CSE 4621
Machine Learning

Lecture 12

Md. Hasanul Kabir, PhD.

Professor, CSE Department

Islamic University of Technology (IUT)

Decision Tree

- *A decision tree is a hierarchical data structure implementing the divide-and-conquer strategy.*
  - *It is an efficient nonparametric method,*
  - *can be used for both classification and regression.*
- A decision tree is composed of internal decision nodes and terminal leaves.
- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Advantage of the decision tree is interpretability. As the tree can be converted to a set of *IF-THEN rules.*
- Tree induction is the construction of the tree given a training sample. For a given training set, there exists many trees that code it with no error.

Decision Tree

- Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome.
- Each *decision node m* implements a test function $f_m(\boldsymbol{x})$ with discrete outcomes labeling the branches.
- This process starts at the root and is repeated recursively until a *leaf node* is hit, at which point the value written in the leaf constitutes the output.

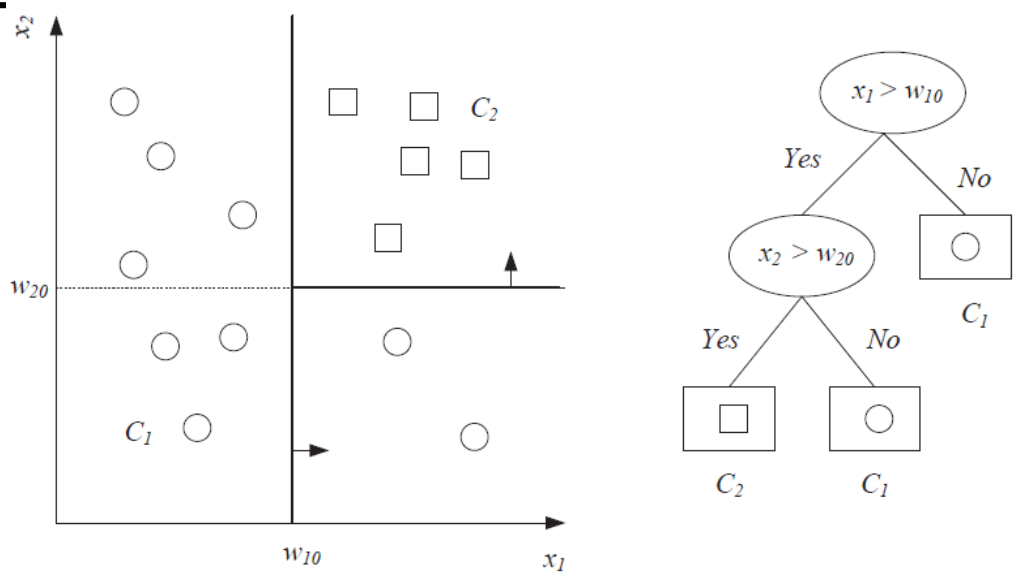


Figure 9.1 Example of a dataset and the corresponding decision tree. Oval nodes are the decision nodes and rectangles are leaf nodes. The univariate decision node splits along one axis, and successive splits are orthogonal to each other. After the first split, $\{\boldsymbol{x}|x_1 < w_{10}\}$ is pure and is not split further.

Test Function

- Each $f_m(\boldsymbol{x})$ defines a discriminant in the d-dimensional input space dividing it into smaller regions that are further subdivided down the tree.
- Different decision tree methods assume different models for $f_m(\cdot)$, and the model class defines the shape of the discriminant and the shape of regions.
- The boundaries of the regions are defined by the discriminants that are coded in the internal nodes.
- Hierarchical placement of decisions allows a fast localization of the region covering an input.
  - in best case, the correct region can be found in $\log_2 b$ decisions.

Univariate Tree

- In a *univariate tree*, in each internal node, the test uses only one of the input dimensions/attributes x_j.
  - is discrete, taking one of n possible values, the decision node checks the value of x_j and takes the corresponding branch (n-way split)
  - is numeric, input should be discretized.

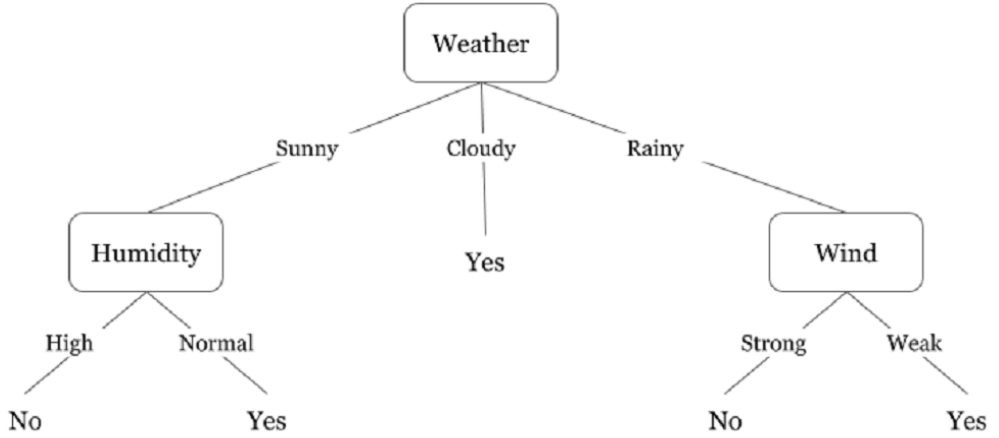
$$f_m(\boldsymbol{x}) : x_j > w_{m0}$$

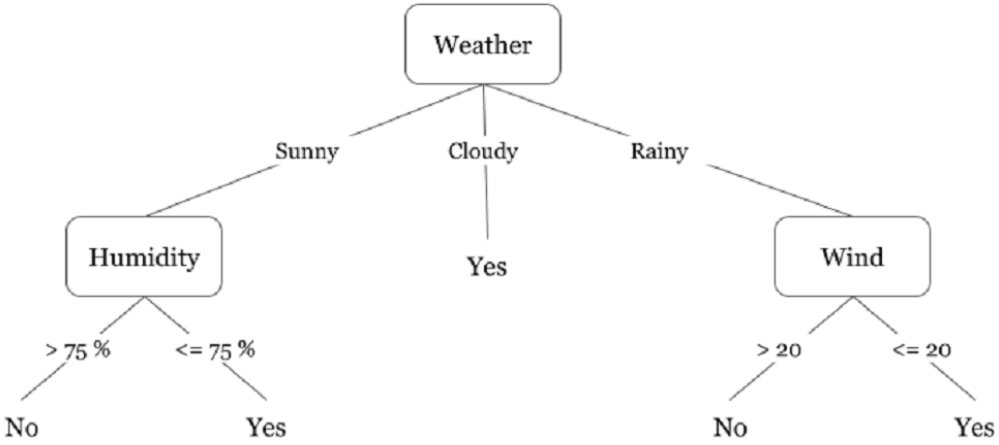
where w_{m0} is a suitably chosen threshold value. The decision node divides the input space into two: $L_m = \{\boldsymbol{x} | x_j > w_{m0}\}$ and $R_m = \{\boldsymbol{x} | x_j \leq w_{m0}\}$; this is called a *binary split*. Successive decision nodes on a path

- Tree learning algorithms are greedy and, at each step, we look for the best split

Example

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No





Classification Tree

- A decision tree for classification, namely, a *classification tree*, the goodness of a split is quantified by an *impurity measure*.
  - a nonnegative function measuring the impurity of a split if it satisfies the following properties

  - $\phi(1/2, 1/2) \geq \phi(p, 1-p)$, for any $p \in [0, 1]$.
  - $\phi(0, 1) = \phi(1, 0) = 0$.
  - $\phi(p, 1-p)$ is increasing in p on $[0, 1/2]$ and decreasing in p on $[1/2, 1]$.

- A split is pure if after the split, for all branches, all the instances choosing a branch belong to the same class.

Impurity Test: Entropy

- Entropy in information theory specifies the minimum number of bits needed to encode the class code of an instance.
- Our measure impurity is *entropy*

1. Entropy

$$\phi(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

$$\mathcal{I}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

$$p_m^i = \frac{N_m^i}{N_m}$$

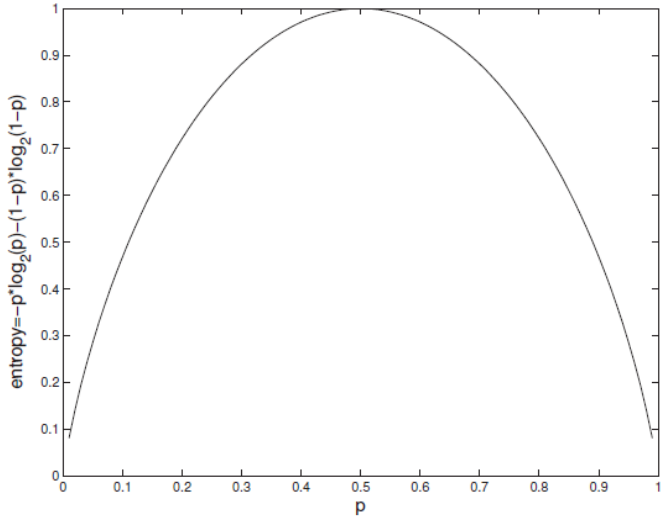


Figure 9.2 Entropy function for a two-class problem.

Impurity Test

2. *Gini index* (Breiman et al. 1984)

$$\phi(p, 1-p) = 2p(1-p)$$

3. Misclassification error

$$\phi(p, 1-p) = 1 - \max(p, 1-p)$$

Split and Sub-tree generation

- If node m is not pure, then the instances should be split to decrease impurity
  - there are multiple possible attributes on which we can split.
  - For a numeric attribute, multiple split positions are possible.
- Among all, take the split that minimizes impurity after the split because we want to generate the smallest tree.
  - this is locally optimal, and we have no guarantee of finding the smallest decision tree.
  - Tree size is measured as the number of nodes in the tree and the complexity of the decision nodes.
  - Finding the smallest tree is NP-complete.

Split and Sub-tree generation

Then given that at node m, the test returns outcome j, the estimate for the probability of class C_i is

$$\hat{P}(C_i|\boldsymbol{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

and the total impurity after the split is given as

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- In the case of a numeric attribute, here are $N_m - 1$ possible w_{m0} between N_m data points
  - Do not need to test for all. the best split is always between adjacent points belonging to different classes.

Split and Sub-tree generation

- For all attributes, discrete and numeric, and for a numeric attribute for all split positions, we calculate the impurity and choose the one that has the minimum entropy.
- Choose the split that causes the largest decrease in impurity, which is the difference between the impurity of data reaching node *m* and the total entropy of data reaching its branches after the split.
- Then tree construction continues recursively and in parallel for all the branches that are not pure, until all are pure.
- Algorithms:
  - CART
  - ID3
  - C4.5

Important Factors

- One problem is that such splitting favors attributes with many values
- Nodes with many branches are complex and go against our idea of splitting class discriminants into simple decisions.
- Methods may be proposed to penalize such attributes and to balance the impurity drop and the branching factor.

- Second when there is noise, growing the tree until it is purest, we may grow a very large tree and it overfits.
- To alleviate such overfitting, tree construction ends when nodes become pure enough, namely, a subset of data is not split further if $I<\theta_I$
  - *do not require that p^i_{mj} be exactly 0 or 1 but close enough.*

- *Generally advised that in a leaf, one stores the posterior probabilities of classes.*

Classification Tree

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* equation 9.3 */
      Create leaf labelled by majority class in X
      Return
    i ← SplitAttribute(X)
    For each branch of x_i
      Find X_i falling in branch
      GenerateTree(X_i)
```

```
SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
      If x_i is discrete with n values
        Split X into X_1,...,X_n by x_i
        e ← SplitEntropy(X_1,...,X_n) /* equation 9.8 */
        If e<MinEnt MinEnt ← e; bestf ← i
      Else /* x_i is numeric */
        For all possible splits
            Split X into X_1, X_2 on x_i
            e←SplitEntropy(X_1, X_2)
            If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Figure 9.3 Classification tree construction.

Step-by-Step Example

- Sample # 14

Play Golf(14)	
Yes	No
9	5

Fig. Frequency Table

$$E(PlayGolf) = E(5,9)$$

$$= -\left(\frac{9}{14}\log_2\frac{9}{14}\right) - \left(\frac{5}{14}\log_2\frac{5}{14}\right)$$

$$= -(0.357\ \log_2 0.357) - (0.643\ \log_2 0.643)$$

$$= 0.94$$

Entropy at Root Node

Attributes | Classes

Outlook	Temperature	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

Step-by-Step: Calculate Entropy for Attributes After Split

We need to calculate the entropy after each of the split.

- E(PlayGolf, Outloook)

$$\boldsymbol{E}(PlayGolf, Outlook) = \boldsymbol{P}(Sunny)\boldsymbol{E}(Sunny) + \boldsymbol{P}(Overcast)\boldsymbol{E}(Overcast) + \boldsymbol{P}(Rainy)\boldsymbol{E}(Rainy)$$

- E(PlayGolf, Temperature)

- E(PlayGolf, Humidity)

- E(PlayGolf, Windy)

E(PlayGolf, Outlook)

		PlayGolf(14)		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5

Fig. Frequency Table

$$E(Sunny) = E(3,2)$$

$$= -\left(\frac{3}{5}\log_2\frac{3}{5}\right) - \left(\frac{2}{5}\log_2\frac{2}{5}\right)$$

$$= -(0.60\ \log_2 0.60) - (0.40\log_2 0.40)$$

$$= -(0.60 * 0.737) - (0.40 * 0.529)$$

$$= \mathbf{0.971}$$

$$E(Overcast) = E(4,0)$$

$$= -\left(\frac{4}{4}\log_2\frac{4}{4}\right) - \left(\frac{0}{4}\log_2\frac{0}{4}\right)$$

$$= -(0) - (0)$$

$$= \mathbf{0}$$

$$E(Rainy) = E(2,3)$$

$$= -\left(\frac{2}{5}\log_2\frac{2}{5}\right) - \left(\frac{3}{5}\log_2\frac{3}{5}\right)$$

$$= -(0.40\ \log_2 0.40) - (0.6\log_2 0.60)$$

$$= \mathbf{0.971}$$

$$E(PlayGolf, Outlook) = \frac{5}{14}E(3,2) + \frac{4}{14}E(4,0) + \frac{5}{14}E(2,3)$$

$$= \frac{5}{14}0.971 + \frac{4}{14}0.0 + \frac{5}{14}0.971$$

$$= 0.357 * 0.971 + 0.0 + 0.357 * 0.971$$

$$= \mathbf{0.693}$$

E(PlayGolf, Temperature)

		PlayGolf(14)		
		Yes	No	
Temperature	Hot	2	2	4
	Cold	3	1	4
	Mild	4	2	6

Fig. Frequency Table

$$E(\text{PlayGolf, Temperature}) = \mathbf{4/14} * \mathbf{E}(\text{Hot}) + \mathbf{4/14} * \mathbf{E}(\text{Cold}) + \mathbf{6/14} * \mathbf{E}(\text{Mild})$$

$$E(\text{PlayGolf, Temperature}) = \mathbf{4/14} * \mathbf{E}(2, 2) + \mathbf{4/14} * \mathbf{E}(3, 1) + \mathbf{6/14} * \mathbf{E}(4, 2)$$

$$E(\text{PlayGolf, Temperature}) = 4/14 * -(2/4 \log 2/4) - (2/4 \log 2/4)$$
$$+ 4/14 * -(3/4 \log 3/4) - (1/4 \log 1/4)$$
$$+ 6/14 * -(4/6 \log 4/6) - (2/6 \log 2/6)$$

$$E(\text{PlayGolf, Temperature}) = 5/14 * 1.0$$
$$+ 4/14 * 1.811$$
$$+ 5/14 * 0.918$$
$$= \mathbf{0.911}$$

E(PlayGolf, Humidity)

		PlayGolf(14)		
		Yes	No	
Humidity	High	3	4	7
	Normal	6	1	7

Fig. Frequency Table

E (PlayGolf, Humidity) = **7/14** * **E**(High) + **7/14** * **E**(Normal)

E (PlayGolf, Humidity) = 7/14 * -(3/7 log 3/7) – (4/7 log 4/7)

+ 7/14 * -(6/7 log 6/7) – (1/7 log 1/7)

E (PlayGolf, Humidity) = 7/14 * 0.985

+ 7/14 * 0.592

= 0.788

E(PlayGolf, Windy)

		PlayGolf(14)		
		Yes	No	
Windy	TRUE	3	3	6
	FALSE	6	2	8

Fig. Frequency Table

E (PlayGolf, Windy) = **6/14 * E**(True) + **8/14 * E**(False)

E (PlayGolf, Windy) = **6/14 * E**(3, 3) + **8/14 * E**(6, 2)

E (PlayGolf, Windy) = 6/14 * -(3/6 log 3/6) – (3/6 log 3/6)

+ 8/14 * -(6/8 log 6/8) – (2/8 log 2/8)

E (PlayGolf, Windy) = 6/14 * 1.0

+ 8/14 * 0.811

= **0.892**

Calculating Information Gain for Each Split

- E(PlayGolf, Outloook) = **0.693**
- E(PlayGolf, Temperature) = **0.911**
- E(PlayGolf, Humidity) = **0.788**
- E(PlayGolf,Windy) = **0.892**

- *Gain(S,T) = Entropy(S)*
  – Entropy(S,T)

- *Gain(PlayGolf, Outlook) = Entropy(PlayGolf)*
  – Entropy(PlayGolf, Outlook)
  = 0.94 – 0.693 **= 0.247**

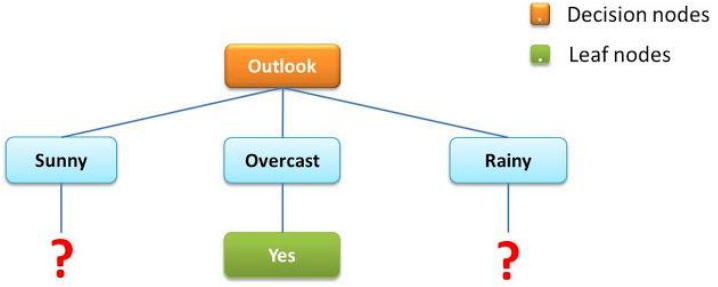
- *Gain(PlayGolf, Temperature) = Entropy(PlayGolf)*
  – Entropy(PlayGolf, Temparature)
  = 0.94 – 0.911 ***= 0.029***

- *Gain(PlayGolf, Humidity) = Entropy(PlayGolf)*
  – Entropy(PlayGolf, Humidity)
  = 0.94 – 0.788 ***= 0.152***

- *Gain(PlayGolf, Windy) = Entropy(PlayGolf)*
  – Entropy(PlayGolf, Windy)
  = 0.94 – 0.892 ***= 0.048***

Perform the First Split & Repeat

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



Regression Tree

- Constructed in almost the same manner as a classification tree
- Impurity measure is replaced by a measure appropriate for regression.
  - In regression, the goodness of a split is measured by the mean square error from the estimated value.

$$E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(\boldsymbol{x}^t) \qquad b_m(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathcal{X}_m\text{: } \boldsymbol{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

- Let us say g_m is the estimated value in node m.
- Use the mean (median if there is too much noise) of the required outputs

$$g_m = \frac{\sum_t b_m(\boldsymbol{x}^t) r^t}{\sum_t b_m(\boldsymbol{x}^t)}$$

- If at a node, the error is acceptable, that is, $E_m < \theta_r$, then a leaf node is created and it stores the g_m value.

Split and Sub-tree generation

- If the error is not acceptable, data reaching node *m* is split further such that the sum of the errors in the branches is minimum.
  - we look for the attribute (and split threshold for a numeric attribute) that minimizes the error.

Let us define $\mathcal{X}_{mj}$ as the subset of $\mathcal{X}_m$ taking branch j: $\cup_{j=1}^{n} \mathcal{X}_{mj} = \mathcal{X}_m$. We define

$$b_{mj}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathcal{X}_{mj}\text{: } \boldsymbol{x} \text{ reaches node } m \text{ and takes branch } j \\ 0 & \text{otherwise} \end{cases}$$

g_{mj} is the estimated value in branch j of node m.

$$g_{mj} = \frac{\sum_t b_{mj}(\boldsymbol{x}^t) r^t}{\sum_t b_{mj}(\boldsymbol{x}^t)}$$

and the error after the split is

$$E'_m = \frac{1}{N_m} \sum_j \sum_t (r^t - g_{mj})^2 b_{mj}(\boldsymbol{x}^t)$$

Impact of Error Threshold

- Acceptable error threshold θ_r is the complexity parameter
  - when it is small, we generate large trees and risk overfitting;
  - when it is large, we underfit and smooth too much

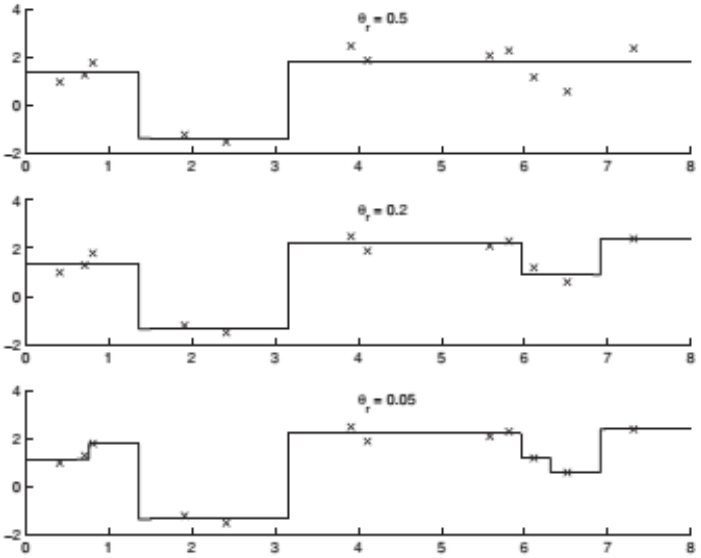


Figure 9.4 Regression tree smooths for various values of θ_r. The corresponding trees are given in figure 9.5.