Decision Trees

(Classification and Regression)

CS 6316 – Machine Learning
Fall 2017

OUTLINE

- Decision Tree Classification – Overview
- The Algorithm (brief)
- Entropy
- Information Gain
- Building the Decision Tree
- Decision Tree to Decision Rules
- CART Trees (brief)

Decision Tree Classifier

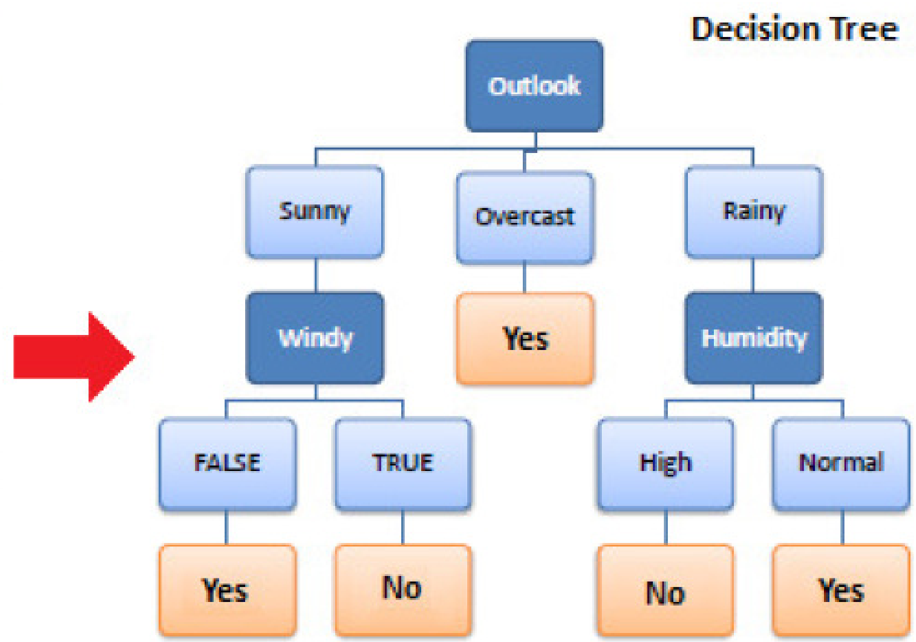
Overview

Decision Tree

- Decision tree builds classification or regression models in the form of a tree structure
- It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed
- The end product is a tree that consists of two kinds of nodes:
  - **Decision nodes** – two or more branches
  - **Leaf nodes** – represents a classification / decision
- Root node: top-most decision node corresponding to the best predictor

Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



The Algorithm

ID3

The Algorithm – ID3

- Core algorithm for building decision trees: **ID3** (by J. R. Quinlan)
- Employs a top-down, greedy search through the space of possible branches with no backtracking
- Decision tree can handle both categorical and numerical data

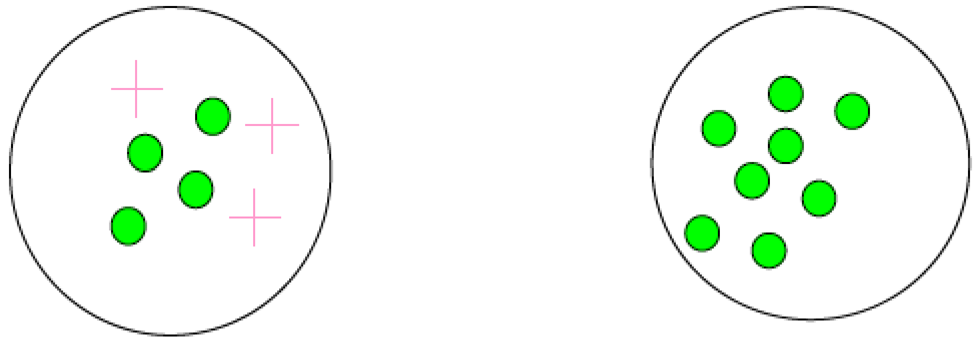
- ID3 uses **Entropy** and **Information Gain** to construct a decision tree

Entropy

- Which test is more informative?

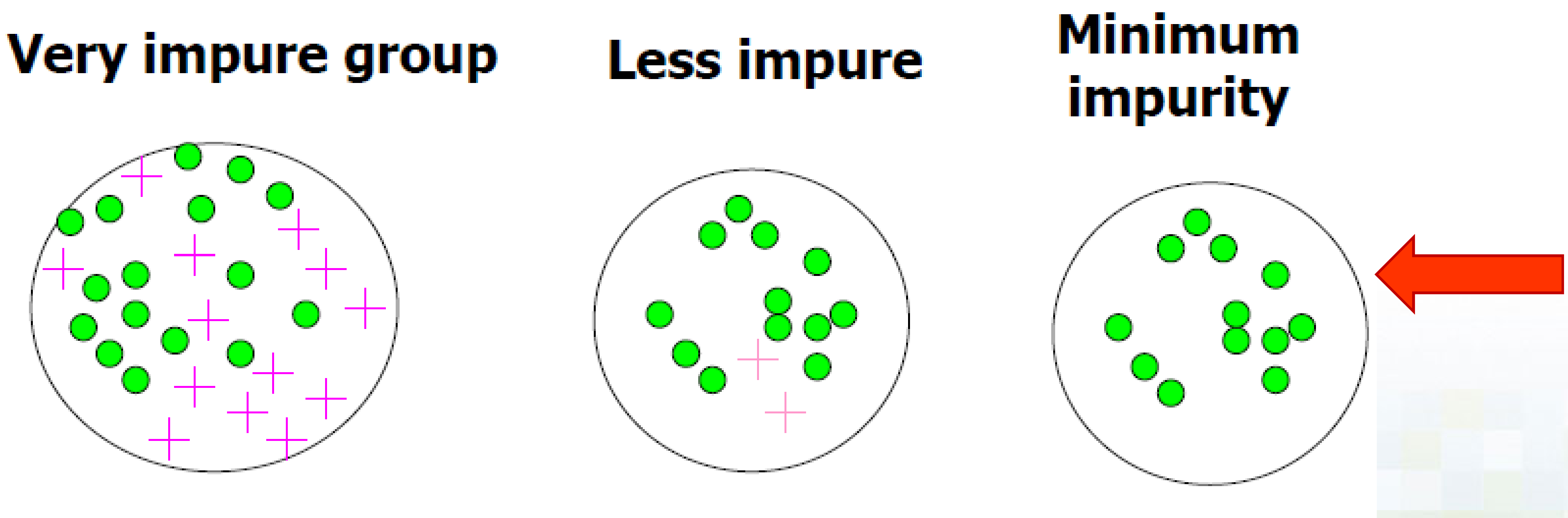
Entropy / Impurity (informal)

- Measures the level of **impurity** in a group of examples



Impurity

- Very impure: lots of examples from both classes
- Less impure: majority from one class, some from other class
- Minimum impurity: all from one class, none from other class



Entropy

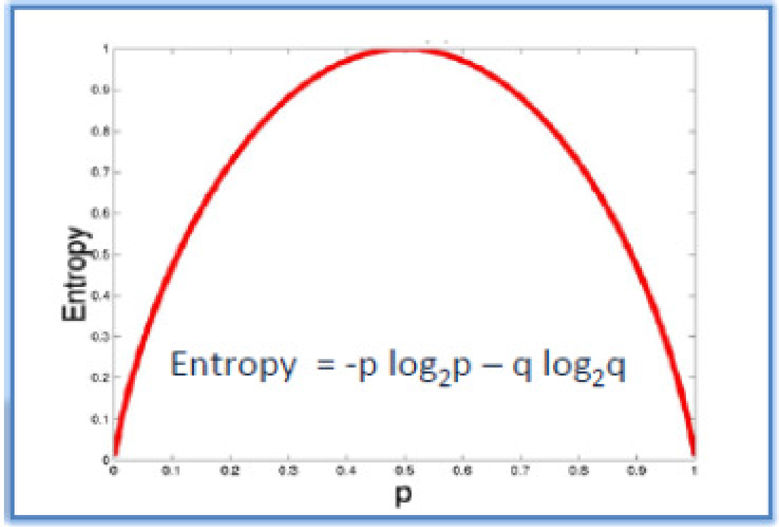
Entropy

- Entropy: a common way to measure impurity

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous)
- ID3 algorithm uses **entropy** to calculate the homogeneity (purity) of a sample

Entropy

- If the sample is completely homogeneous (pure) the entropy is **ZERO** and if the sample is equally divided it has entropy of **ONE**

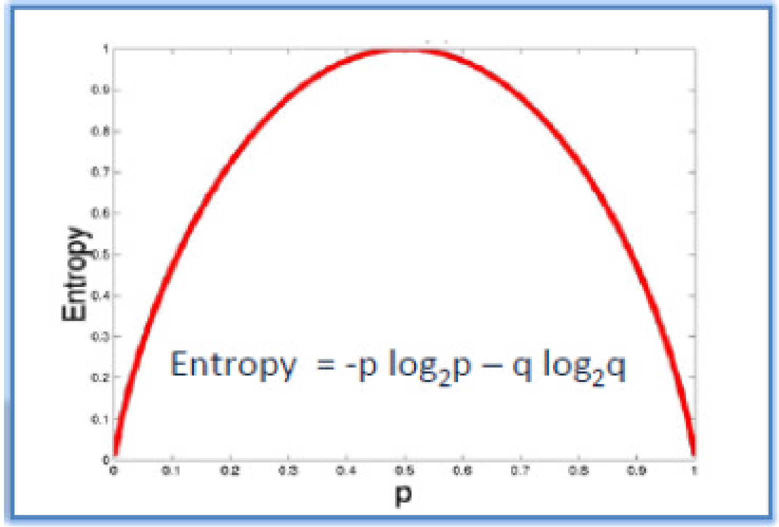


$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

- What is better, lower entropy or higher entropy?

Entropy

- If the sample is completely homogeneous (pure) the entropy is **ZERO** and if the sample is equally divided it has entropy of **ONE**



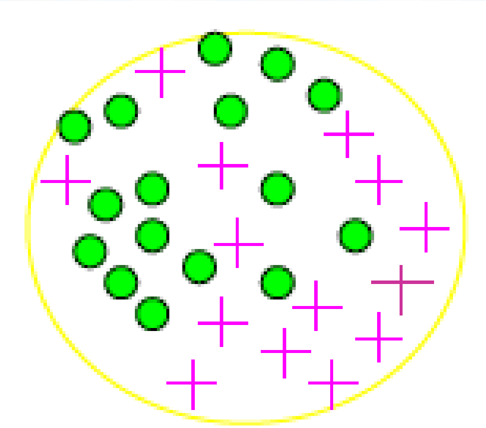
$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

- What is better, lower entropy or higher entropy? **LOW !**

Entropy

Entropy =

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



- P_i is the probability of class i, and c is total number of classes

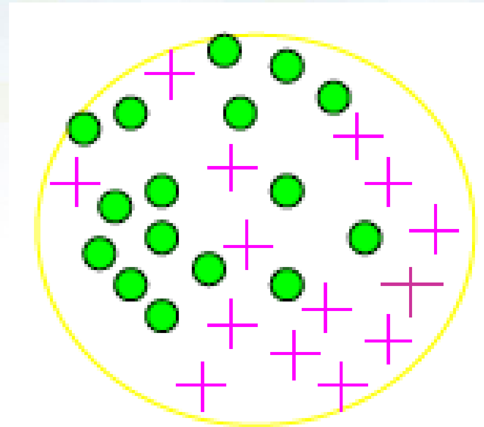
  Compute it as the proportion of class i in the set
- 16/30 are green circles; 14/30 are pink plus signs

  $\log_2(16/30)$ = -0.9; $\log_2(14/30)$ = -1.1
- Entropy = -(16/30)(-0.9) – (14/30)(-1.1) = **0.99**

Entropy

Entropy =

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



- P_i is the ber of classes

  Comput t

Entropy is high (0.99) showing the *impurity* of the given example

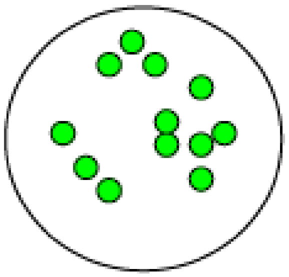
- 16/30 are green circles; 14/30 are pink plus signs

  $\log_2(16/30) = -0.9$; $\log_2(14/30) = -1.1$
- Entropy = -(16/30)(-0.9) – (14/30)(-1.1) = **0.99**

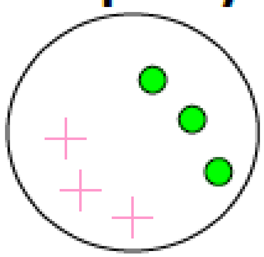
Entropy & Information Theory

- Entropy comes from information theory
- The **higher** the entropy the more the *information content*
- *What does that mean for learning from examples?*
- What is the entropy of a group in which all examples belong to the same class?
  - Entropy = $-1 \log_2 1 = \mathbf{0.0}$ (NOT a good training set for learning)
- What is the entropy of a group with 50% in either class?
  - Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = \mathbf{1}$ (Good training set for learning)

Minimum impurity



Maximum impurity



Information Gain

Constructing a decision tree is all about finding an attribute that returns the highest information gain (most homogeneous branches)

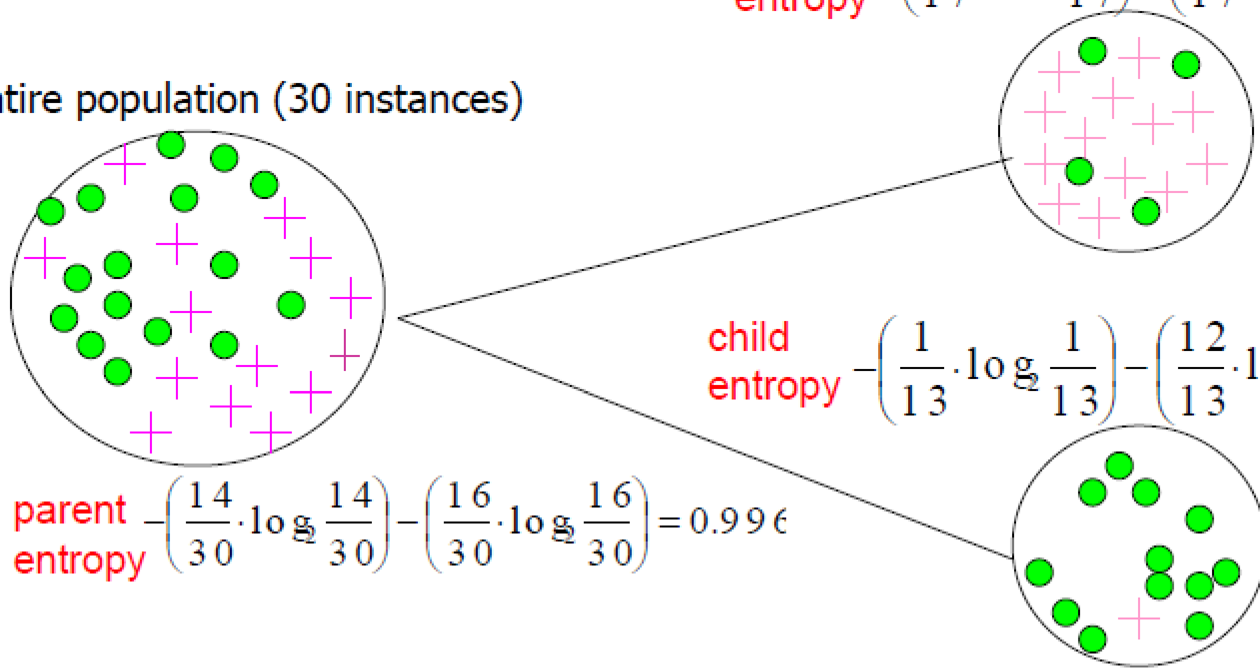
Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned
- The information gain is based on the decrease in entropy after a dataset is split on an attribute
- Information gain tells us how important a given attribute of the feature vector is
  → Used to decide the **ordering** of attributes in the nodes of a decision tree

Calculating Information Gain

Information Gain = Entropy(parent) – [average Entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38 for this split

Simple Example

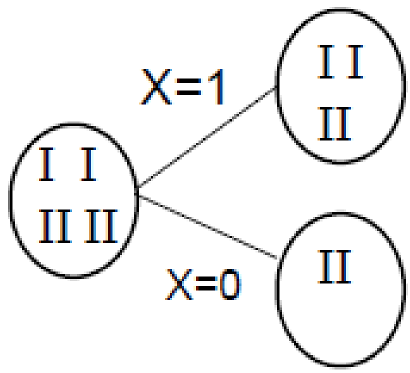
- How would you distinguish class I from class II?

Training Set: 3 features and 2 classes

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute X



If X is the best attribute,
this node would be further split.

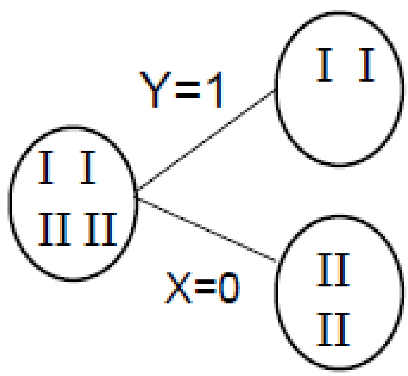
$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= .5284 + .39$
$= .9184$

$E_{child2} = 0$

$E_{parent} = 1$
GAIN = 1 – (3/4)(.9184) – (1/4)(0) = .3112

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Y



$E_{child1} = 0$

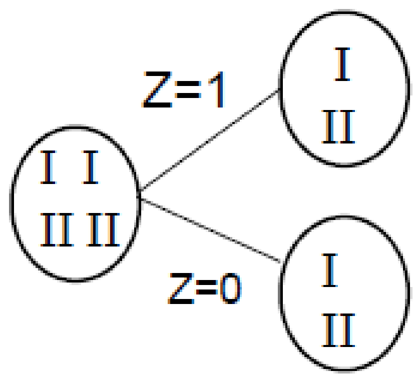
$E_{child2} = 0$

$E_{parent} = 1$

GAIN = 1 –(1/2) 0 – (1/2)0 = 1; BEST ONE

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Z



$E_{child1} = 1$

$E_{child2} = 1$

$E_{parent} = 1$

GAIN = 1 – (1/2)(1) – (1/2)(1) = 0 ie. NO GAIN; WORST

Building the Decision Tree

Reminder of Data Set

Predictors				Target
Outlook	**Temp.**	**Humidity**	**Windy**	**Play Golf**
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Building Decision Tree

- To build a decision tree, two types of entropy need to be calculated (using frequency tables):
  - (a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

$$\text{E(PlayGolf)} = -(\mathbf{9}/14 \log_2 \mathbf{9}/14) - (\mathbf{5}/14 \log_2 \mathbf{5}/14)$$

"Yes" "No"

Play Golf	
Yes	No
9	5

$$= -(0.64 \log_2 0.64) - (0.36 \log_2 0.36)$$

$$= -(-0.41) - (-0.53) = 0.41 + 0.53 = \mathbf{0.94}$$

Building Decision Tree

– (b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

E(PlayGolf, Outlook) =

P(Sunny)E(Sunny) + P(Overcast)E(Overcast) + P(Rainy)E(Rainy)

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

Building Decision Tree

– (b) Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

E(PlayGolf, Outlook) =

P(Sunny)E(Sunny) + P(Overcast)E(Overcast) + P(Rainy)E(Rainy)

E(Sunny) = – $(3/5 \log_2 3/5)$ – $(2/5 \log_2 2/5)$

= – (0.6)(-0.74) – (0.4)(-1.32)

= 0.44 + 0.53 = **0.97** = E(Rainy) (*Do you see why?*)

E(Overcast) = –$(4/4 \log_2 4/4)$ – $(0 \log_2 0)$ = **0.0**

E(PG,O) = (5/14)*0.97+(4/14)*0.0+(5/14)*0.97 = **<u>0.69</u>**

Step 1: Calculate Entropy of Target

- Calculate entropy of the **target** ("PlayGolf")

$$E(\text{PlayGolf}) = -(\mathbf{9}/14 \log_2 \mathbf{9}/14) - (\mathbf{5}/14 \log_2 \mathbf{5}/14)$$

"Yes" "No"

$$= -(0.64 \log_2 0.64) - (0.36 \log_2 0.36)$$

$$= -(-0.41) - (-0.53) = 0.41 + 0.53 = \mathbf{0.94}$$

At the start, the entire data set is used to calculate the entropy of the target ("PlayGolf" in this case)

Step 2: Calculate Information Gain

- The data set is split on the different attributes
- The entropy for each branch is calculated
- Then it is added proportionally, to get total entropy for the split → E(T,X)
- The resulting entropy is subtracted from the entropy before the split. The result is the **Information Gain** (G, Gain or IG), or decrease in entropy

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

- G(PlayGolf, Outlook) = 0.94 – 0.69 = **0.25**

Now, calculate IG for EACH attribute

Exercise: Information Gain

- Calculate Information Gain for
  - Humidity
  - Temperature
  - Windy
  - (Outlook has already been calculated in preceding slides)

- Which attribute has the largest information gain?
- This will be the decision node (root of DT)

Information Gain

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

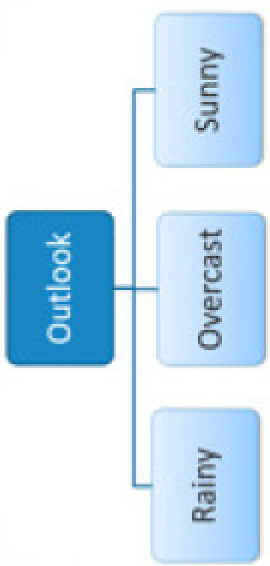
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

Step 3: Choose Attribute with Largest Information Gain

- Decision node (first node in the DT) will be **Outlook** (Gain = 0.247)

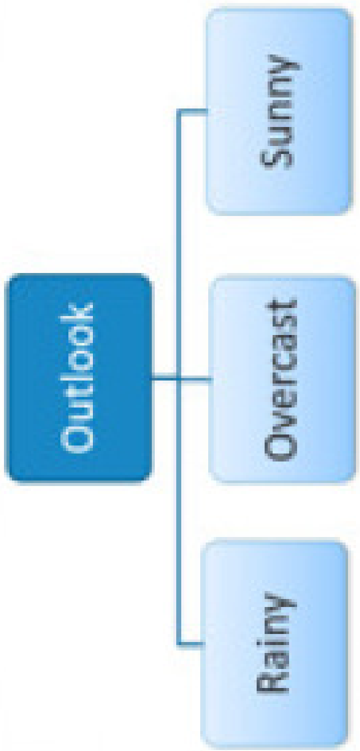
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Step 3: Choose Attribute with Largest Information Gain

- Then divide the dataset by its branches and repeat the same process on every branch



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

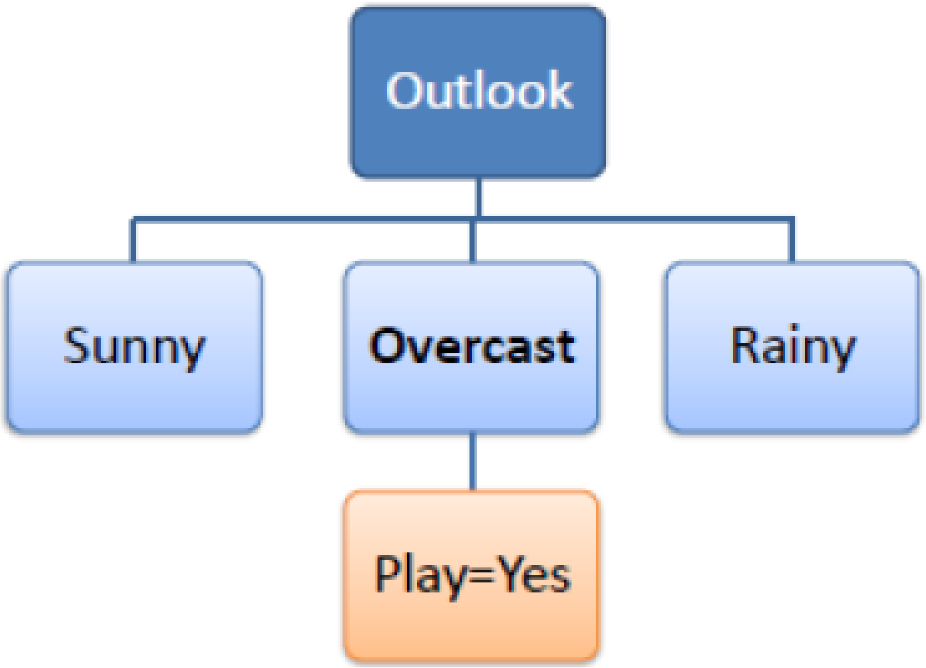
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes

Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Step 4(a): End or Split Further?

- A branch with entropy of 0 is a leaf node
  - E(Overcast) = 0.0 → Leaf node

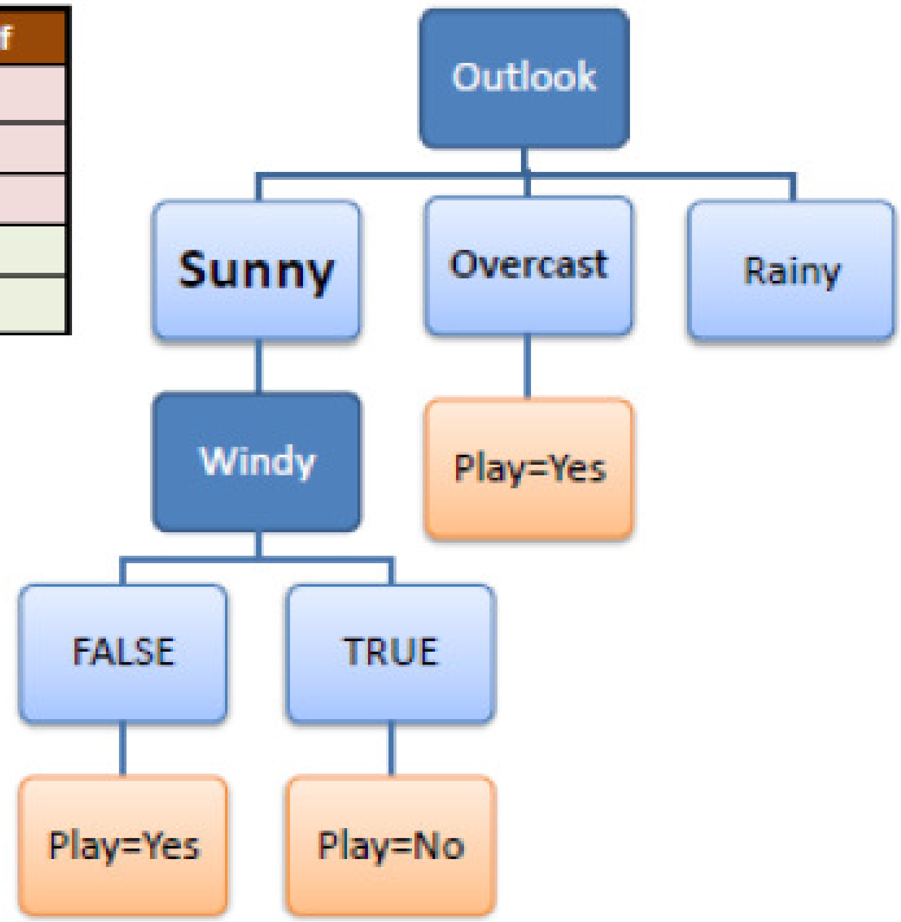
Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4(b): End or Split Further?

- A branch with entropy > 0 needs further splitting

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: Run Recursively!

- The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified

Further Splitting on Windy

		Play Golf		
		Yes	No	
Windy	TRUE	**0**	**2**	**2**
	FALSE	**3**	**0**	**3**
			Total:	**5**

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

Recalculate $E(\text{PlayGolf}) = -(3/5 \log_2 3/5) - (2/5 \log_2 2/5)$

$= -(0.6 \log_2 0.6) - (0.4 \log_2 0.4) = 0.442 + 0.529 = \mathbf{0.971}$

Further Splitting on Windy

		Play Golf		
		Yes	No	
Windy	TRUE	**0**	**2**	**2**
	FALSE	**3**	**0**	**3**
			Total:	**5**

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

E(PlayGolf, Windy) = P(True)E(True) + P(False)E(False)

E(True) = $-(0/2 \log_2 2/2) - (2/2 \log_2 2/2)$ = **0.0** (leaf)

E(False) = $-(3/3 \log_2 3/3) - (0/3 \log_2 0/3)$ = **0.0** (leaf)

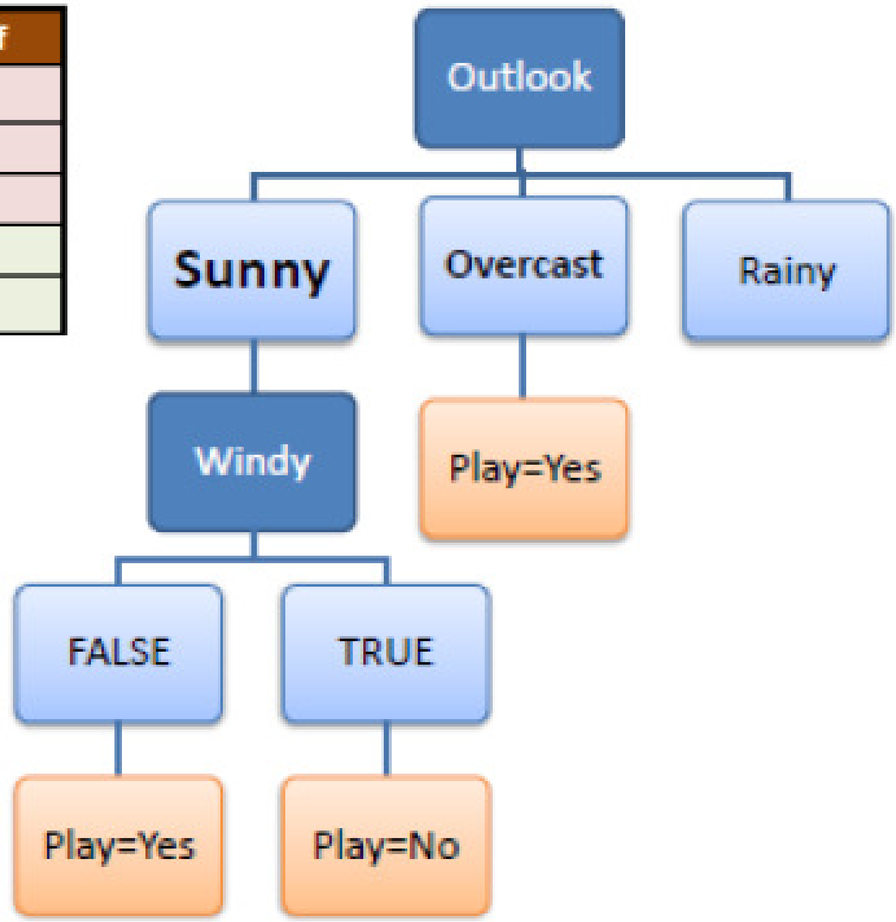
E(PG, W) = (2/5)*0.0 + (3/5)*0.0 = **0.0**

Gain(PG, W) = E(PG) – E(PG,W) = **0.97** – 0.0 = **0.97**

- Will do same for Temp and Humidity -> will have Gains < 0.97. So **next node** after **Sunny** will be ***Windy***

Further Splitting on Windy

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Further Splitting on Rainy

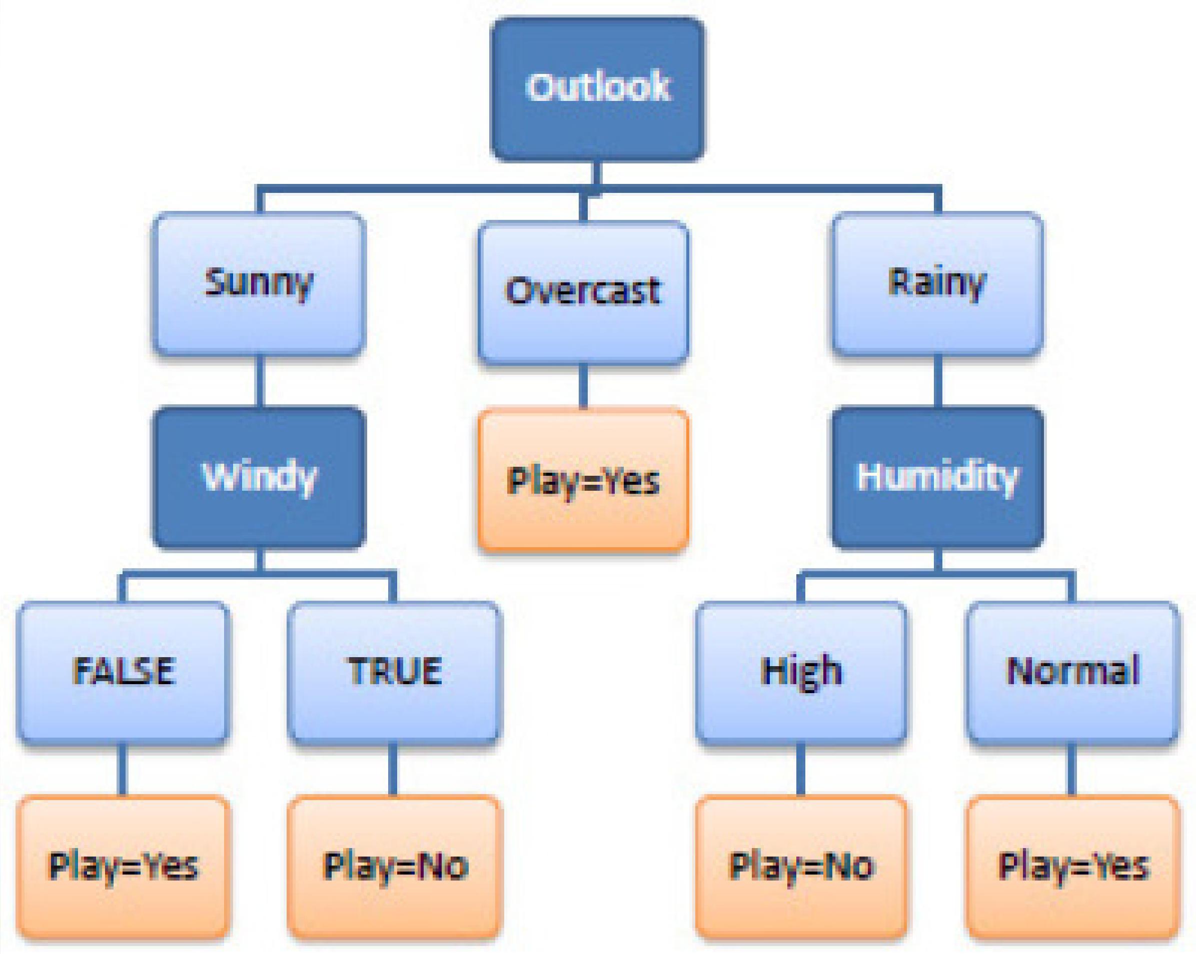
- Notice if we do E(PG, Humidity) we will get **0.0**

		Play Golf		
		Yes	No	
Humidity	High	0	3	3
	Normal	2	0	2
			Total:	5

- E(High) = E(Normal) = **0.0**
- Gain for Temperature will be < Gain for Humidity
- So **next node** after **Rainy** will be ***Humidity***
- Given both branches have entropy = 0.0, both are leaves
- TREE IS DONE! ☺

Tree is DONE!

- Final Decision Tree:



Decision Tree to Decision Rules

How a Decision Tree can be interpreted

Decision Tree to Decision Rules

- A decision tree can easily be transformed to a set of rules by following every path from root to leaf and mapping it to a rule:

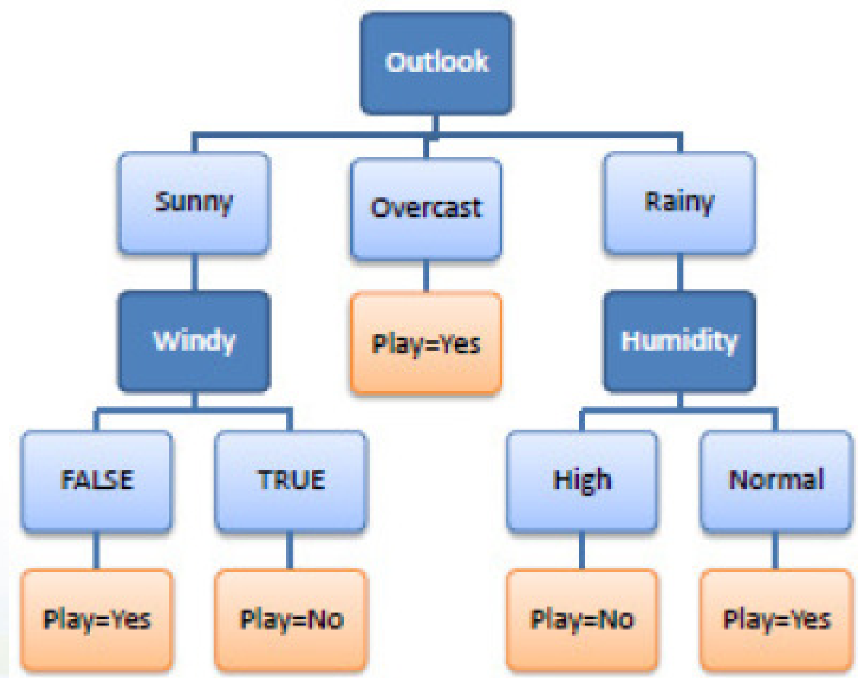
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

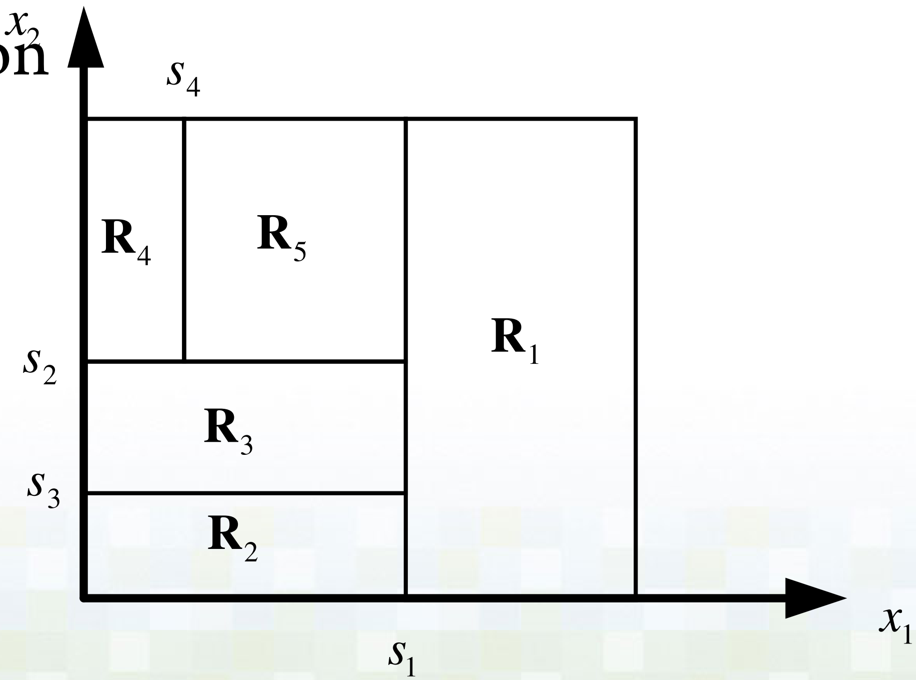


CART Trees

Classification And Regression Trees

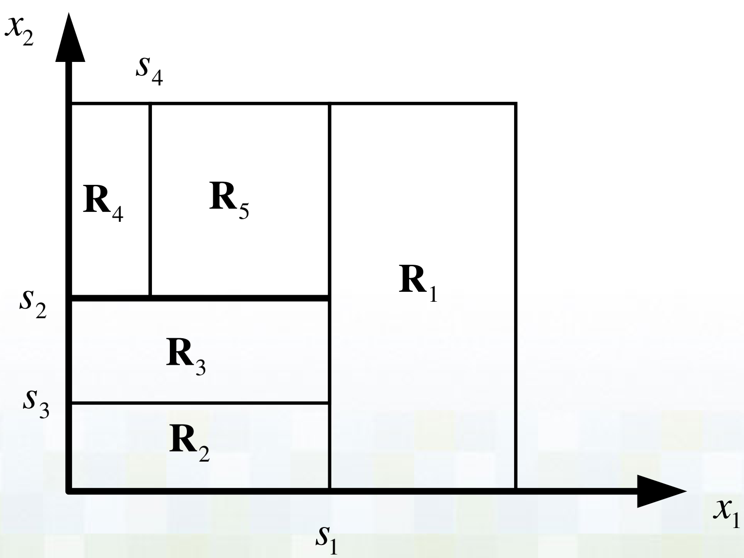
Other kinds of Trees

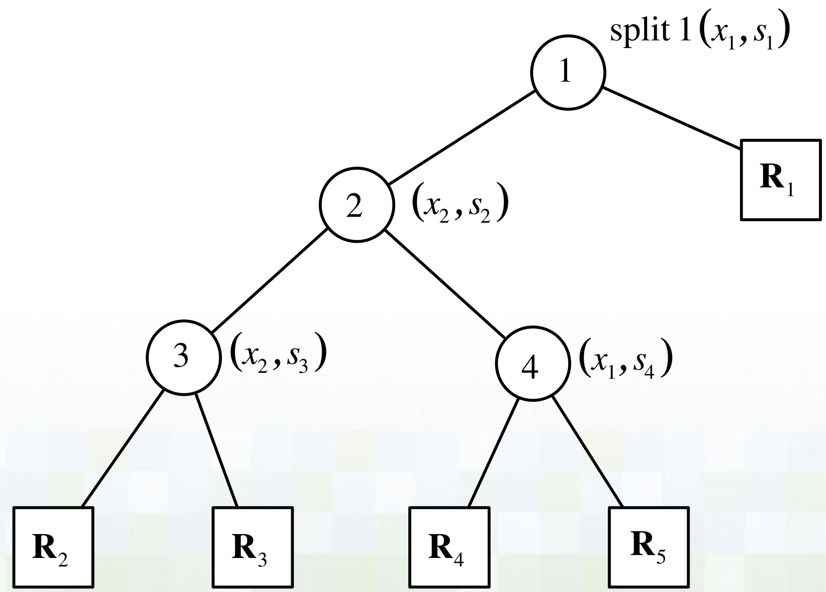
- Example of **CART** Partitioning
  - *Classification And Regression Trees*
- Example of nonlinear parameterization
- CART Partitioning in 2D space
  - each region ~ basis function
  - piecewise-constant estimate of y (in each region)
  - number of regions ~ model complexity



Regression Trees (CART)

- Minimization of empirical risk (squared error) via partitioning of the input space into regions
  $$f(\mathbf{x}) = \sum_{j=1}^{m} w_j I(\mathbf{x} \in \mathbf{R}_j) \quad \text{where} \quad w_j = \frac{1}{n_j} \sum_{\mathbf{x}_i \in \mathbf{R}_j} y_i$$
- Example of CART partitioning for a function of 2 inputs:





Classification Trees (CART)

- Binary classification example (2D input space)
- Algorithm *similar* to regression trees (tree growth via binary splitting + model selection), BUT using different