343H: Honors AI

Lecture 24:

ML: Decision trees and neural networks

4/22/2014

Kristen Grauman

UT Austin

Slides courtesy of Dan Klein, UC Berkeley

Last time

- Perceptrons
  - MIRA
  - Dual/kernelized perceptron
- Support vector machines
- Nearest neighbors
- Clustering
  - K-means
  - Agglomerative

Quiz

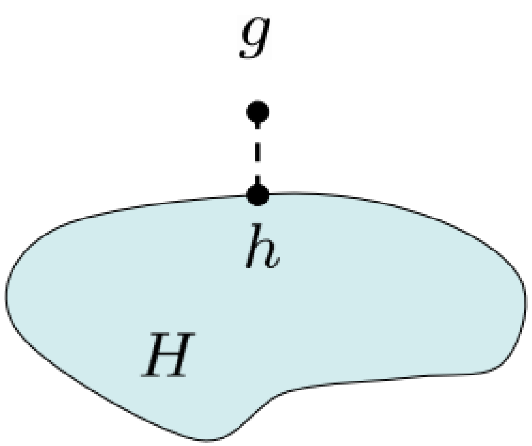
- What distinguishes the learning objectives for MIRA and SVMs?
- What is a support vector?
- Why do we care about kernels?
- Does k-means converge?
- How would we know which of two runs of k-means is better?
- What does it mean to have a parametric vs. non-parametric model?
- How would clusters with k-means differ from those found with agglomerative using “closest-pair” similarity?
- How can clustering achieve feature space discretization?

Today

- Formalizing learning
  - Consistency
  - Simplicity
- Decision trees
  - Expressiveness
  - Information gain
  - Overfitting
- Neural networks

Inductive learning

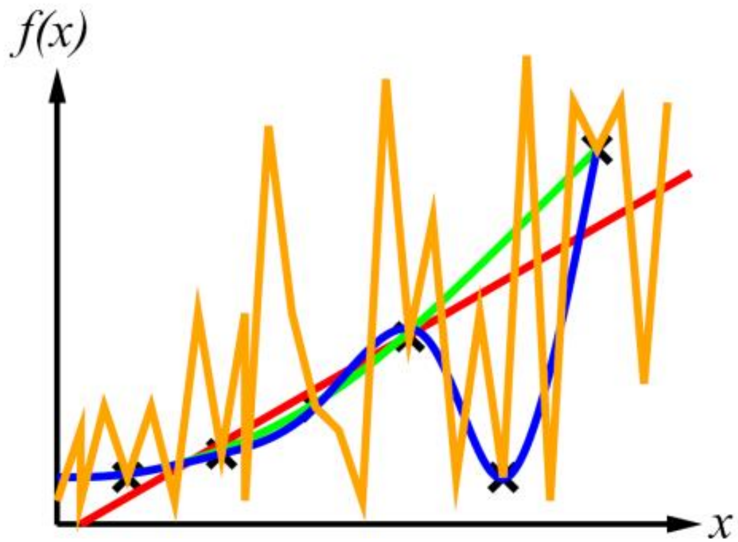
- Simplest form: learn a function from examples
  - A target function: g
  - Examples: input-output pairs (x, g(x))
    - E.g., x is an email and g(x) is spam/ham
    - E.g., x is a house and g(x) is its selling price
- Problem:
  - Given a hypothesis space H
  - Given a training set of examples x_i
  - Find a hypothesis h(x) such that h~g



- Includes
  - Classification, Regression
- How do perceptron and naïve Bayes fit in?

Inductive learning

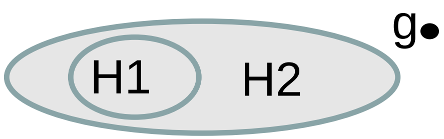
- Curve fitting (regression, function approximation)



- Consistency vs. simplicity
- Ockham's razor

Consistency vs. simplicity

- Fundamental tradeoff: bias vs. variance



- Usually algorithms prefer consistency by default
- Several ways to operationalize "simplicity"
  - Reduce the hypothesis space
    - Assume more: e.g., independence assumptions, as in Naïve Bayes
    - Have fewer, better features/attributes: feature selection
    - Other structural limitations
  - Regularization
    - Smoothing: cautious use of small counts
    - Many other generalization parameters (pruning cutoffs today)
    - Hypothesis space stays big, but harder to get to the outskirts

Reminder: features

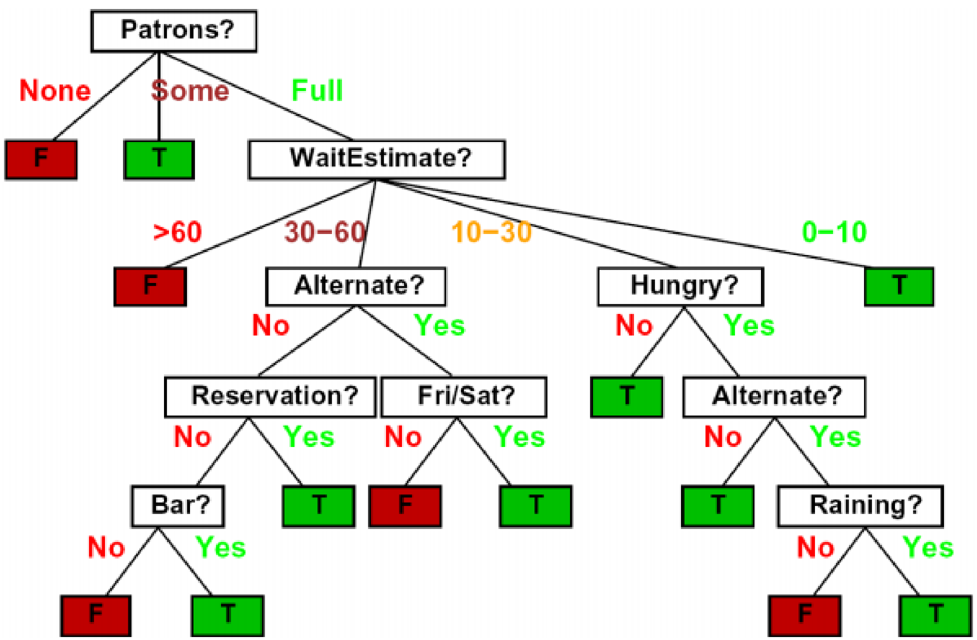
- Features, aka attributes
  - Sometimes: TYPE = French
  - Sometimes $f_{\text{TYPE=French}}(x) = 1$

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision trees

- Compact representation of a function
  - Truth table
  - Conditional probability table
  - Regression values

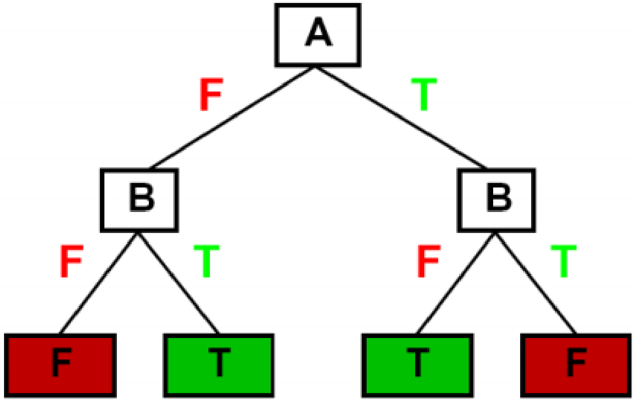
- True function
  - Realizable: in H



Expressiveness of DTs

- Can express any function of the features

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- However, we hope for compact trees

Comparison: Perceptrons

- What is expressiveness of perceptron over these features?

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*

- For a perceptron, feature's contribution either pos or neg
  - If you want one feature's effect to depend on another, you have to add a new conjunction feature
- DTs automatically conjoin features/attributes
  - Features can have different effects in different branches of the tree!

Hypothesis spaces

- How many distinct decision trees with n Boolean attributes?
  - = number of Boolean functions over n attributes
  - = number of distinct truth tables with 2^n rows
  - = $2^{(2^n)}$
    - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees
- How many trees of depth 1 (decision stumps)?
  - = number of Boolean functions over 1 attribute
  - = number of truth tables with 2 rows, times n
  - =4n
    - E.g. with 6 Boolean attributes, there are 24 decision stumps

Hypothesis spaces

- More expressive hypothesis space:
  - Increases chance that target function can be expressed (good)
  - Increases number of hypotheses consistent with training set (bad)
  - Means we can get better predictions (lower bias)
  - But we may get worse predictions (higher variance)

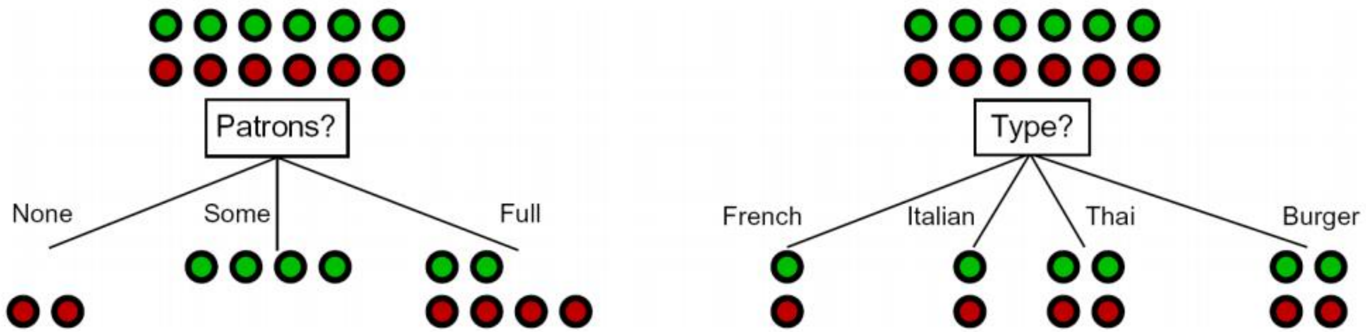
Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- So: we need a measure of how "good" a split is, even if the results aren't perfectly separated

Entropy and information

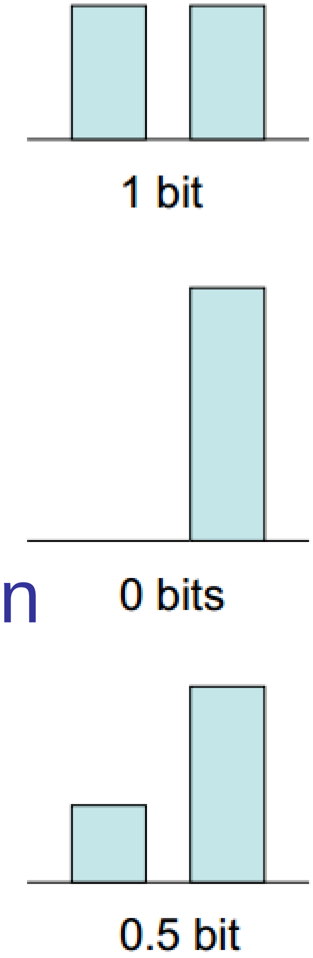
- Information answers questions
  - The more uncertain about the answer initially, the more information in the answer
  - Scale: bits
    - Answer to a Boolean question with prior <1/2,1/2>?
    - Answer to a 4-way question with prior <¼, ¼, ¼, ¼>?
    - Answer to a 4-way question with prior <0,0,0,1>?
    - Answer to a 3-way question with prior <1/2,1/4,1/4>?
- A probability p is typical of:
  - A uniform distribution of size 1/p
  - A code of length log 1/p

Entropy

- General answer: if prior is <$p_1,\ldots,p_n$>
  - Information is the expected code length

$$H(\langle p_1,\ldots,p_n\rangle) = E_p \log_2 1/p_i$$

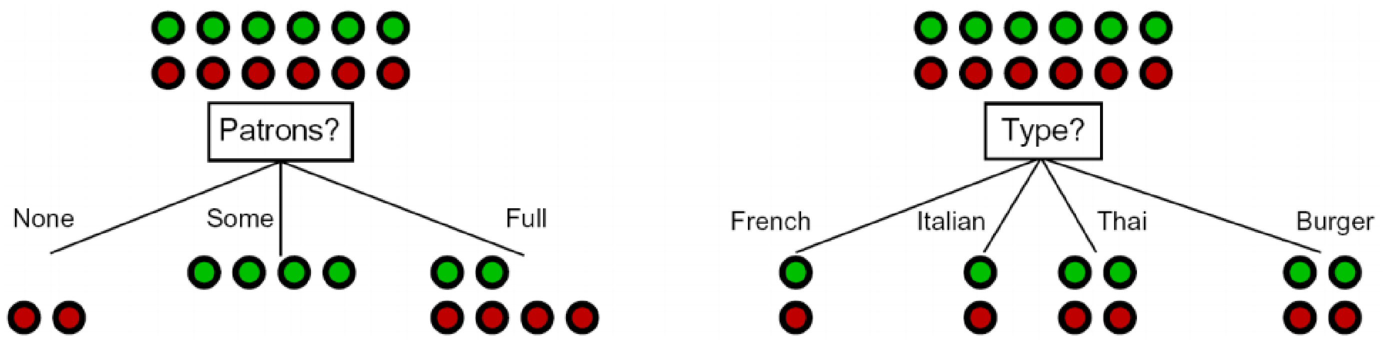
$$= \sum_{i=1}^{n} -p_i \log_2 p_i$$



- Also called the entropy of the distribution
  - More uniform = higher entropy
  - More values = higher entropy
  - More peaked = lower entropy
  - Rare values almost “don’t count”

Information gain

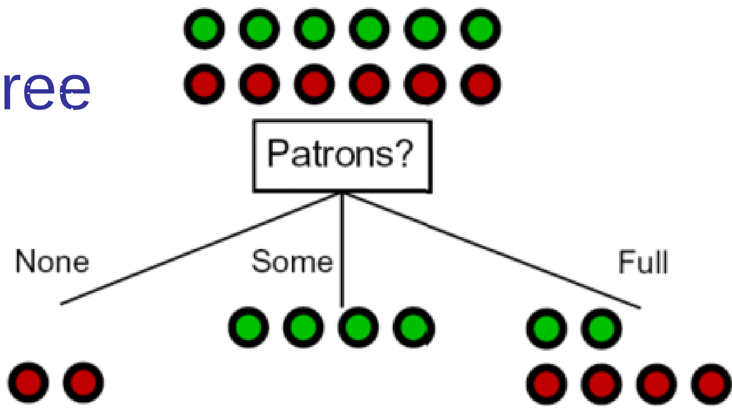
- Back to decision trees!
- For each split, compare entropy before and after
  - Difference is the information gain



  - Problem: there's more than one distribution after split!
  - Solution: use expected entropy, weighted by the number of samples

Next step: Recurse

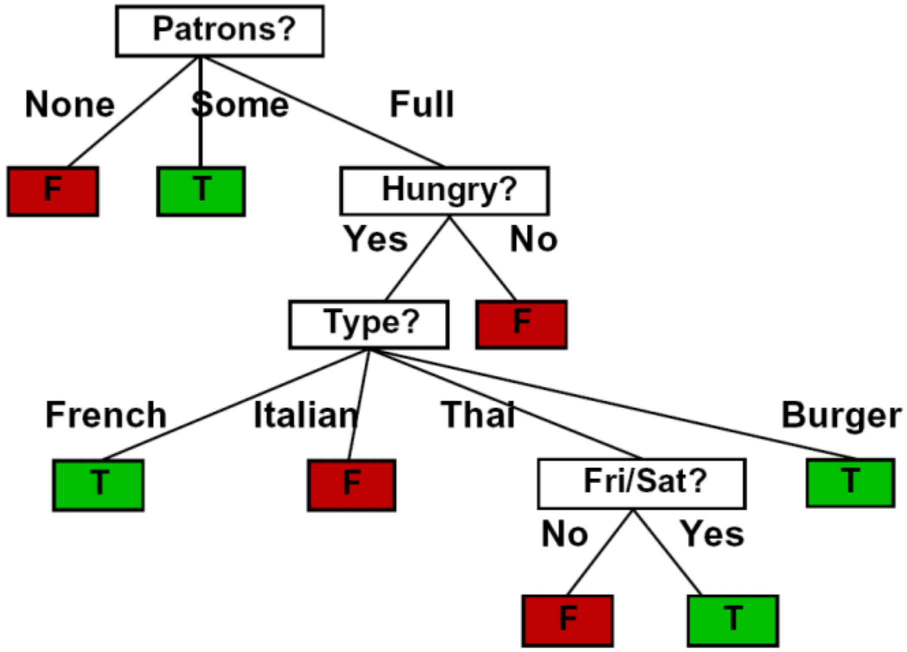
- Now we need to keep growing the tree
- What to do under "full"?



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

Example: learned tree

- Decision tree learned from these 12 examples:



- Substantially simpler than “true” tree
  - A more complex hypothesis isn’t justified by data

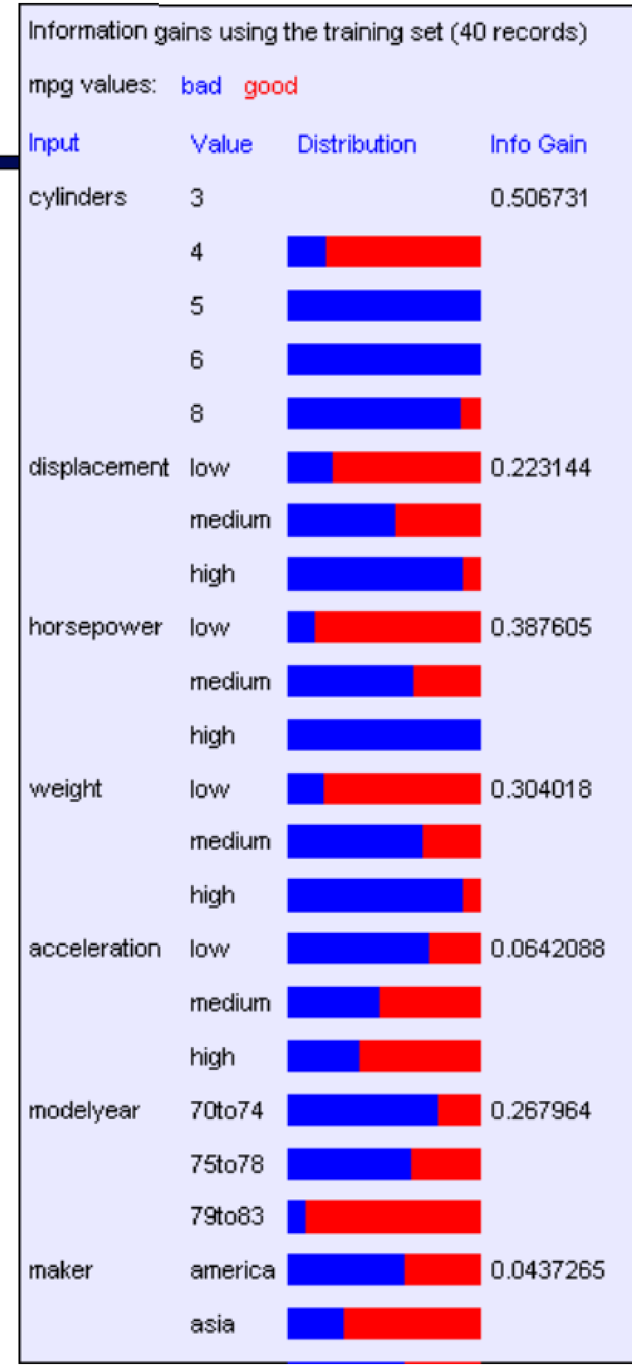
Example: Miles per gallon

40 Examples

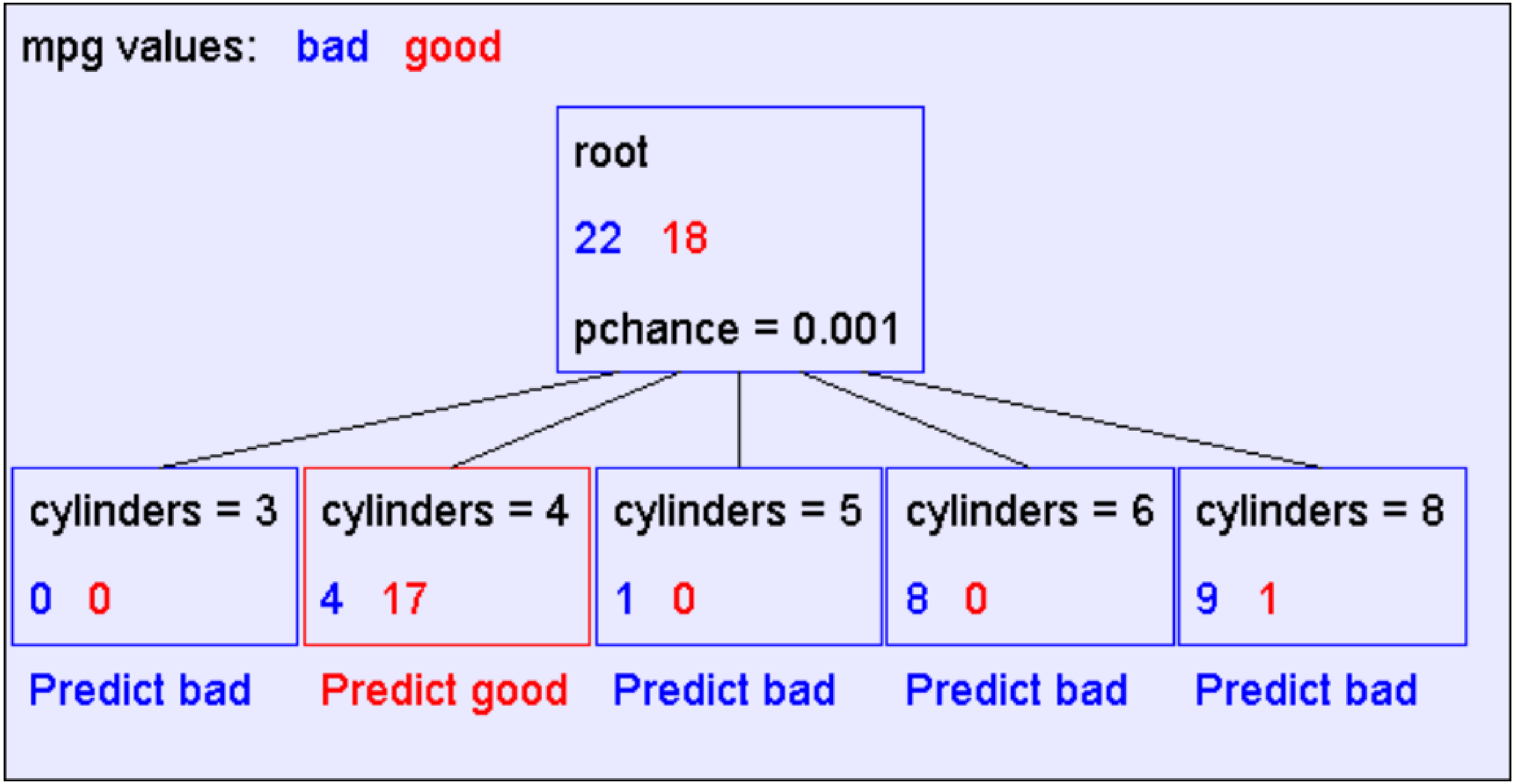
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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

Find the first split

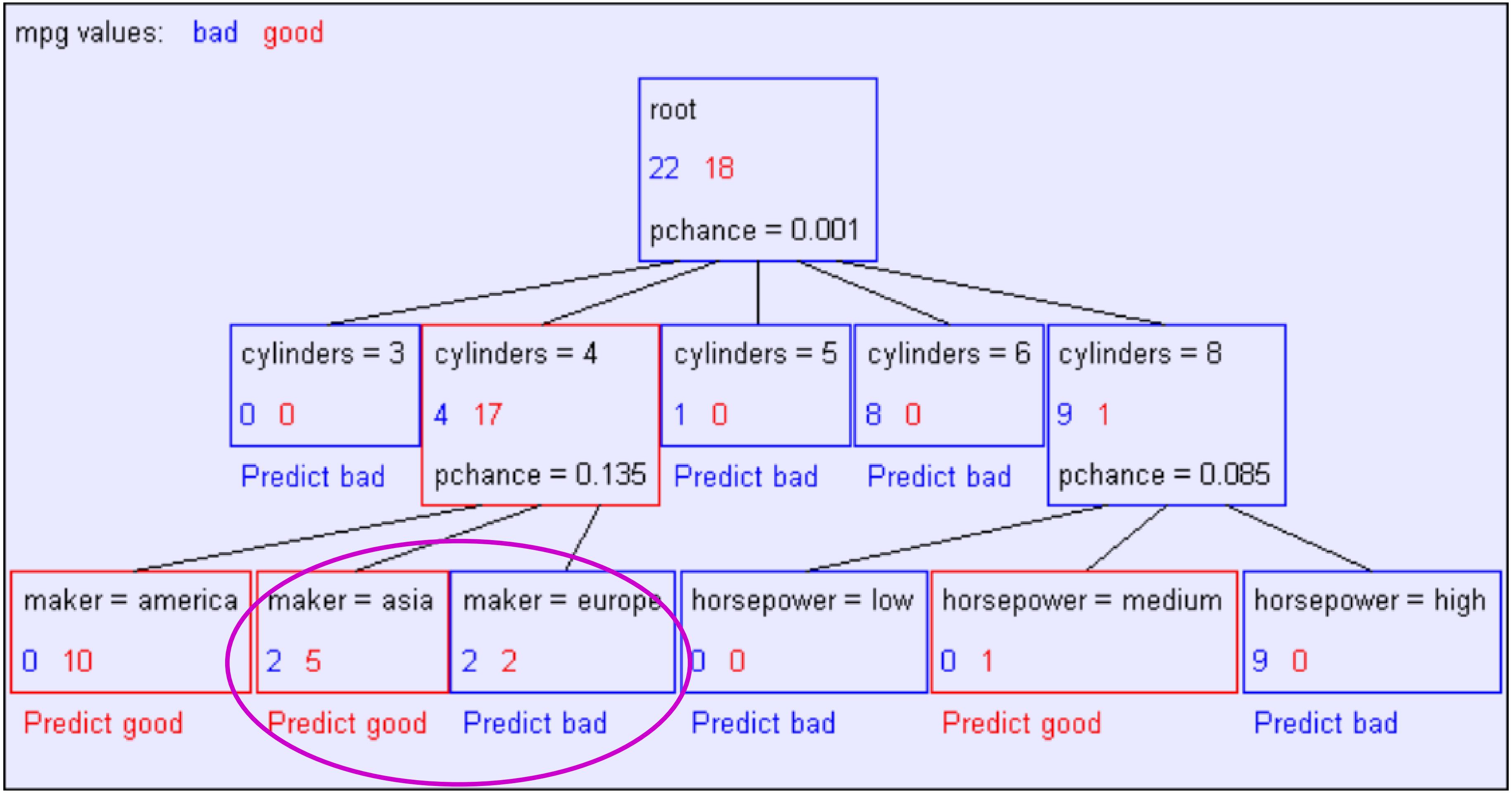
- Look at information gain for each attribute
- Note that each attribute is correlated with the target
- What do we split on?



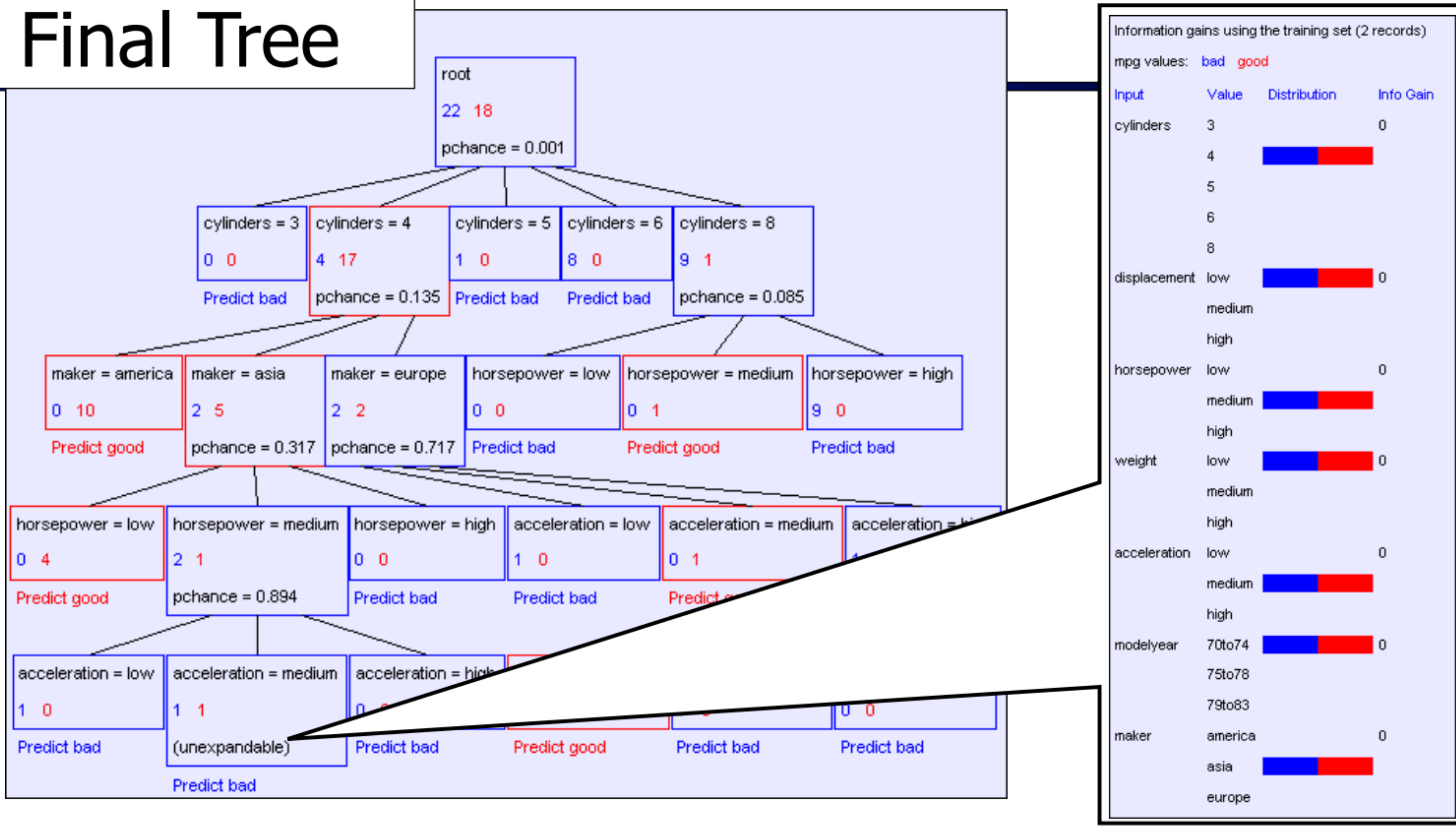
Result: Decision stump



Second level



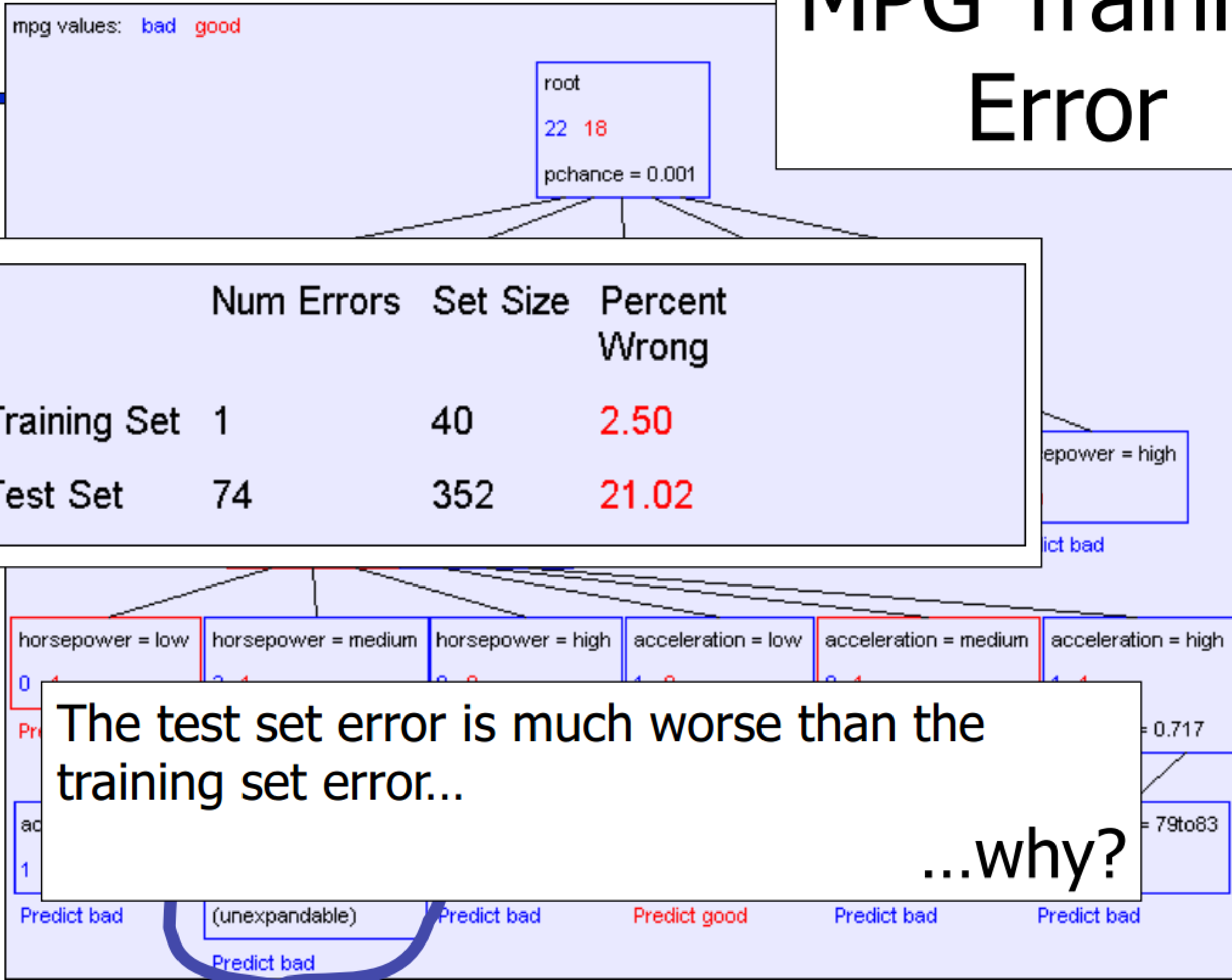
Final Tree


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
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
Predict bad
Predict good
Predict bad
0 0
Predict bad
Information gains using the training set (2 records)
mpg values: bad good
Input
Value
Distribution
Info Gain
cylinders
3
0
4
5
6
8
displacement
low
0
medium
high
horsepower
low
0
medium
high
weight
low
0
medium
high
acceleration
low
0
medium
high
modelyear
70to74
0
75to78
79to83
maker
america
0
asia
europe

Reminder: overfitting

- Overfitting:
  - When you stop modeling the patterns in the training data (which generalize)
  - And start modeling the noise (which doesn’t)
- We had this before:
  - Naïve Bayes: needed to smooth
  - Perceptron: early stopping

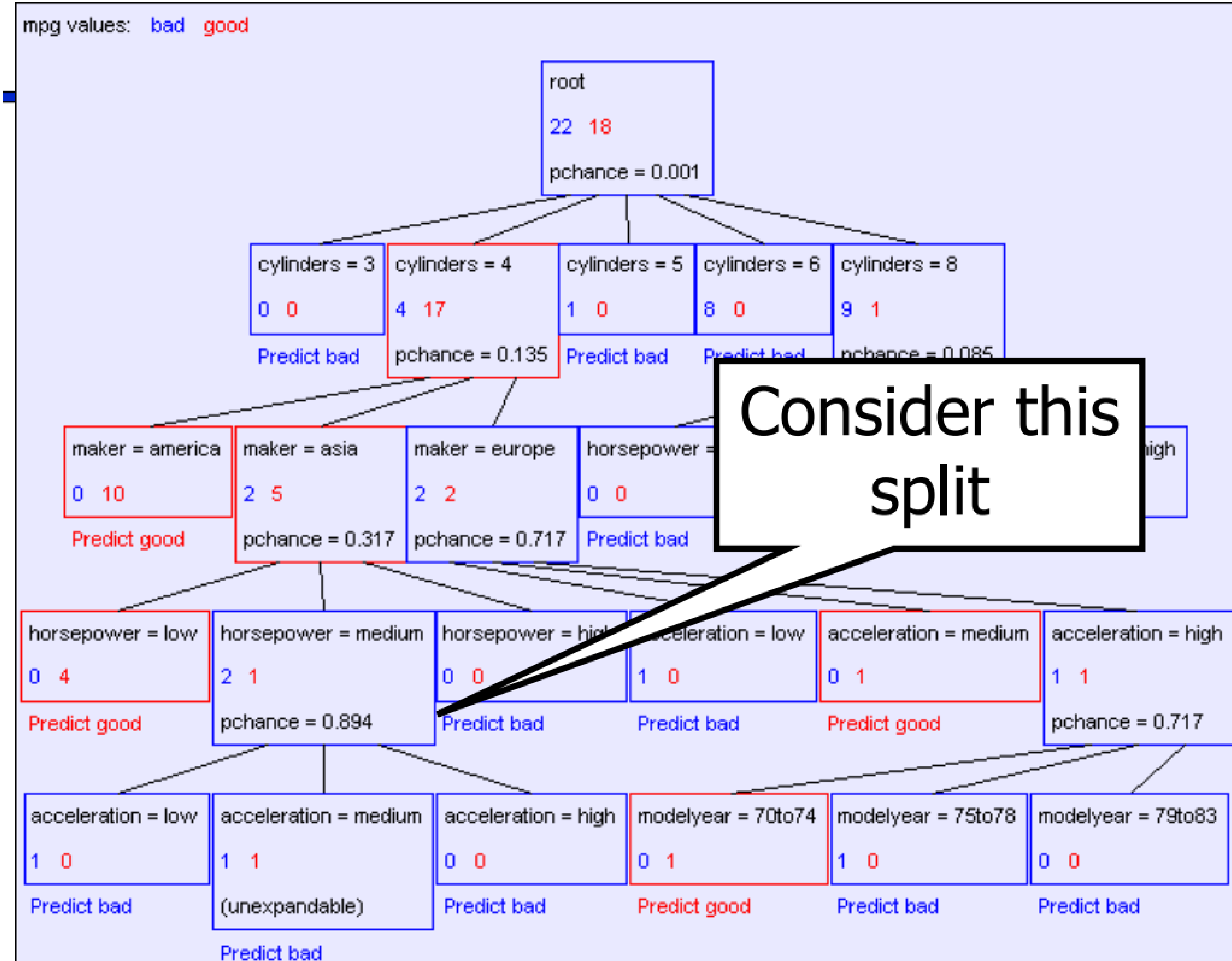
MPG Training Error



	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?



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
cylinders = 8
9 1
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
0 0
Predict bad
Consider this split
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
0 0
Predict bad
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

Significance of a split

- Starting with:
  - Three cars with 4 cylinders, from Asia, with medium HP
  - 2 bad MPG, 1 good MPG
- What do we expect from a three-way split?
  - Maybe each example in its own subset?
  - Maybe just what we saw on the last slide?
- Probably shouldn't split if the counts are so small they could be due to chance
- A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance
- Each split will have a significance value, p_{CHANCE}

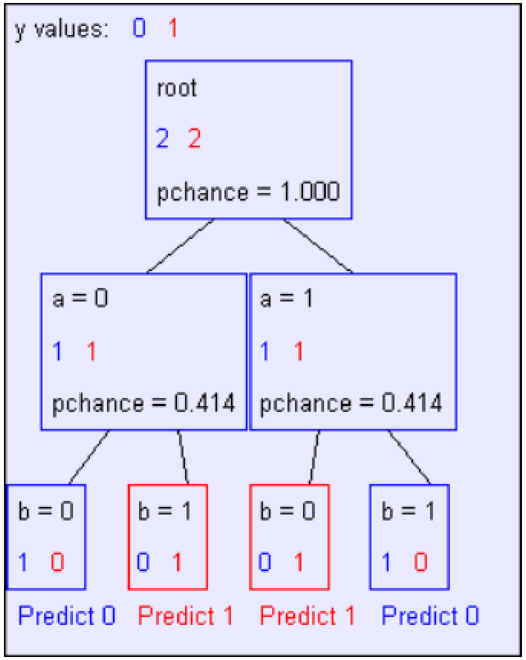
Keeping it general

- Pruning:
  - Build the full decision tree
  - Begin at the bottom of the tree
  - Delete splits in which

    $p_{CHANCE} > \text{Max } p_{CHANCE}$
  - Continue working upward until there are no prunable nodes
  - Note: some chance nodes may not get pruned because they were "redeemed" later

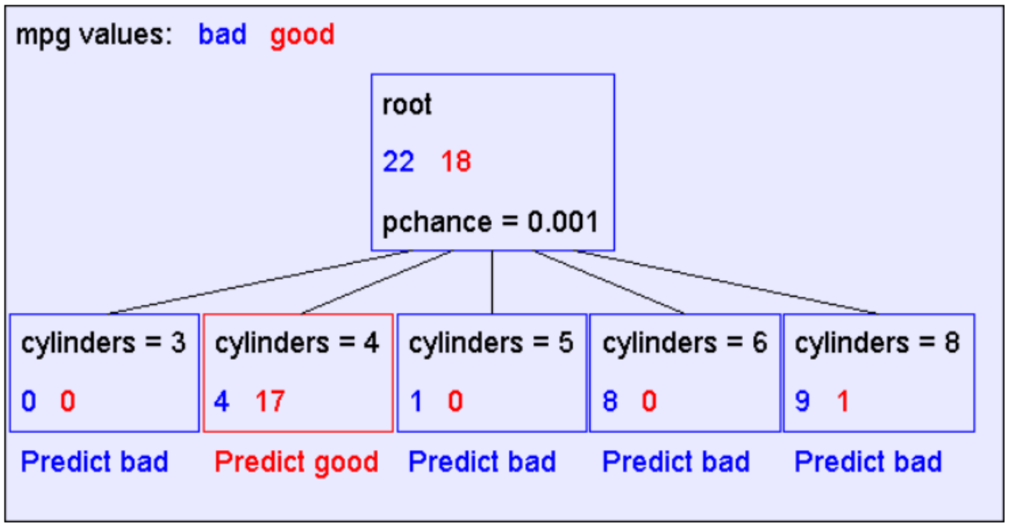
y = a XOR b

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



Pruning example

- With Max $p_{CHANCE} = 0.1$:

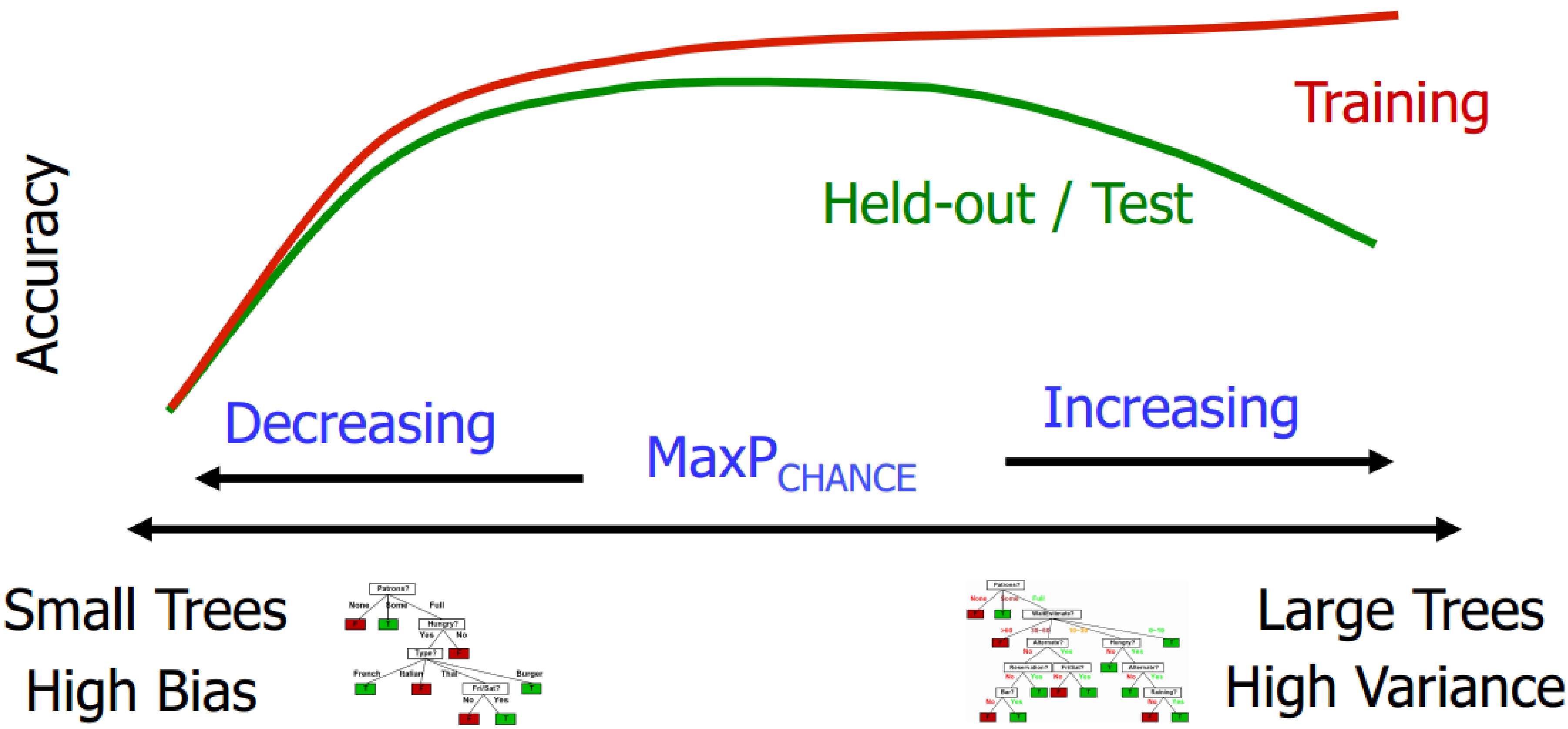


Note the improved test set accuracy compared with the unpruned tree

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

Regularization

- Max p_{CHANCE} is a regularization parameter
- Generally, set it using held-out data (as usual)



Two ways to control overfitting

- Limit the hypothesis space
  - E.g., limit the max depth of trees

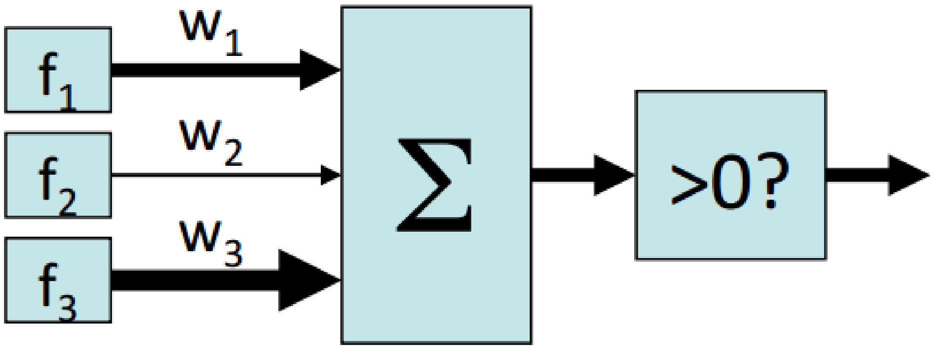
- Regularize the hypothesis selection
  - E.g., chance cutoff
  - Disprefer most of the hypotheses unless data is clear
  - Usually done in practice

Reminder: Perceptron

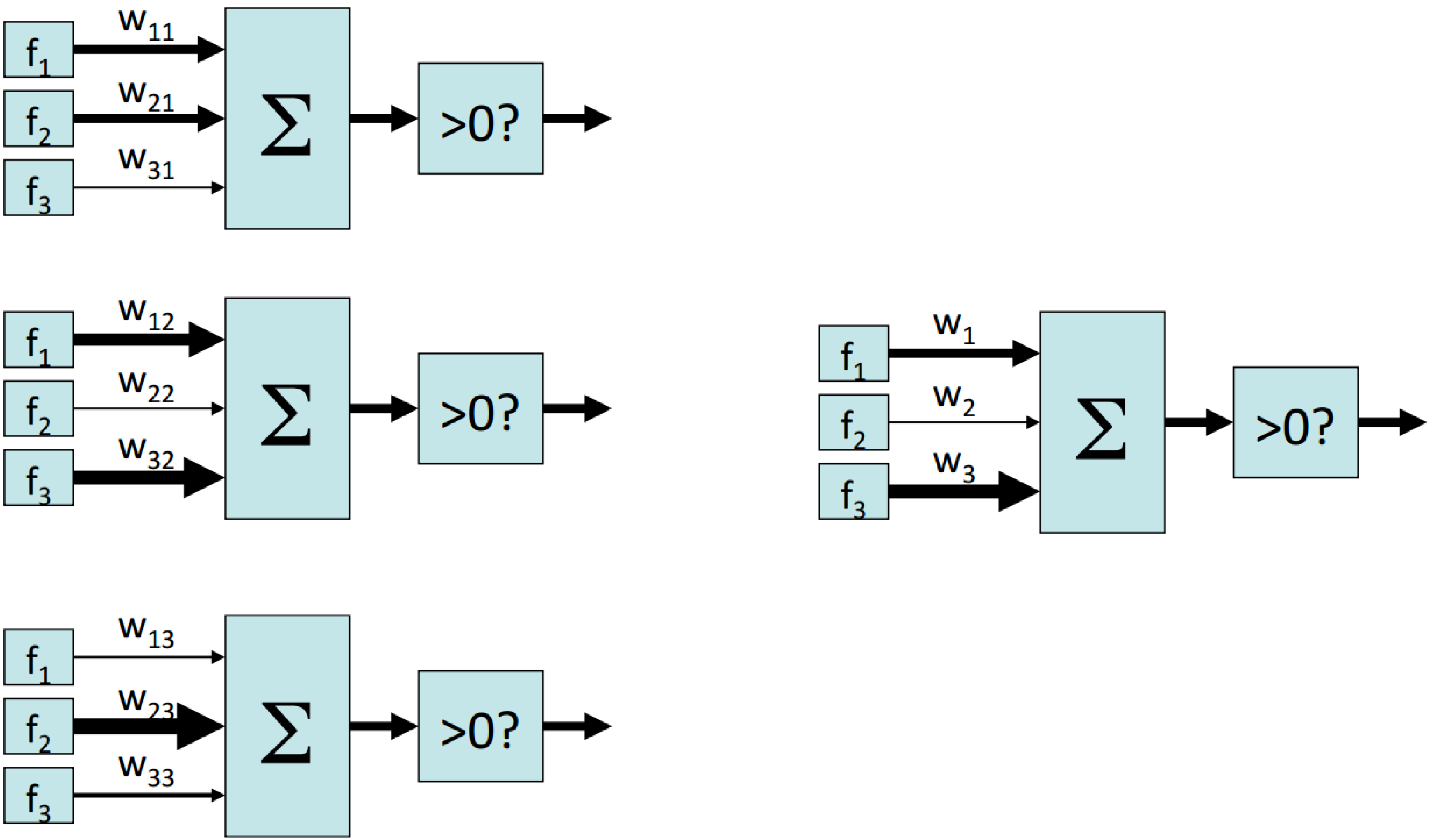
- Inputs are feature values
- Each feature has a weight
- Sum is the activation

$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

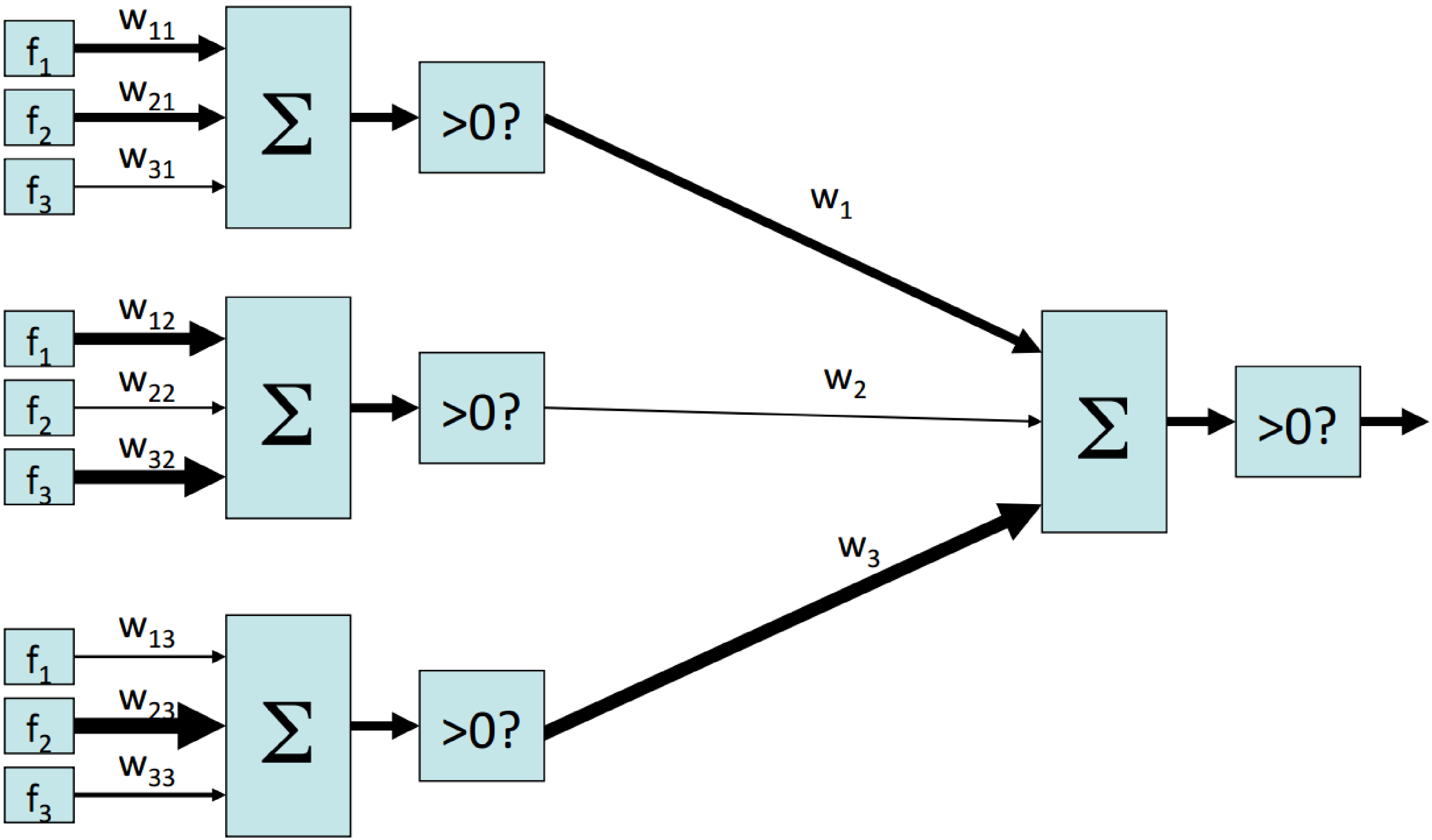
- If the activation is:
  - Positive, output +1
  - Negative, output -1



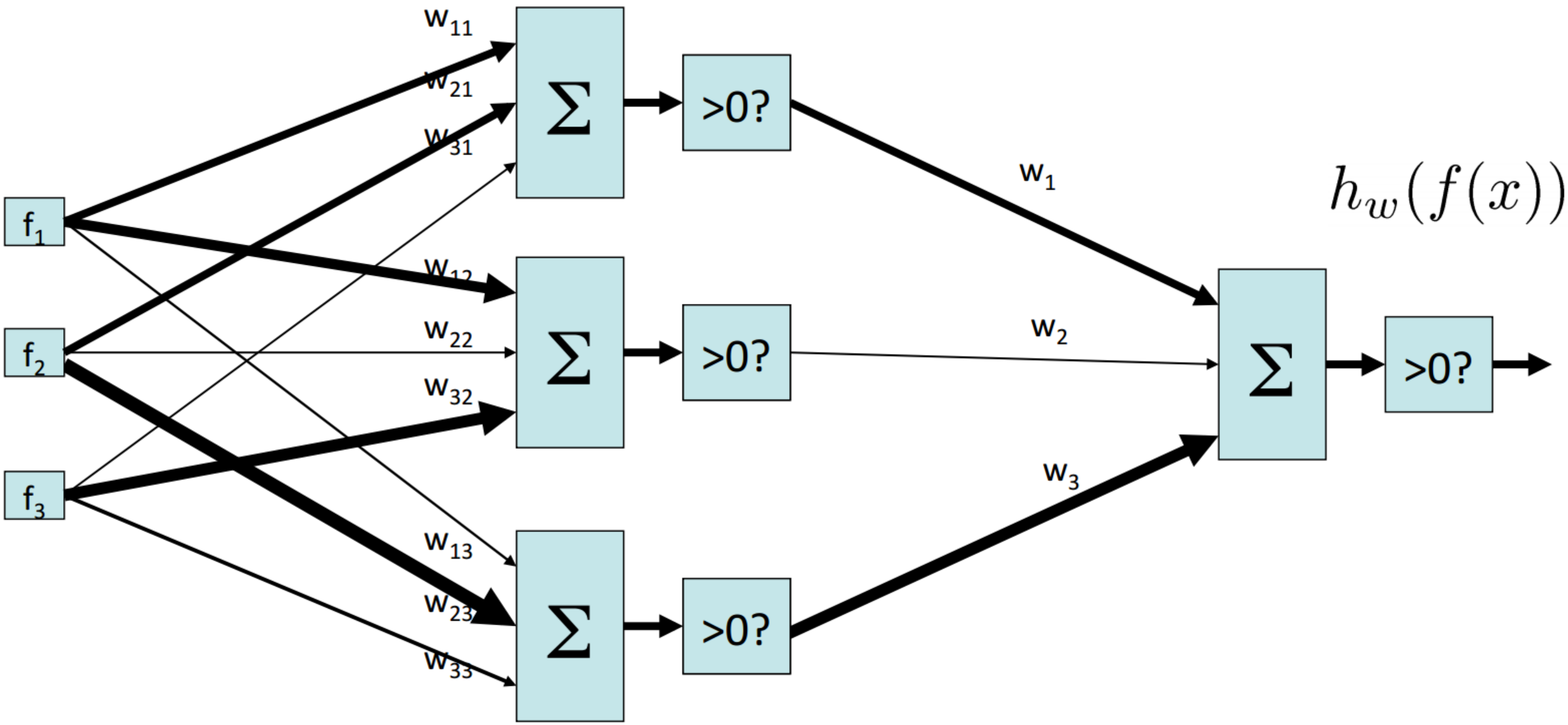
Two-layer perceptron network



Two-layer perceptron network



Two-layer perceptron network



Learning w

- Training examples

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$$

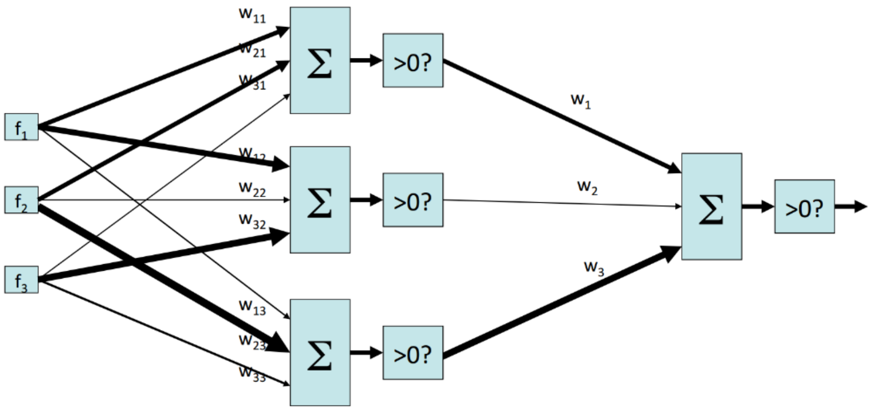
- Objective:

$$\min_w \sum_{i=1}^{m} \left(y^{(i)} - h_w(f(x^{(i)})) \right)^2$$

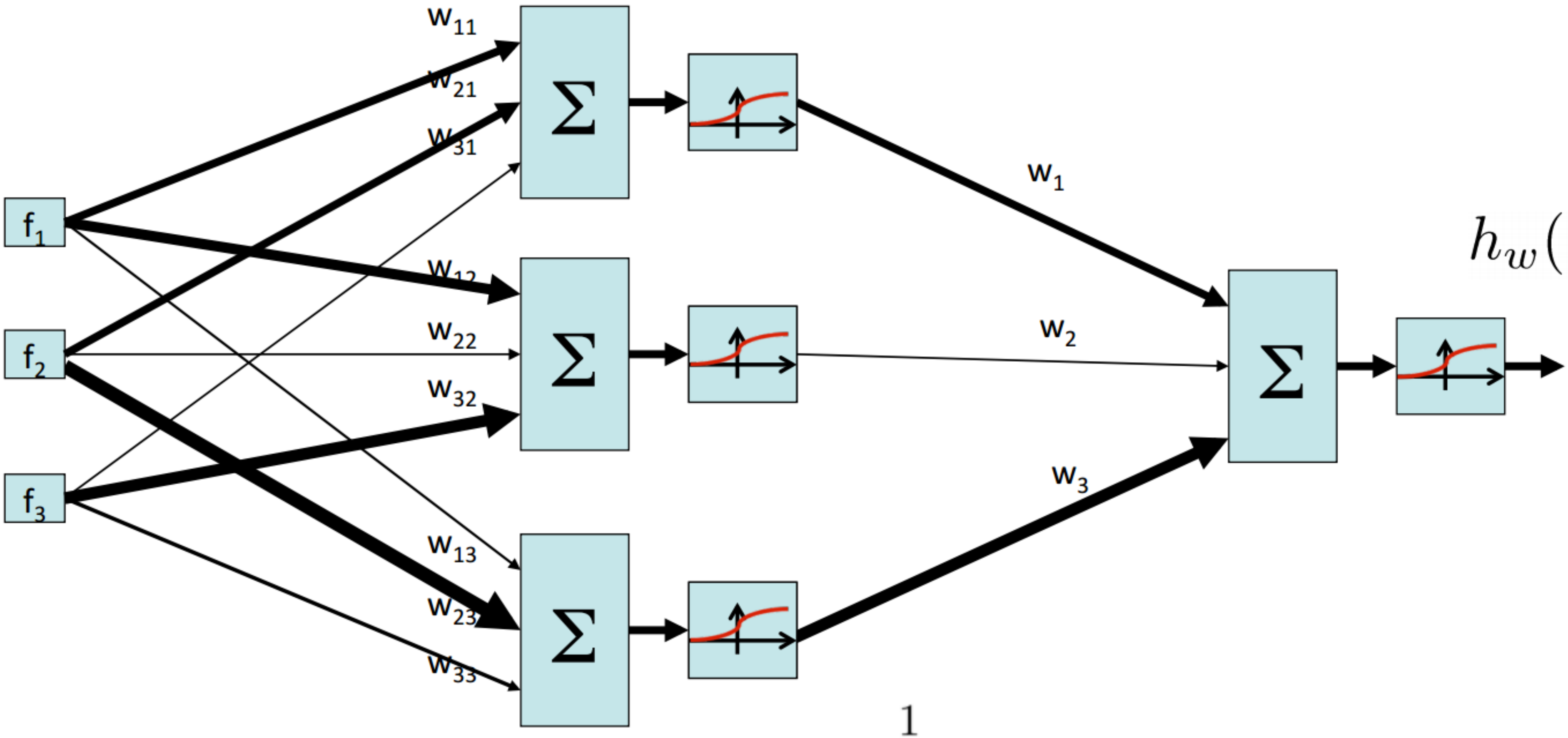
- Procedure:
  - Hill climbing

Hill climbing

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit
  - Neighbors = small perturbations of w
- What's bad?
  - Complete?
  - Optimal?



Two-layer neural network



$$h_w(f$$

$$z \rightarrow \frac{1}{1+e^{-z}}$$

Neural network properties

- Theorem (Universal function approximators): A two-layer network with a sufficient number of neurons can approximate any continuous function to any desired accuracy

- Practical considerations:
  - Can be seen as learning the features
  - Large number of neurons
    - Danger for overfitting
  - Hill-climbing procedure can get stuck in bad local optima

Summary

- Formalization of learning
  - Target function
  - Hypothesis space
  - Generalization
- Decision trees
  - Can encode any function
  - Top-down learning (not perfect!)
  - Information gain
  - Bottom-up pruning to prevent overfitting
- Neural networks
  - Learn features
  - Universal function approximators
  - Difficult to train