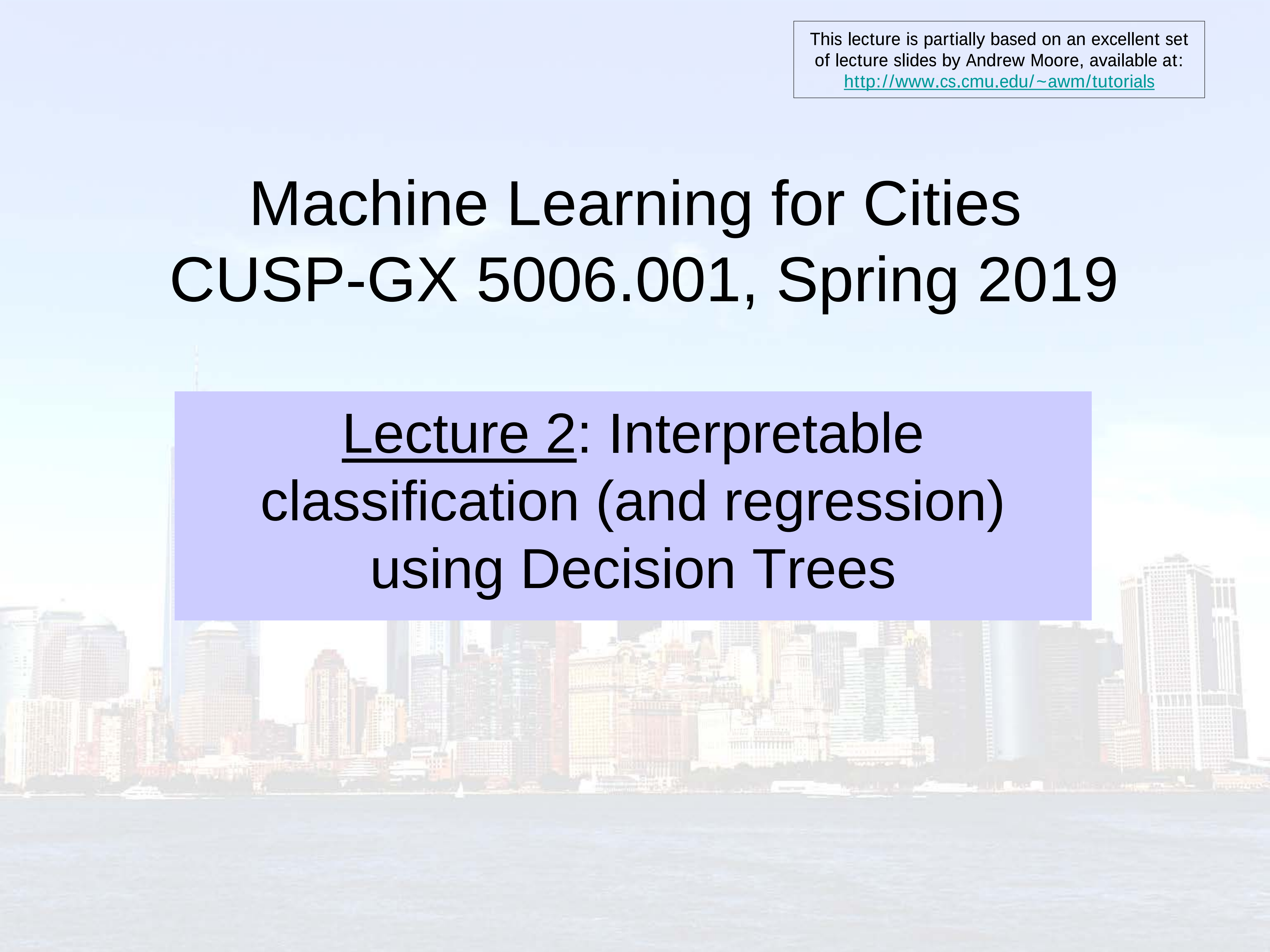


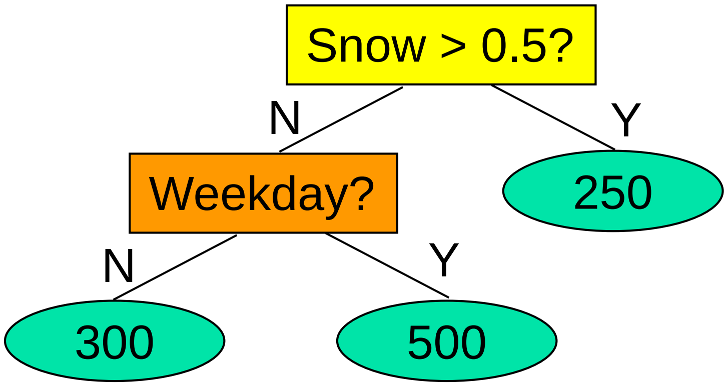
This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Machine Learning for Cities
CUSP-GX 5006.001, Spring 2019

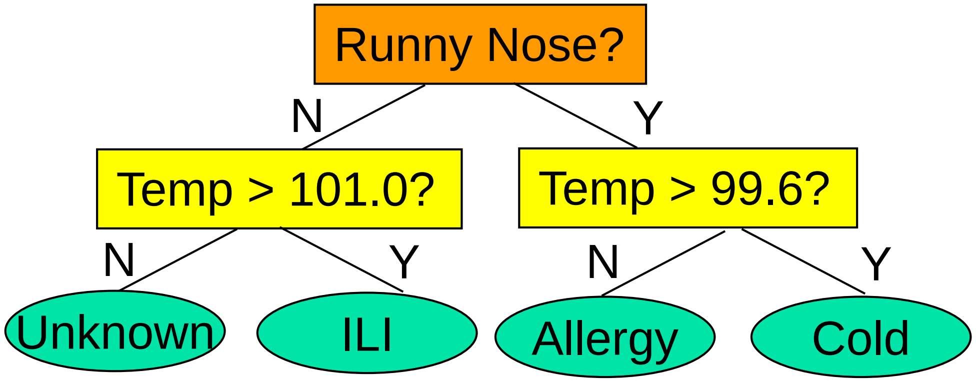
Lecture 2: Interpretable classification (and regression) using Decision Trees

Rule-based learning with decision trees

- A decision tree is a set of rules that can be learned from data and used to predict an unknown value.
  - Unknown real value (regression): What is the expected incidence of car thefts in NYC on a given day?
  - Unknown category value (classification): What type of illness does patient X have, given their symptoms and demographic data?



How many thefts on Tuesday, January 3 (0.2 inches of snow)?



What do we predict for a patient with Temp = 100 and a runny nose?

Learning binary decision trees

Example dataset:
Predicting whether a car is fuel-efficient, given its number of cylinders (4, 6, or 8), weight (light, medium, or heavy), and horsepower (real-valued).

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

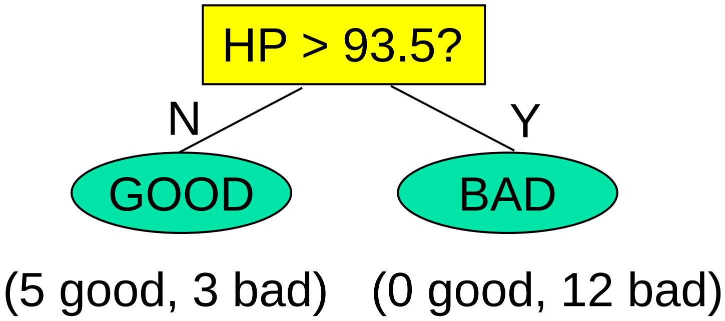
- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).

(5 good, 15 bad)

MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.

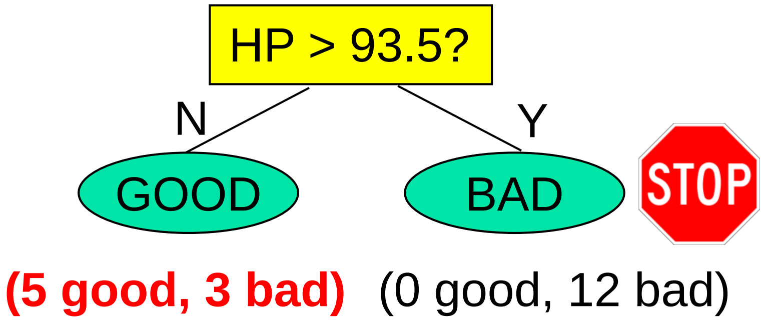


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.

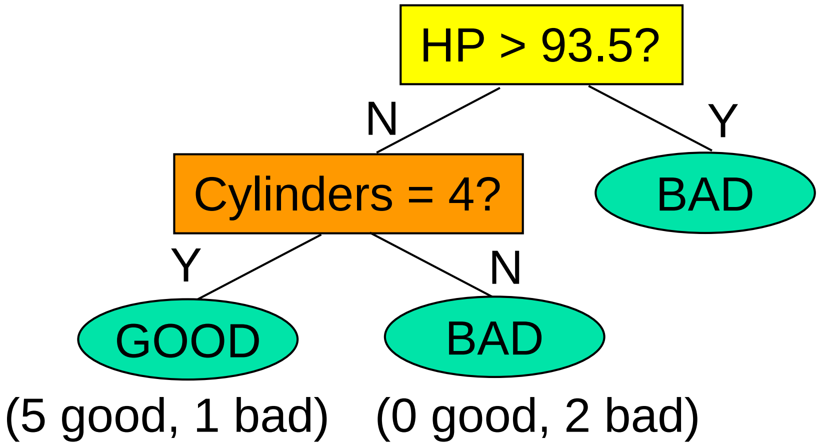


<u>MPG, cylinders, HP, weight</u>

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.

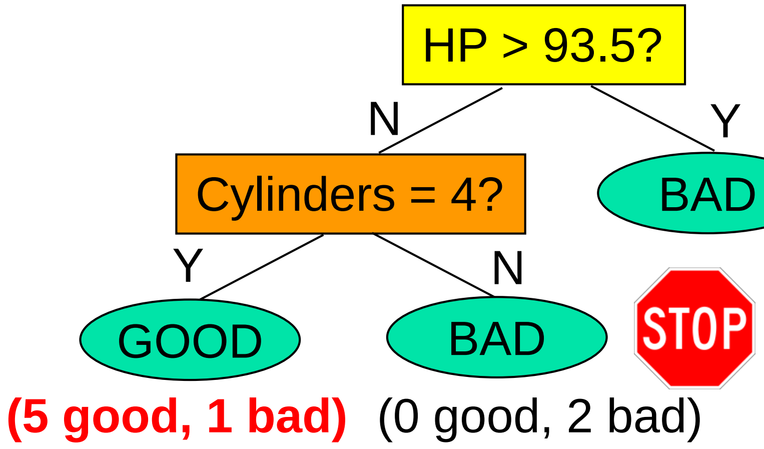


MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

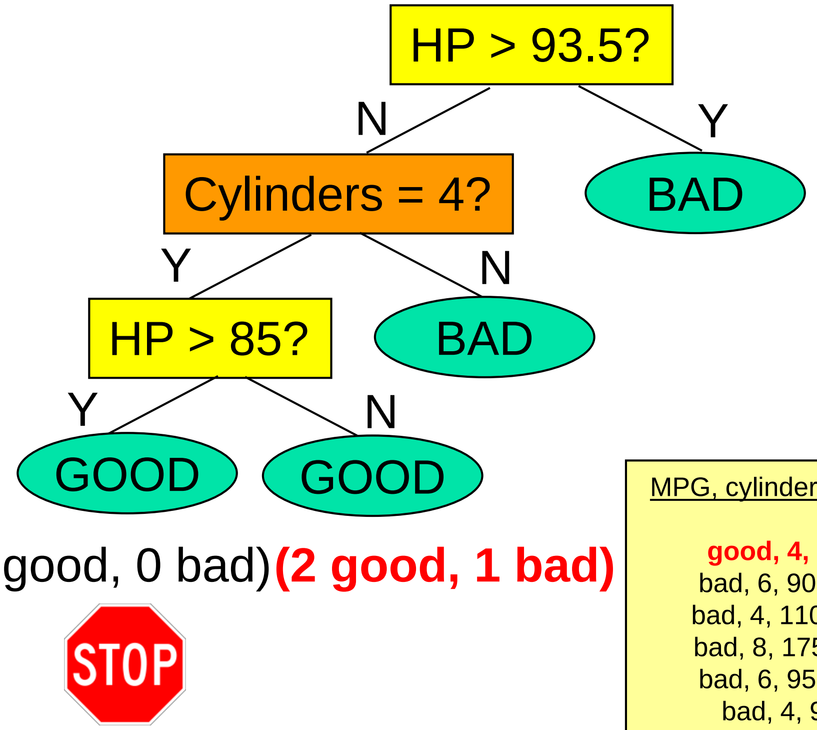
- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.



MPG	cylinders	HP	weight
good	**4**	**75**	**light**
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	**4**	**92**	**medium**
bad	6	100	weighty
bad	8	170	weighty
good	**4**	**89**	**medium**
good	**4**	**65**	**light**
bad	6	85	medium
bad	**4**	**81**	**light**
bad	6	95	medium
good	**4**	**93**	**light**

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.



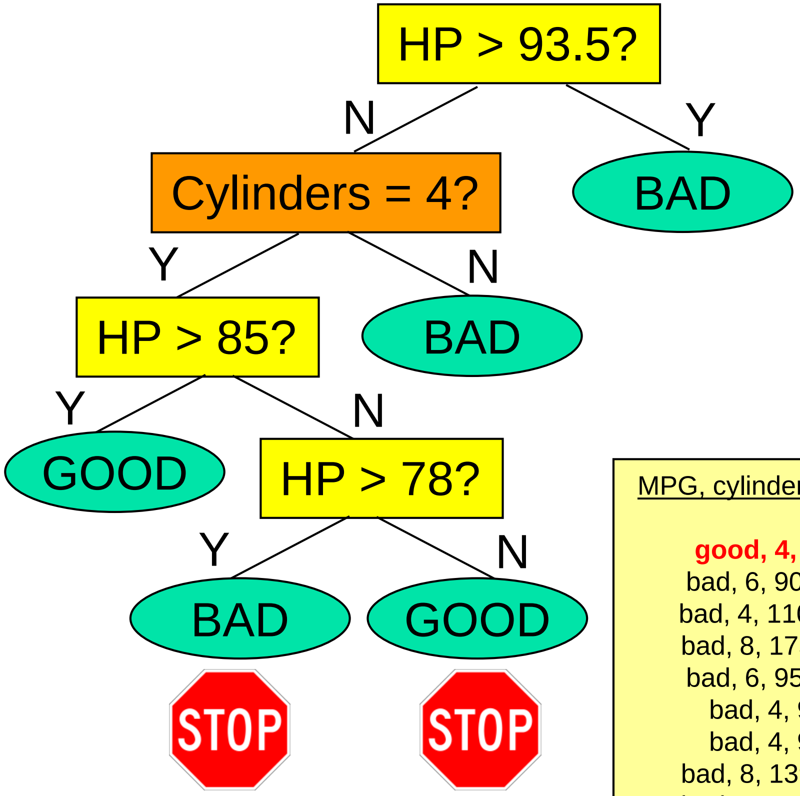
(3 good, 0 bad)**(2 good, 1 bad)**

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.

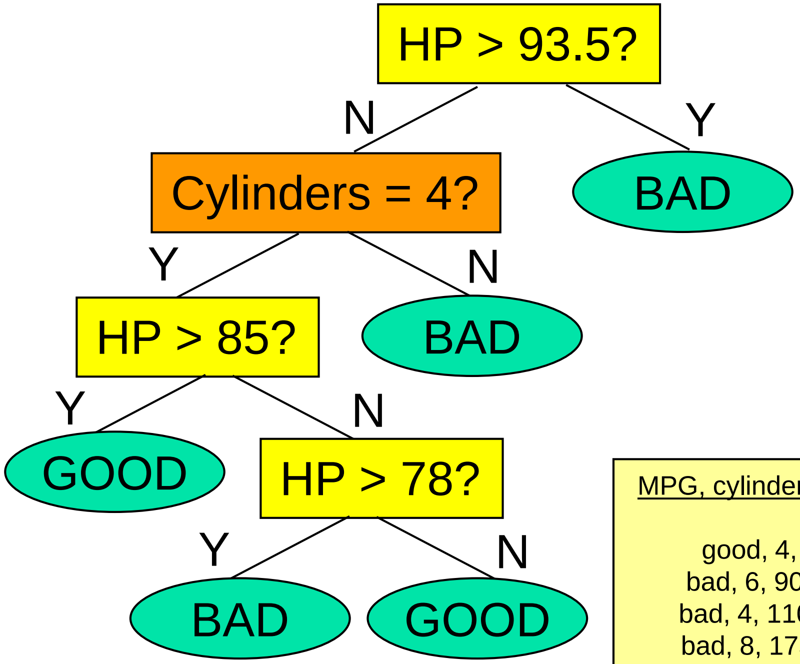


MPG	cylinders	HP	weight
good	**4**	**75**	**light**
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	**4**	**65**	**light**
bad	6	85	medium
bad	**4**	**81**	**light**
bad	6	95	medium
good	4	93	light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
  - **All outputs same?**
  - **All inputs same?**

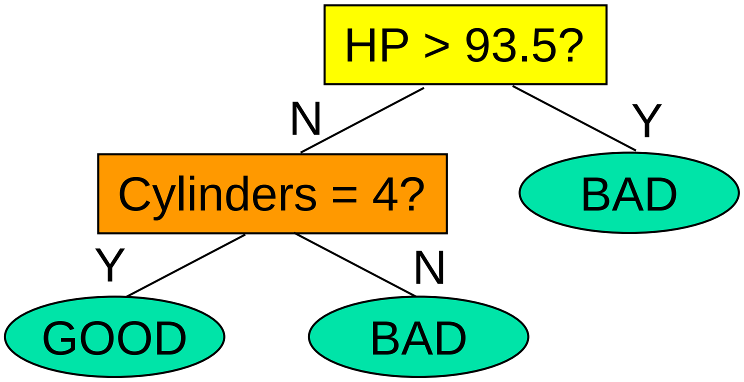
bad, 4, 81, light
good, 4, 81, light



MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

Learning binary decision trees

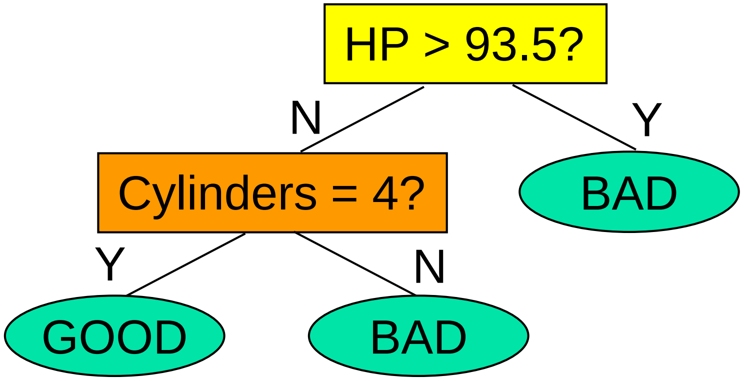
- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
- Step 4: Prune the tree to remove irrelevant rules.



MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

Learning binary decision trees

- Step 1: Start with all data points in a single node. Predict the most common value (classification) or mean value (regression).
- Step 2: Choose the “best” binary decision rule, and use it to split the data into two groups.
- Step 3: Repeat step 2 on each group, until some stopping criterion is reached.
- Step 4: Prune the tree to remove irrelevant rules.



<u>Question 1</u>: How to choose the best decision rule for a given node?

<u>Question 2</u>: How to prune the tree (and why bother?)

MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

Choosing a decision rule

- We can use any input attribute to split.
  - If discrete: choose a class, split into = and ≠.
  - If real: choose a threshold, split into > and ≤.
- To choose a threshold for a real attribute: sort the values, and use midpoints.

Split	Group Y	Group N
Cylinders = 4?	5+ / 4-	0+ / 11-
Cylinders = 6?	0+ / 6-	5+ / 9-
Cylinders = 8?	0+ / 5-	5+ / 10-
HP > 78?	2+ / 0-	3+ / 15-
HP > 87?	2+ / 2-	3+ / 13-
HP > 89.5?	3+ / 2-	2+ / 13-
HP > 91?	3+ / 3-	2+ / 12-
HP > 93.5?	5+ / 3-	0+ / 12-
Weight = light?	3+ / 3-	2+ / 12-
Weight = medium?	2+ / 6-	3+ / 9-
Weight = heavy?	0+ / 6-	5+ / 9-

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We can use any input attribute to split.
    - If discrete: choose a class, split into = and ≠.
    - If real: choose a threshold, split into > and ≤.
- To choose a threshold for a real attribute: sort the values, and use midpoints.
- Choose the split with highest information gain.

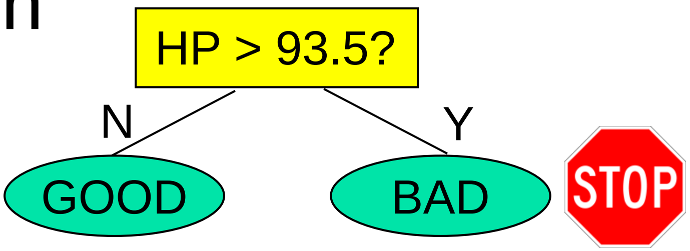
Split	Group Y	Group N	Gain
Cylinders = 4?	5+ / 4-	0+ / 11-	0.365
Cylinders = 6?	0+ / 6-	5+ / 9-	0.153
Cylinders = 8?	0+ / 5-	5+ / 10-	0.123
HP > 78?	2+ / 0-	3+ / 15-	0.226
HP > 87?	2+ / 2-	3+ / 13-	0.054
HP > 89.5?	3+ / 2-	2+ / 13-	0.144
HP > 91?	3+ / 3-	2+ / 12-	0.097
HP > 93.5?	**5+ / 3-**	**0+ / 12-**	**0.430**
Weight = light?	3+ / 3-	2+ / 12-	0.097
Weight = medium?	2+ / 6-	3+ / 9-	0.000
Weight = heavy?	0+ / 6-	5+ / 9-	0.153

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



(5 good, 3 bad) (0 good, 12 bad)

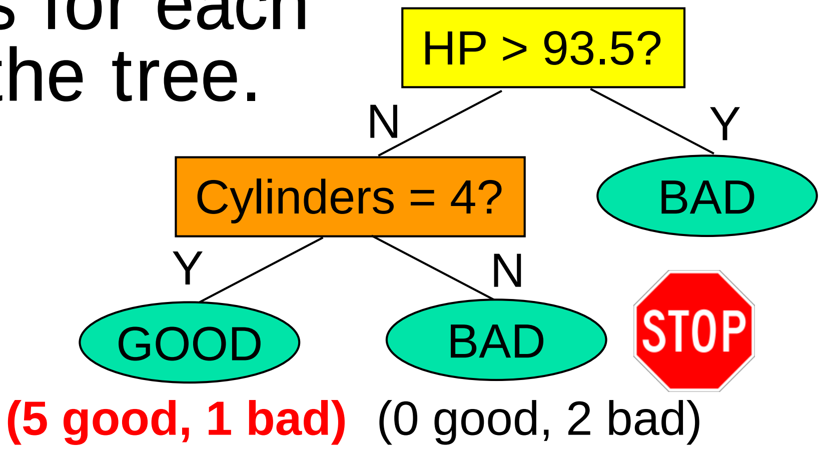
Split	Group Y	Group N	Gain
Cylinders = 4?	**5+ / 1-**	**0+ / 2-**	**0.467**
HP > 78?	2+ / 0-	3+ / 3-	0.204
HP > 87?	2+ / 2-	3+ / 1-	0.049
HP > 89.5?	3+ / 2-	2+ / 1-	0.003
HP > 91?	3+ / 3-	2+ / 0-	0.204
Weight = light?	3+ / 1-	2+ / 2-	0.049

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



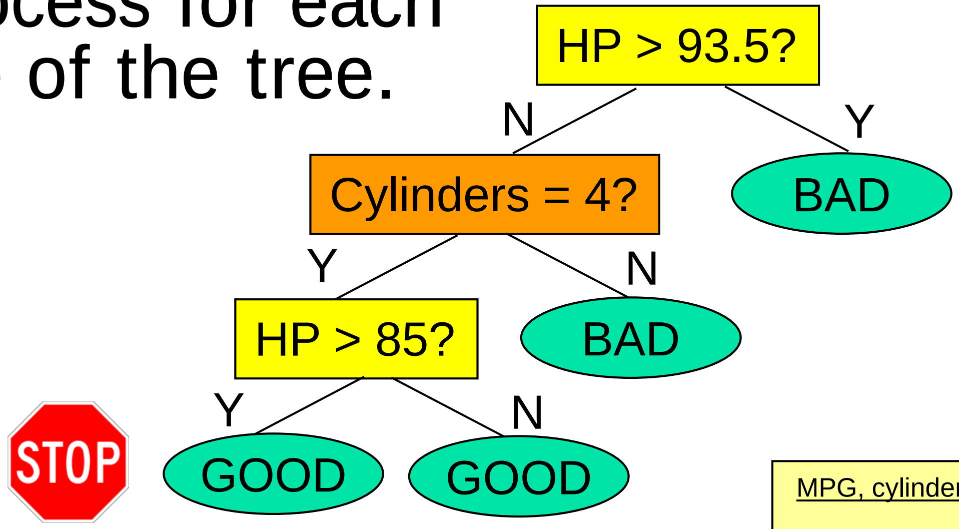
Split	Group Y	Group N	Gain
HP > 78?	3+ / 1-	2+ / 0-	0.109
HP > 85?	**3+ / 0-**	**2+ / 1-**	**0.191**
Weight = light?	3+ / 1-	2+ / 0-	0.109

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



(3 good, 0 bad)**(2 good, 1 bad)**

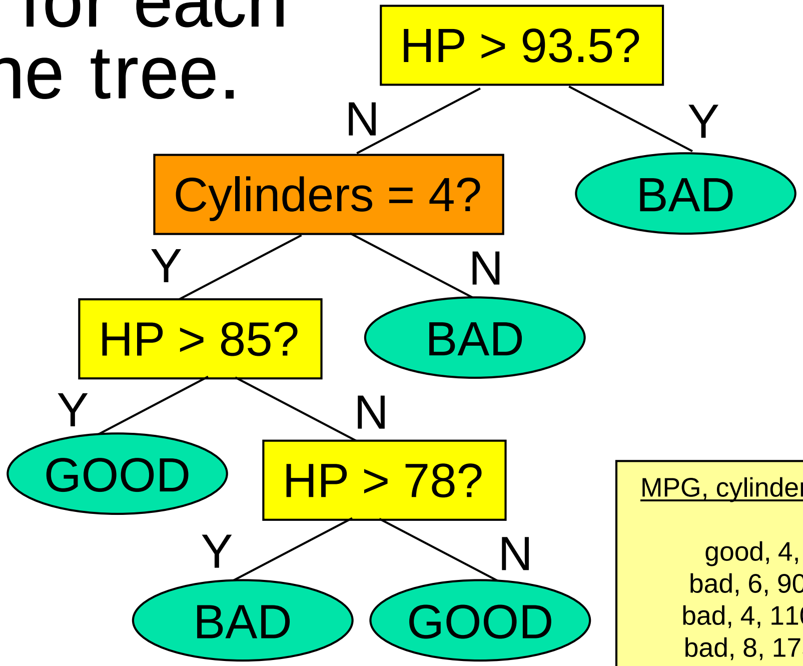
Split	Group Y	Group N	Gain
HP > 78?	**0+ / 1-**	**2+ / 0-**	**0.918**

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

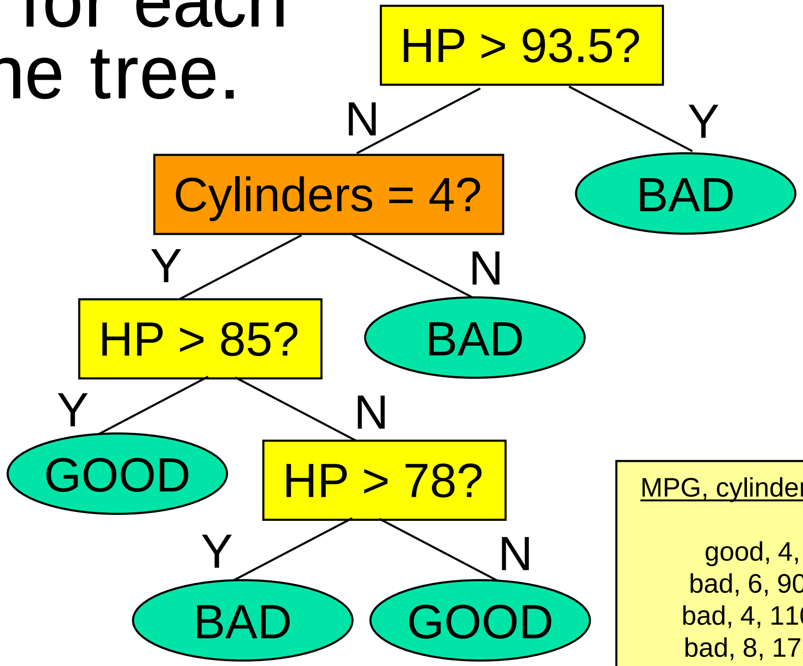
Group Y	Group N
A+ / B-	C+ / D-

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

Group Y	Group N
A+ / B-	C+ / D-

$$\text{Gain} = \frac{F((A+C),(B+D)) - F(A,B) - F(C,D)}{A+B+C+D}$$

where: $F(X,Y) = X\log_2 \frac{X+Y}{X} + Y\log_2 \frac{X+Y}{Y}$

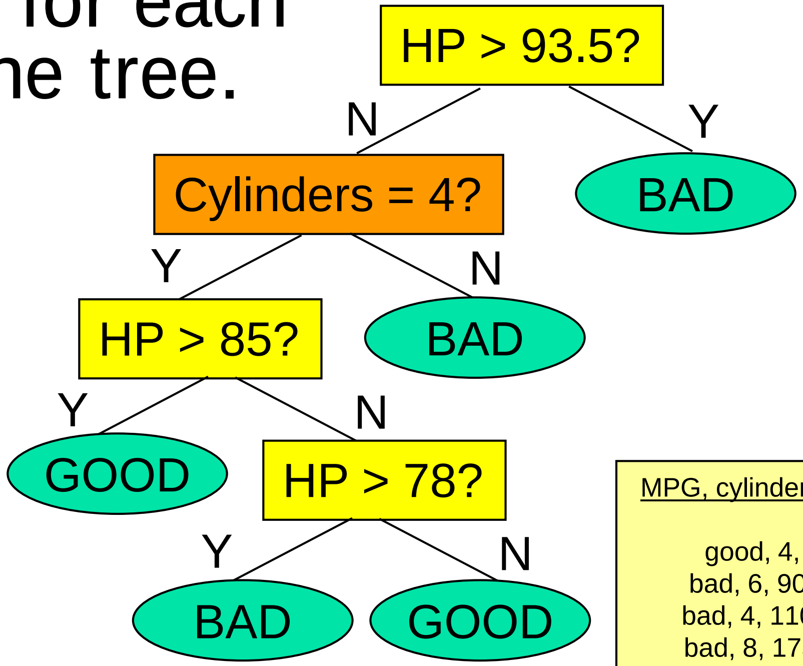
(You don't have to memorize this formula.)

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Choosing a decision rule

- We repeat this process for each non-terminal node of the tree.



Information gain is an information-theoretic measure of how well the split separates the data.

It can be computed as a function of the numbers of + and – examples in each group.

Group Y	Group N	Gain
3+ / 1-	6+ / 2-	0.000
8+ / 1-	1+ / 2-	0.204
9+ / 0-	0+ / 3-	0.811

Intuitively, the information gain is large when the proportions of positive examples in the two groups are very different, and zero when they are the same.

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Mathematical formulation

From http://scikit-learn.org/stable/modules/tree.html

Given training vectors x_i and a label vector y.
Let data at tree node m be represented by Q.

$$x_i \in R^n \qquad y \in R^l$$

Consider a set of candidate binary splits, each consisting of a feature j and threshold t_m, and partitioning Q into subsets Q_{left} and Q_{right}.

$$\theta = (j, t_m)$$

$$Q_{left}(\theta) = (x, y) | x_j <= t_m$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta)$$

Select the split that minimizes the average *impurity* of Q_{left} and Q_{right}, and recurse.

$$\theta^* = \operatorname{argmin}_\theta G(Q, \theta)$$

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

For classification, use *cross-entropy.* (p_{mk} are class proportions at node m).

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k)$$

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$$

For regression, use *mean squared error* (c_m is the mean at node m).

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2$$

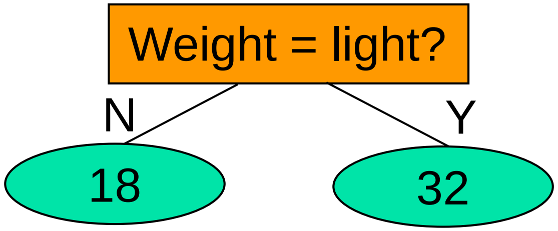
Choosing a decision rule

- If we are trying to predict a real-valued attribute (i.e. doing regression), minimize the sum of squared errors instead of maximizing information gain.

MPG, cylinders, HP, weight

32, 4, 75, light
20, 6, 95, medium
20, 4, 115, medium
14, 6, 95, medium

Split	Group Y	Group N	Total SSE
Cylinders = 4?	Predict: 26 SSE: 72	Predict: 17 SSE: 18	90
Weight = light?	**Predict: 32** **SSE: 0**	**Predict: 18** **SSE: 24**	**24**
HP > 85?	**Predict: 18** **SSE: 24**	**Predict: 32** **SSE: 0**	**24**
HP > 105?	Predict: 20 SSE: 0	Predict: 22 SSE: 168	168



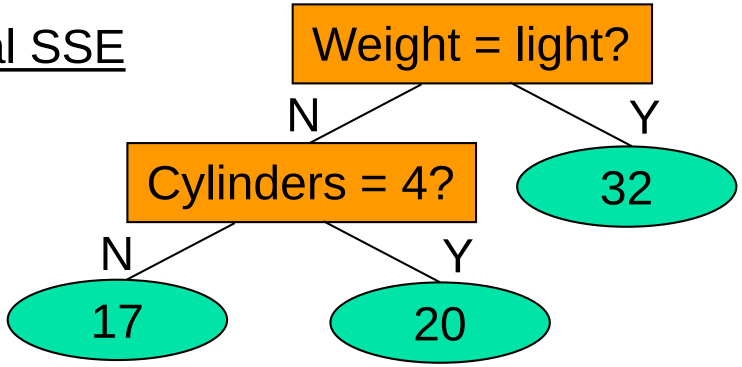
Choosing a decision rule

- If we are trying to predict a real-valued attribute (i.e. doing regression), minimize the sum of squared errors instead of maximizing information gain.

MPG, cylinders, HP, weight

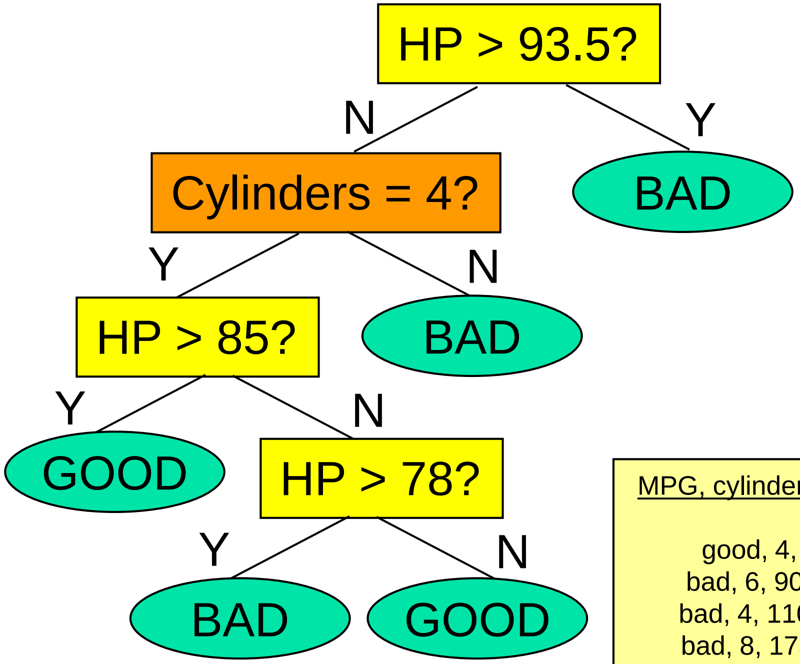
32, 4, 75, light
20, 6, 95, medium
20, 4, 115, medium
14, 6, 95, medium

Split	Group Y	Group N	Total SSE
Cylinders = 4?	**Predict: 20** **SSE: 0**	**Predict: 17** **SSE: 18**	**18**
HP > 105?	**Predict: 20** **SSE: 0**	**Predict: 17** **SSE: 18**	**18**



Pruning decision trees to prevent overfitting

- Notice that the unpruned decision tree classifies every training example perfectly.
- This will always be the case (unless there are records with the same input values and different output values, as in the regression example).
- Does this mean we've found the best tree?

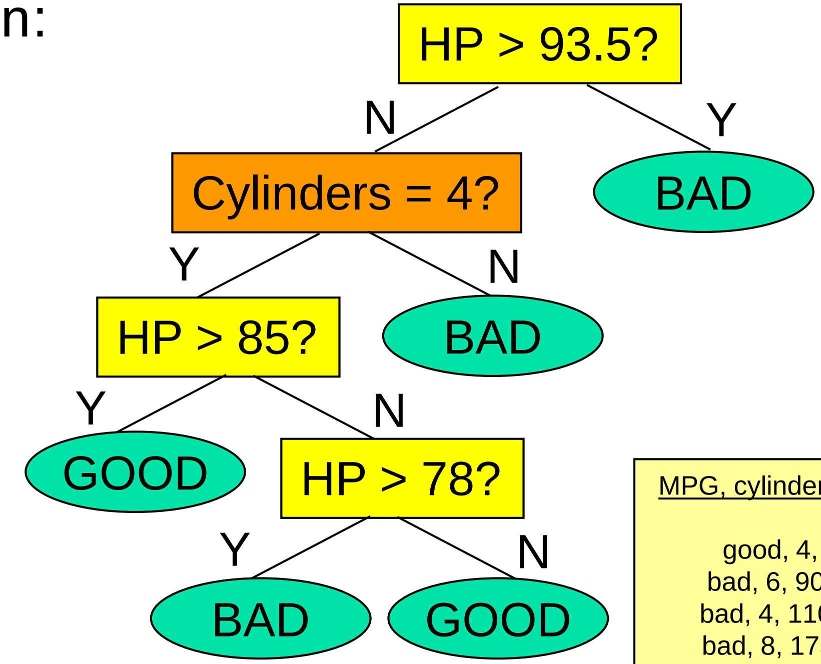


MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

What we really want to know:
How well does this classifier predict values for new data that we haven't already seen?

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the “test set”).
  - Learn a tree using the rest of the data (the “training set”).
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
100% correct, 0% incorrect

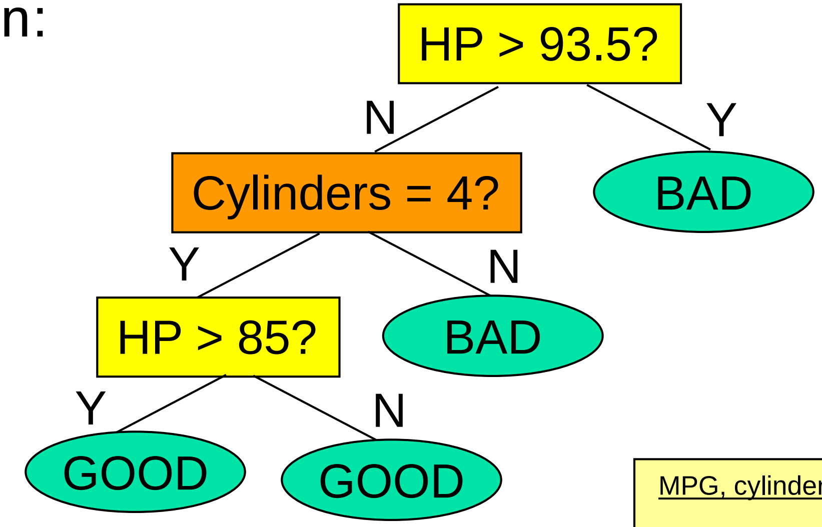
Test set (100 examples):
86% correct, 14% incorrect

What we really want to know:
How well does this classifier predict values for new data that we haven’t already seen?

MPG	cylinders	HP	weight
good	4	75	light
bad	6	90	medium
bad	4	110	medium
bad	8	175	weighty
bad	6	95	medium
bad	4	94	light
bad	4	95	light
bad	8	139	weighty
bad	8	190	weighty
bad	8	145	weighty
bad	6	100	medium
good	4	92	medium
bad	6	100	weighty
bad	8	170	weighty
good	4	89	medium
good	4	65	light
bad	6	85	medium
bad	4	81	light
bad	6	95	medium
good	4	93	light

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the “test set”).
  - Learn a tree using the rest of the data (the “training set”).
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

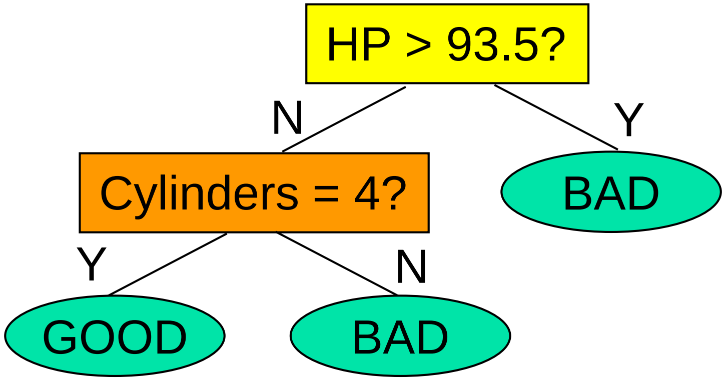
MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

What we really want to know:
How well does this classifier predict values for new data that we haven’t already seen?

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the “test set”).
  - Learn a tree using the rest of the data (the “training set”).
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

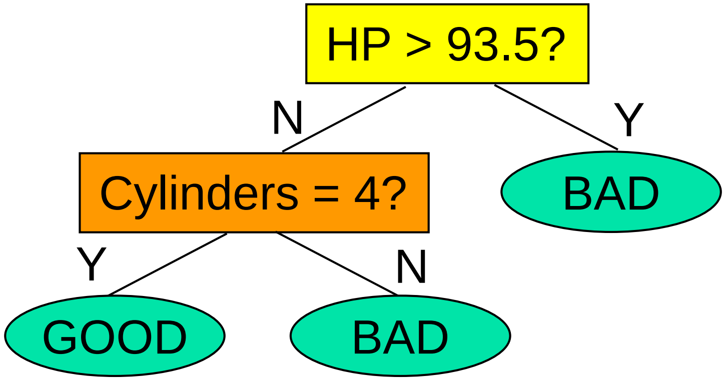
MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

What we really want to know:
How well does this classifier predict values for new data that we haven’t already seen?

Pruning decision trees to prevent overfitting

- One way to answer this question:
  - Hide part of your data (the “test set”).
  - Learn a tree using the rest of the data (the “training set”).
  - See what proportion of the test set is classified correctly.
- Consider pruning each node to see if it reduces test set error.



Training set (20 examples):
95% correct, **5%** incorrect

Test set (100 examples):
89% correct, **11%** incorrect

If we pruned Cylinders = 4, test set error would increase to 16%, so stop here!

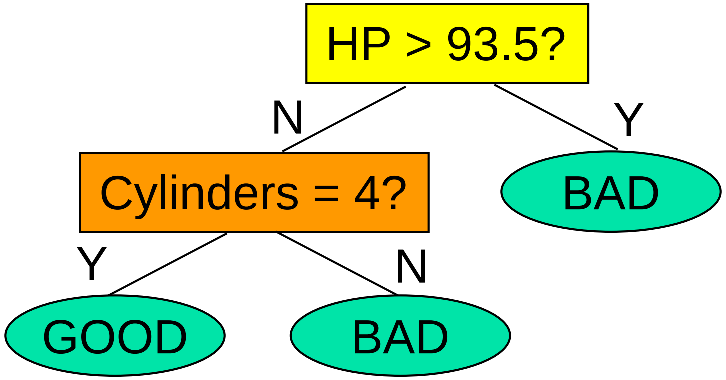
What we really want to know:
How well does this classifier predict values for new data that we haven’t already seen?

MPG, cylinders, HP, weight

good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

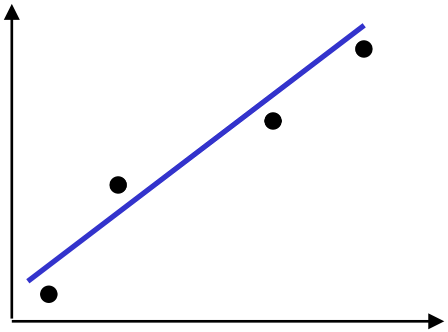
Pruning decision trees to prevent overfitting

- Q: Why does pruning the tree reduce test set error?
- A: Because the unpruned tree was <u>overfitting</u> the training data (paying attention to parts of the data that are not relevant for prediction).

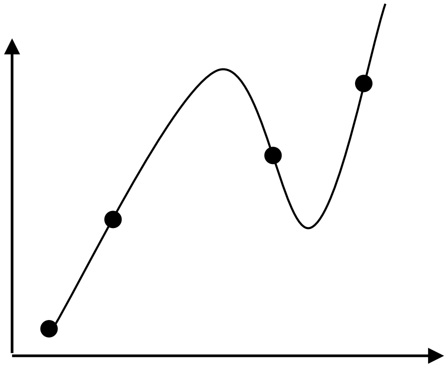


MPG, cylinders, HP, weight
good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Here's another example of overfitting, this time for regression:



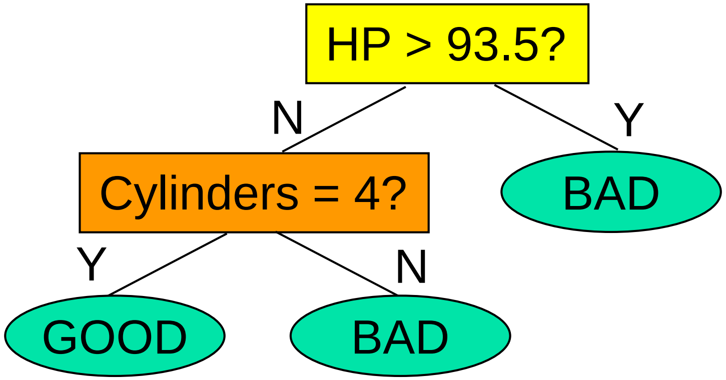
A line fits pretty well…



A cubic is probably overfitting.

Pruning decision trees to prevent overfitting

- Q: What if we don’t have enough data points?
- A: Another way to prevent overfitting is to do a certain kind of significance test (chi-squared) for each node, and prune any nodes that are not significant.



MPG, cylinders, HP, weight

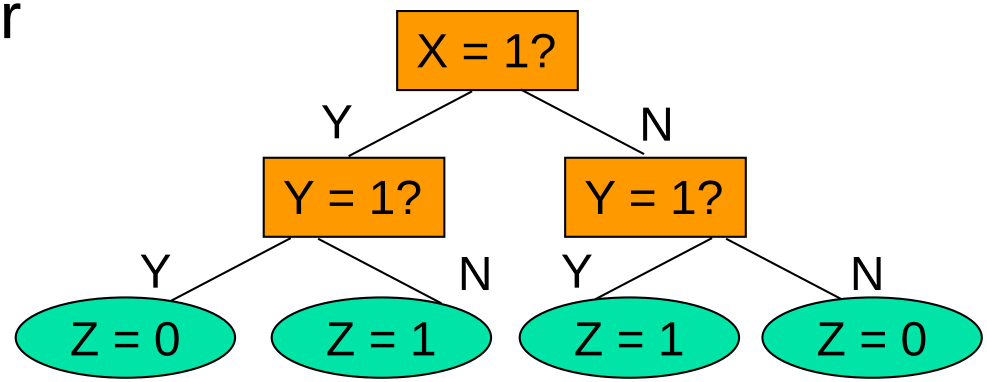
good, 4, 75, light
bad, 6, 90, medium
bad, 4, 110, medium
bad, 8, 175, weighty
bad, 6, 95, medium
bad, 4, 94, light
bad, 4, 95, light
bad, 8, 139, weighty
bad, 8, 190, weighty
bad, 8, 145, weighty
bad, 6, 100, medium
good, 4, 92, medium
bad, 6, 100, weighty
bad, 8, 170, weighty
good, 4, 89, medium
good, 4, 65, light
bad, 6, 85, medium
bad, 4, 81, light
bad, 6, 95, medium
good, 4, 93, light

Pruning decision trees to prevent overfitting

- Q: Why bother building the whole tree, if we’re just going to prune it?
- A: We could do significance testing while building the tree, but for many datasets, post-pruning gives better performance.

Consider the XOR function:

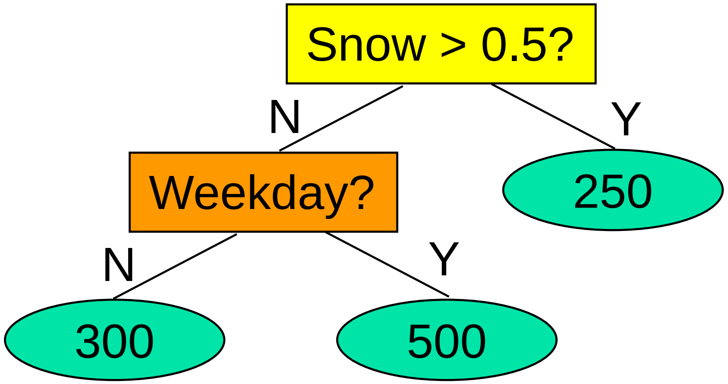
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

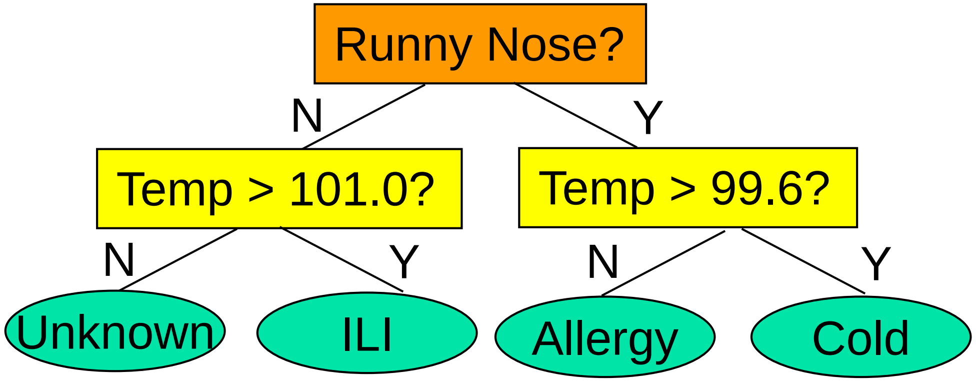


Each second level split has high information gain. The first level split has no information gain but is needed to make the second level splits possible.

Some advantages of decision trees

- Easy to learn the tree automatically from a dataset.
- Very easy to predict the target value given the tree.
- Generally good classification performance (though some fancier methods may do better, depending on the dataset).
- Can do both classification and regression, and can use both real and discrete inputs.
- Gives an idea of which variables are important in predicting the target value.
  - More important variables tend to be toward the top of the tree.
  - Unimportant variables are not included.





When to use decision trees?

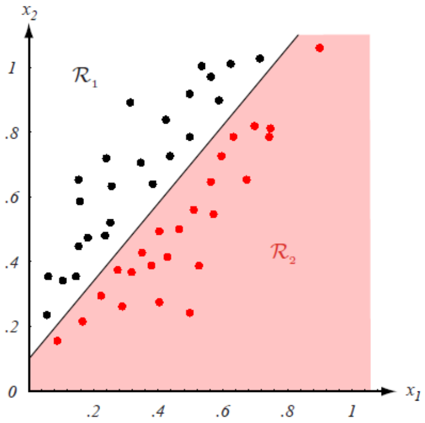
- We have a dataset, with some attribute we want to predict.
  - We can do either classification or regression.
  - Datasets with lots of attributes, and/or lots of records, are okay. But if lots of attributes and not that many records, should probably use feature selection first to avoid overfitting.
  - Datasets with discrete or real values, or both, are okay.
- We want to be able to explain our prediction using a simple and interpretable set of rules.
  - We want to distinguish relevant from irrelevant attributes.
  - We want to provide a set of decision steps (e.g. "expert system" for medical diagnosis, don't want to perform irrelevant tests).
  - If all we care about is prediction accuracy and not interpretability, we might want to use some "black box" classifier instead (e.g. neural network, support vector machine).
- Can represent complex, non-linear functions.
- Performance is better when a tree structure makes sense (see next slide).

Inductive bias

Duda, Hart, and Stork 2nd ed., fig. 8.5.

Inductive bias describes how a prediction method generalizes to previously unseen examples.

For example: binary logistic regression assumes a linear decision boundary between Pr(y = 1) > 0.5 and Pr(y = 1) < 0.5.

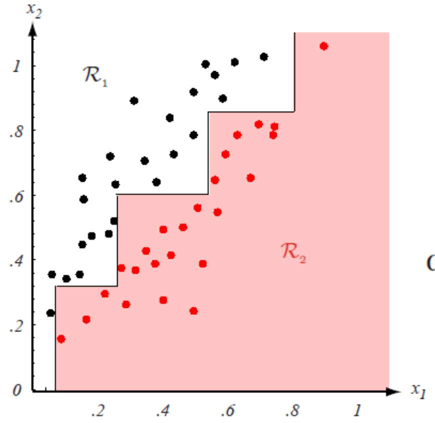


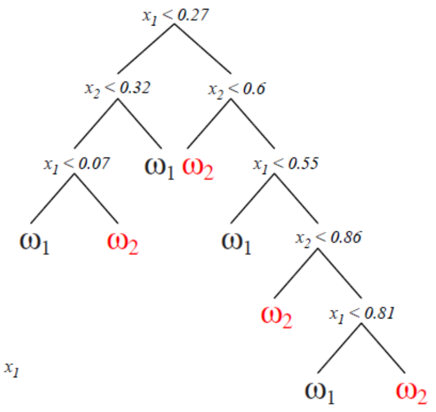
What are the inductive biases of a decision tree?

- Flexible – any function can be represented.
- Prefers shorter trees to longer trees.
- Prefers attributes with higher information gain.
- Splits on one attribute at a time.
- Splits are axis-aligned.

Suboptimal for linear functions

Hard to fit simple logical functions like XOR or majority vote.





Other issues with decision trees

Optional material, time permitting

1) Multi-way instead of binary splits are okay (e.g., Quinlan's ID3) but we generally prefer binary.

If using multi-way, use *gain ratio* instead of information gain to adjust for the arity (# of distinct values) of an attribute. Otherwise, dataset gets fragmented, leading to poor generalization accuracy (e.g., splitting on country name in the diarrheal illness dataset).

2) Decision trees can be easily used to predict **multiple outputs**.

Create a single tree where the split criterion is the average impurity over all outputs.

See http://scikit-learn.org/stable/modules/tree.html section 1.10.3.

3) Lots of different ways to handle **missing data**:

- Delete missing observations (not so good)
- Treat "missing" as its own value (good only if missingness is informative)
- Surrogate splits (Duda et al. Ch. 8.3.10)
- Propagate examples with missing values down both branches as partial observations

4) Can create a *rule list* from the leaves of a decision tree (e.g., Quinlan's C4.5).

Advantages: possibly more interpretable; can prune the rule list directly.

References

- For next time: read Python documentation at http://scikit-learn.org/stable/modules/tree.html
- L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*, Wadsworth, 1984.
- J.R. Quinlan. *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, 1993.
- T.M. Mitchell. *Machine Learning*, McGraw-Hill, 1997. (Chapter 3)
- T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*, 2001.
- A.W. Moore. *Decision Trees*. Available at http://www.cs.cmu.edu/~awm/tutorials.