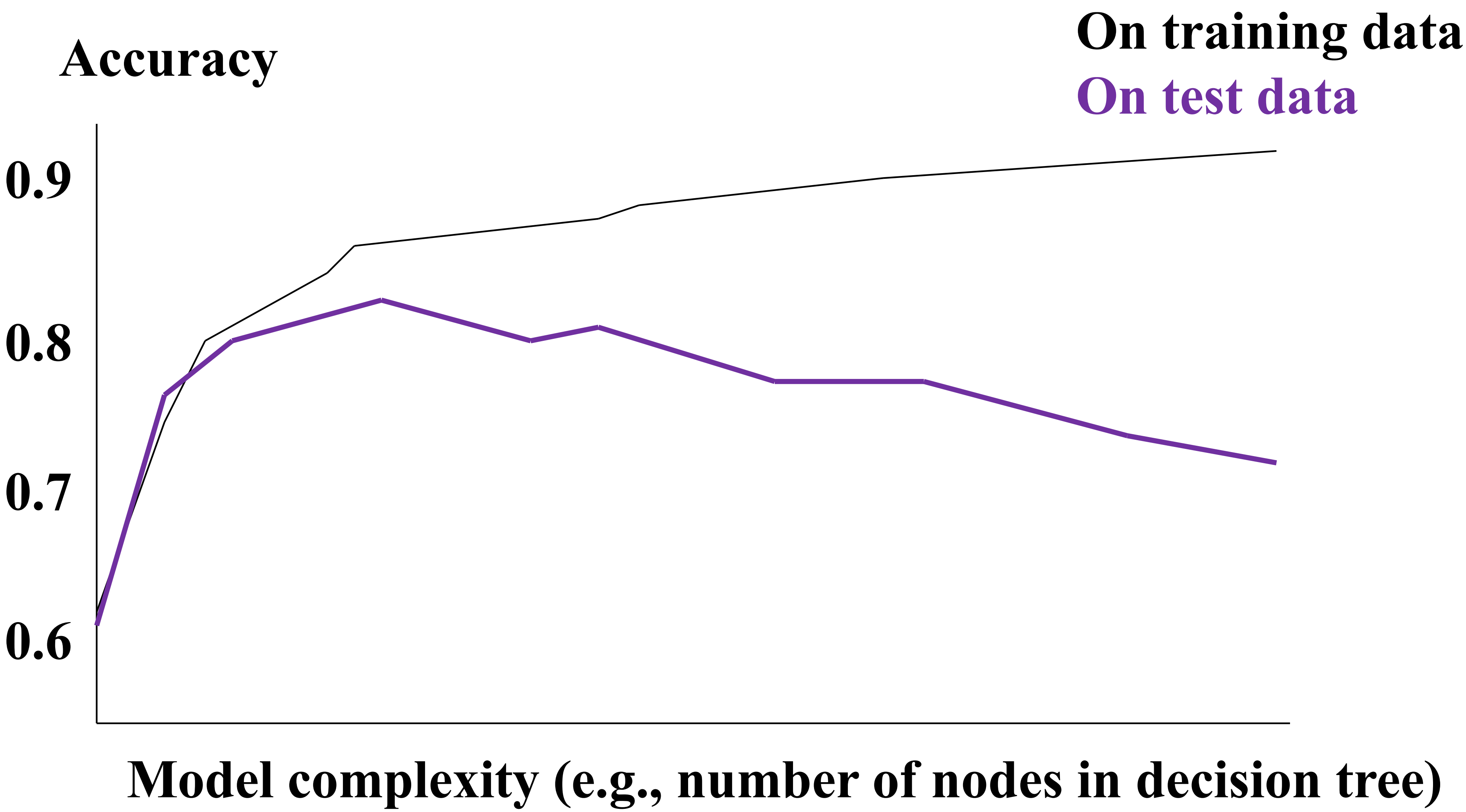
- Training set is too small
- Large number of features

Some solutions

- Validation set
- Regularization

Big Problem for ML

Overfitting



A learning problem: predict fuel efficiency

From the UCI repository (thanks to Ross Quinlan)

- 40 Records
- Discrete data (for now)
- 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

Y X

Need to find “Hypothesis”: $f : X \rightarrow Y$

How Represent Function?

$f\Big($

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
4	low	low	low	high	75to78	asia

$\Big) \rightarrow$

mpg
good

General Propositional Logic?

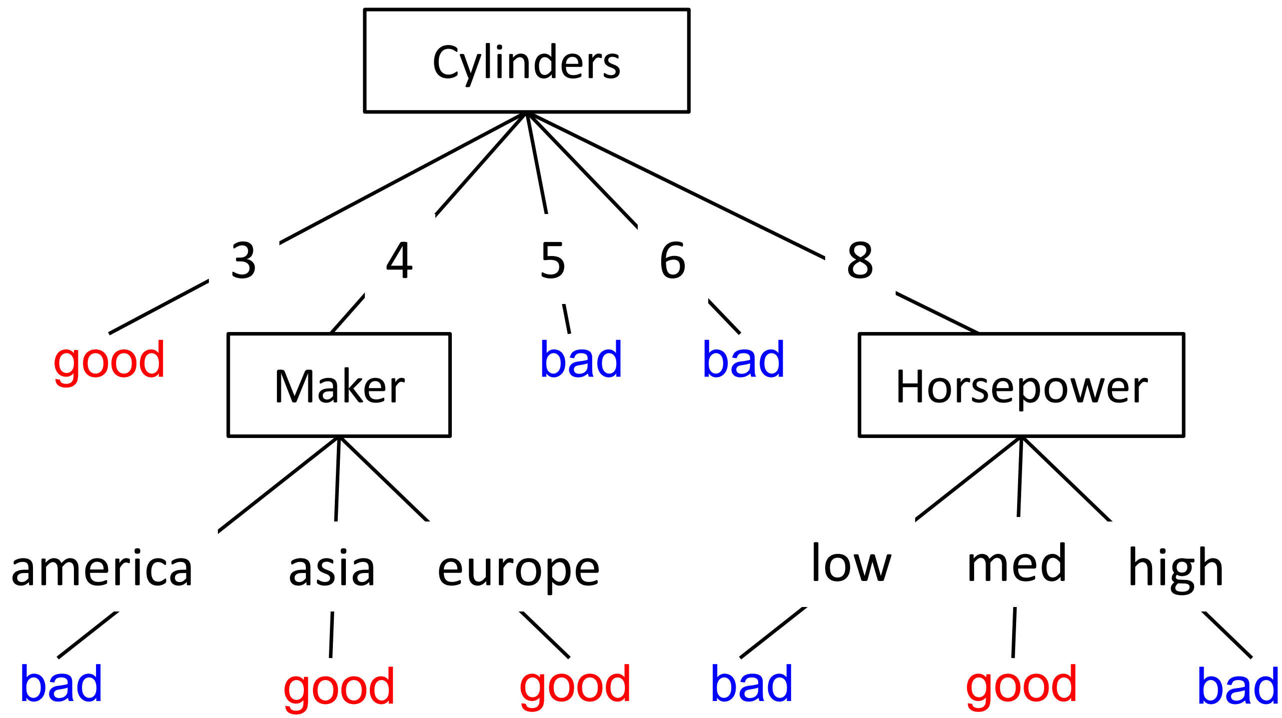
maker=asia $\vee$ weight=low

Need to find "Hypothesis": $f : X \rightarrow Y$

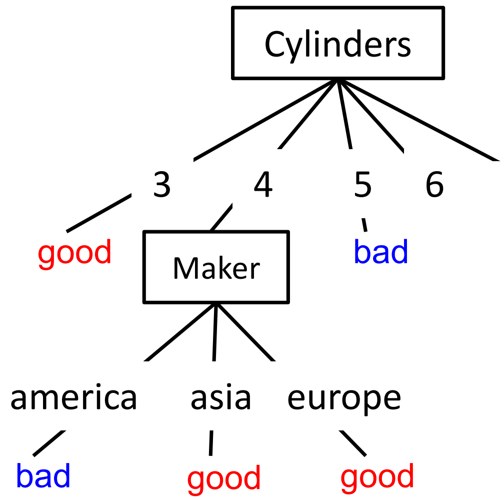
Hypotheses: decision trees $f : X \rightarrow Y$

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x?

  traverse the tree from root to leaf, output the labeled y



What functions can be represented?



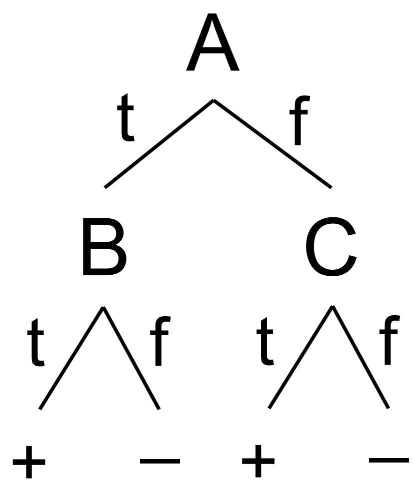
$$cyl=3 \vee (cyl=4 \wedge (maker=asia \vee maker=europe)) \vee \ldots$$

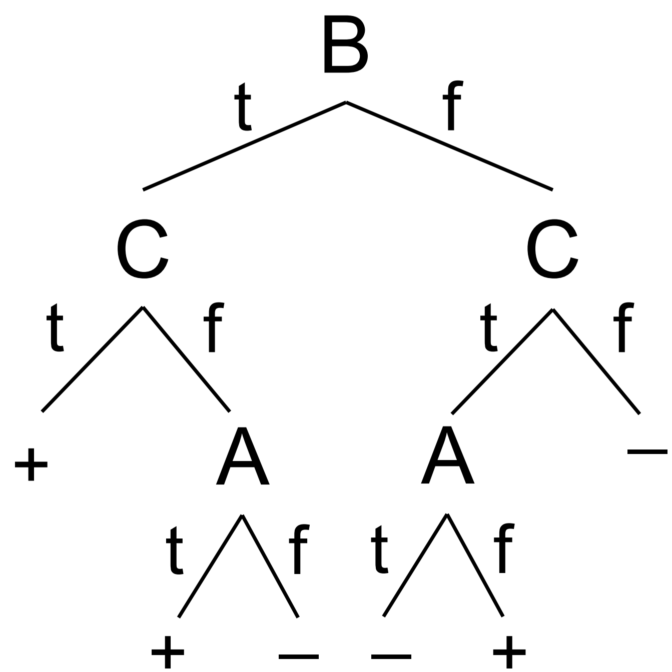
Are all decision trees equal?

Many trees can represent the same concept

But, not all trees will have the same size!

e.g., $\phi = (A \wedge B) \vee (\neg A \wedge C)$





How to find the best tree?

Learning decision trees is hard!!!

Finding the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

What to do?

Learning as Search

Nodes?

Operators?

Start State?

Goal?

Search Algorithm?

Heuristic?

The Starting Node: What is the Simplest Tree?

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

predict
mpg=bad

Is this a good tree?

[22+, 18-]

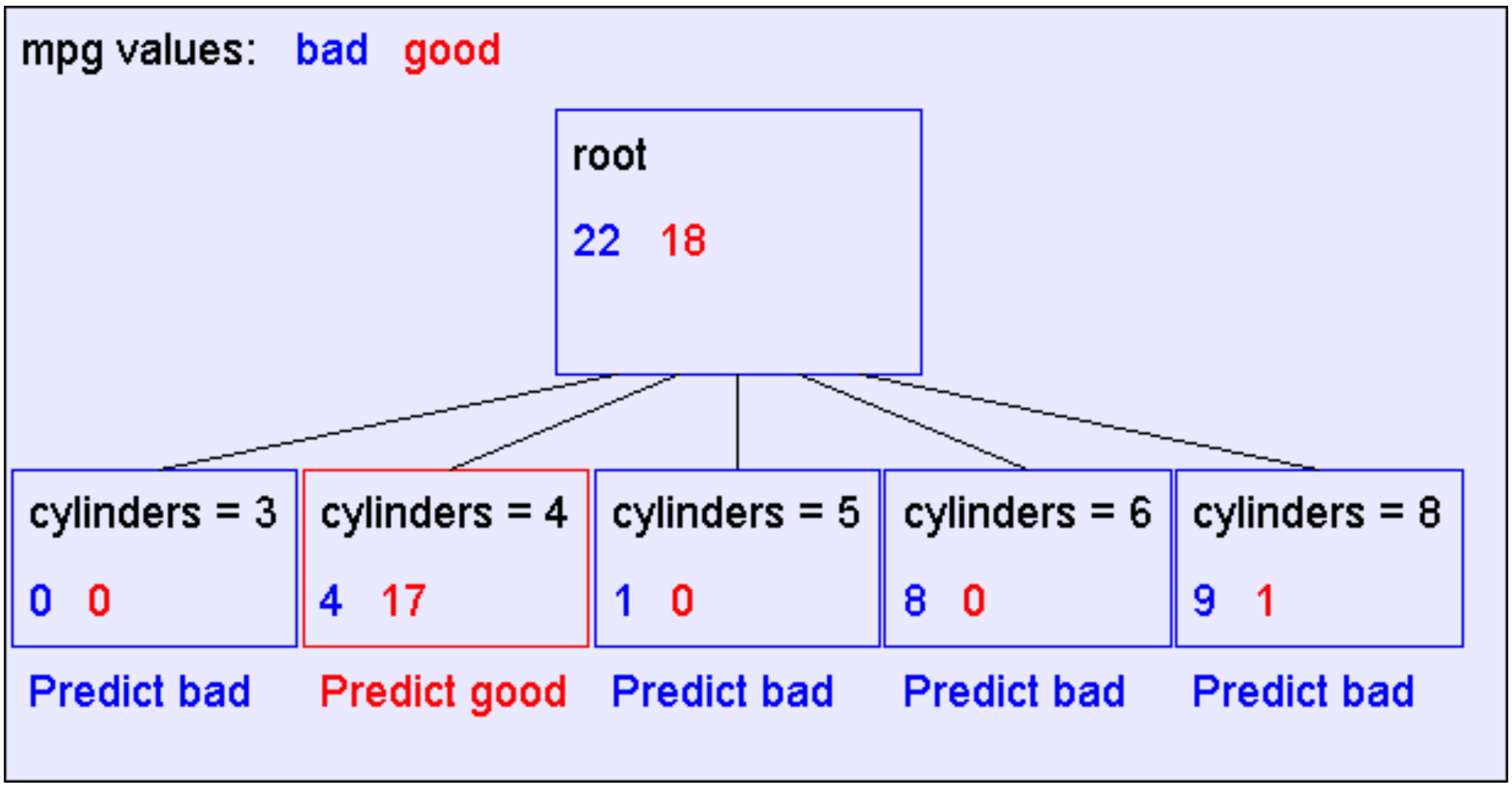
Means:

correct on 22 examples

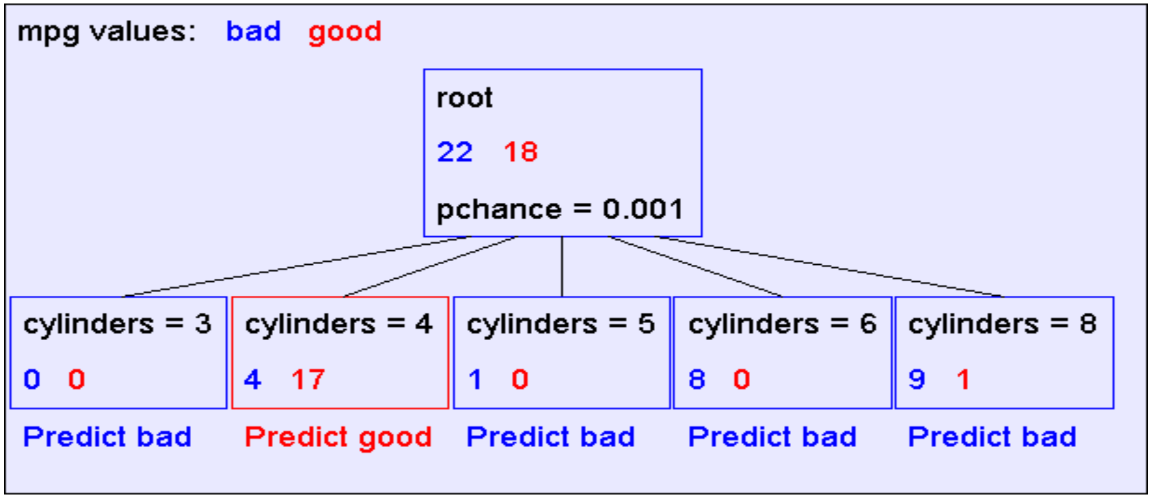
incorrect on 18 examples

Operators: Improving the Tree

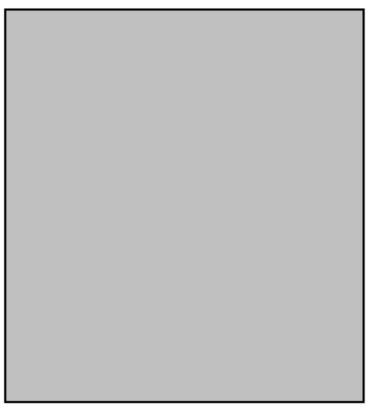
predict
mpg=bad



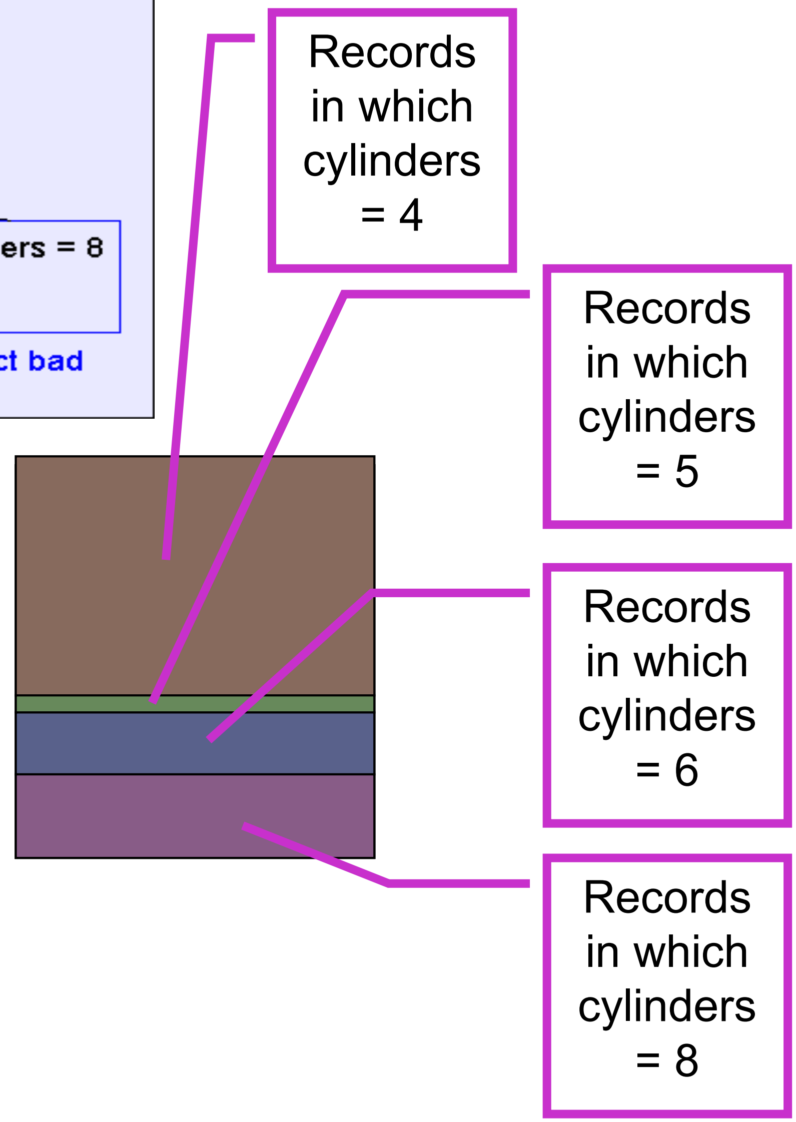
Recursive Step



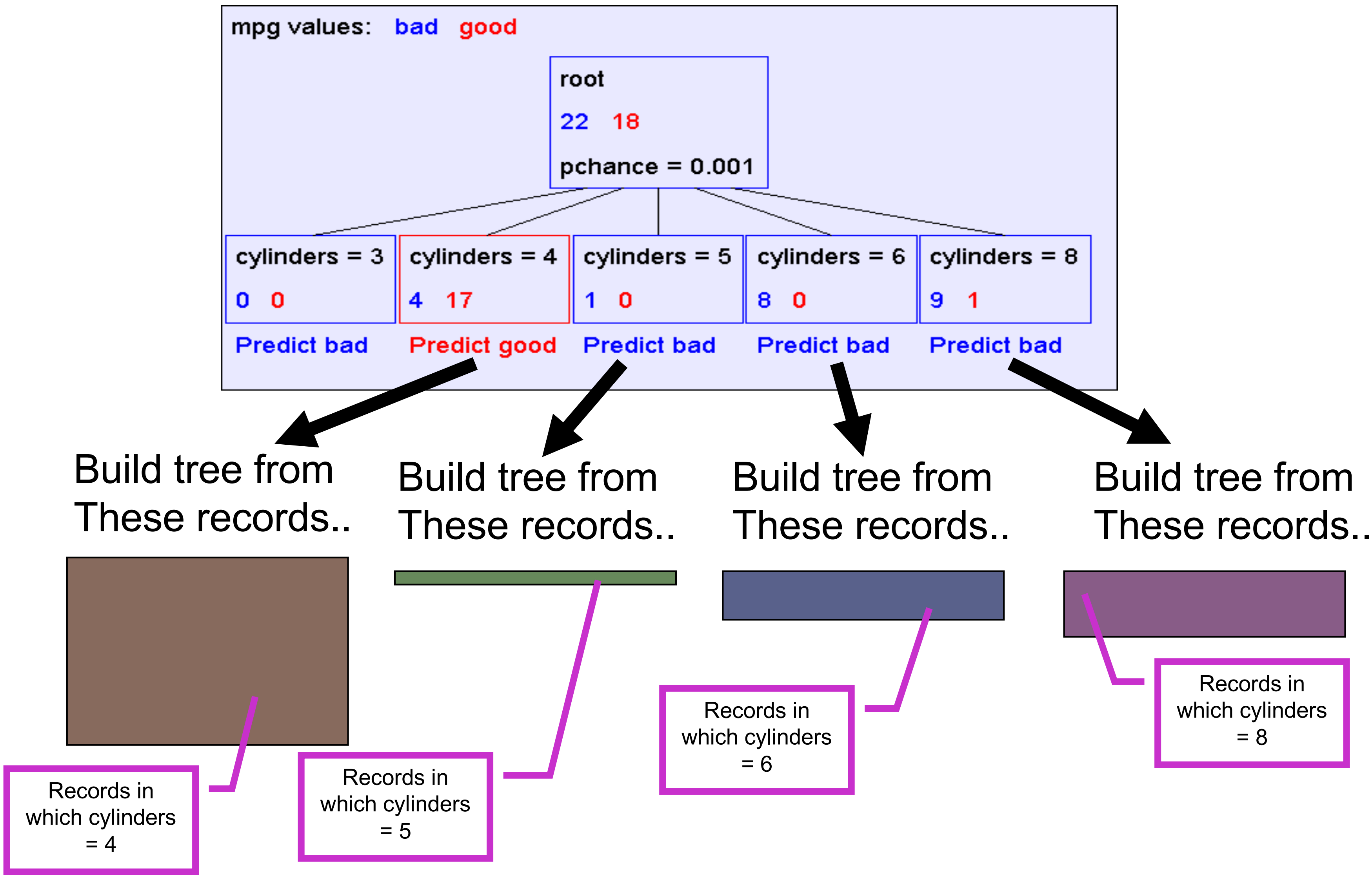
Take the Original Dataset..



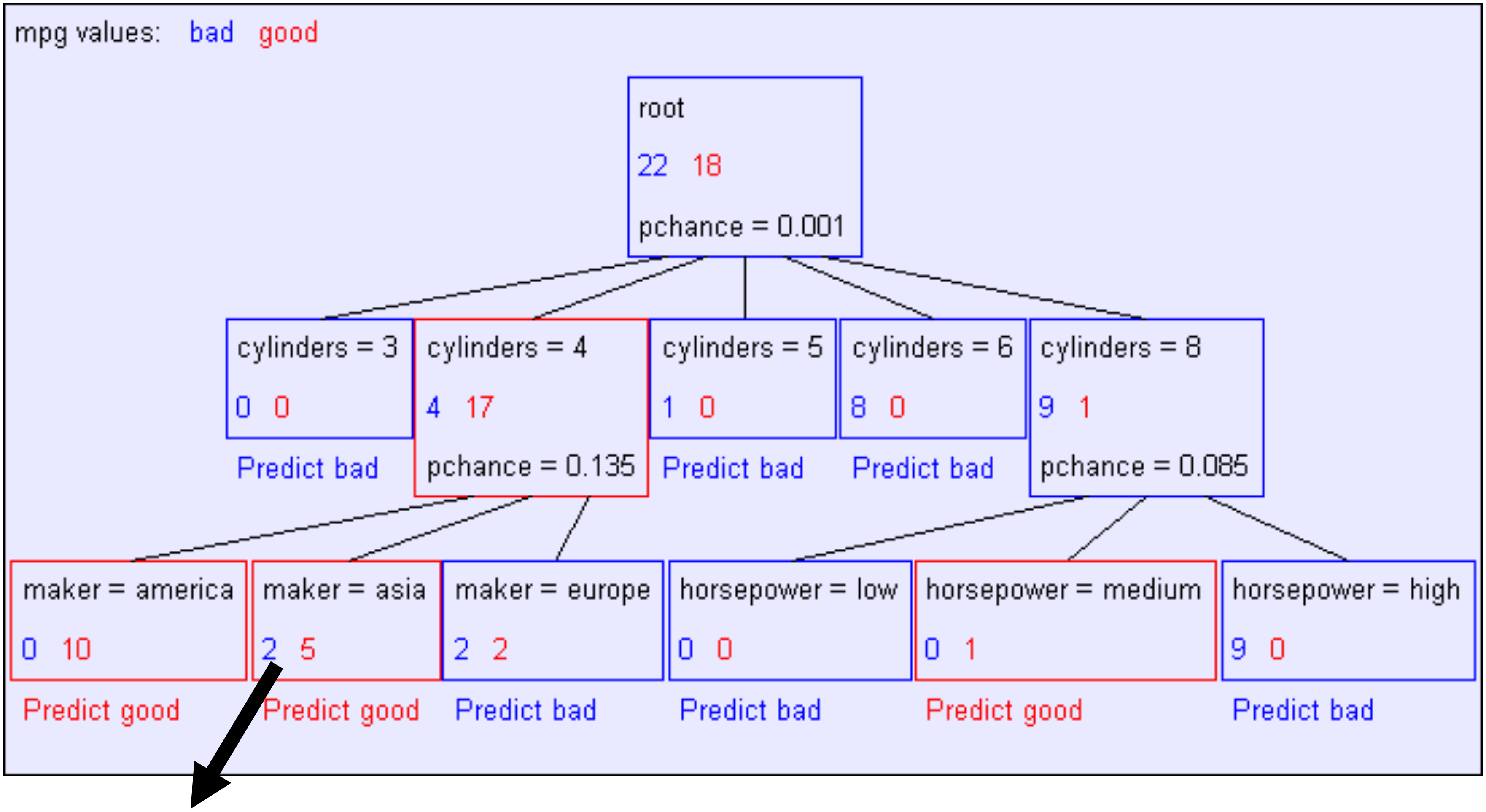
And partition it according to the value of the attribute we split on



Recursive Step



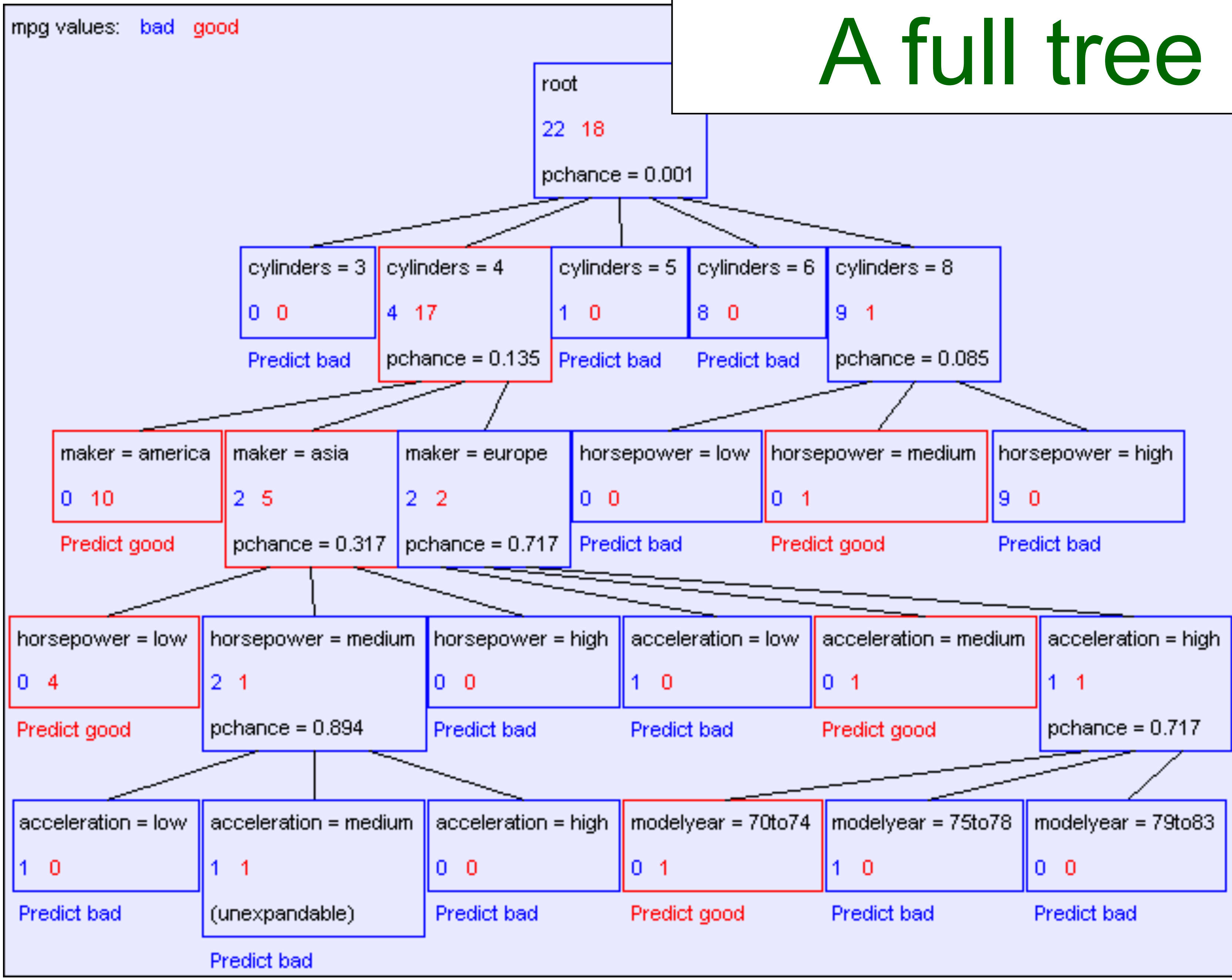
Second level of tree



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)

A full tree



mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad

Two Questions

Hill Climbing Algorithm:

– Start from empty decision tree

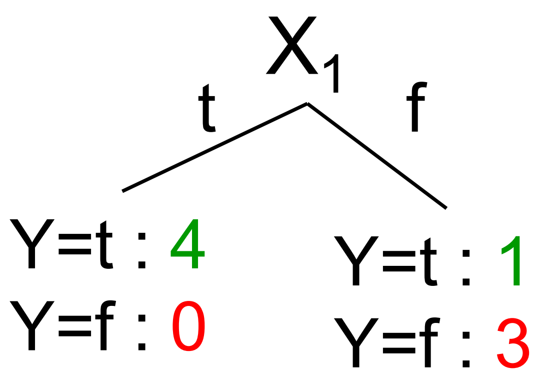
– Split on the **best attribute (feature)**

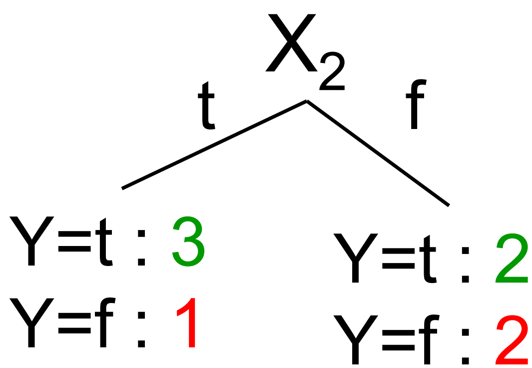
– Recurse

1. Which attribute gives the best split?
2. When to stop recursion?

Splitting: choosing a good attribute

Would we prefer to split on X_1 or X_2?





Idea: use counts at leaves to define probability distributions so we can measure uncertainty!

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
- What about distributions in between?

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

P(Y=A) = 1/3	P(Y=B) = 1/4	P(Y=C) = 1/4	P(Y=D) = 1/6

Which attribute gives the best split?

A_1: The one with the highest ***information gain***

Defined in terms of ***entropy***

A_2: Actually many alternatives, eg, ***accuracy***

Seeks to reduce the ***misclassification rate***

Entropy

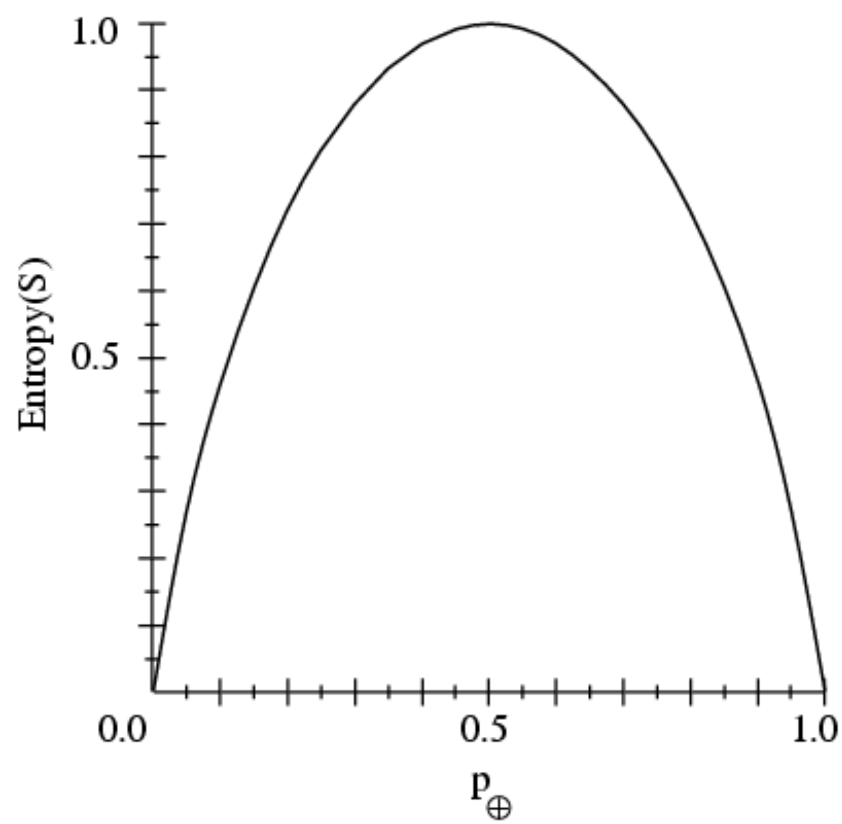
Entropy *H(Y)* of a random variable *Y*

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

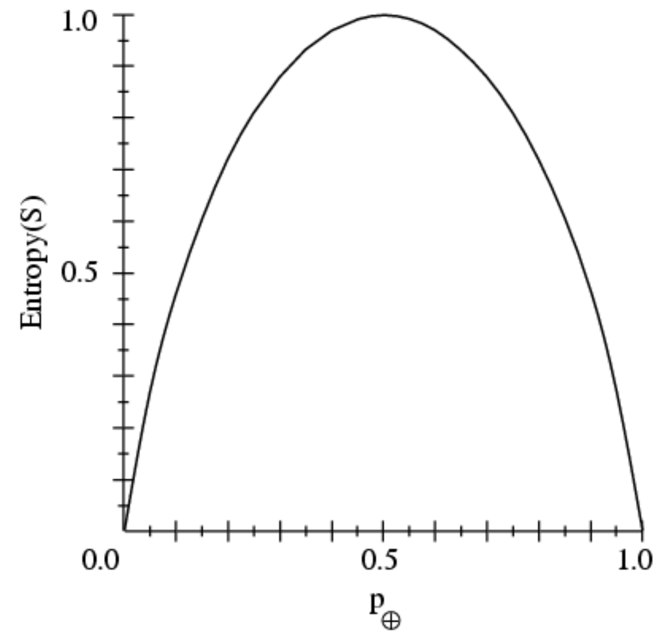
More uncertainty, more entropy!

Information Theory interpretation:

H(Y) is the expected number of bits needed to encode a randomly drawn value of *Y* (under most efficient code)



Entropy Example



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

P(Y=t) = 5/6

P(Y=f) = 1/6

H(Y) = - 5/6 $\log_2$ 5/6 - 1/6 $\log_2$ 1/6

= 0.65

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

Conditional Entropy

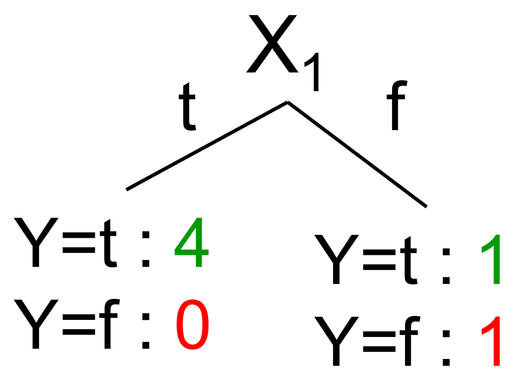
Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X

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

Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$



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

$H(Y|X_1) = -\ 4/6\ (1 \log_2 1 + 0 \log_2 0)$

$-\ 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$= 2/6$

$= 0.33$

Information Gain

Advantage of attribute – decrease in entropy (uncertainty) after splitting

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

In our running example:

$IG(X_1) = H(Y) – H(Y|X_1)$
$= 0.65 – 0.33$

$IG(X_1) > 0$ → we prefer the split!

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 information gain (or…?) to select attribute:

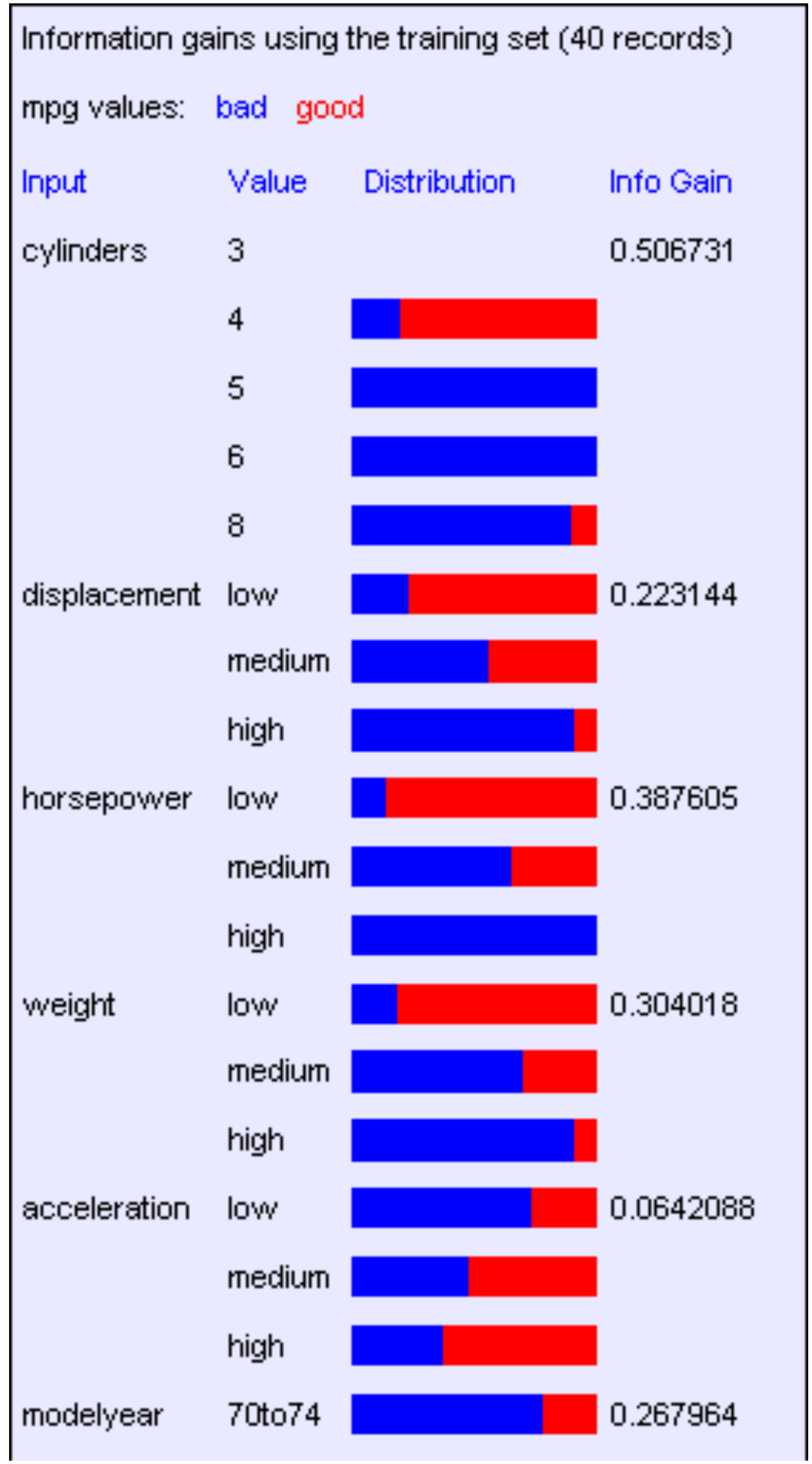
$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

Recurse

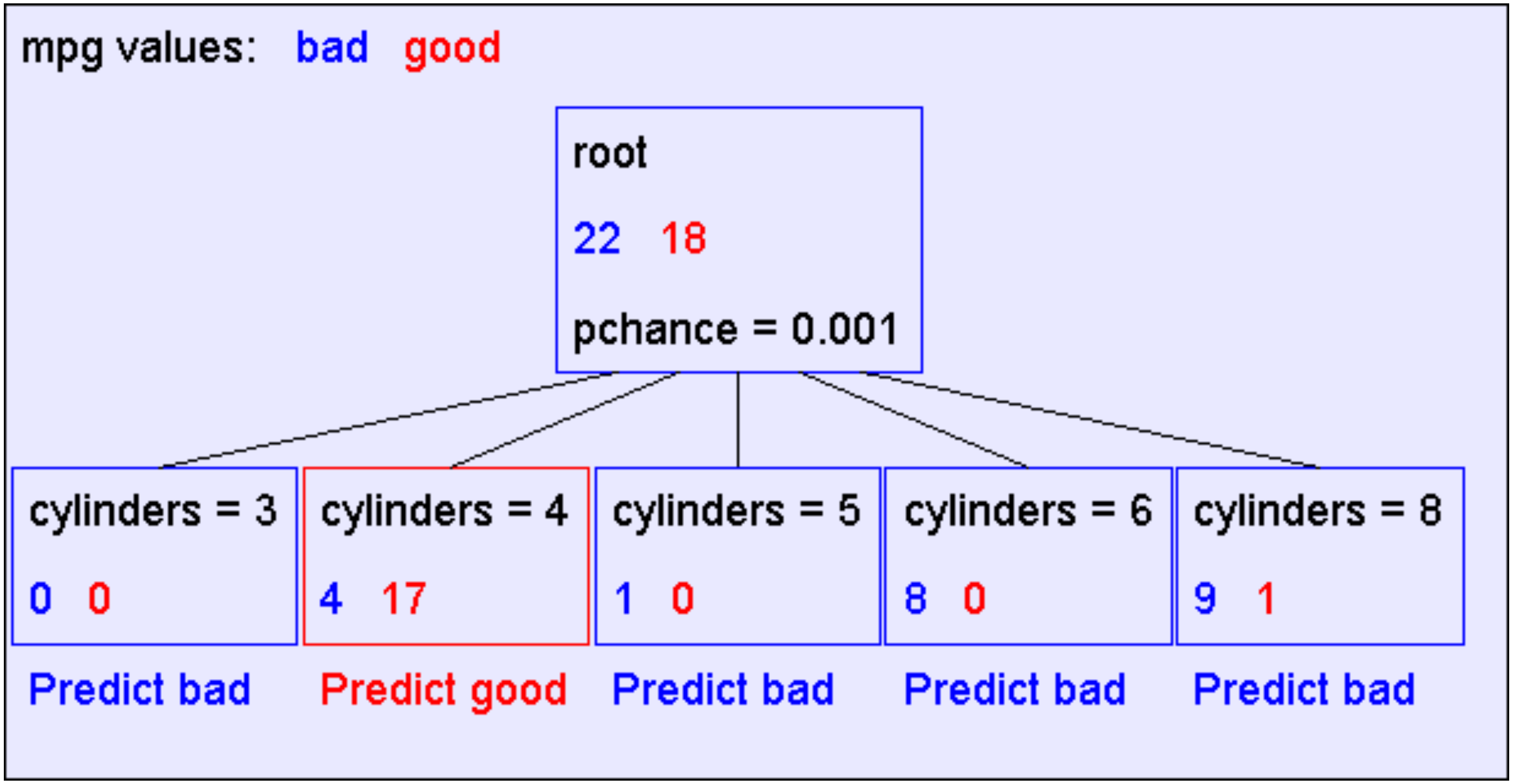
Suppose we want to predict MPG

predict
mpg=bad

Now, Look at all the information gains…

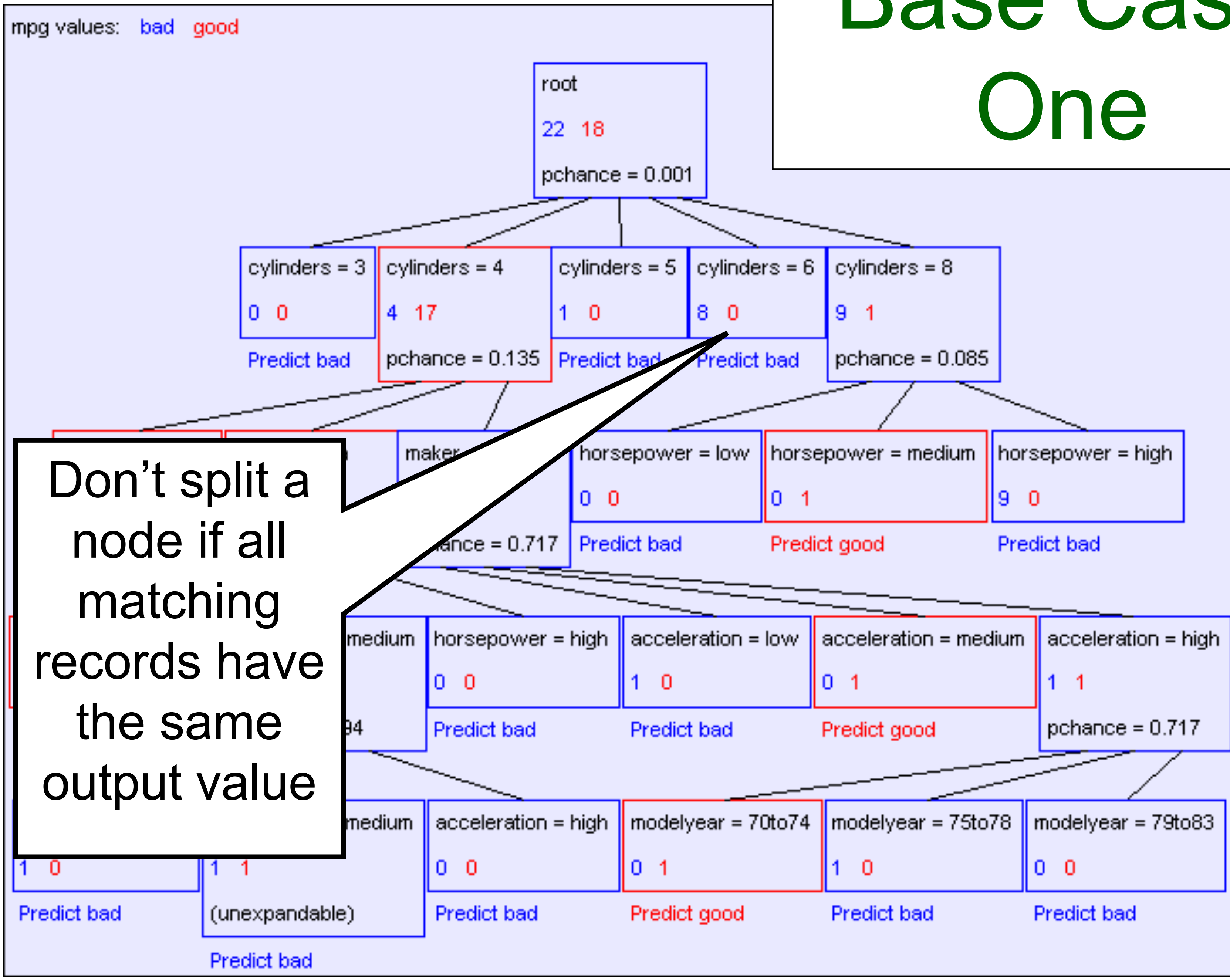


Tree After One Iteration

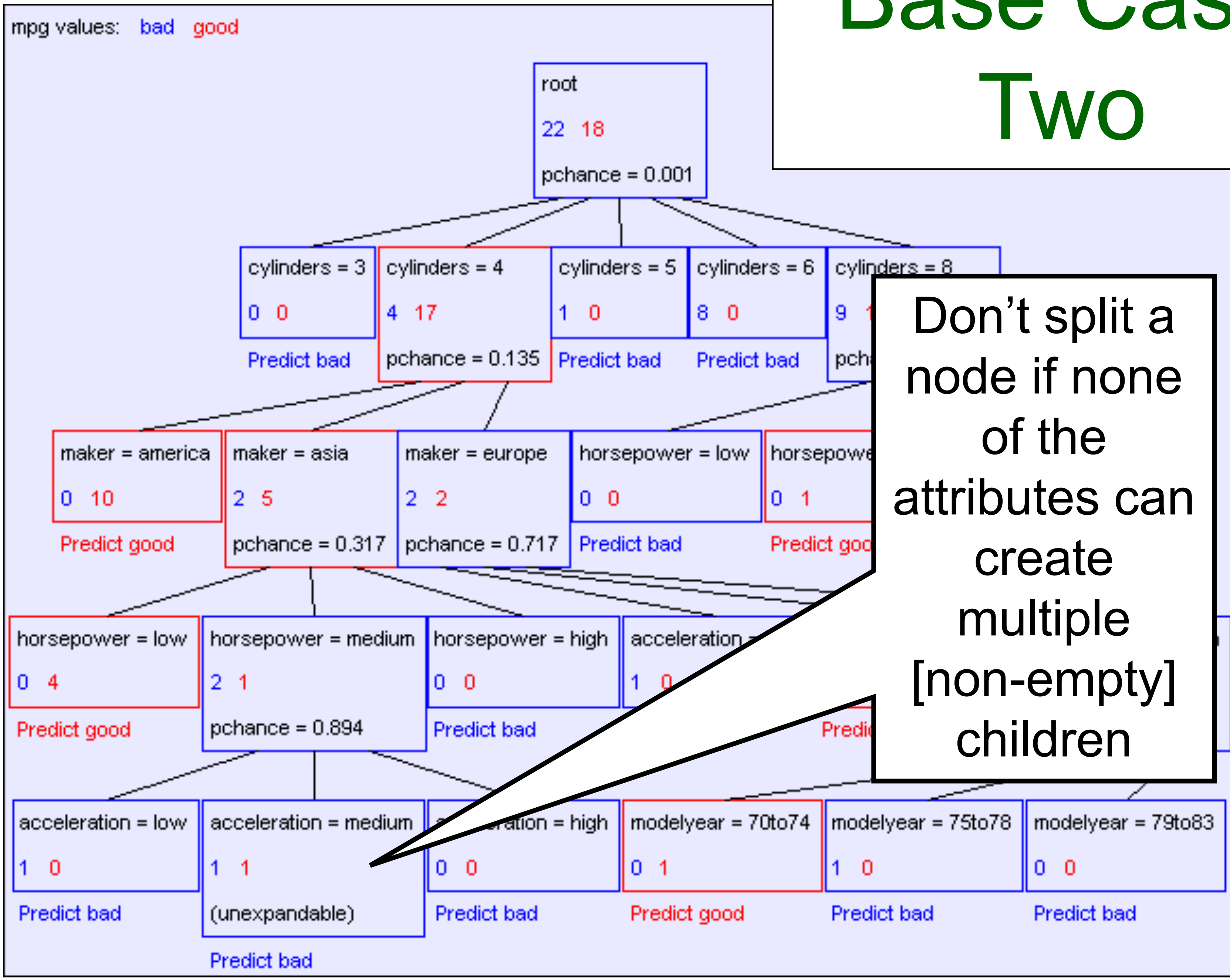


When to Terminate?

Base Case One



Base Case Two



Don’t split a node if none of the attributes can create multiple [non-empty] children

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

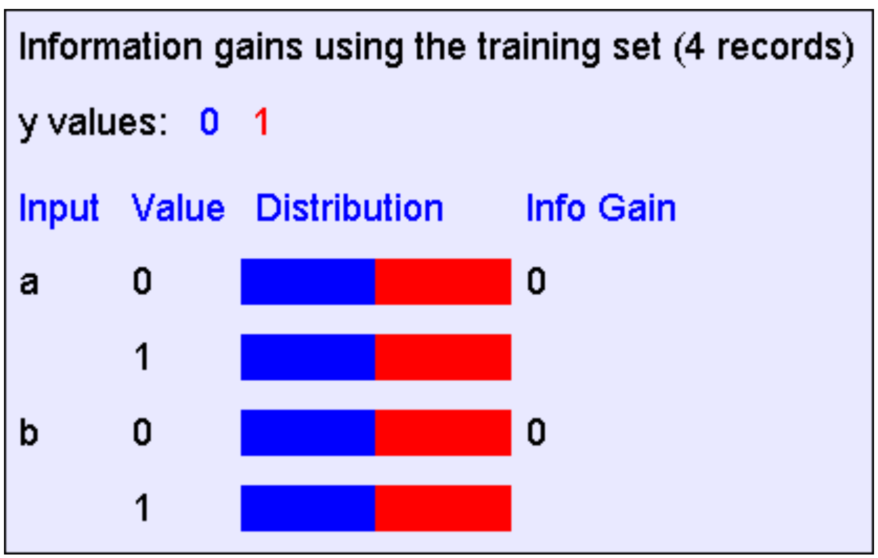
Is this a good idea?

The problem with Base Case 3

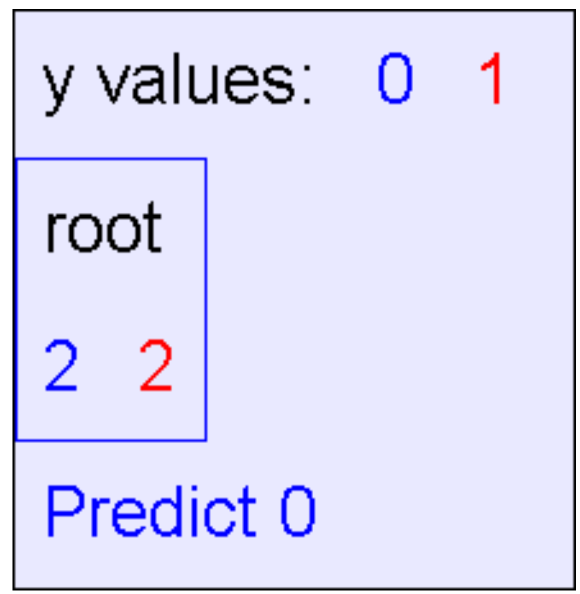
$y = a \text{ XOR } b$

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

The information gains:



The resulting decision tree:

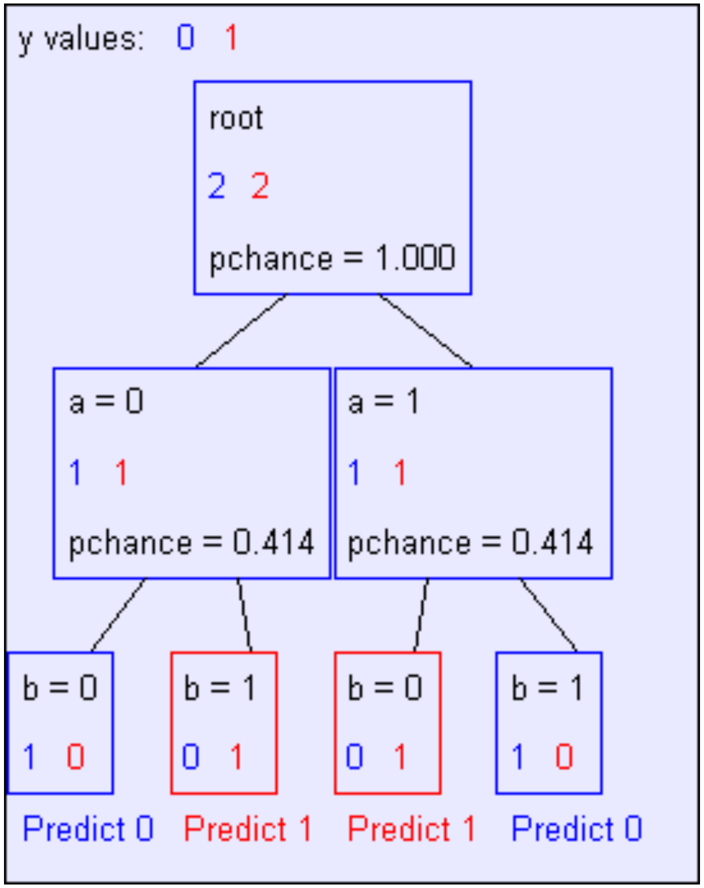


But *Without* Base Case 3:

y = a XOR b

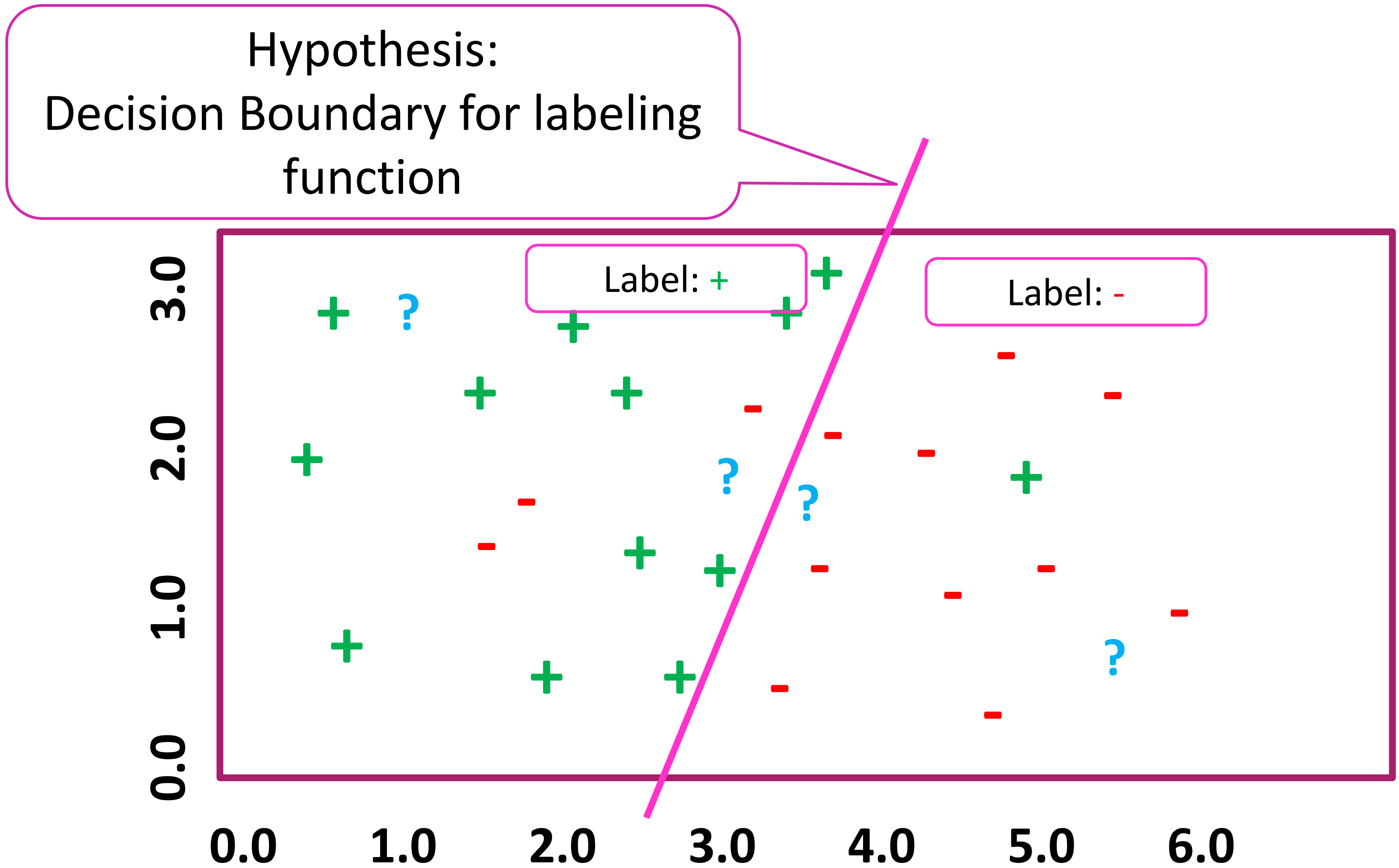
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

The resulting decision tree:



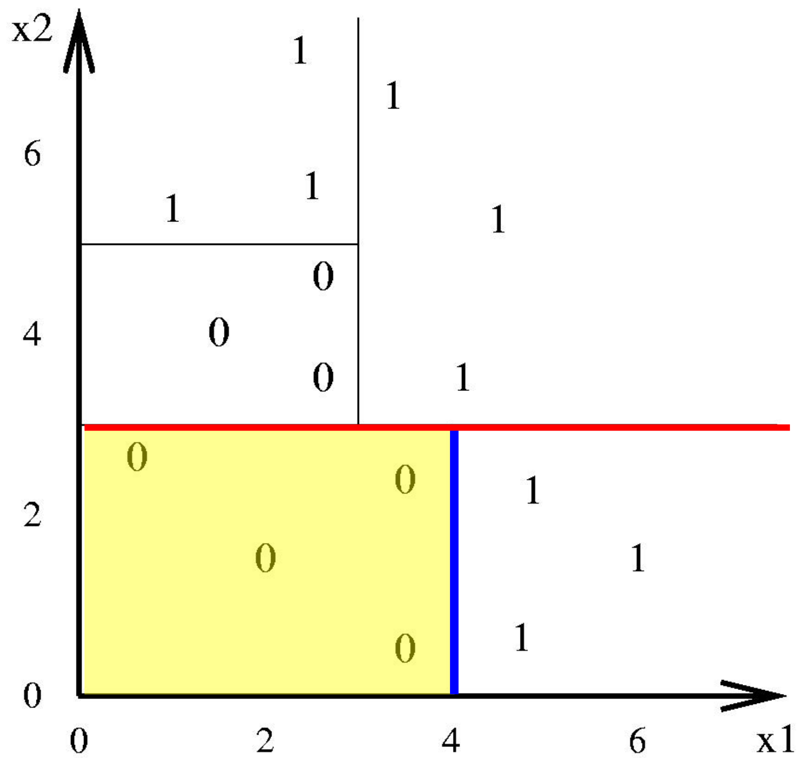
So: Base Case 3?
Include or Omit?

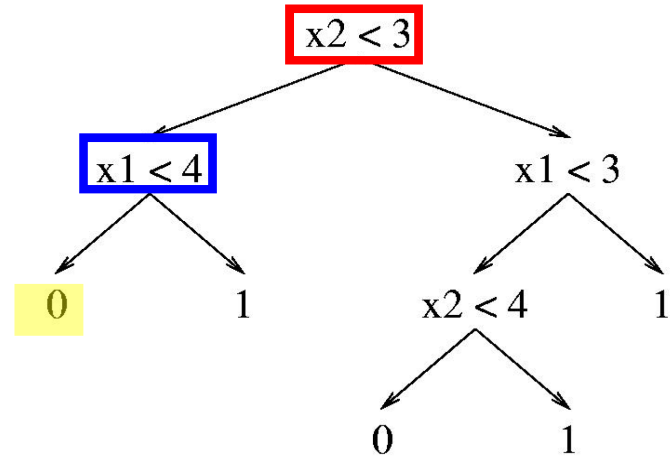
General View of a Classifier



Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

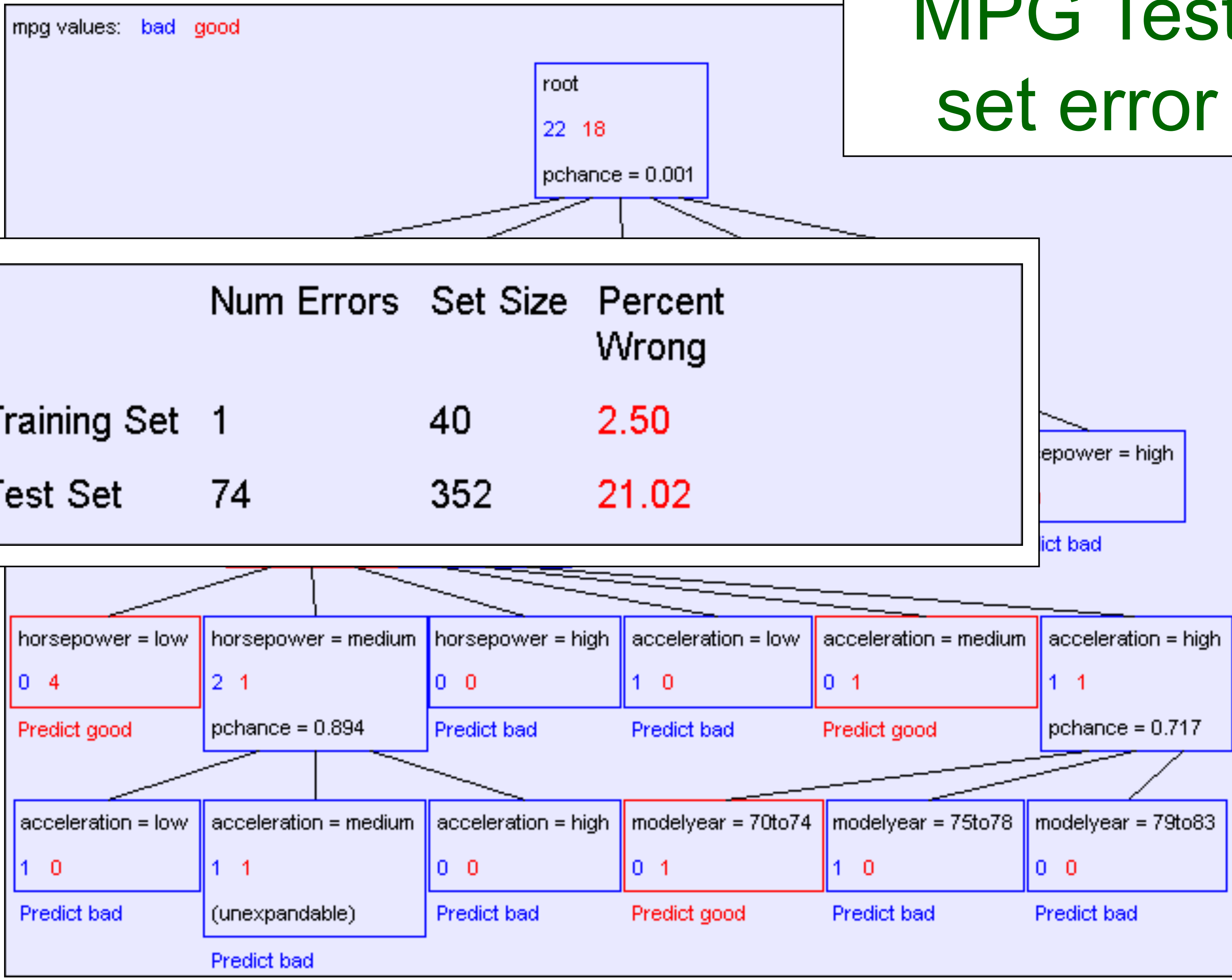




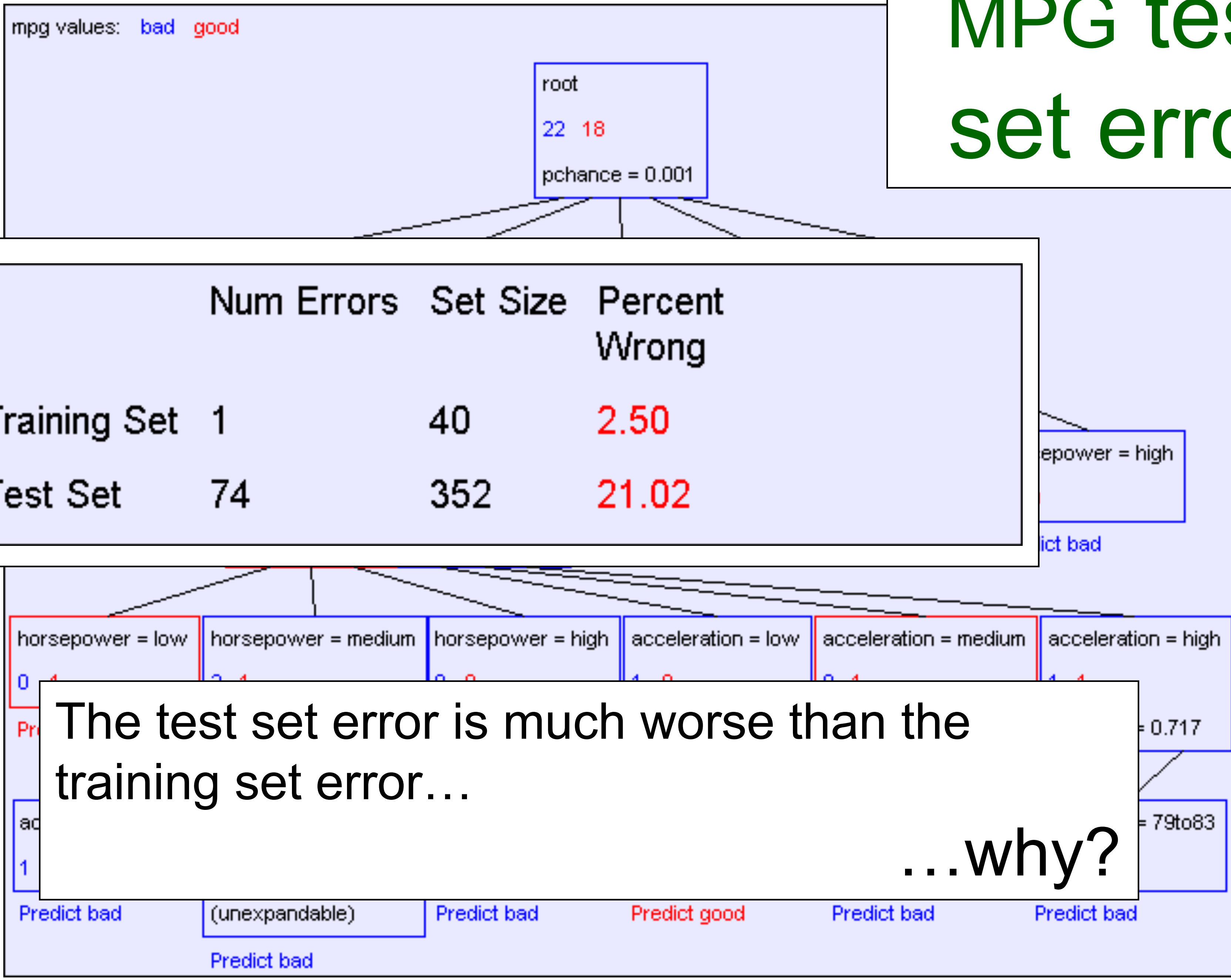
Ok, so how does it perform?

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?

Decision trees will overfit

Our 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 introduce some bias towards *simpler* trees

Why might one pick simpler trees?

Occam's Razor

Why Favor Short Hypotheses?

Arguments for:

- Fewer short hypotheses than long ones
  - →A short hyp. less likely to fit data by coincidence
  - →Longer hyp. that fit data may might be coincidence

Arguments against:

- Argument above uses fact that hypothesis ***space*** is small !
- What is so special about small sets based on the ***complexity*** of each ***hypothesis***?

How to Build Small Trees

Several reasonable approaches:

Stop growing tree before overfit

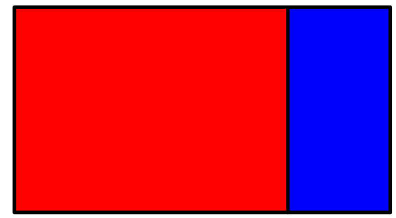
- Bound depth or # leaves
- Base Case 3
- *Doesn't work well in practice*

Grow full tree; then prune

- **Optimize on a held-out (development set)**
  - If growing the tree hurts performance, then cut back
  - Con: Requires a larger amount of data…
- **Use statistical significance testing**
  - Test if the improvement for any split is likely due to noise
  - If so, then prune the split!
- **Convert to logical rules**
  - Then simplify rules

Reduced Error Pruning

Split data into ***training*** & ***validation*** sets (10-33%)



Train on training set (overfitting)

Do until further pruning is harmful:

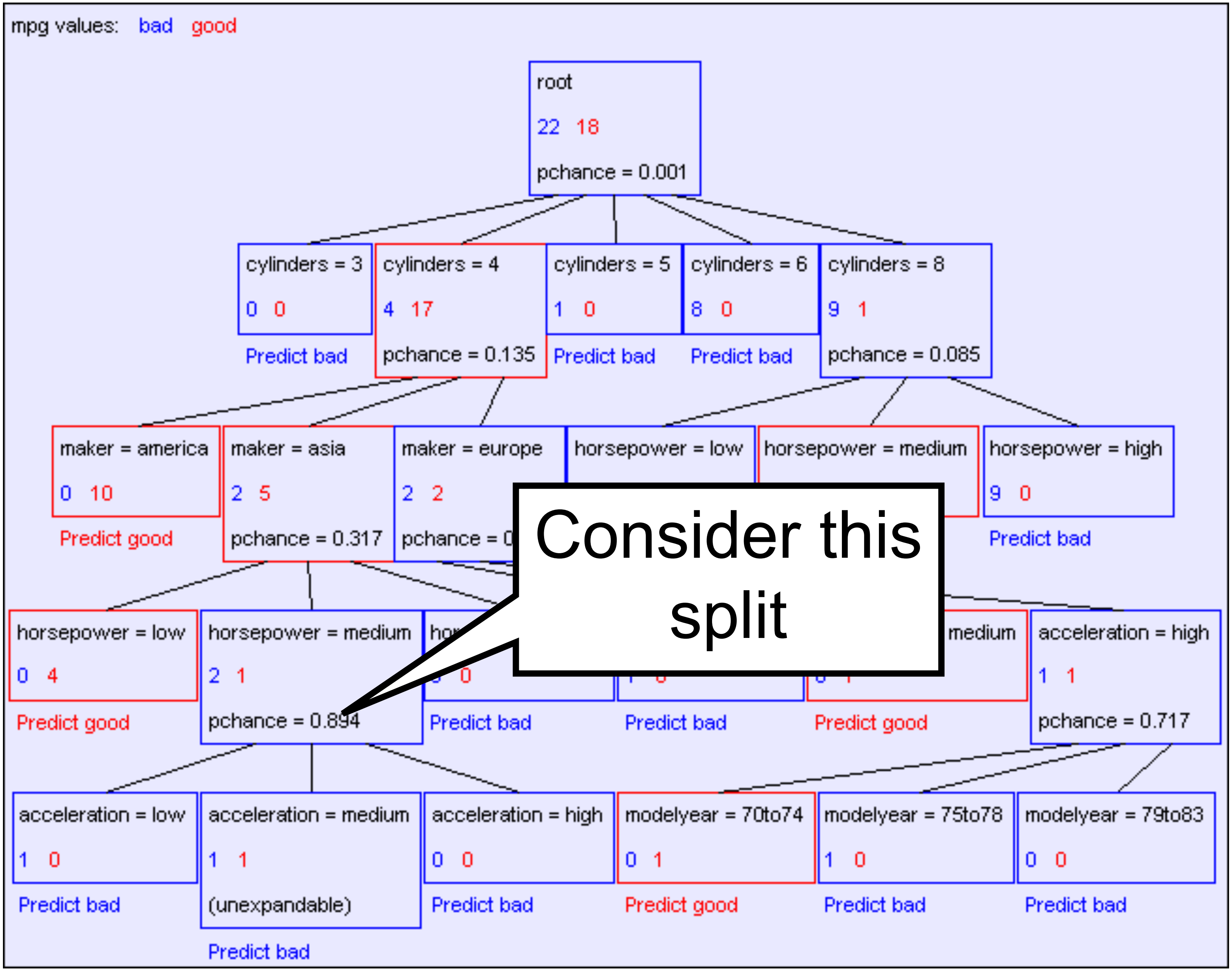
1) Evaluate effect on validation set of pruning ***each*** possible node (and tree below it)
2) Greedily remove the node that ***most improves accuracy of validation set***

Alternatively

Chi-squared pruning

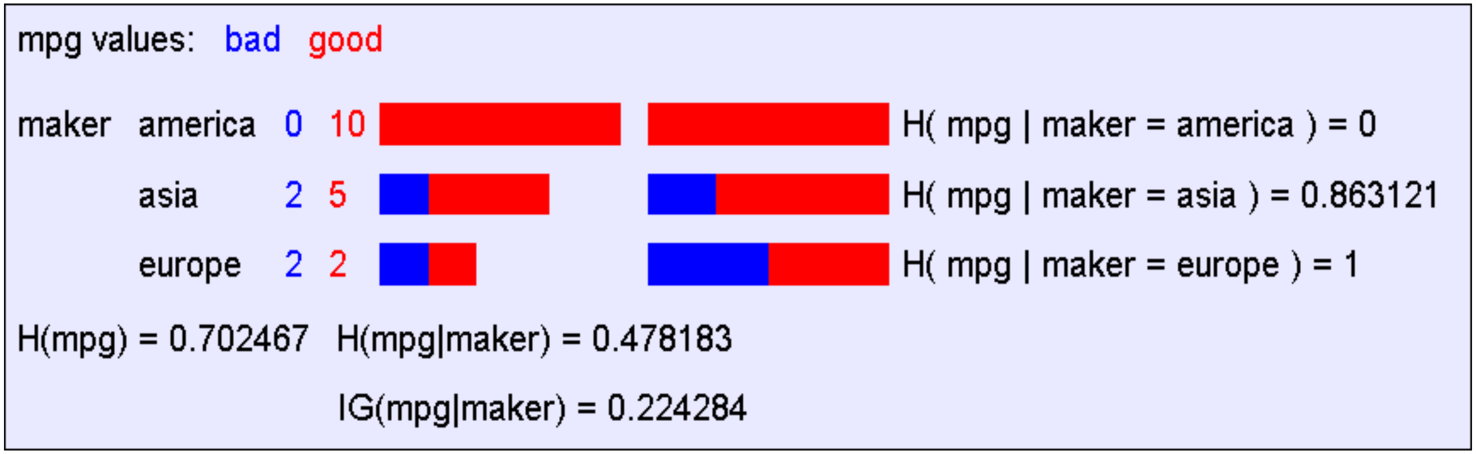
- Grow tree fully
- Consider leaves in turn
  - Is parent split worth it?

Compared to Base-Case 3?



mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
horsepower = low
horsepower = medium
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
Predict bad
Predict bad
medium
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad
Consider this split

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 13.5%

Such hypothesis tests are relatively easy to compute, but involved

Using Chi-squared to avoid overfitting

Build the full decision tree as before

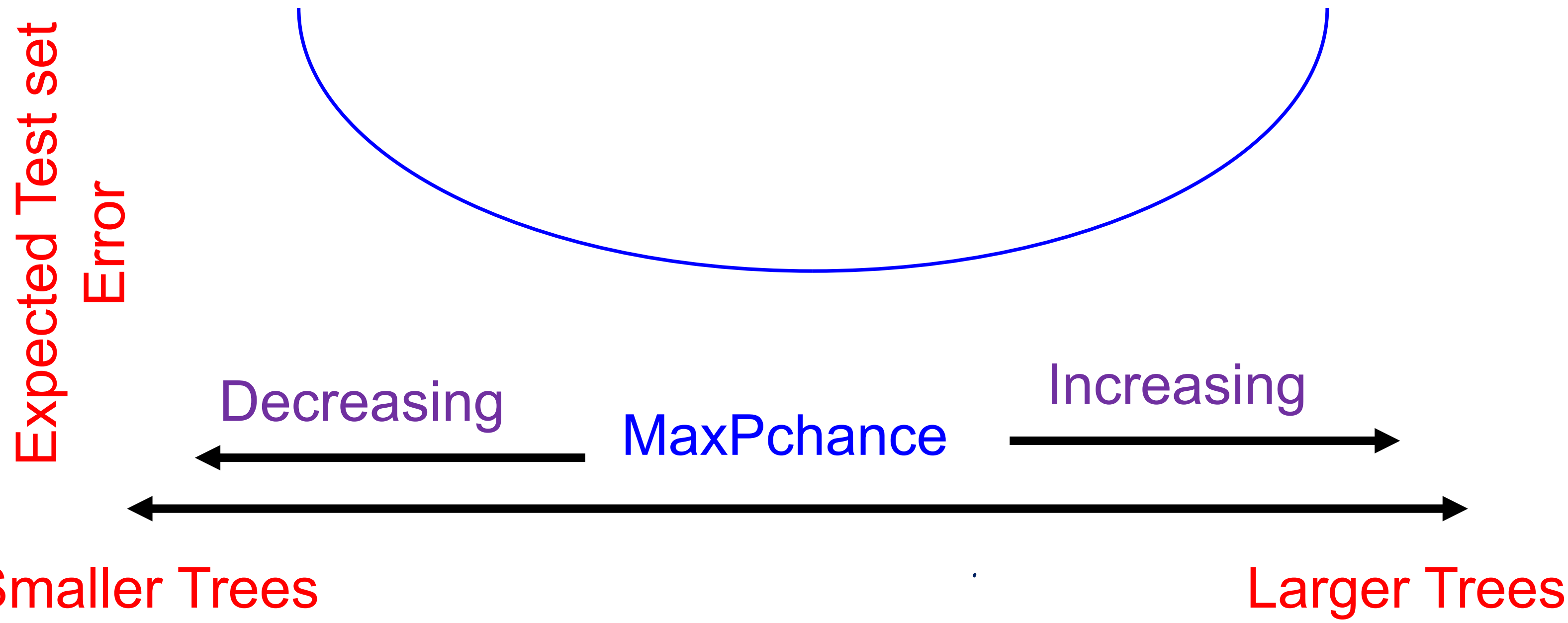
But when you can grow it no more, start to prune:

- 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

Regularization

Note for Future: MaxPchance is a regularization parameter that helps us bias towards simpler models



We'll learn to choose the value of magic parameters like this one later!

ML as Optimization

Greedy search for best ***scoring*** hypothesis

Where ***score*** =

- Fits training data most accurately?
- Sum: ***training accuracy – complexity penalty***

regularization

Advanced Decision Trees

Attributes with:

- Numerous Possible Values
- Continuous (Ordered) Values
- Missing Values

decision tree summary

Decision trees are one of the most popular ML tools

- Easy to understand, implement, and 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!!!

- Must use tricks to find “simple trees”, e.g.,
  - Fixed depth/Early stopping
  - Pruning
  - Hypothesis testing

Loss Functions

How measure quality of hypothesis?

Loss Functions

How measure quality of hypothesis?

$L(x, y, \hat{y})$ = utility(result of using y given input of x)
- utility(result of using $\hat{y}$ given input of x)

L(edible, poison)

L(poison, edible)

Common Loss Functions

0/1 loss	0 if $y=\hat{y}$ else 1
Absolute value loss	$\|y-\hat{y}\|$
Squared error loss	$\|y-\hat{y}\|^2$

Overview of Learning

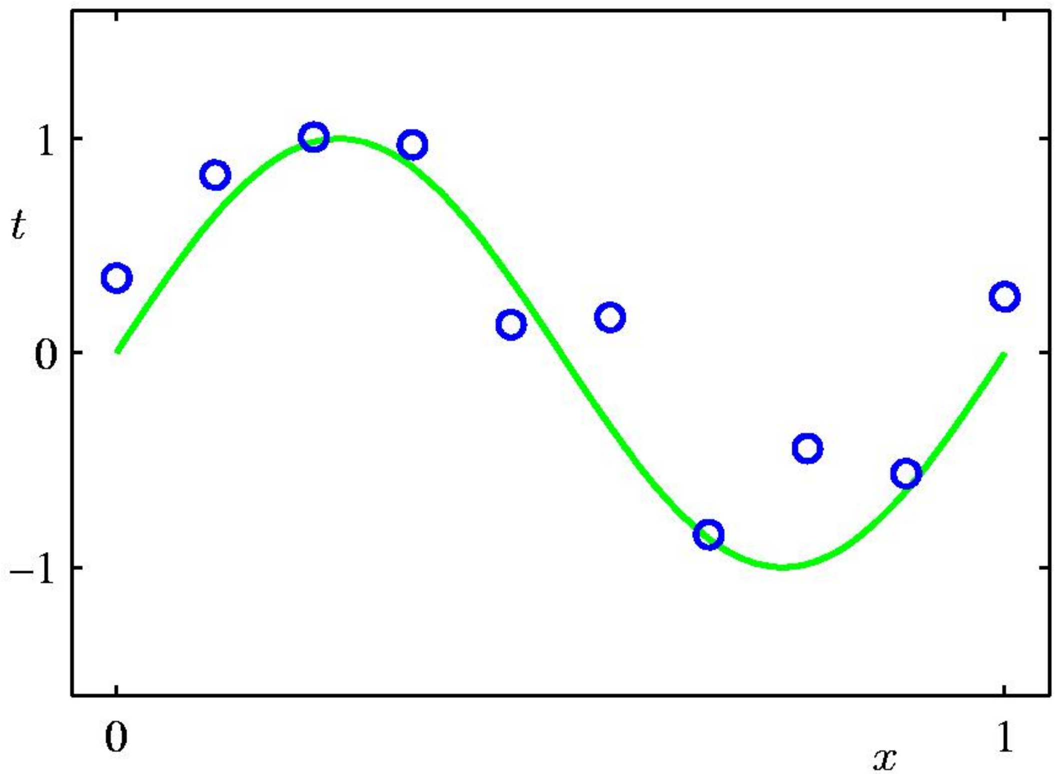
Type of Supervision

(eg, Experience, Feedback)

What is Being Learned?

	Labeled Examples	**Reward**	**Nothing**
Discrete Function	Classification		Clustering
Continuous Function	$\boxed{\text{Regression}}$		
Policy	Apprenticeship Learning	Reinforcement Learning	

Polynomial Curve Fitting

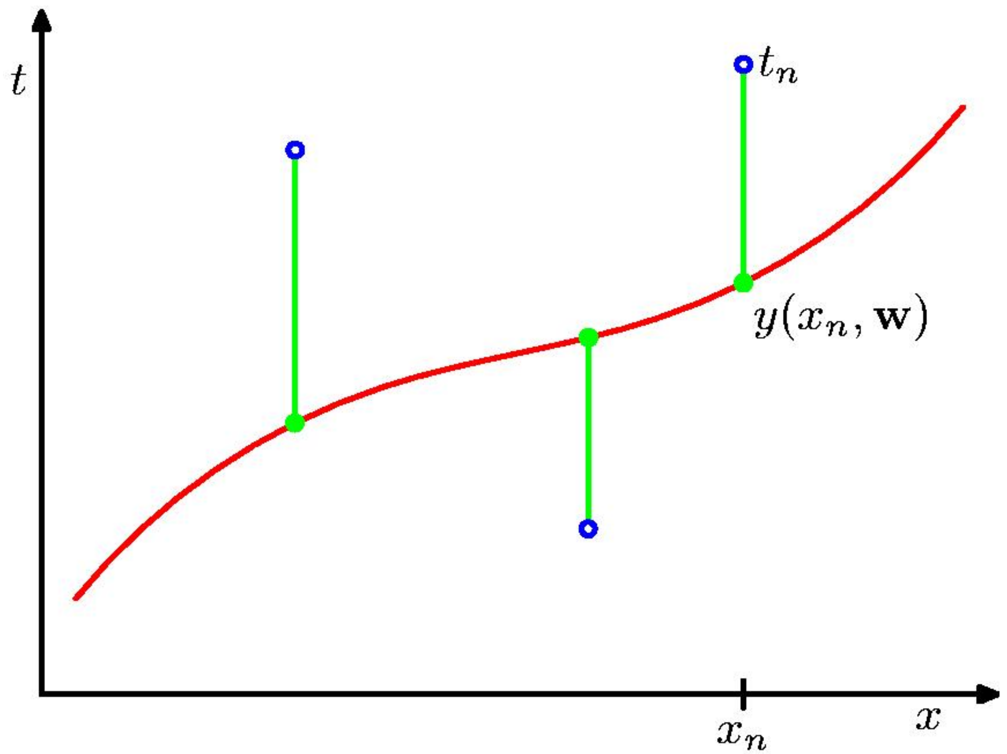


Sin(2πx)

Hypothesis Space

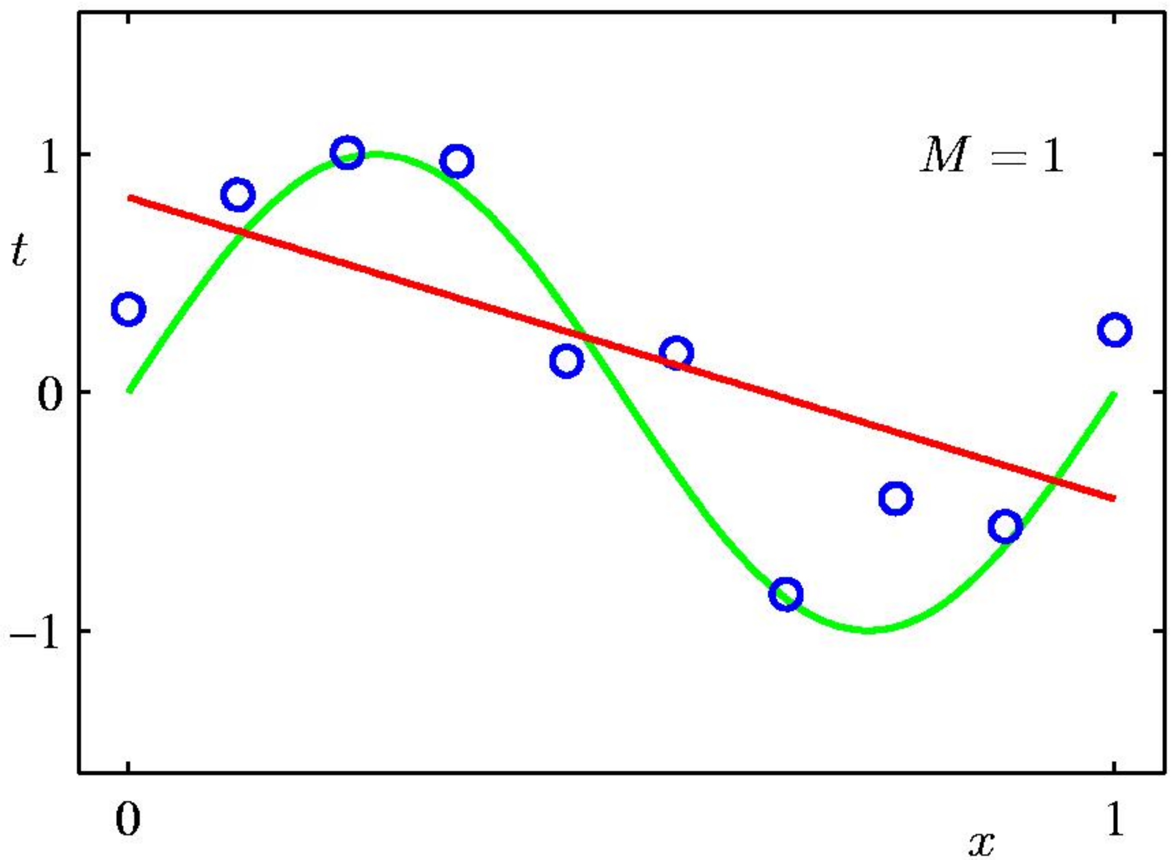
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Sum-of-Squares Error Function

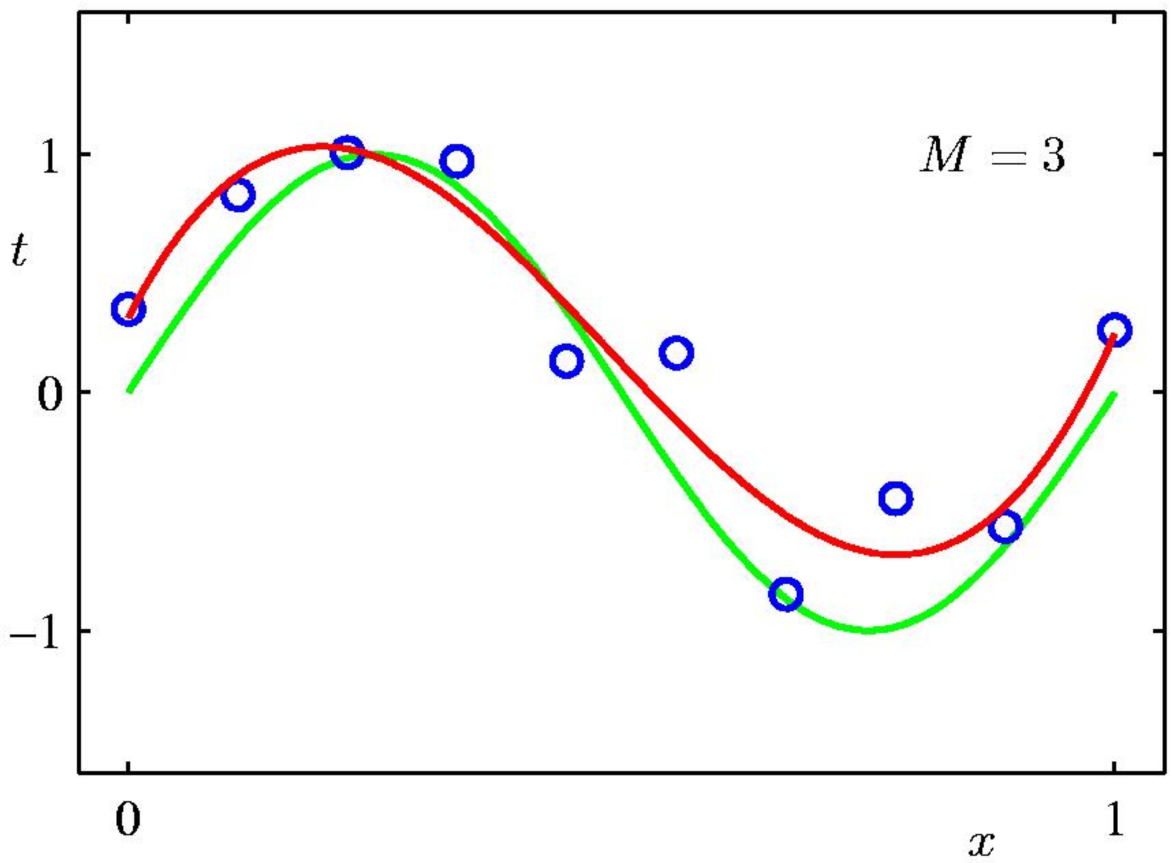


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

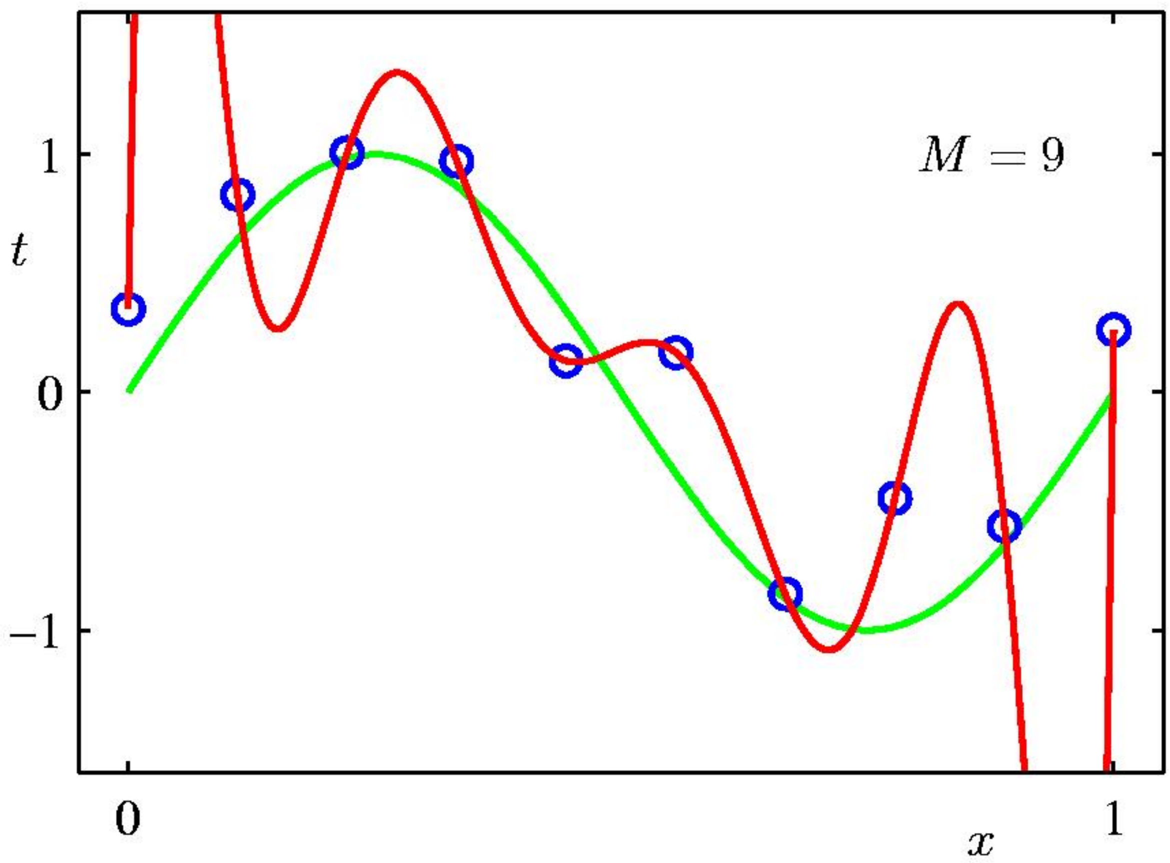
1^{st} Order Polynomial



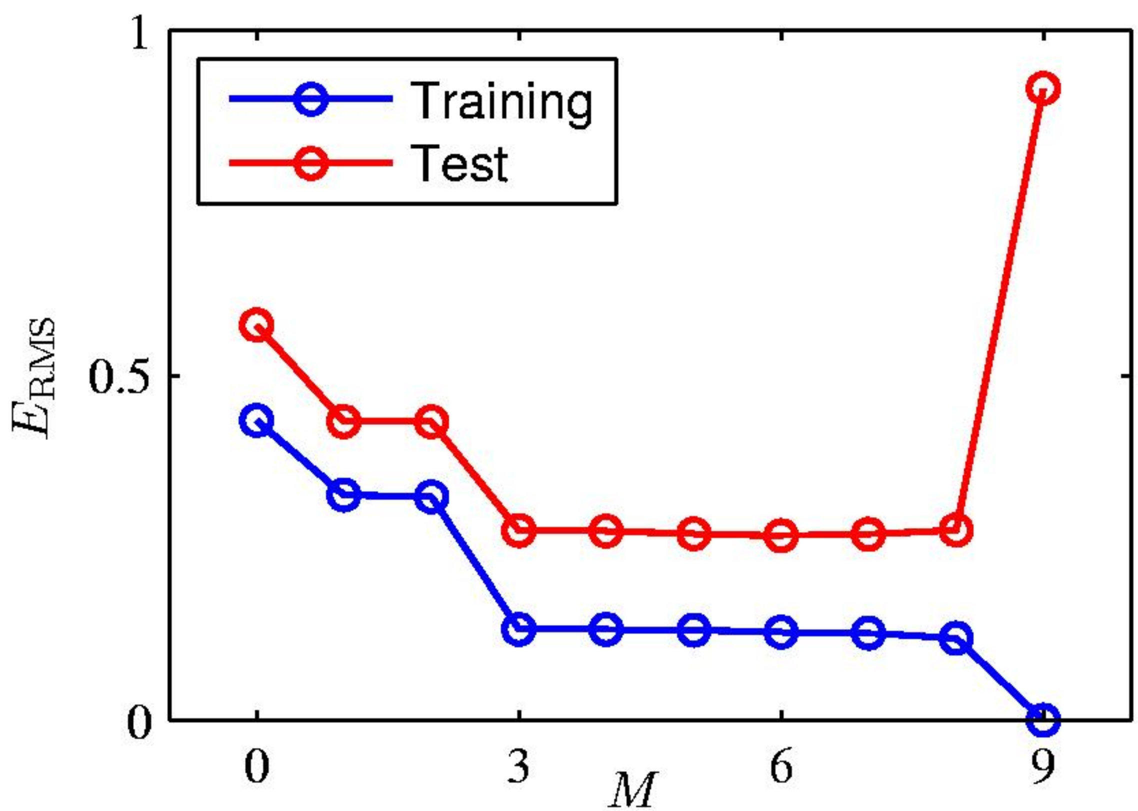
3rd Order Polynomial



9^{th} Order Polynomial



Over-fitting



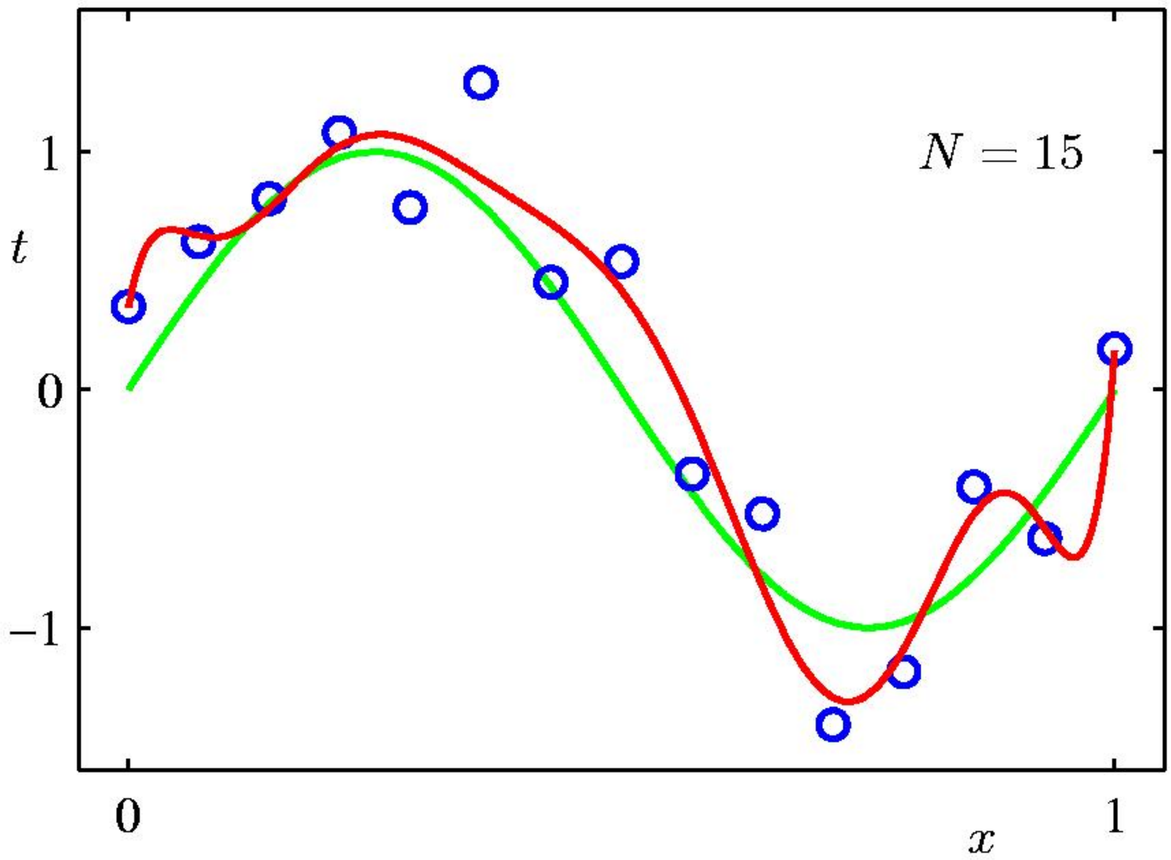
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

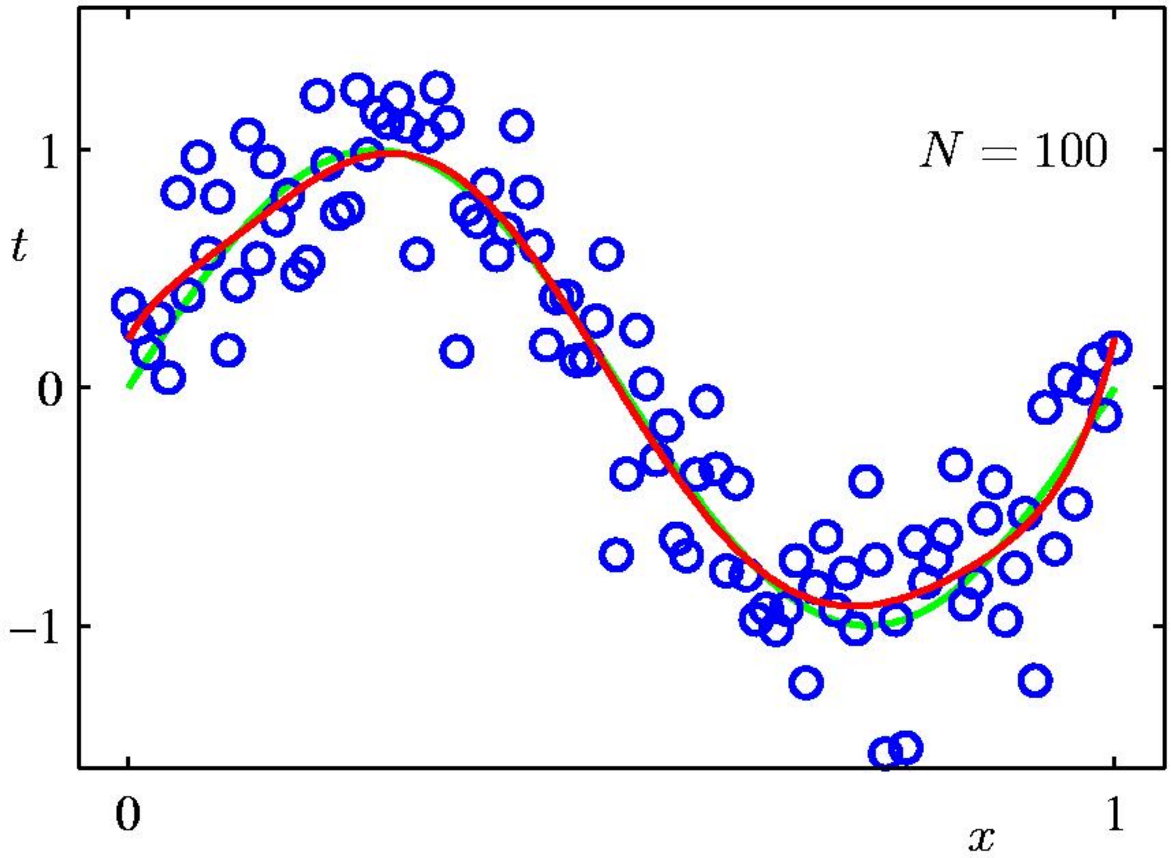
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial

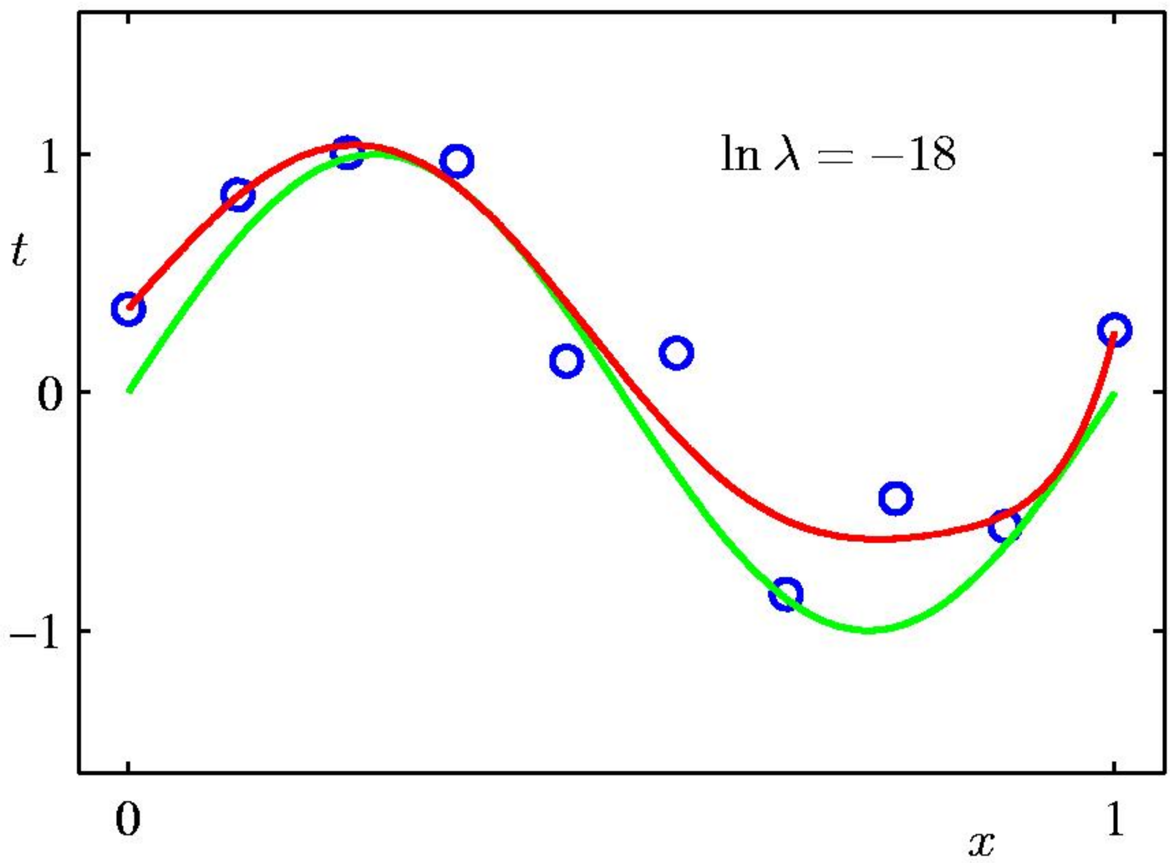


Regularization

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

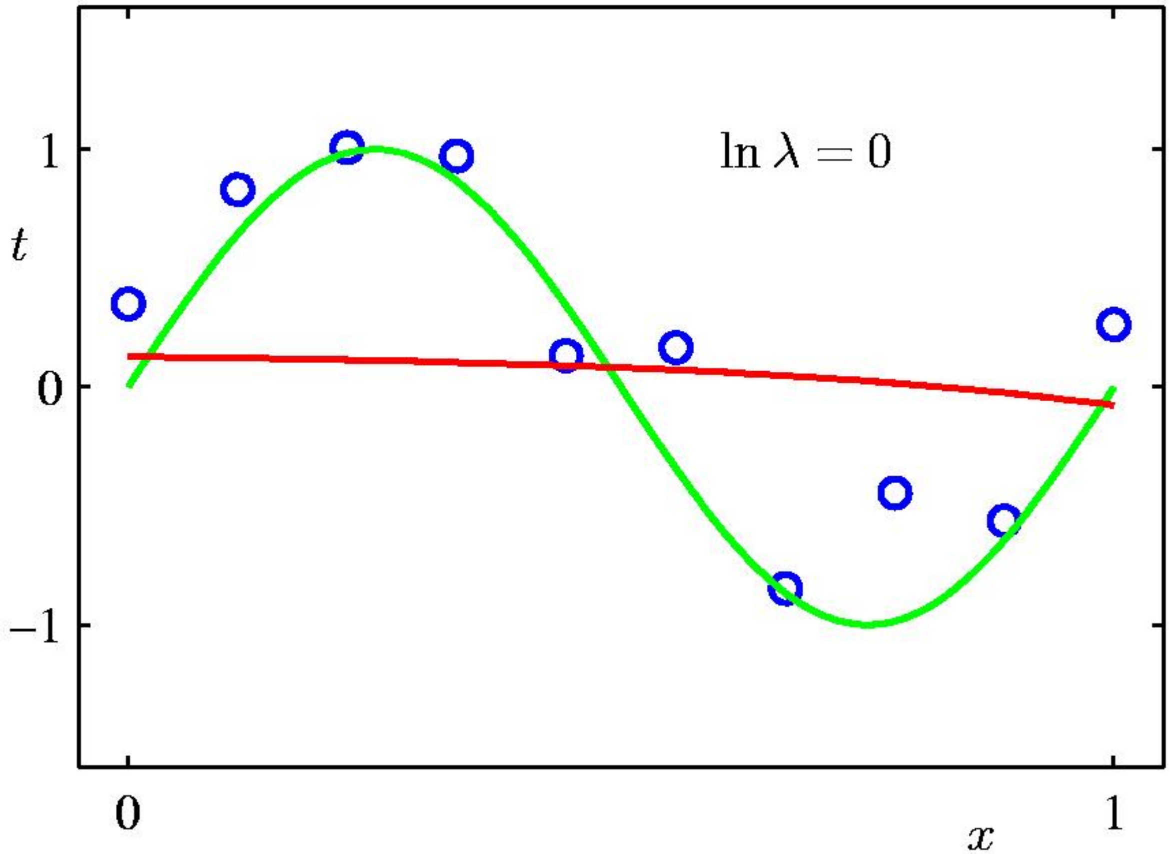
Penalize large coefficient values

Regularization: $\ln \lambda = -18$

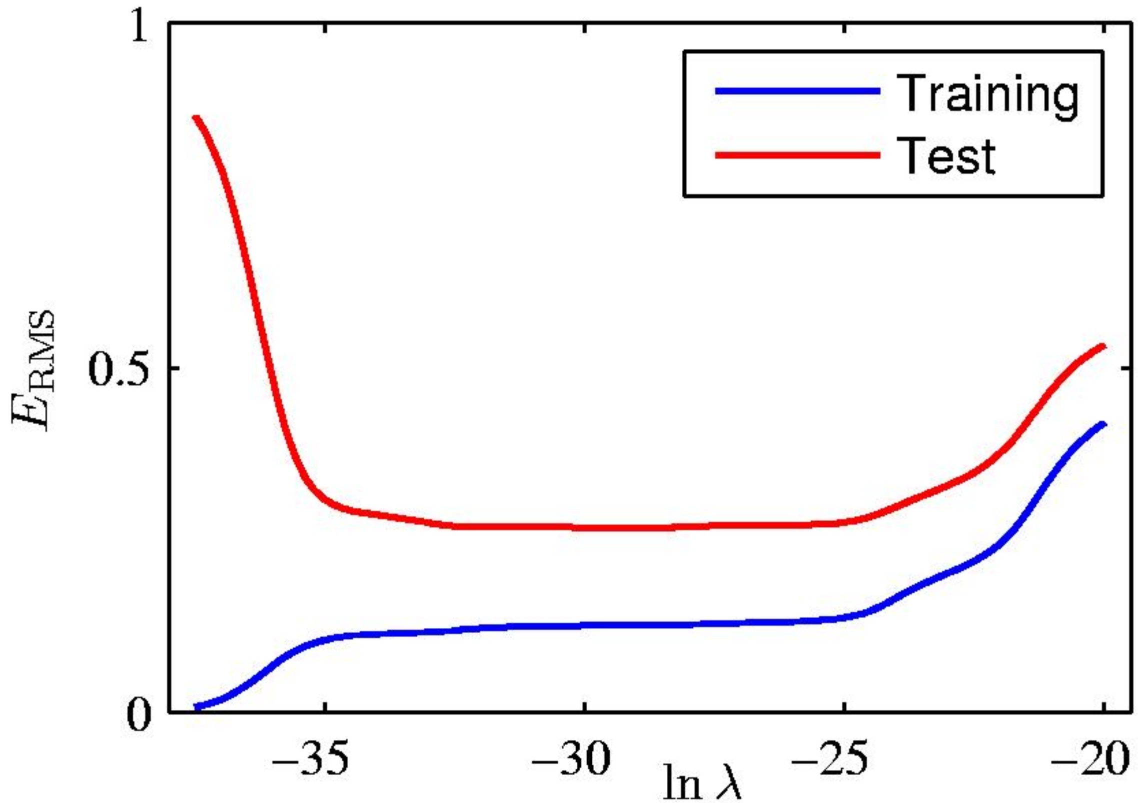


Regularization:

$$\ln \lambda = 0$$



Regularization: E_{RMS} *vs.* $\ln \lambda$



Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Part 2

Continuous Optimization

Machine Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Parametric

Non-parametric

Y Continuous

Y Discrete

Gaussians

Learned in closed form

Linear Functions

1. Learned in closed form
2. Using gradient descent

Decision Trees

Greedy search; pruning

Probability of Class | Features

1. Learn P(Y), P(X|Y); apply Bayes
2. Learn P(Y|X) w/ gradient descent

Non-probabilistic Linear Classifier

Perceptron – w/ gradient descent

Hypothesis Expressiveness

LINEAR

Naïve Bayes

Logistic Regression

Perceptron

Support Vector Machines

NONLINEAR

Decision Trees

Neural Networks

Ensembles

Kernel Methods

Nearest Neighbor

Graphical Models

Logistic Regression

Want to Learn: h:**X** $\mapsto$ Y

- **X** – features
- Y – target classes

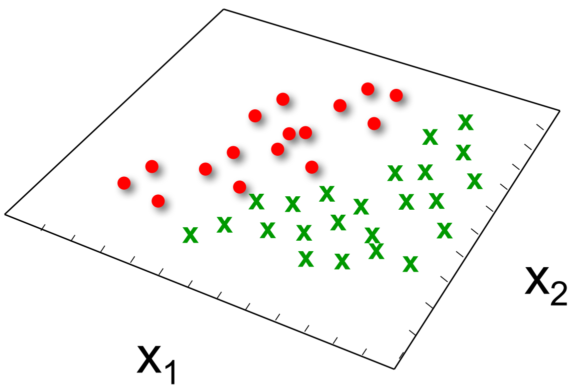
Probabilistic Discriminative Classifier

- Assume some **functional form for P(Y|X)**
  - **Logistic Function**
  - **Accepts both discrete & continuous features**
- Estimate parameters of P(Y|X) directly from training data
- This is the **‘*discriminative*’ model**
  - Directly learn P(Y|X)
  - But **cannot generate a sample of the data**,
  - No way to compute P(X)

Earthquake or Nuclear Test?

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

implies



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

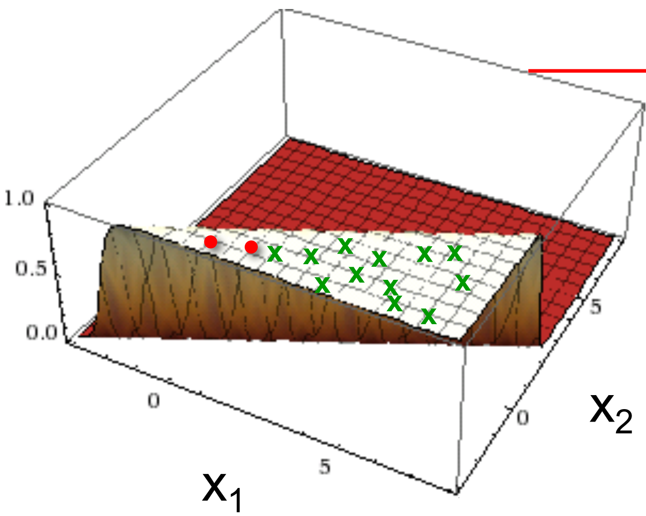
linear classification rule!

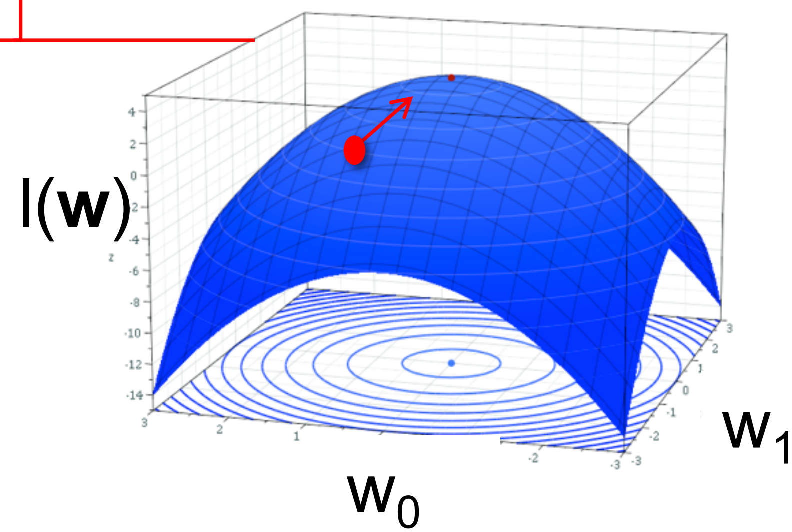
Logistic w/ Initial Weights

w_0=20 w_1= -5 w_2 =10

Loss(H_w) = Error(H_w, data)

Minimize Error → Maximize l(**w**) = ln P(D_Y | D_x, H_w)





Update rule:

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

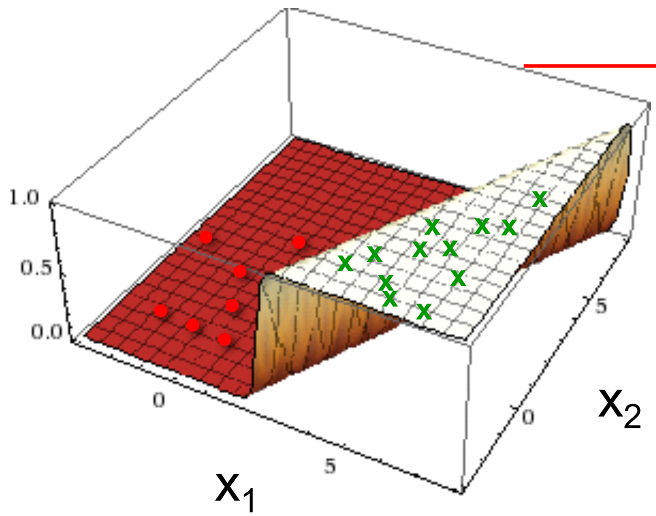
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

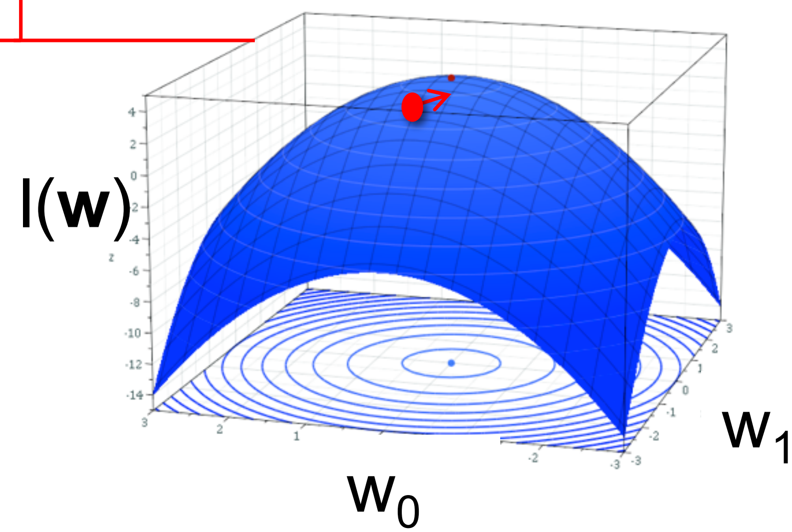
Step size

Gradient Ascent

w_0=40 w_1= -10 w_2 =5

Maximize l(**w**) = ln P(D_Y | D_x, H_w)





Update rule:

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

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

Root Finding

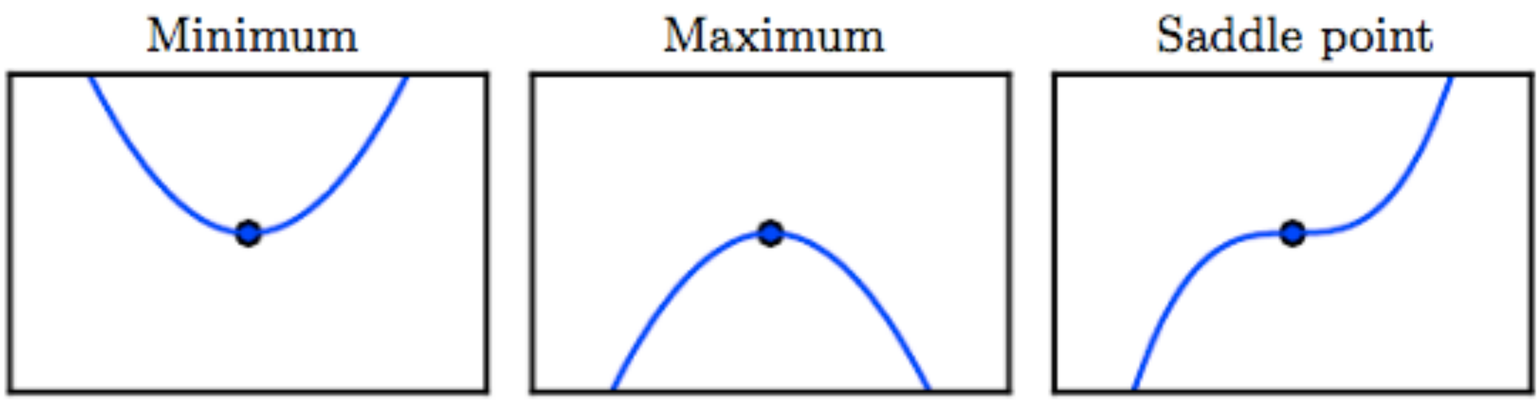
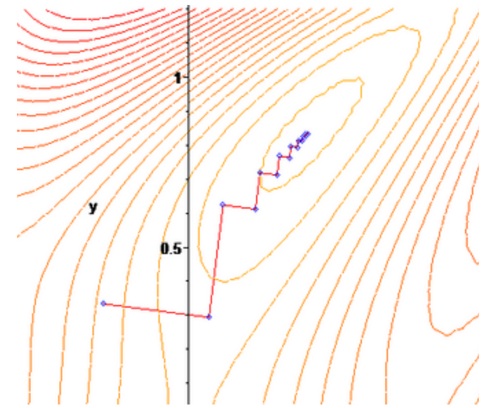


Fig from “Deep Learning” by Goodfellow et al.
http://www.deeplearningbook.org/contents/numerical.html

Gradient Descent

Assume we have a continuous function: $f(x_1,x_2,\ldots,x_N)$
and we want minimize over continuous variables X1,X2,..,Xn



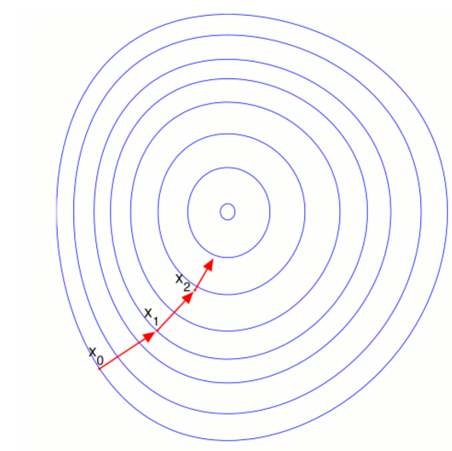
1. Compute the *gradients* for all *i:* $\partial f(x_1,x_2,\ldots,x_N) /\partial x_i$

2. Take a small step downhill in the direction of the gradient:

$$x_i \leftarrow x_i - \lambda \partial f(x_1,x_2,\ldots,x_N) /\partial x_i$$

3. Repeat.

- How to select step size, λ
  - Line search: successively double
    - until *f* starts to increase again



Higher Order Derivatives

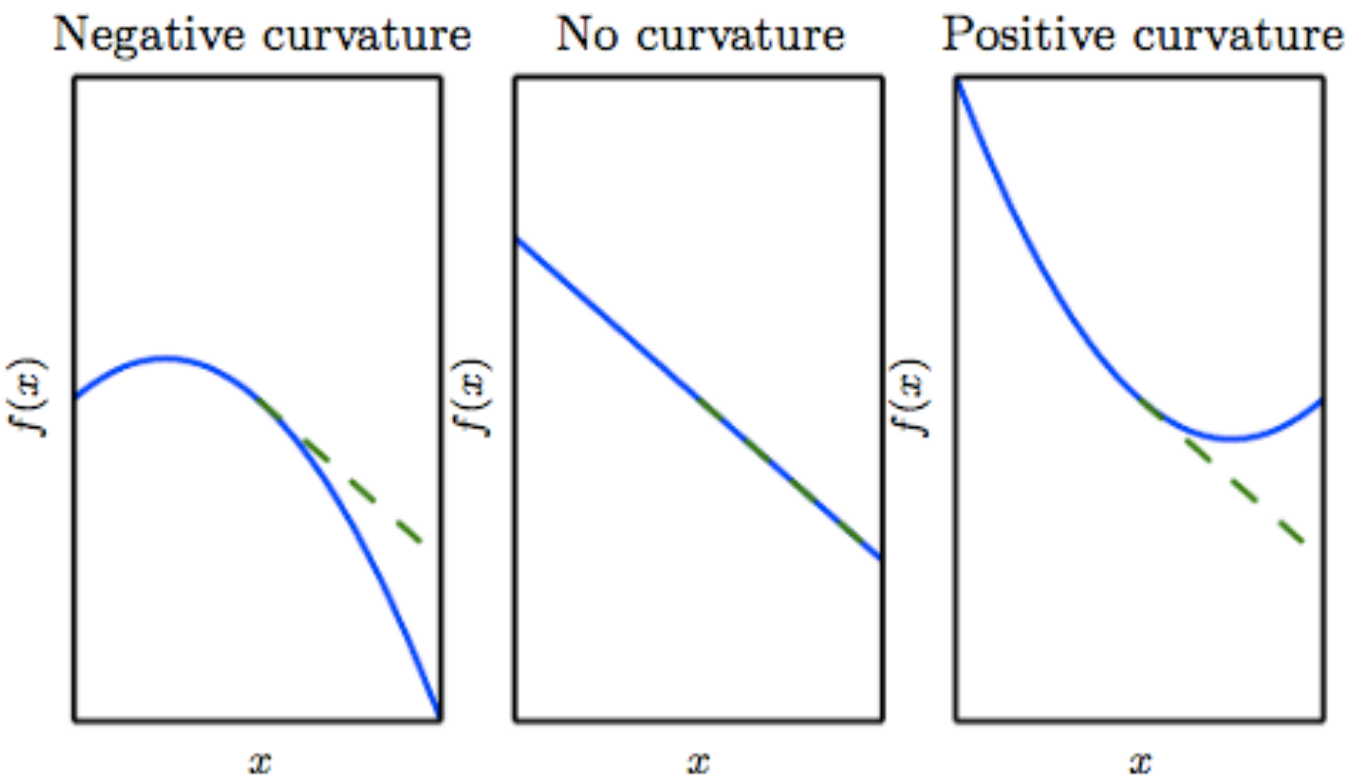
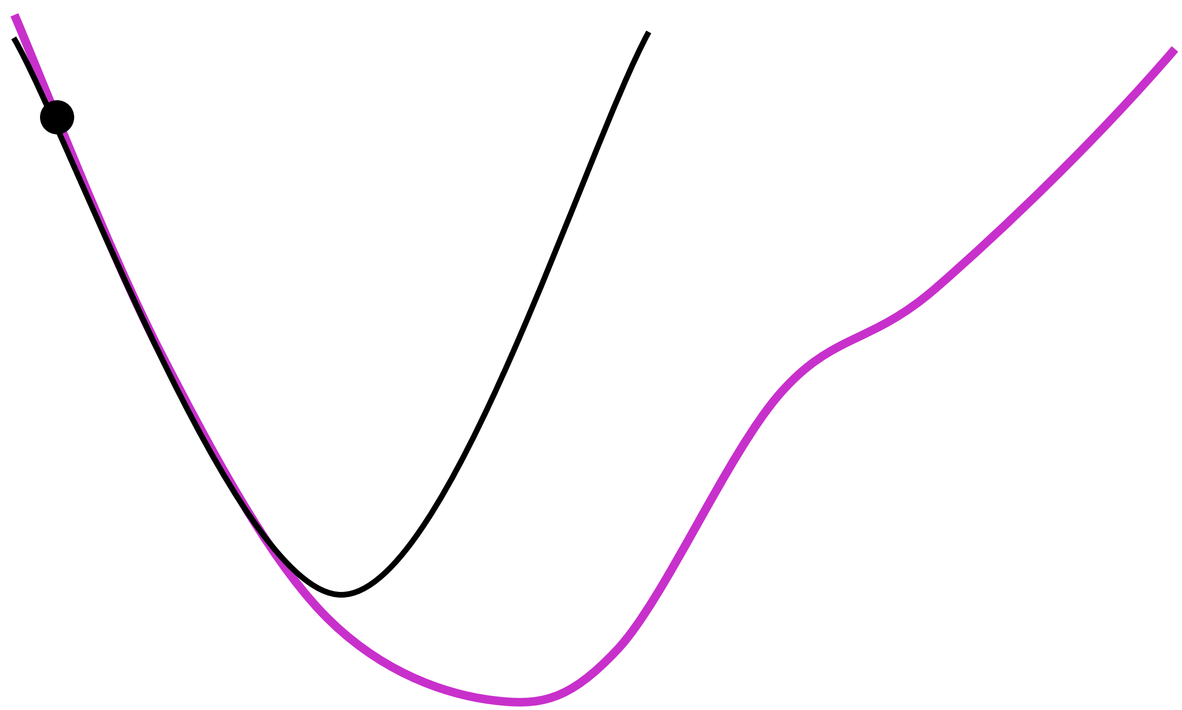


Fig from “Deep Learning” by Goodfellow et al.
http://www.deeplearningbook.org/contents/numerical.html

Newton's Method

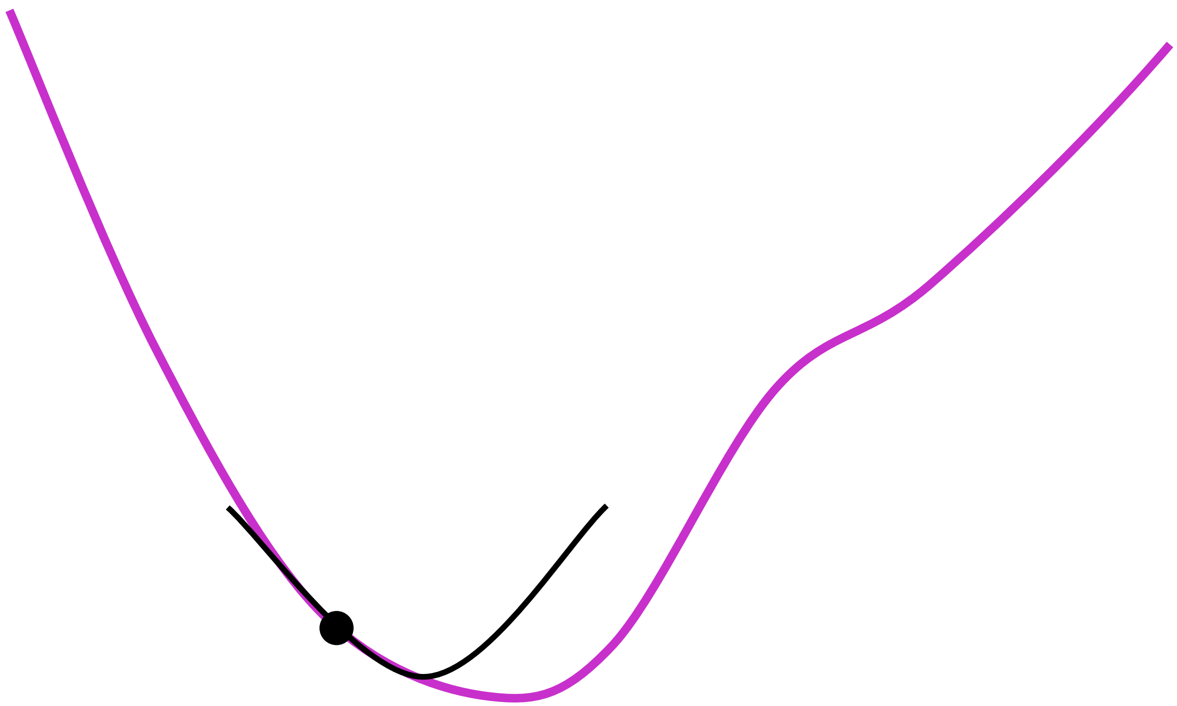
Assume function can be locally approximated with quadratic

Use both first & second derivatives



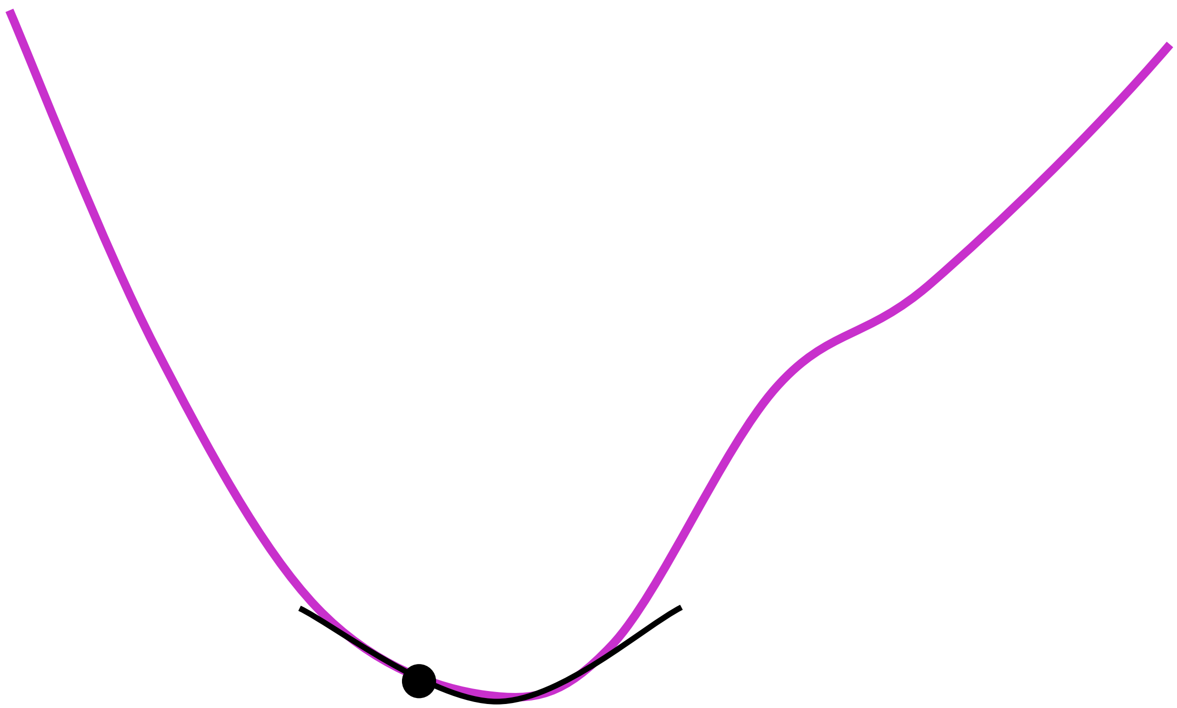
Slide from Princeton COS323 / Szymon Rusinkiewicz

Newton's Method



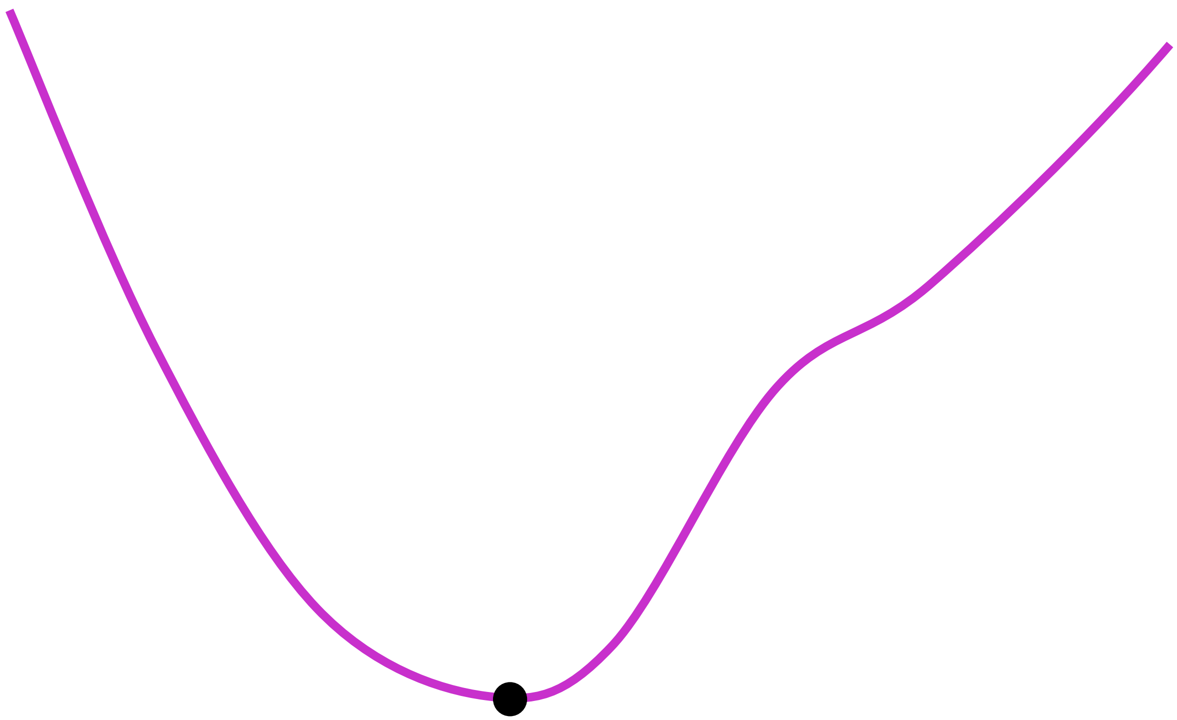
Slide from Princeton COS323 / Szymon Rusinkiewicz

Newton's Method



Slide from Princeton COS323 / Szymon Rusinkiewicz

Newton's Method



Slide from Princeton COS323 / Szymon Rusinkiewicz

Newton's Method

At each step:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Requires 1st and 2nd derivatives

Quadratic convergence

Newton's Method in Multiple Dimensions

Replace 1st derivative with gradient, 2nd derivative with Hessian

$$f(x,y)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

Slide from Princeton COS323 / Szymon Rusinkiewicz

Newton's Method in Multiple Dimensions

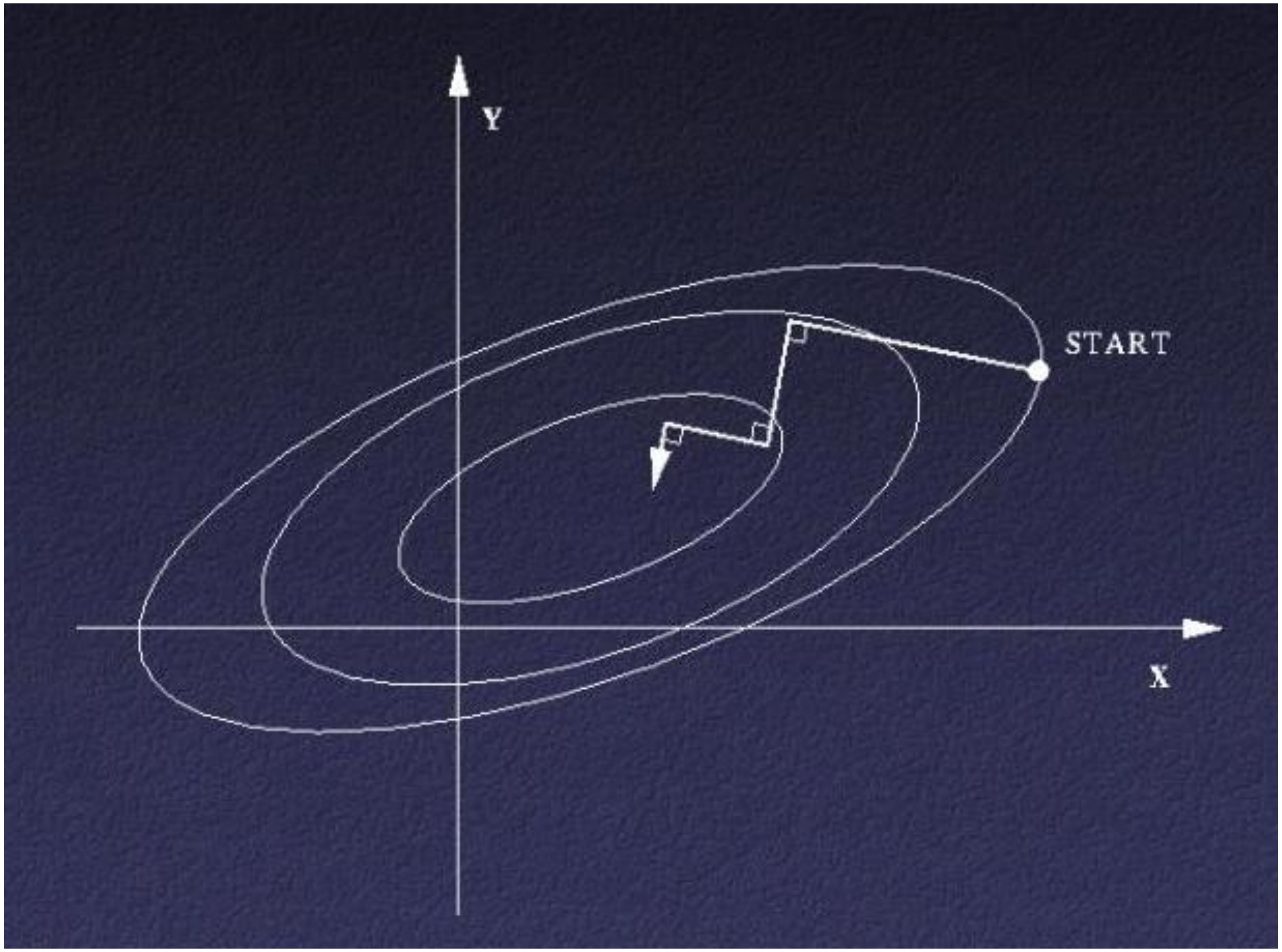
Replace 1st derivative with gradient,
2nd derivative with Hessian

So,

$$\vec{x}_{k+1} = \vec{x}_k - H^{-1}(\vec{x}_k)\nabla f(\vec{x}_k)$$

Tends to be extremely fragile unless function very smooth and starting close to minimum

Problem With Steepest Descent



Slide from Princeton COS323 / Szymon Rusinkiewicz

Conjugate Gradient Methods

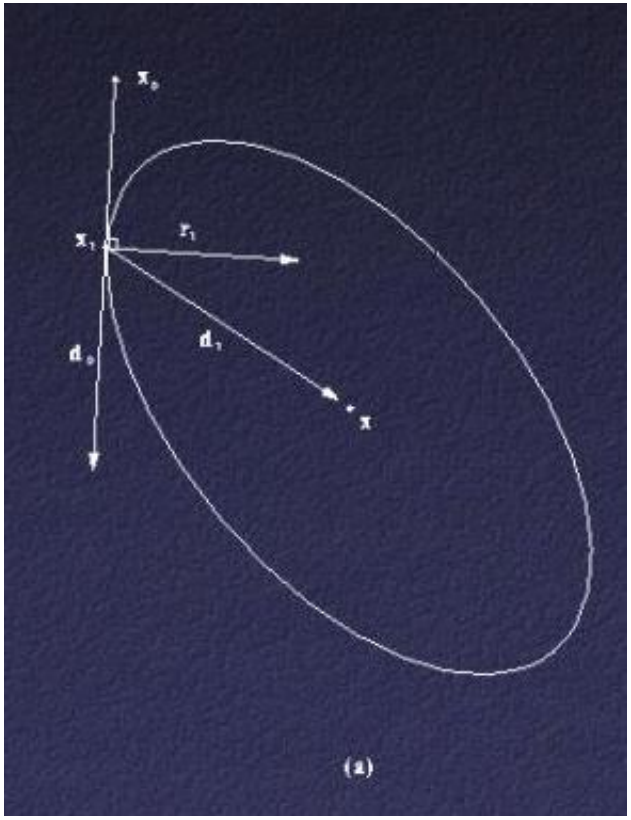
Idea: avoid "undoing" minimization that's already been done

Walk along direction

$$d_{k+1} = -g_{k+1} + \beta_k d_k$$

Polak and Ribiere formula:

$$\beta_k = \frac{g_{k+1}^{T}(g_{k+1} - g_k)}{g_k^{T} g_k}$$



Conjugate Gradient Methods

Conjugate gradient implicitly obtains information about Hessian

For quadratic function in *n* dimensions, gets *exact* solution in *n* steps (ignoring roundoff error)

Works well in practice…