DATA MINING AND DATA WAREHOUSING

(BCSDM515)

By,
Anusha K S
Assistant Professor, Dept. of CSE
VVCE, Mysuru.

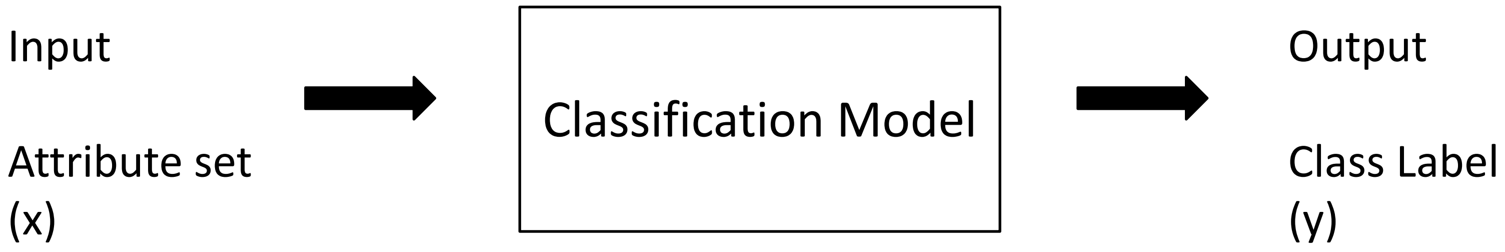
MODULE 3

Classification: General Approach to solving a Classification Problem, Decision Tree Induction, Model Overfitting, Nearest-Neighbour Classifiers, Bayesian Classifiers, Artificial Neural Network(ANN).

SLT: Evaluating the performance of a Classifier

CLASSIFICATION

- It is a task of assigning objects to one of several predefined categories.
- "Classification is the task of learning a target function "f" that maps each attribute set "x" to one of predefined class labels "y".

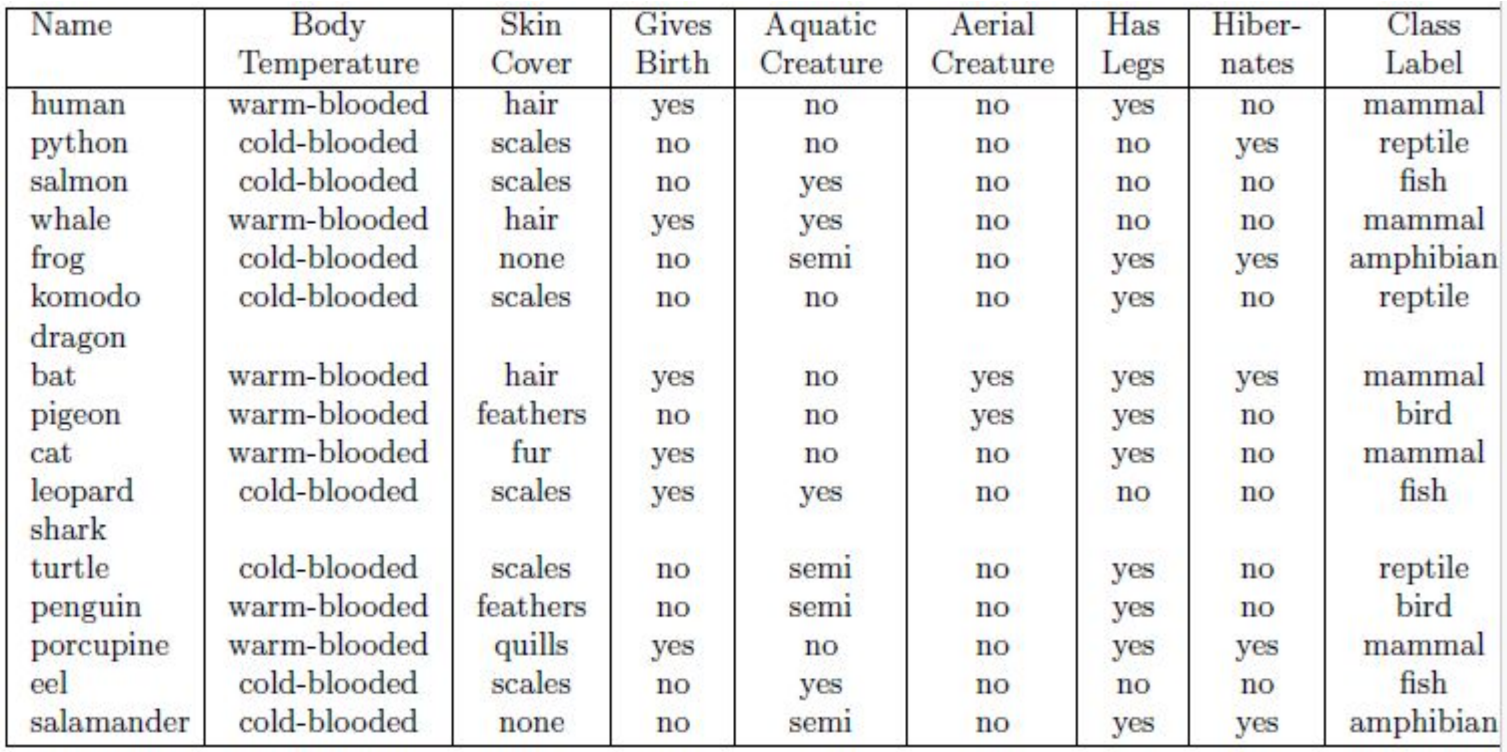


- Example: Detecting spam email message based upon the message header and content.
- Classification techniques are most suited for predicting or describing data sets with binary or nominal categories, less effective for ordinal categories.

Preliminaries

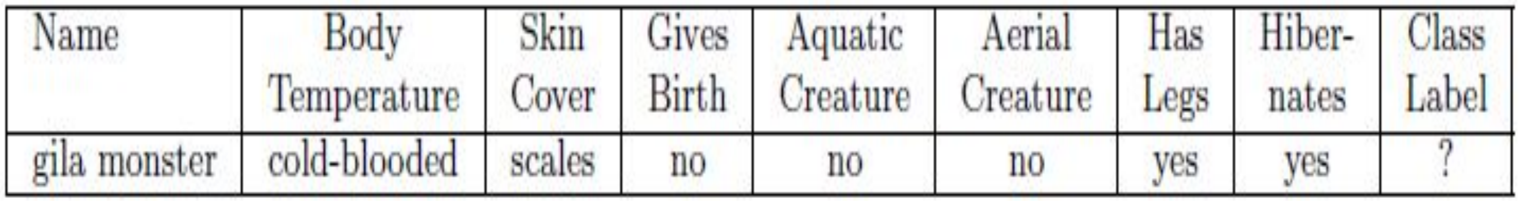
- A Classification model is useful for the following purpose

Descriptive Modeling



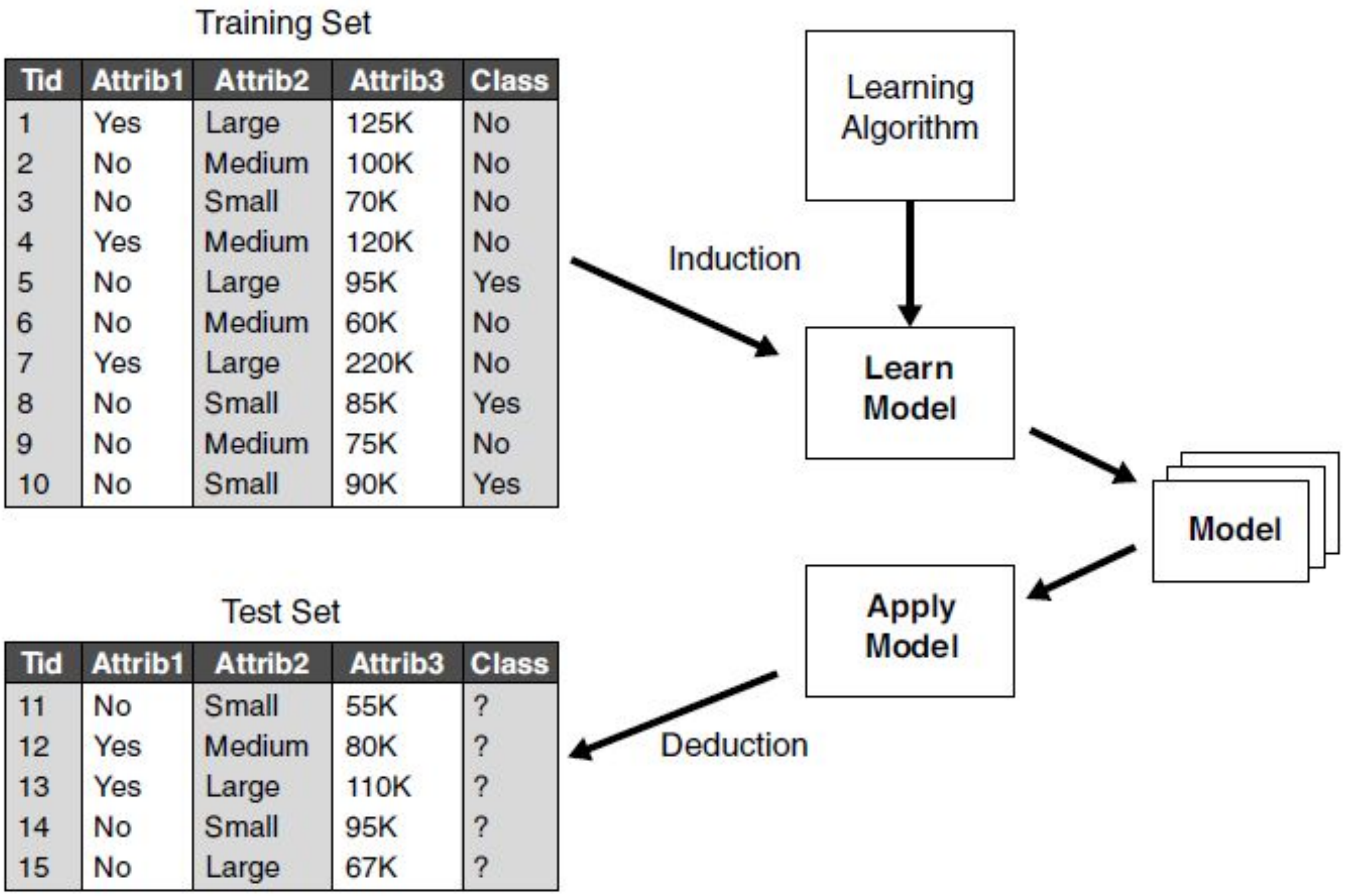
Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Predictive Modeling



Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
gila monster	cold-blooded	scales	no	no	no	yes	yes	?

GENERAL APPROACH TO SOLVING A CLASSIFICATION PROBLEM



Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Figure 4.3. General approach for building a classification model.

Confusion Matrix

Table 4.2. Confusion matrix for a 2-class problem.

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

performance metric

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

Classification algorithms seek highest accuracy, or equivalently, the lowest error rate

Decision Tree Induction

The tree has three types of nodes:

- **Root node,** that has no incoming edges and zero or more outgoing edges
- **Internal nodes**, each of which has exactly one incoming edge and two or more outgoing edges
- **Leaf or terminal nodes,** each of which has exactly one incoming edge and no outgoing edges

In a decision tree, each leaf node is assigned a class label. The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics

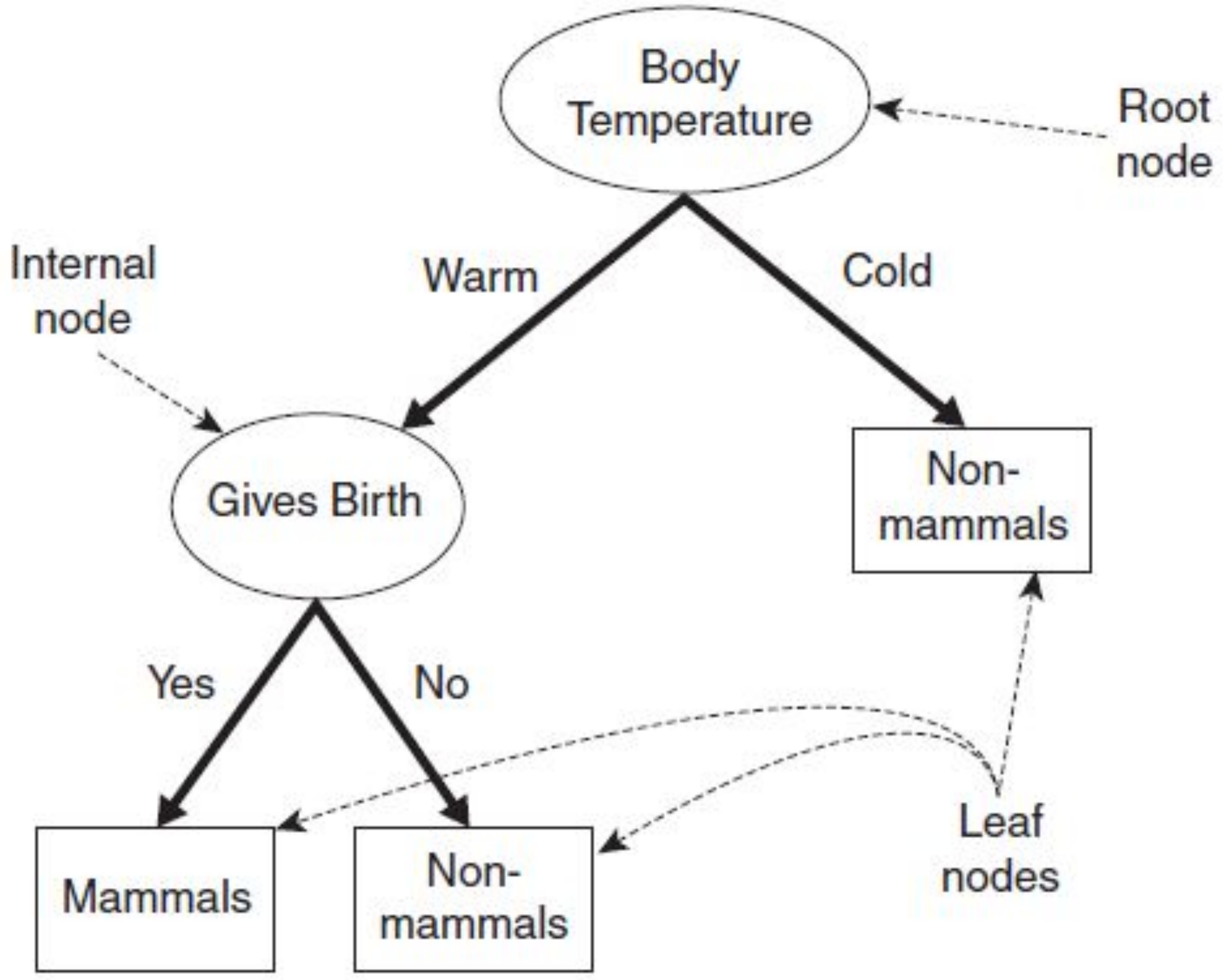
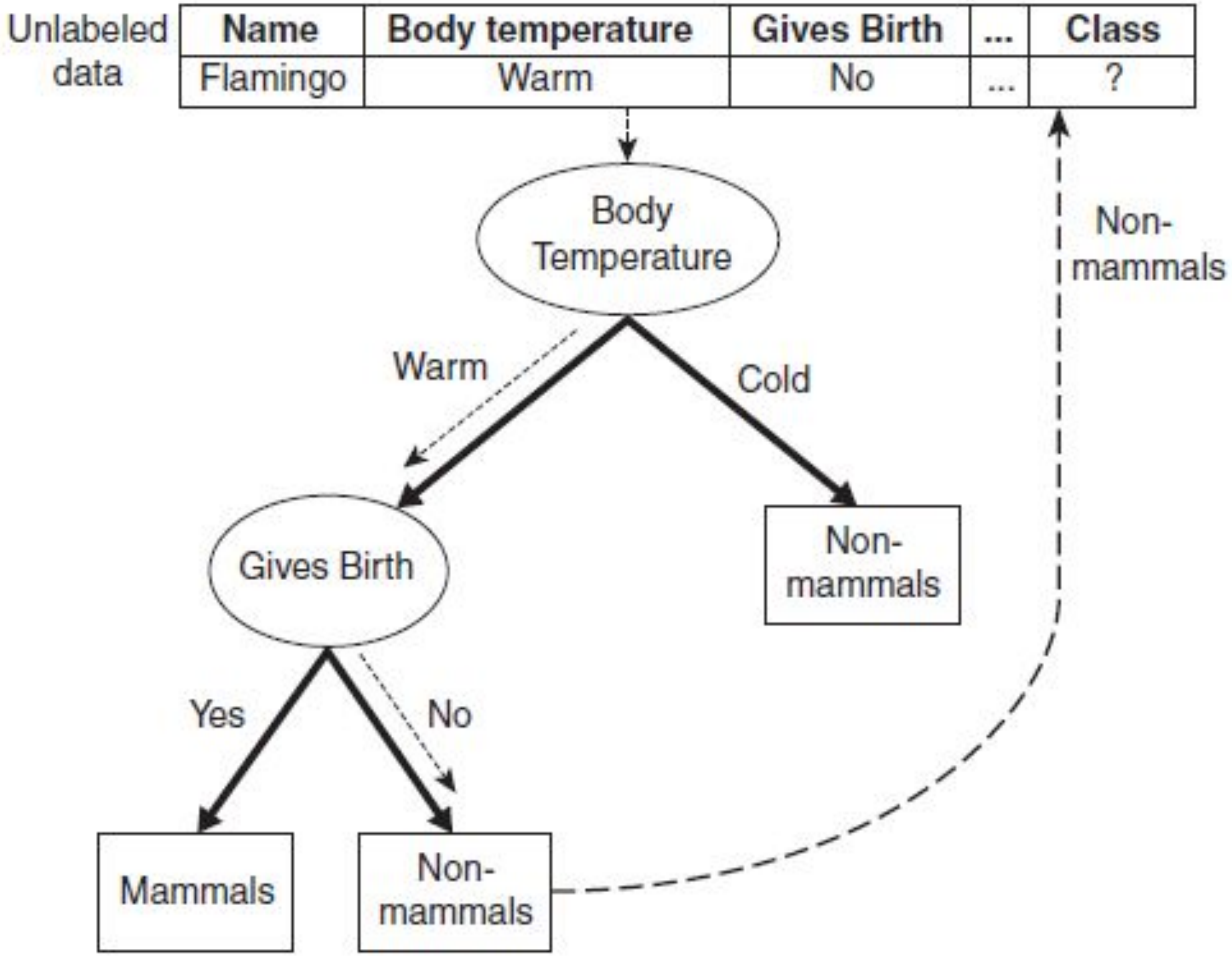


Figure 3.4. A decision tree for the mammal classification problem.



Unlabeled data

Name	Body temperature	Gives Birth	...	Class
Flamingo	Warm	No	...	?

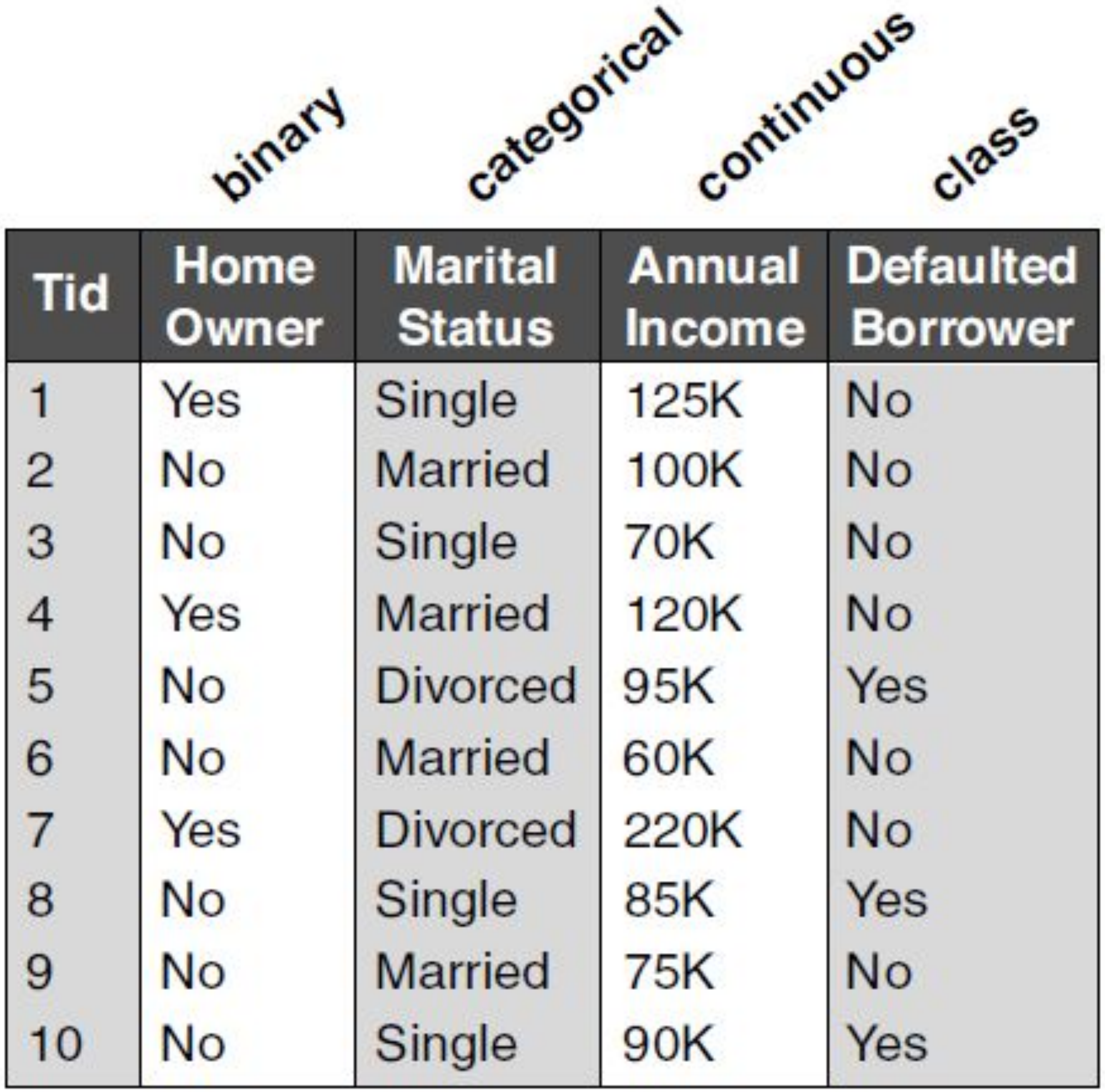
Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let Di be the set of training records that are associated with node t and y= {“y1,y2,y3….yc"} be the class labels.

The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in Dt belong to the same class yt, then t is a leaf node labeled as yt.

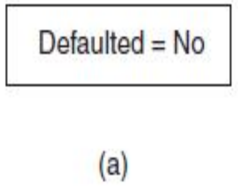
Step 2: If Di contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each ***child node.***

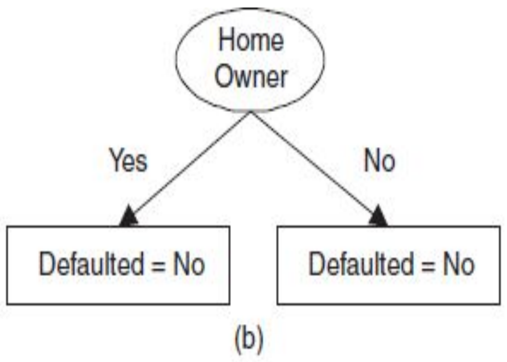


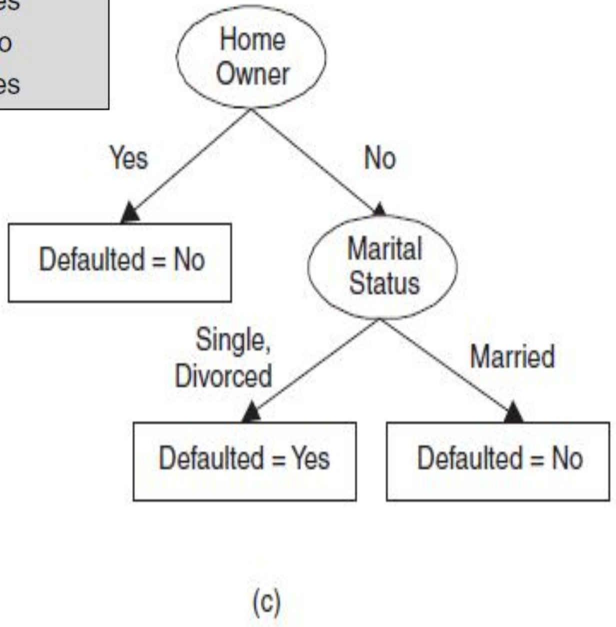
	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

binary categorical continuous class

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes







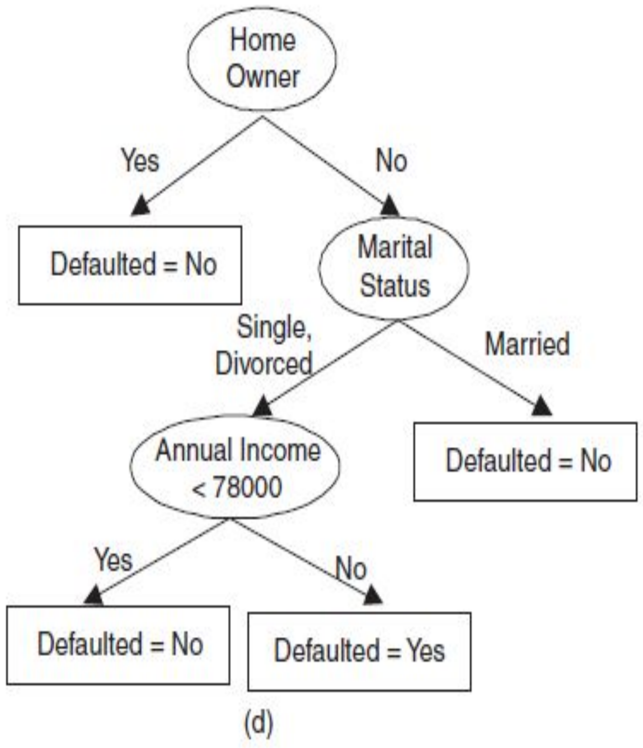
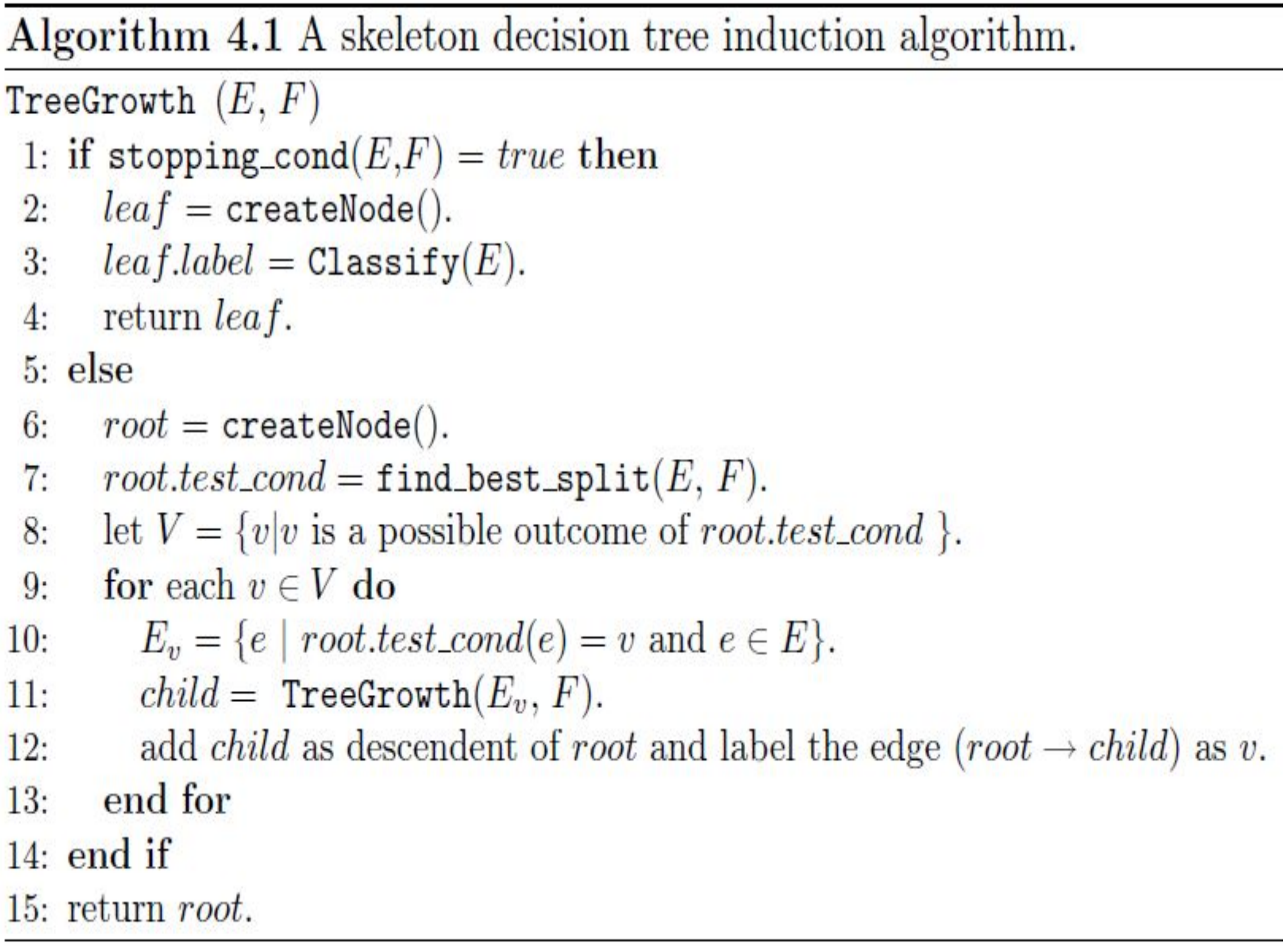


Figure 3.6. Hunt's algorithm for building decision trees.

ALGORITHM FOR DECISION TREE INDUCTION



Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth $(E, F)$
if stopping_cond$(E,F)$ = $true$ then
    $leaf$ = createNode().
    $leaf.label$ = Classify$(E)$.
    return $leaf$.
else
    $root$ = createNode().
    $root.test\_cond$ = find_best_split$(E, F)$.
    let $V = \{v | v$ is a possible outcome of $root.test\_cond\}$.
    for each $v \in V$ do
        $E_v = \{e \mid root.test\_cond(e) = v$ and $e \in E\}$.
        $child$ = TreeGrowth$(E_v, F)$.
        add $child$ as descendent of $root$ and label the edge $(root \rightarrow child)$ as $v$.
    end for
end if
return $root$.
```

Design Issues of Decision Tree Induction

Hunt's algorithm is a generic procedure for growing decision trees in a greedy fashion. To implement the algorithm, there are two key design issues that must be addressed.

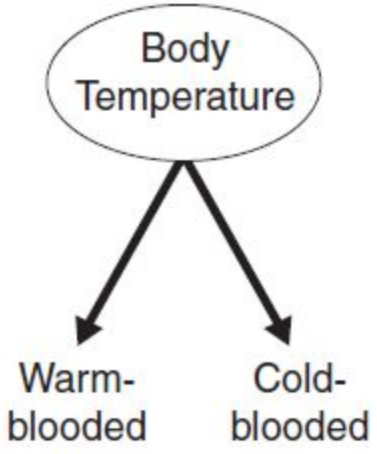
- **What is the splitting criterion?**
- **What is the stopping criterion?**

CHARACTERSTICS OF DECISION TREE INDUCTION

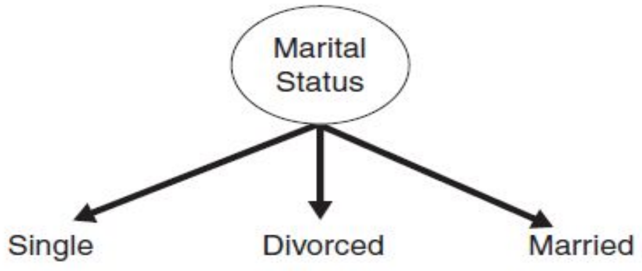
- It is non-parametric approach for building classification model
- Decision tree are computationally inexpensive making it possible to quickly construct even when training set size is very large
- Decision trees, especially smaller-sized tree are relatively easy to interpret
- Decision tree algorithm are quite robust to presence of noise
- Since decision tree uses “top-down” recursive approach the number of records become smaller
- “Sub tree” can be replicated many times
- “test condition” involves using only one attributes at a time

Methods for expressing attributes test condition

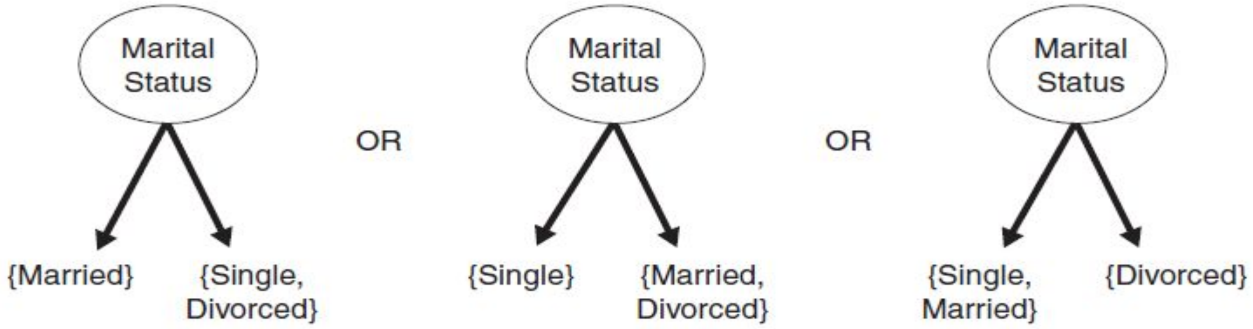
- Binary attributes



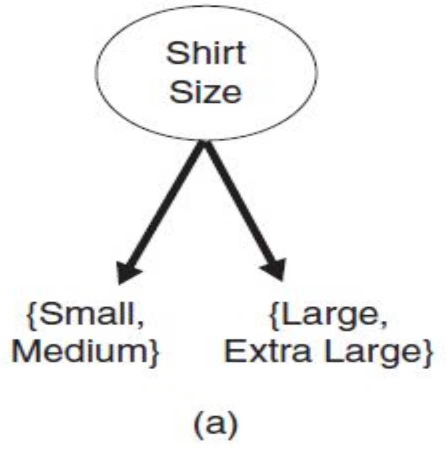
- Nominal attributes

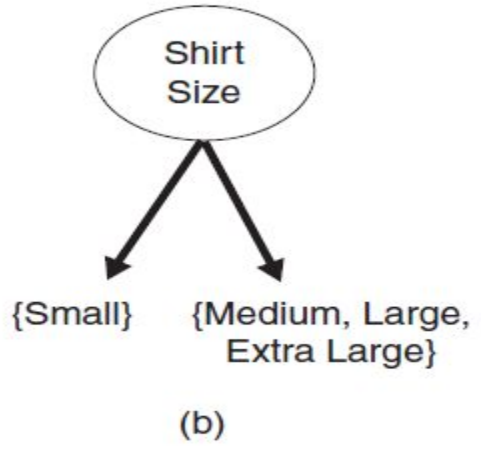


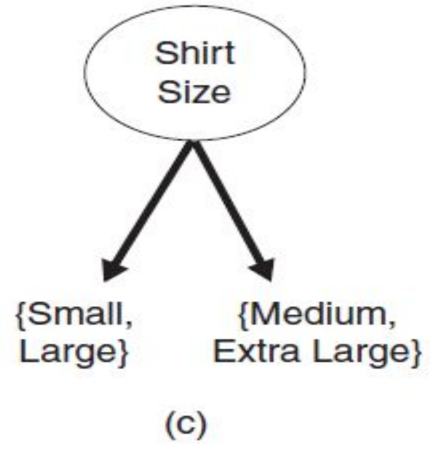
(a) Multiway split



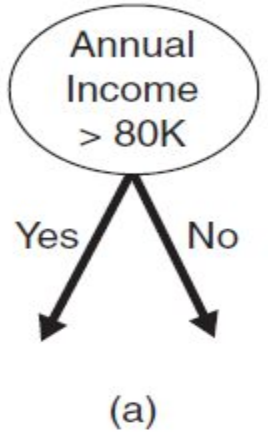
- Ordinary attributes

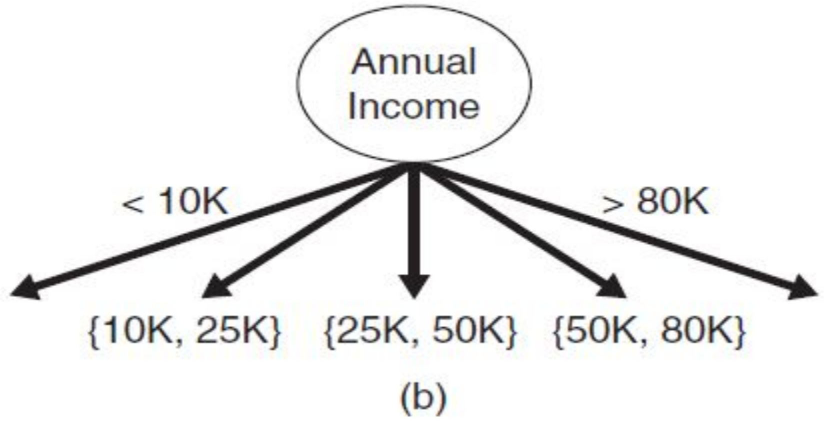






- Continuous attributes





Measures for Selecting the Best Split

- The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
- The impurity measure include the following

$$\begin{aligned}\text{Entropy}(t) &= -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t),\\ \text{Gini}(t) &= 1-\sum_{i=0}^{c-1}[p(i|t)]^2,\\ \text{Classification error}(t) &= 1-\max_i[p(i|t)],\end{aligned}$$

where c is the number of classes and $0\log_2 0 = 0$ in entropy calculations.
$P(i/t)$ denotes the fraction of records belonging to class i at a given node t

Example

Node N_1	Count
Class=0	0
Class=1	6

$$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$$
$$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$
$$\text{Error} = 1 - \max[0/6, 6/6] = 0$$

Node N_2	Count
Class=0	1
Class=1	5

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$
$$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$$
$$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$$

Node N_3	Count
Class=0	3
Class=1	3

$$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$$
$$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$
$$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$$

- To determine how well a test condition performs, we need to compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting). The larger their difference, the better the test condition. The gain Δ, is a criterion that can be used to determine the goodness of a split

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j),$$

- Where,

- I(·) is the impurity measure of a given node
- N is the total number of records at the parent node
- k is the number of attribute values
- N(vj) is the number of records associated with the child node, vj

MODEL OVERFITTING

- The errors committed by a classification model are generally divided into two types

  1. **Training errors** - The number of misclassification errors committed on training records

  2. **Generalization errors** - The expected error of the model on previously unseen records

     I. Overfitting Due to the presence of Noise
     II. Overfitting Due to lack of Representative samples
     III. Overfitting and the Multiple Comparison Procedure
     IV. Estimation of Generalization Errors

- A Good model must have low training error as well as low generalization error.

Evaluating the Performance of a Classifier (4.5 - SLT)

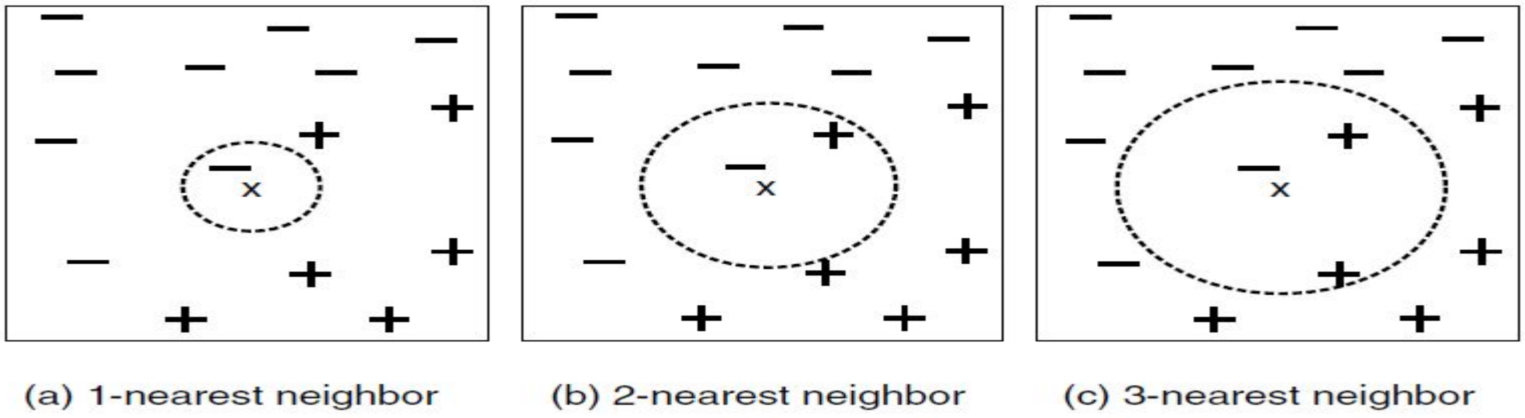
- Holdout Method
- Random Subsampling
- Cross-Validation
- Bootstrap

Nearest Neighbour Classifiers

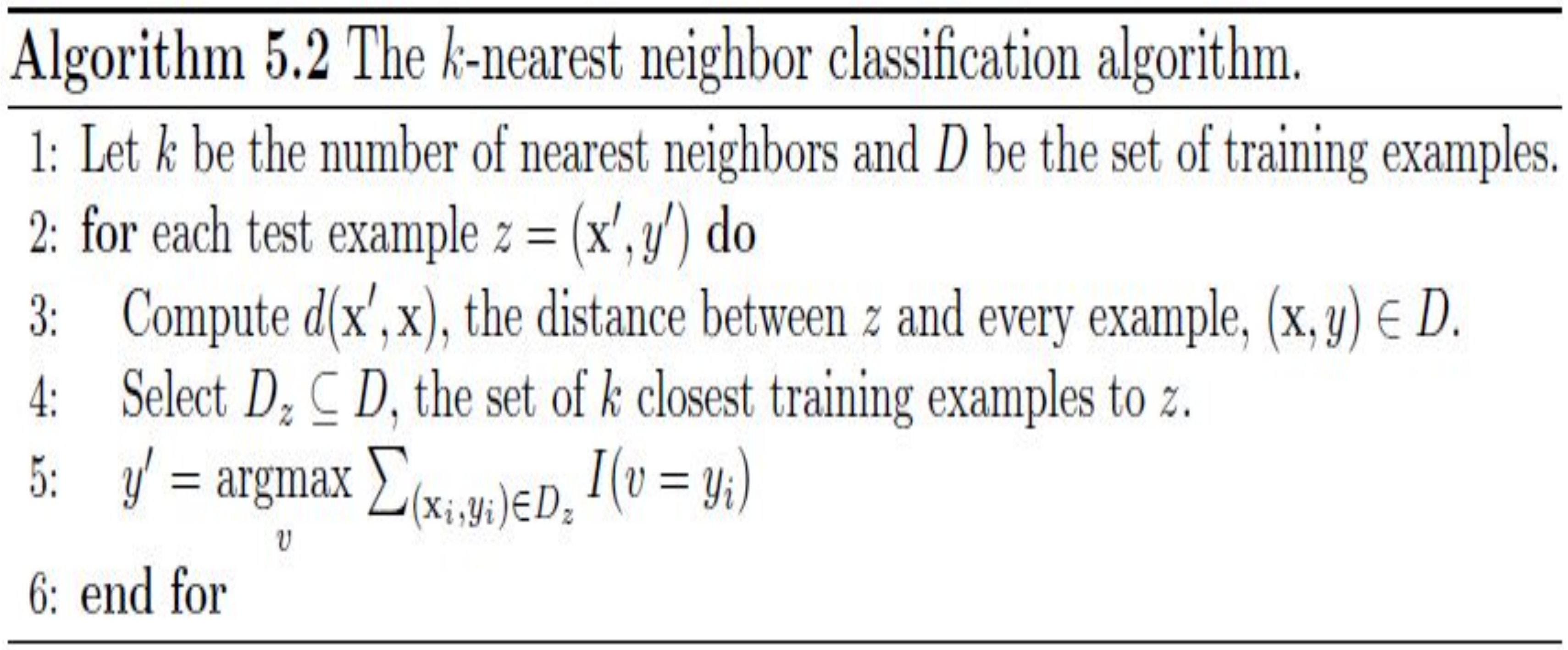
- The classification framework involves a two-step process
  - An **inductive step** for constructing a classification model from data
  - A **deductive step** for applying model to test example
- The decision tree classifier and rule-based classifier are good example of **“Eager learners”**
- The techniques that employee the nearest-neighbour classification are known as **“Lazy learner”**
  - Ex: Rote classifier (Advantage and disadvantage)

Steps that need to be carried out during the KNN algorithm

1. Divide the data into training and test data.
2. Select a value of K
3. Determine which distance function is to be used.
4. Choose a sample from the test data that needs to be classified and compute the distance to its n training samples.
5. Sort the distance obtained and take the k-nearest data samples.
6. Assign the test class to the class based on the majority vote of its K neighbours.



- The algorithm computes the distance or similarity between each test examples z = (x' , y') and all the training examples (x,y) ∈ D to determine its nearest-neighbour list.



Algorithm 5.2 The k-nearest neighbor classification algorithm.

```
1: Let $k$ be the number of nearest neighbors and $D$ be the set of training examples.
2: for each test example $z = (\mathbf{x}', y')$ do
3:    Compute $d(\mathbf{x}', \mathbf{x})$, the distance between $z$ and every example, $(\mathbf{x}, y) \in D$.
4:    Select $D_z \subseteq D$, the set of $k$ closest training examples to $z$.
5:    $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6: end for
```

- Once the nearest neighbour list is obtained, the test examples classified based on the majority class of its nearest neighbour

Characteristics of Nearest-Neighbor Classifiers

- Nearest-neighbor classification is part of a more general technique known as instance-based learning, which uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data
- Lazy learners such as nearest-neighbor classifiers do not require model building
- Nearest-neighbor classifiers make their prediction based on local information
- Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries. The decision boundaries of nearest – neighbor classifier have “high variability” because they depend on the composition of training examples.
- Nearest-neighbor classifiers can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken

Bayesain Classifiers

- **Bayes theorem** is a statistical principle for combining prior knowledge of classes with new evidence gathered from data.

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}$$

- **Navies bayes Classifiers**
  Estimates the lass-conditional probability by assuming that the attributes are conditionally independent, given the class label y

- **Bayesian Belief Network (BBN)**
  Provides the graphical representation of the probabilistic relationships among a set of random variables.

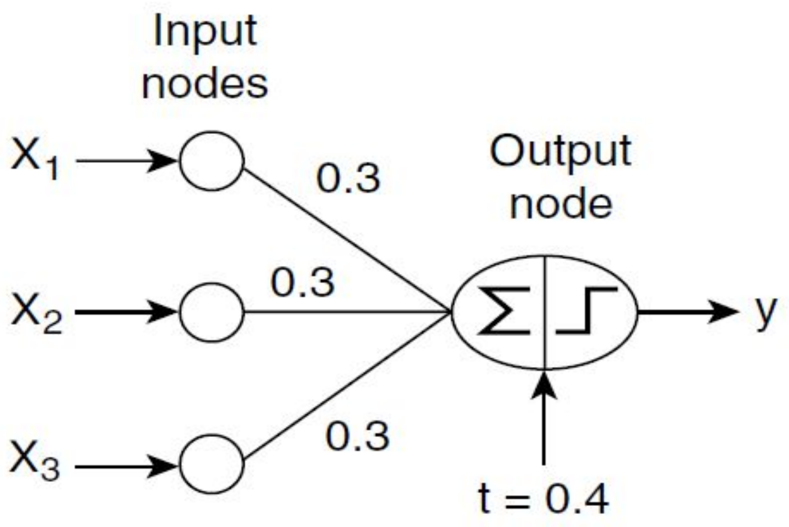
Artificial Neural Network(ANN)

- The study of artificial neural networks (ANN) was inspired by attempts to simulate biological neural systems.
- The human brain consists primarily of nerve cells called **neurons**, linked together with other neurons via strands of fiber called **axons**.
- Axons are used to transmit nerve impulses from one neuron to another whenever the neurons are stimulated.
- A neuron is connected to the axons of other neurons via **dendrites**, which are extensions from the cell body of the neuron.
- The contact point between a dendrite and an axon is called a **synapse.**
- Neurologists have discovered that the human brain learns by changing the strength of the **synaptic connection** between neurons upon repeated stimulation by the same impulse.

Perceptron

X_1	X_2	X_3	y
1	0	0	−1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	−1
0	1	0	−1
0	1	1	1
0	0	0	−1

(a) Data set.



(b) Perceptron.

Figure 5.14. Modeling a boolean function using a perceptron.

- The weighted link is used to emulate the strength of synaptic connection between neurons.
- As in biological neural systems, training a perceptron model amounts to adapting the weights of the links until they fit the input-output relationships of the underlying data.

- The model shown in Figure 5.14(b) has three input nodes, each of which has an identical weight of 0.3 to the output node and a **bias factor of *t* = 0.4**. The output computed by the model is

$$\hat{y} = \begin{cases} 1, & \text{if } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0; \\ -1, & \text{if } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0. \end{cases}$$

- Ex: x1=1, x2=1, x3=0 is **+ve i.e., 1** and x1=0, x2=1, x3=0 is **–ve i.e., -1**
- An **input node** simply transmits the value it receives to the outgoing link without performing any transformation.
- The output node, on the other hand, is a mathematical device that computes the weighted sum of its inputs, subtracts the bias term, and then produces an output that depends on the sign of the resulting sum.
- O/p of perceptron model: $\hat{y} = sign\big(w_d x_d + w_{d-1} x_{d-1} + \ldots + w_2 x_2 + w_1 x_1 - t\big),$

Learning Perceptron Model

- During the training phase of a perceptron model, the weight parameters **w** are adjusted until the outputs of the perceptron become consistent with the true outputs of training examples.

Algorithm 5.4 Perceptron learning algorithm.

```
1: Let $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, N\}$ be the set of training examples.
2: Initialize the weight vector with random values, $\mathbf{w}^{(0)}$
3: repeat
4:     for each training example $(\mathbf{x}_i, y_i) \in D$ do
5:         Compute the predicted output $\hat{y}_i^{(k)}$
6:         for each weight $w_j$ do
7:             Update the weight, $w_j^{(k+1)} = w_j^{(k)} + \lambda\big(y_i - \hat{y}_i^{(k)}\big)x_{ij}$.
8:         end for
9:     end for
10: until stopping condition is met
```

Where,

w(k) - is the weight parameter associated with the ith input link after the kth iteration

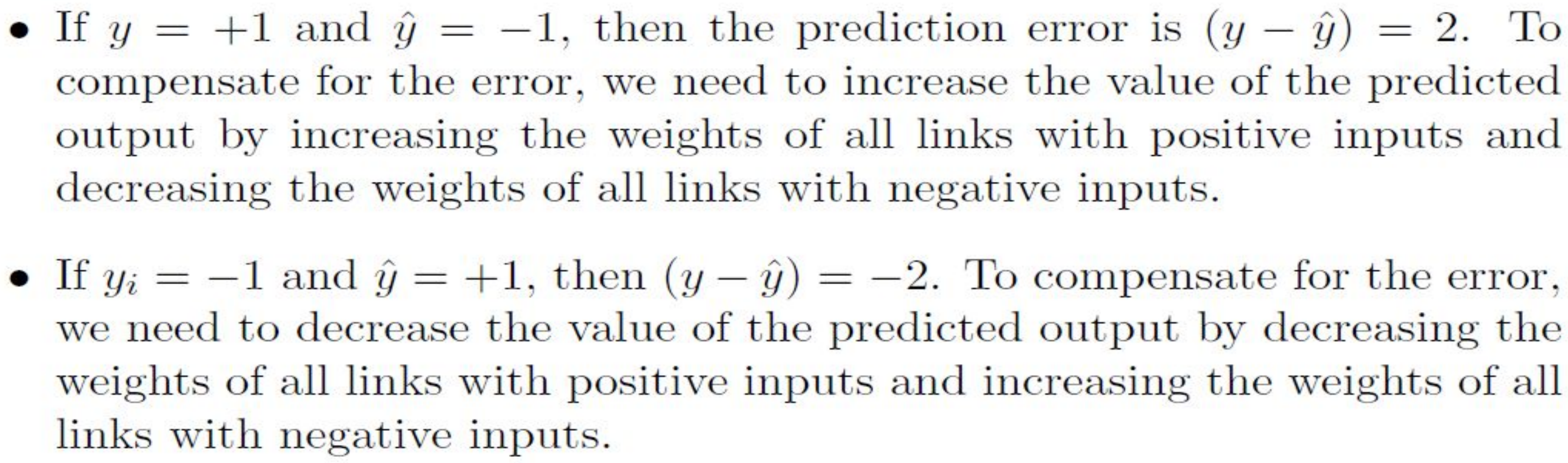
λ - is a parameter known as the **learning rate**

x_{ij} - is the value of the jth attribute of the training example x_i

***w*(*k*+1)** - is a combination of the old weight $w(k)$ and a term proportional to the prediction

error, $(y - \hat{y})$

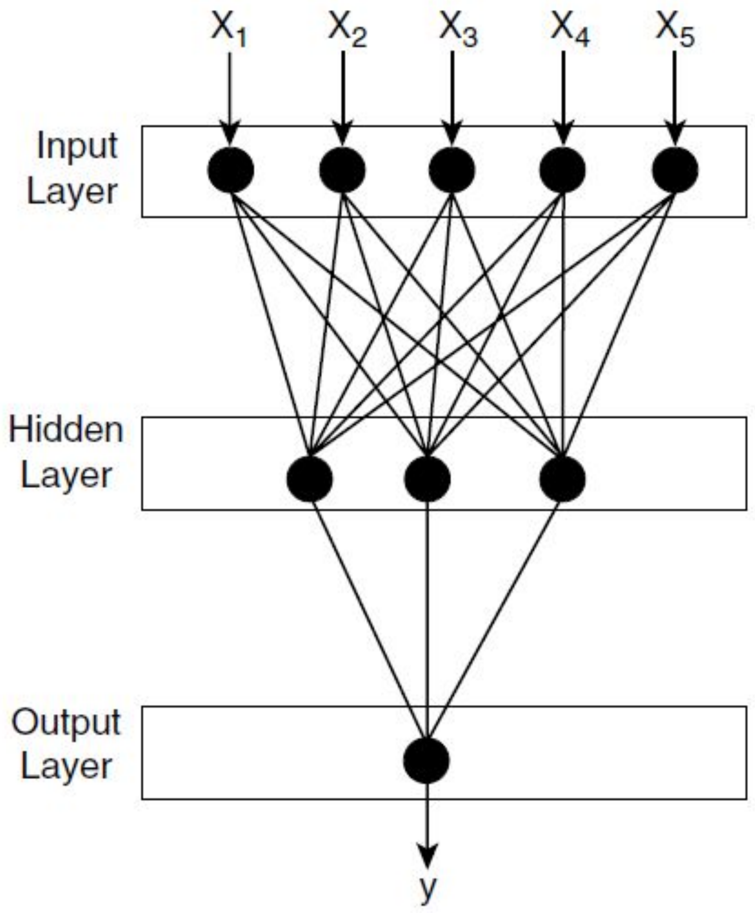
- If the prediction is correct, then the weight remains unchanged. Otherwise, it is

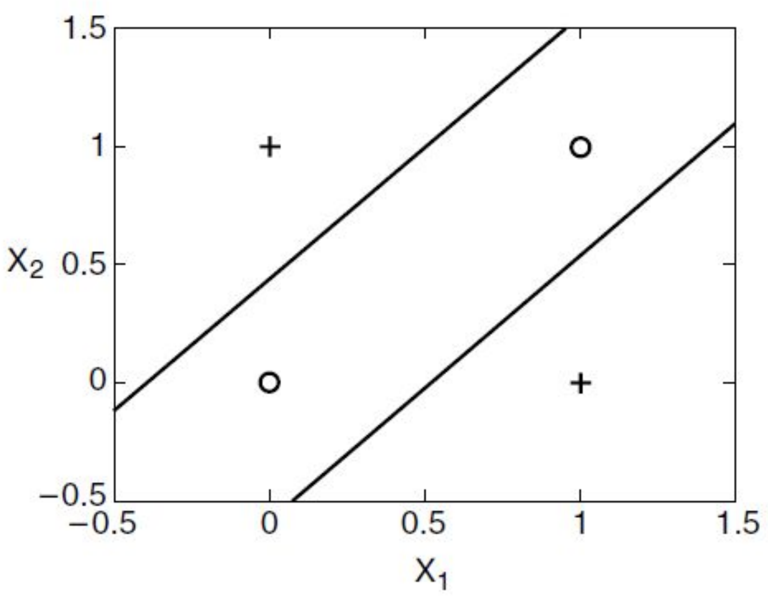


- If $y = +1$ and $\hat{y} = -1$, then the prediction error is $(y - \hat{y}) = 2$. To compensate for the error, we need to increase the value of the predicted output by increasing the weights of all links with positive inputs and decreasing the weights of all links with negative inputs.

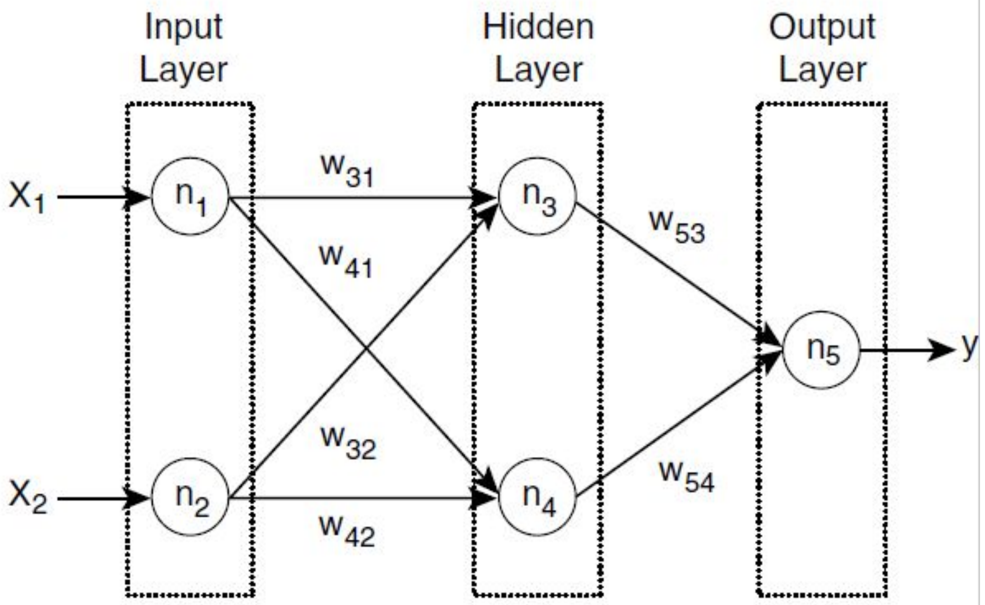
- If $y_i = -1$ and $\hat{y} = +1$, then $(y - \hat{y}) = -2$. To compensate for the error, we need to decrease the value of the predicted output by decreasing the weights of all links with positive inputs and increasing the weights of all links with negative inputs.

Multilayer Artificial Neural Network





(a) Decision boundary.



(b) Neural network topology.

Figure 5.19. A two-layer, feed-forward neural network for the XOR problem.

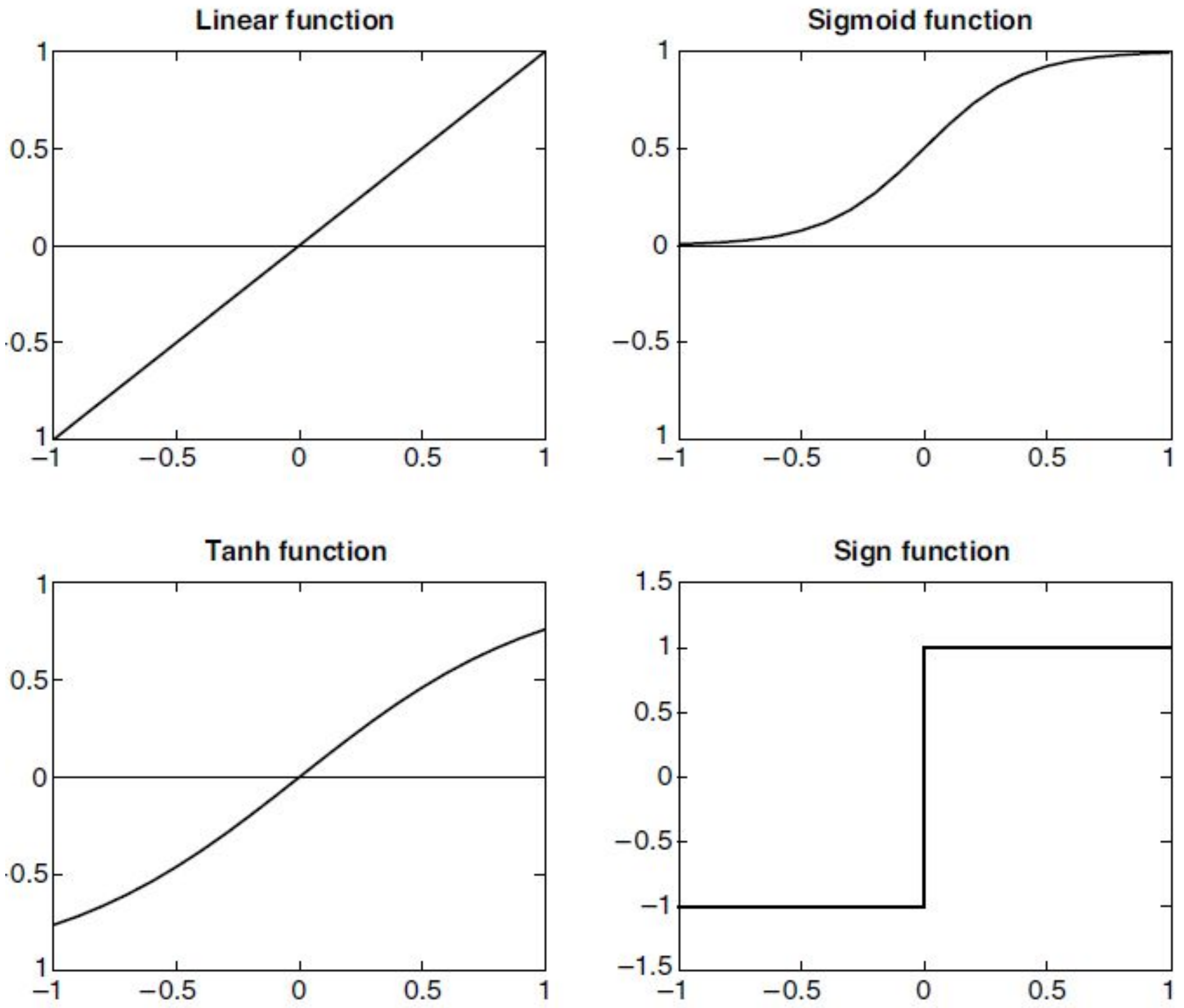


Figure 5.18. Types of activation functions in artificial neural networks.

Design Issues in ANN Learning

Before we train a neural network to learn a classification task, the following design issues must be considered.

- The number of nodes in the input layer should be determined.
- The number of nodes in the output layer should be established.
- The network topology (e.g., the number of hidden layers and hidden nodes, and feed-forward or recurrent network architecture) must be selected.
- The weights and biases need to be initialized. Random assignments are usually acceptable.
- Training examples with missing values should be removed or replaced with most likely values.

Characteristics of ANN

1. Multilayer neural networks with at least one hidden layer are **universal approximators**

2. ANN can handle redundant features because the weights are automatically learned during the training step. The weights for redundant features tend to be very small.

3. Neural networks are quite sensitive to the presence of noise in the training data. One approach to handling noise is to use a validation set to determine the generalization error of the model. Another approach is to decrease the weight by some factor at each iteration.

4. The gradient descent method used for learning the weights of an ANN often converges to some local minimum. One way to escape from the local minimum is to add a momentum term to the weight update formula.

5. Training an ANN is a time-consuming process, especially when the number of hidden nodes is large. Nevertheless, test examples can be classified rapidly.

Thank you