MODULE 4
DECISION TREE & NEURAL NETWORKS

SLIDES ADAPTED (AND EXTENDED) FROM ETHEM ALPAYDIN © THE MIT PRESS, 2004

SYLLABUS

Decision Trees- Entropy, Information Gain, Tree construction, ID3, Issues in Decision Tree learning- Avoiding Over-fitting, Reduced Error Pruning, The problem of Missing Attributes, Gain Ratio, Classification by Regression (CART), Neural Networks- The Perceptron, Activation Functions, Training Feed Forward Network by Back Propagation.

DECISION TREE

Decision Tree Learning

- method for approximating discrete valued target functions
- learned function is represented by a decision tree
- one of the most widely used and practical methods for inductive inference

Candidate looking for a job:

- monthly salary is at least Rs.50,000
- commuting time is less than one hour
- free coffee every morning

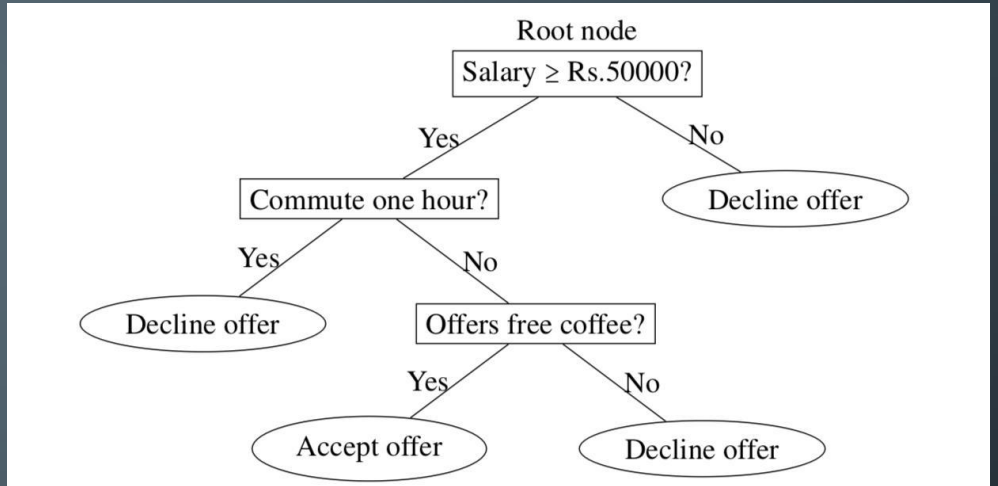
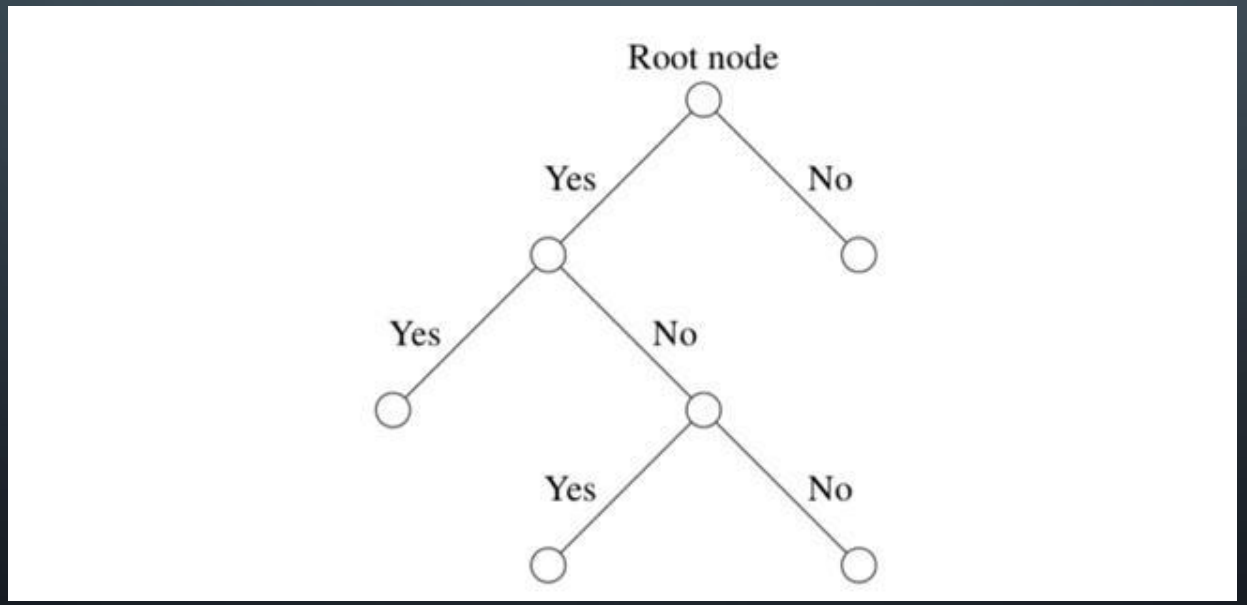


Figure 8.1: Example for a decision tree



TYPES OF DECISION TREES

1. Classification trees
   - target variable can take a discrete set of values
   - leaves represent class labels
   - branches represent conjunctions of features that lead to those class labels

2. Regression trees
   - target variable can take continuous values (real numbers)

     Eg. price of a house, or a patient's length of stay in a hospital

CLASSIFICATION TREE

8.3.1 Example

Data

Nam	Features				Class label
	gives birth	aquatic animal	aerial animal	has legs	
human	yes	no	no	yes	mammal
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
bat	yes	no	yes	yes	bird
pigeon	no	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.1: The vertebrate data set

TREES CONSTRUCTION

Step 1:

- Split into disjoint subsets according to the values of the feature "gives birth"

Nam	Features				Class label
	gives birth	aquatic animal	aerial animal	has legs	
human	yes	no	no	yes	mammal
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
bat	yes	no	yes	yes	bird
pigeon	no	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Name	Gives birth	Aquatic animal	Aerial animal	Has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish

Table 8.2: The subset of Table 8.1 with "gives birth" = "yes"

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
pigeon	no	no	yes	yes	bird
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.3: The subset of Table 8.1 with "gives birth" = "no"

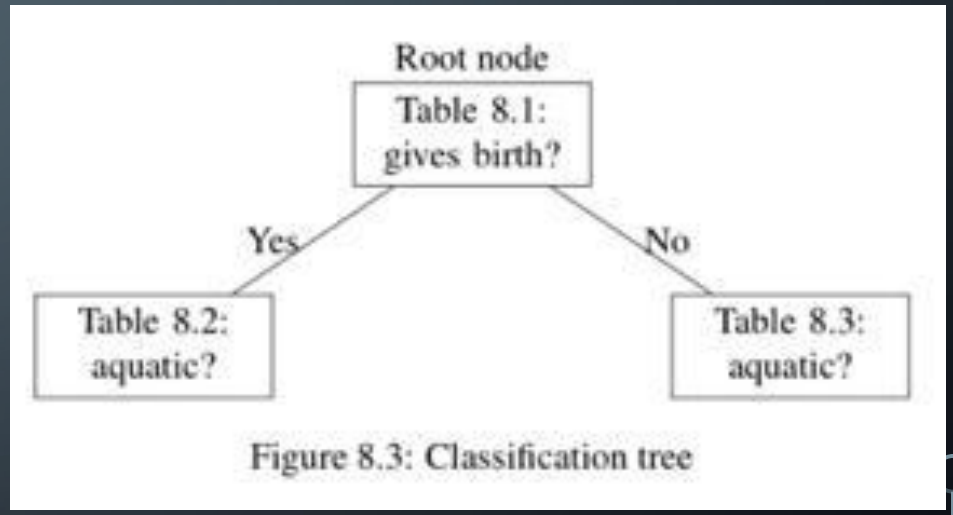


Figure 8.3: Classification tree

Step 2:

- Consider table 8.2
- Split examples based on the values of the feature "aquatic animal"

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal

Table 8.5: The vertebrate data set

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
shark	yes	yes	no	no	fish

Table 8.4: The vertebrate data set

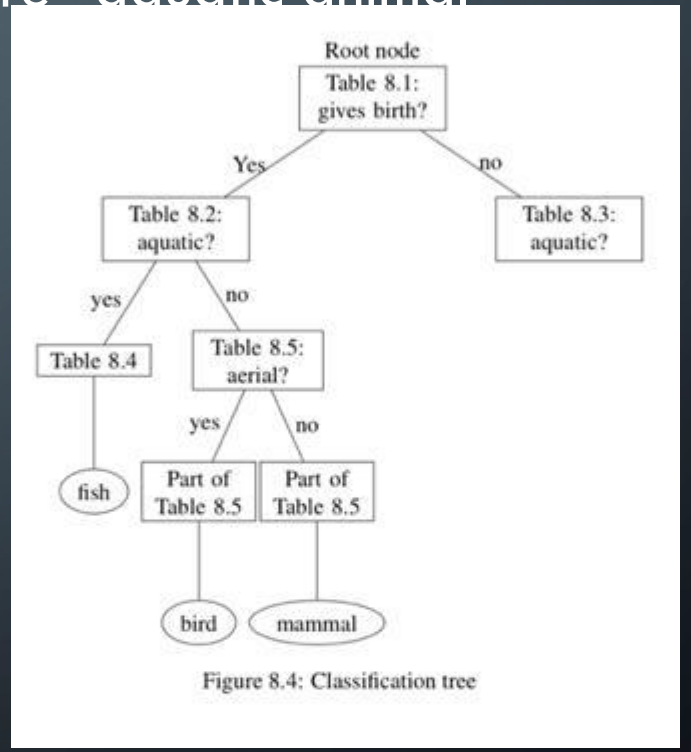


Figure 8.4: Classification tree

Step 3:

- Consider table 8.3
- Split examples based on the values of the feature “aquatic animal”

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
salmon	no	yes	no	no	fish

Table 8.6: The vertebrate data set

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
frog	no	semi	no	yes	amphibian
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.7: The vertebrate data set

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
python	no	no	no	no	reptile
pigeon	no	no	yes	yes	bird

Table 8.8: The vertebrate data set

Complete tree structure:

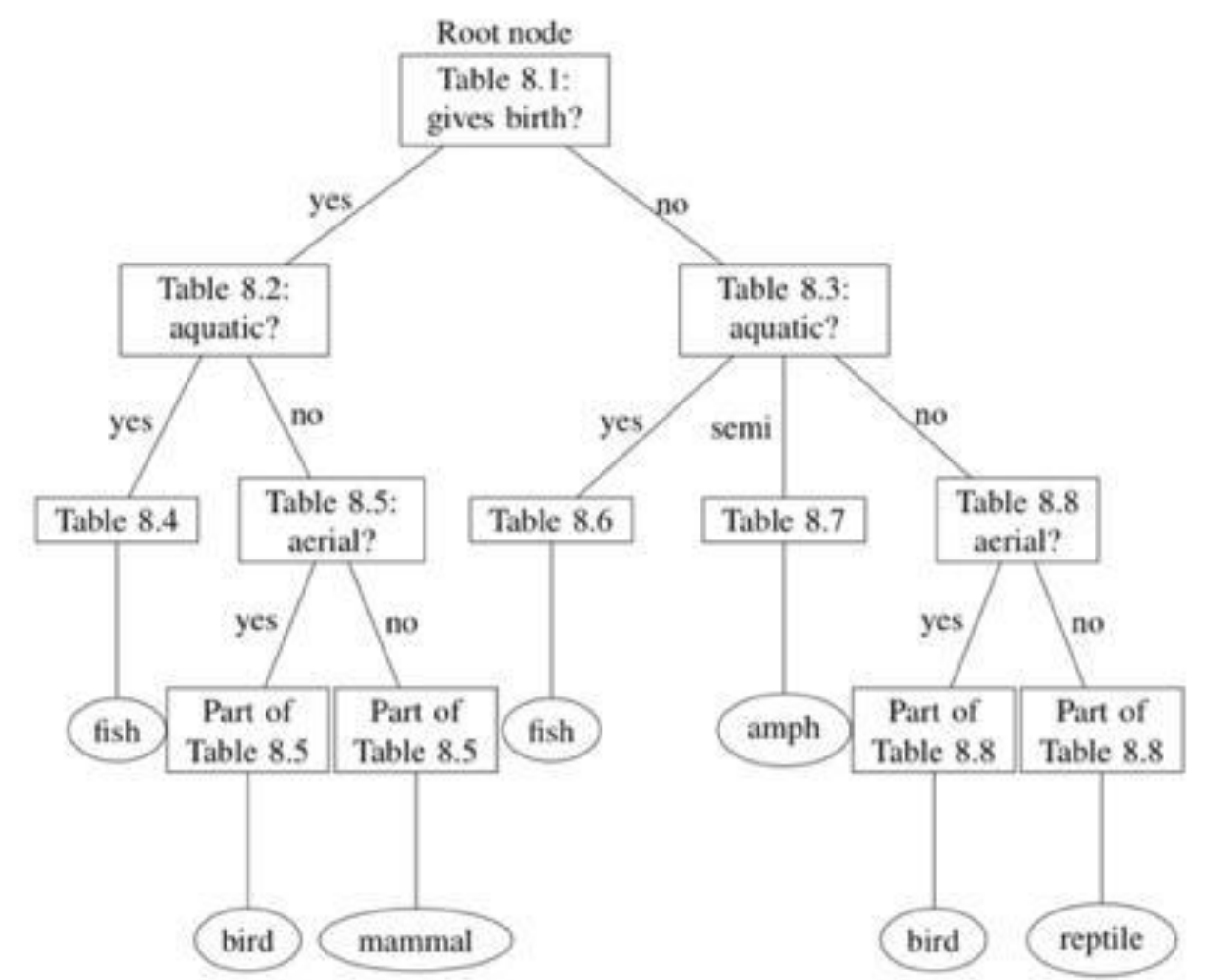


Figure 8.5: Classification tree

ALGORITHM

Algorithm for classification of vertebrates

```
1.  if give birth = "yes" then
2.      if aquatic = "yes" then
3.          return class = "fish"
4.      else
5.          if aerial = "yes" then
6.              return class = "bird"
7.          else
8.              return class = "mammal"
9.          end if
10.     end if
11. else
12.     if aquatic = "yes" then
13.         return class = "fish"
14.     end if
15.     if aquatic = "semi" then
16.         return class = "amphibian"
17.     else
18.         if aerial = "yes" then
19.             return class = "amphibian"
20.         else
21.             return class = "reptile"
22.         end if
23.     end if
24. end if
```

CLASSIFICATION TREES

The various elements in a classification tree are identified as follows:

- Nodes in the classification tree are identified by the feature names of the given data.
- Branches in the tree are identified by the values of features.
- The leaf nodes identified by are the class labels.

On the order in which the features are selected:

- The classification tree depends on the order in which the features are selected for partitioning the data

Stopping criteria:

Classification trees will naturally be more complex in real world

- All (or nearly all) of the examples at the node have the same class.
- There are no remaining features to distinguish among the examples.
- The tree has grown to a predefined size limit.

FEATURE SELECTION MEASURES

- Complicated problem to Deciding which attribute is to be to placed at
  - the root
  - different levels of the tree as internal nodes
- Random selection not enough
- **Feature selection measures**
  - Methods to assign numerical values to the various features exist
  - Values reflect the relative importance of the various features
- Two of the popular feature selection measures are
  - Information gain
  - Gini index

ENTROPY

- **Purity:** degree to which a subset of examples contains only a single class
- **Pure class:** subset composed of only a single class
- *Informally*: entropy is a measure of "impurity" in a dataset
- High entropy
  - very diverse
  - provide little information about other items in the set
  - No commonality
- Entropy is measured in bits

- Entropy values range from:
  - For 2 classes – 0 to 1
  - For n classes – 0 to $\log_2(n)$
- minimum value indicates that the sample is completely homogeneous
- maximum value indicates that the data are as diverse as possible

Formal Definition

Consider a segment S of a dataset having c number of class labels.

Let p_i be the proportion of examples in S having the i^{th} class label.

The entropy of S is defined as $$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

EXAMPLES

1. **Entropy of data in Table 8.1**

Let S be the data in Table 8.1. The class labels are "amphi", "bird", "fish", "mammal" and "reptile". In S we have the following numbers.

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

Therefore, we have:

$$\text{Entropy } (S) = \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}})$$

$$
\begin{aligned}
&= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&\quad - p_{\text{fish}} \log_2(p_{\text{fish}}) - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&= -(3/10) \log_2(3/10) - (2/10) \log_2(2/10) \\
&\quad - (2/10) \log_2(2/10) - (2/10) \log_2(2/10) \\
&\quad - (1/10) \log_2(1/10) \\
&= 2.2464
\end{aligned}
$$

2. **Entropy of data in Table 8.2**

Consider the segment S of the data in Table 8.1 given in Table 8.2. For quick reference, the table has been reproduced below:

Name	Gives birth	Aquatic animal	Aerial animal	Has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish

Three class labels appear in this segment, namely, "bird", "fish" and "mammal". We have:

Number of examples with class label "bird"	1
Number of examples with class label "fish"	1
Number of examples with class label "mammal"	2
Total number of examples	4

Therefore we have

$$\begin{aligned}
\text{Entropy}\,(S) &= \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}}) \\
&= -p_{\text{bird}} \log_2(p_{\text{bird}}) - p_{\text{fish}} \log_2(p_{\text{fish}}) \\
&\quad - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&= -(1/4)\log_2(1/4) - (1/4)\log_2(1/4) - (2/4)\log_2(2/4) \\
&= -(1/4) \times (-2) - (1/4) \times (-2) - (2/4) \times (-1) \\
&= 1.5 \qquad (8.1)
\end{aligned}$$

INFORMATION GAIN

- Let
  - S be a set of examples
  - A be a feature (or, an attribute),
  - Sv be the subset of S with A = v,
  - Values (A) be the set of all possible values of A.
- Then the information gain of an attribute A relative to the set S, denoted by Gain(S,A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where |S| denotes the number of elements in S

Example

Consider the data S given in Table 8.1. We have have already seen that

$$|S| = 10$$
$$\text{Entropy}\,(S) = 2.2464.$$

We denote the information gain corresponding to the feature "xxx" by Gain (S, xxx).

1. **Computation of Gain $(S$, gives birth$)$**

$$\begin{aligned}
A_1 &= \text{gives birth}\\
\text{Values of } A_1 &= \{\text{"yes"}, \text{"no"}\}\\
S_{A_1=\text{yes}} &= \text{Data in Table 8.2}\\
|S_{A_1=\text{yes}}| &= 4\\
\text{Entropy}\,(S_{A_1=\text{yes}}) &= 1.5 \quad \text{(See Eq.(8.1))}\\
S_{A_1=\text{no}} &= \text{Data in Table 8.3}\\
|S_{A_1=\text{no}}| &= 6\\
\text{Entropy}\,(S_{A_1=\text{no}}) &= 1.7925 \quad \text{(See Eq.(8.2))}
\end{aligned}$$

Now we have

$$\text{Gain}(S, A_1) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A_1)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$\begin{aligned}
&= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}})\\
&\qquad - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}})\\
&= 2.2464 - (4/10) \times 1.5 - (6/10) \times 1.7925\\
&= 0.5709
\end{aligned}$$

2. **Computation of Gain (S, aquatic)**

$$
\begin{aligned}
A_2 &= \text{aquatic} \\
\text{Values of } A_2 &= \{\text{“yes”}, \text{“no”}, \text{“semi”}\} \\
S_{A_2=\text{yes}} &= \text{See Table 8.1} \\
|S_{A_2=\text{yes}}| &= 2 \\
\text{Entropy}\,(S_{A_2=\text{yes}}) &= -p_{\text{fish}} \log_2(p_{\text{fish}}) \\
&= -(2/2)\log_2(2/2) \\
&= 0 \\
S_{A_2=\text{no}} &= \text{See Table 8.1} \\
|S_{A_2=\text{no}}| &= 5 \\
\text{Entropy}\,(S_{A_2=\text{no}}) &= -p_{\text{mammal}} \log_2(p_{\text{mammal}}) - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&\quad - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&= -(2/5) \times \log_2(2/5) - (1/5) \times \log_2(1/5) \\
&\quad - (2/5) \times \log_2(2/5) \\
&= 1.5219 \\
S_{A_2=\text{semi}} &= \text{See Table 8.1} \\
|S_{A_2=\text{semi}}| &= 3 \\
\text{Entropy}\,(S_{A_2=\text{semi}}) &= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) \\
&= -(3/3) \times \log_2(3/3) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
\text{Gain}(S, A_2) &= \text{Entropy}(S) - \sum_{v \in \text{Values}(A_2)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v) \\
&= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) \\
&\quad - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}}) \\
&\quad - \frac{|S_{A_1=\text{semi}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{semi}}) \\
&= 2.2464 - (2/10) \times 0 - (5/10) \times 1.5219 - (3/3) \times 0 \\
&= 1.48545
\end{aligned}
$$

GINI INDICES

- Feature selection method used in the CART algorithm
- GINI Index:
  - Consider a data set S having r class labels c_1 ,....., c_r.
  - p_i the proportion of examples having the class label c_i
  - The Gini index of the data set S, denoted by Gini(S)
  - Defined by

$$\text{Gini}(S) = 1 - \sum_{i=1}^{r} p_i^2.$$

Example

Let S be the data in Table 8.1. There are four class labels "amphi", "bird", "fish", "mammal" and "reptile". The numbers of examples having these class labels are as follows:

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

The Gini index of S is given by

$$\begin{aligned}\text{Gini}(S) &= 1 - \sum_{i=1}^{r} p_i^2 \\ &= 1 - (3/10)^2 - (2/10)^2 - (2/10)^2 - (2/10)^2 - (1/10)^2 \\ &= 0.78\end{aligned}$$

GINI SPLIT INDEX

- Let
  - S be a set of examples
  - A be a feature (or, an attribute)
  - Sv be the subset of S with A = v
  - Values (A) be the set of all possible values of A
- Then the Gini split index of A relative to S,
  - Denoted by Ginisplit(S,A)
  - Defined as

$$\text{Gini}_{\text{split}}(S, A) = \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Gini}(S_v).$$

where |S| denotes the number of elements in S

GAIN RATIO

- Feature selection measure
- Let
  - S be a set of examples
  - A a feature having c different values
  - Values(A) denotes the set of values of A
- Gain of A relative to S, denoted by Gain(S,A):

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values(A)}} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

- Split information of A relative to S, denoted by SplitInformation(S,A):

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- The gain ratio of A relative to S, denoted by GainRatio(S,A):

$$\text{GainRatio}(S, A) = \frac{\text{Gain(S,A)}}{\text{SplitInformation}(S, A)}$$

8.8.1 Example

Consider the data S given in Table 8.1. Let A denote the attribute "gives birth".We have have already seen that

$$|S| = 10$$
$$\text{Entropy}\ (S) = 2.2464$$
$$\text{Gain}(S, A) = 0.5709$$

Now we have

$$\begin{aligned}\text{SplitInformation}(S, A) &= -\frac{|S_{\text{yes}}|}{|S|}\log_2\frac{|S_{\text{yes}}|}{|S|} - \frac{|S_{\text{no}}|}{|S|}\log_2\frac{|S_{\text{no}}|}{|S|} \\ &= -\frac{4}{10}\times\log_2\frac{4}{10} - \frac{6}{10}\times\log_2\frac{6}{10} \\ &= 0.9710\end{aligned}$$

$$\begin{aligned}\text{GainRatio} &= \frac{0.5709}{0.9710} \\ &= 0.5880\end{aligned}$$

In a similar way we can compute the gain ratios Gain(S, "aquatic"), Gain(S, "aerial") and Gain(S, "has legs"

DECISION TREE ALGORITHMS

- Outline
  1. Place the “best” feature (or, attribute) of the dataset at the root of the tree
  2. Split the training set into subsets-each subset contains data with the same value for a feature
  3. Repeat Step 1 and Step 2 on each subset until we find leaf nodes in all the branches of the tree
- Some well-known decision tree algorithms
  - 1. ID3 (Iterative Dichotomiser 3) developed by Ross Quinlan
  - 2. C4.5 developed by Ross Quinlan
  - 3. C5.0 developed by Ross Quinlan
  - 4. CART (Classification And Regression Trees)
  - 5. 1R (One Rule) developed by Robert Holte in 1993.
  - 6. RIPPER (Repeated Incremental Pruning to Produce Error Reduction)

ID3 ALGORITHM

ASSUMPTIONS:

- Information gain used to select the most useful attribute for classification
- We assume that there are only two class labels "+" and "−".
  - Examples with class labels "+" are called positive examples
  - others negative examples

Notations used in the algorithm:

S	The set of examples
C	The set of class labels
F	The set of features
A	An arbitrary feature (attribute)
Values(A)	The set of values of the feature A
v	An arbitrary value of A
S_v	The set of examples with $A = v$
Root	The root node of a tree

S	The set of examples
C	The set of class labels
F	The set of features
A	An arbitrary feature (attribute)
Values(A)	The set of values of the feature A
v	An arbitrary value of A
S_v	The set of examples with $A = v$
Root	The root node of a tree

Algorithm ID3(S, F, C)

1. Create a root node for the tree.
2. **if** (all examples in S are positive) **then**
3. **return** single node tree Root with label "+"
4. **end if**
5. **if** (all examples are negative) **then**
6. **return** single node tree Root with label "–"
7. **end if**
8. **if** (number of features is 0) **then**
9. **return** single node tree Root with label equal to the most common class label.
10. **else**
11. Let A be the feature in F with the highest information gain.
12. Assign A to the Root node in decision tree.
13. **for all** (values v of A) **do**
14. Add a new tree branch below Root corresponding to v.
15. **if** (S_v is empty) **then**
16. Below this branch add a leaf node with label equal to the most common class label in the set S.
17. **else**
18. Below this branch add the subtree formed by applying the same algorithm ID3 with the values ID3($S_v, C, F - \{A\}$).
19. **end if**
20. **end for**
21. **end if**

[5] *dichotomy*: A division into two parts or classifications especially when they are sharply distinguished or opposed

CART ALGORITHM
(CLASSIFICATION AND REGRESSION TREES METHODOLOGY)

- Umbrella term for Classification and Regression
- Classification trees
- Regression trees
- The main elements of CART are:
  - Rules for splitting data at a node based on the value of one variable
  - Stopping rules for deciding when a branch is terminal and can be split no more
  - A prediction for the target variable in each terminal node

Classification

- Gini index is used as feature selection method in CART

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

Regression

- Variance reduction
  - Often employed in cases where the target variable is continuous (regression tree)
  - Use of many other metrics would first require discretization before being applied
  - Variance reduction of a node *N*:

    the total reduction of the variance of the target variable *x* due to the split at this node

Stopping Rules

- If a node becomes pure
- If all cases in a node have identical values for each predictor
- If the current tree depth reaches the user-specified maximum tree depth limit value
- If the size of a node is less than the user-specified minimum node size value
- If the split of a node results in a child node whose node size is less than the userspecified minimum child node size value

REGRESSION TREES

- Relation between one or more independent variables and an output variable
- Output variable - a real continuous variable
- Predict the values of the dependent variables using the relation
- in general prediction of numerical values of variables
- Trees can also be used to make such predictions
- Prediction tree or a Regression tree

A tree used for making predictions of numerical variables

8.11.2 An algorithm for constructing regression trees

Starting with a learning sample, three elements are necessary to determine a regression tree:

1. A way to select a split at every intermediate node
2. A rule for determining when a node is terminal
3. A rule for assigning a value for the output variable to every terminal node

Notations

$x_1, x_2, \ldots, x_n$	:	The input variables
N	:	Number of samples in the data set
$y_1, y_2, \ldots, y_N$	:	The values of the output variables
T	:	A tree
c	:	A leaf of T
n_c	:	Number of data elements in the leaf c
C	:	The set of indices of data elements which are in the leaf c
m_c	:	The mean of the values of y which are in the leaf c
S_T	:	Sum of squares of errors in T

We have

$$m_c = \frac{1}{n_c} \sum_{i \in C} y_i$$

$$S_T = \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2$$

- Algorithm

Step 1:

- Start with a single node containing all data points.

  Calculate m_c and S_T .

$$m_c = \frac{1}{n_c} \sum_{i \in C} y_i$$

$$S_T = \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2$$

Step 2:

- If all the points in the node have the same value for all the independent variables, stop.

Step 3:

- Otherwise, search over all binary splits of all variables for the one which will reduce S_T as much as possible.

  (a) If the largest decrease in S_T would be less than some **threshold** , or one of the resulting nodes would contain less than q points, stop and

  if c is a node where we have stopped, then assign the value m_c to the node

  (b) Otherwise, take that split, creating two new nodes

Step 4:

- In each new node, go back to Step 1

Remarks

1. Entropy & information defined for discrete variables as well as continuous variables
   - In the case of regression trees, it is more common to use the sum of squares
   - The above algorithm is based on sum of squares of errors

2. The CART algorithm searches
   - every distinct value of every predictor variable to find the predictor variable
   - split value which will reduce S_T as much as possible

3. In the above algorithm, simplest criteria used for stopping growing of trees
   - More sophisticated criteria which produce much less error have been developed

EXAMPLE

x_1	1	3	4	6	10	15	2	7	16	0
x_2	12	23	21	10	27	23	35	12	27	17
y	10.1	15.3	11.5	13.9	17.8	23.1	12.7	43.0	17.6	14.9

Step 1:

- Find m_c and S_T

$$
\begin{aligned}
m_c &= \frac{1}{n_c} \sum_{c \in C} y_i \\
&= \frac{1}{10}(10.1 + 15.3 + \cdots + 14.9) \\
&= 17.99 \\
S_T &= \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2 \\
&= (10.1 - 17.99)^2 + (15.3 - 17.99)^2 + \cdots + (14.9 - 17.99)^2 \\
&= 817.669
\end{aligned}
$$

Step 2:

- Find m_{c1} ,m_{c2} and S_{T1} for all x1

For x_1 =6

- One set defined by x1 < 6

x_1	1	3	4	2	0
x_2	12	23	21	35	17
y	10.1	15.3	11.5	12.7	14.9

Table 8.12: Data for regression tree

- Other set defined by x1 >=6

x_1	6	10	15	7	16
x_2	10	27	23	12	27
y	13.9	17.8	23.1	43.0	17.6

Table 8.13: Data for regression tree

$$\text{leaves}(T) = \{c_1, c_2\}.$$

Let T_1 be the tree corresponding to this partition. Then

$$\begin{aligned} S_{T_1} &= \sum_{c \in \text{leaves}(T_1)} \sum_{i \in C} (y_i - m_c)^2 \\ &= \sum_{i \in C_1} (y_i - m_{c_1})^2 + \sum_{i \in C_2} (y_i - m_{c_2})^2 \end{aligned}$$

$$\begin{aligned} m_{c_1} &= \frac{1}{n_{c_1}} \sum_{i \in C_1} y_i \\ &= \frac{1}{5}(10.1 + 15.3 + 11.5 + 12.7 + 14.9) \\ &= 12.9 \end{aligned}$$

$$\begin{aligned} m_{c_2} &= \frac{1}{n_{c_2}} \sum_{i \in C_2} y_i \\ &= \frac{1}{5}(13.9 + 17.8 + 23.1 + 43.0 + 17.6) \\ &= 23.08 \end{aligned}$$

$$\begin{aligned} S_{T_1} &= [(10.1 - 12.9)^2 + \cdots + (14.9 - 12.9)^2] + \\ &\quad [(13.9 - 23.08)^2 + \cdots + (17.6 - 23.08)^2] \\ &= 558.588 \end{aligned}$$

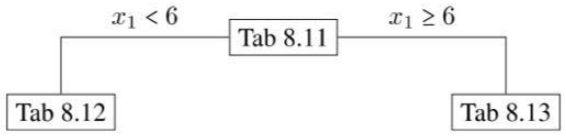


Figure 8.11: Part of a regression tree for Table 8.11

The reduction in sum of squares of errors is

$$S_T - S_{T_1} = 817.669 - 558.588 = 259.081.$$

Step 3: Find the values of m_c and S_T

- Consider the node specified by Table 8.12
  - Split the values of x_2 into two sets:
    - one specified by $x_2 < 21$
    - one specified by $x_2 >= 21$
- Similarly, the node specified by Table 8.13
  - Split the values of x2 into sets:
    - one specified by $x_2 < 23$
    - one specified by $x_2 >= 23$

x_1	1	0
x_2	12	17
y	10.1	14.9

(a)

x_1	3	4	2
x_2	23	21	35
y	15.3	11.5	12.7

(b)

x_1	6	7
x_2	10	12
y	13.9	43.0

(c)

x_1	10	15	16
x_2	27	23	27
y	17.8	23.1	17.6

(d)

Table 8.14: Data for regression tree

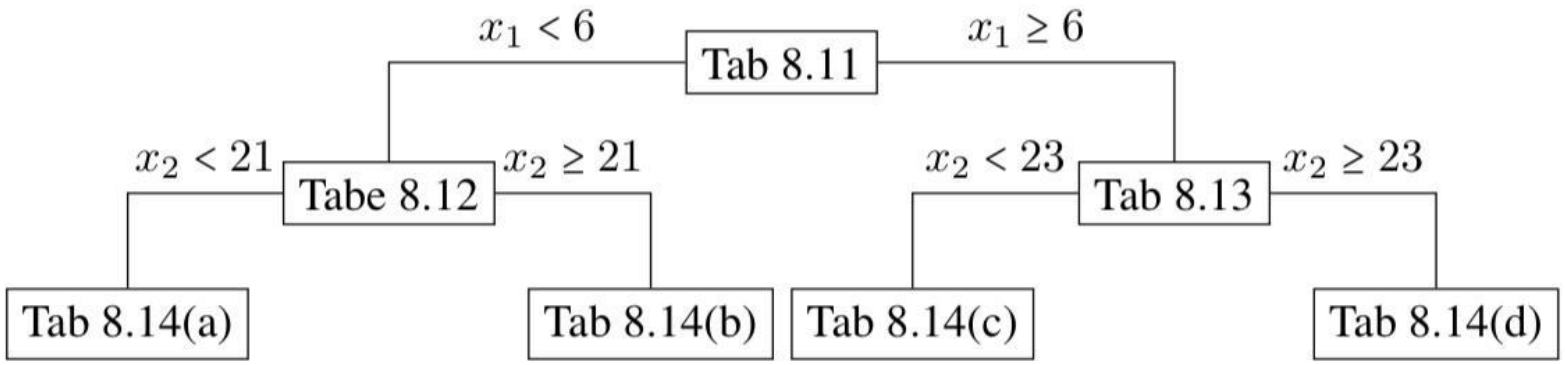


Figure 8.12: Part of regression tree for Table 8.11

Step 3:

- The nodes specified by Table 8.14(a), …, Tab 8.14(d) into leaf nodes
- For each leaf node find the average of the values in the corresponding table

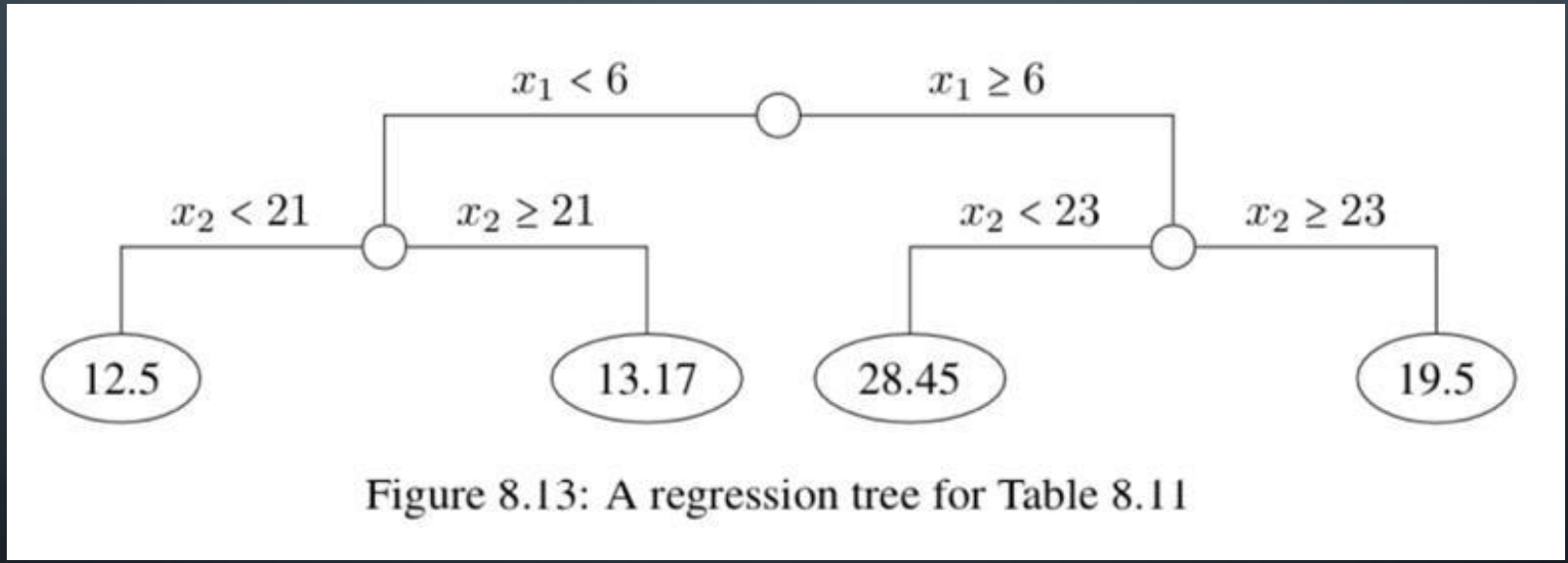


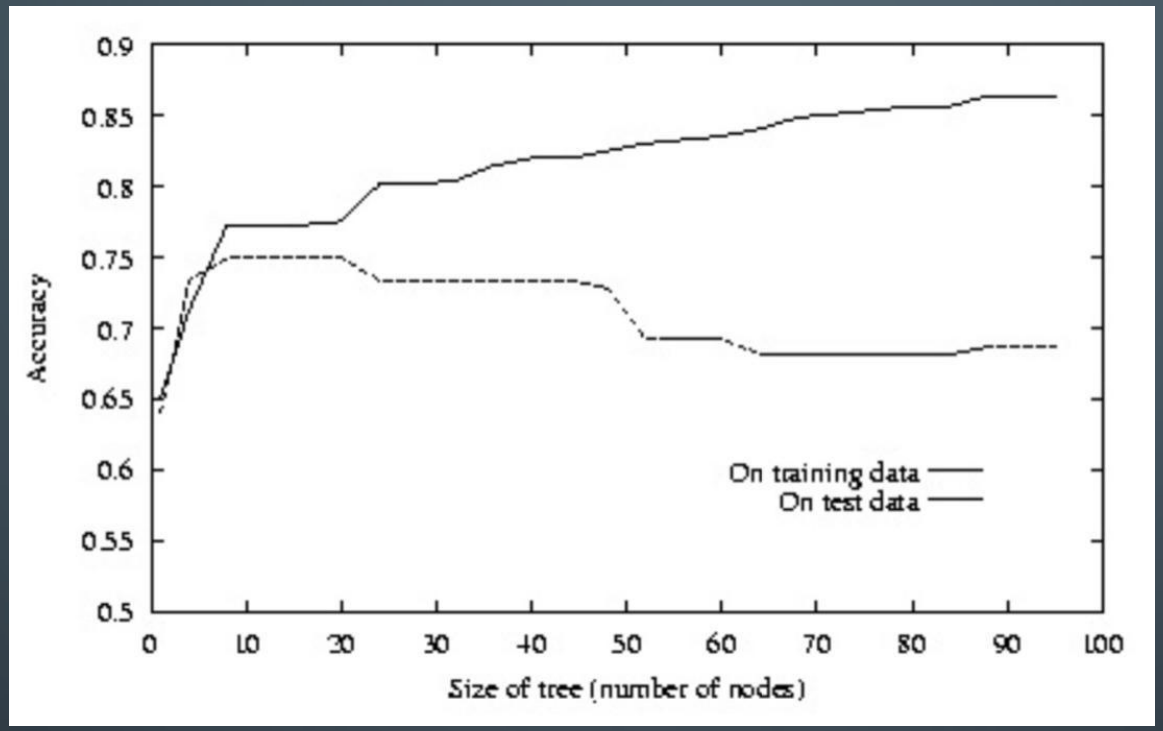
Figure 8.13: A regression tree for Table 8.11

ISSUES IN DECISION TREE LEARNING

AVOIDING OVERFITTING OF DATA

- Branches are grown deep enough to perfectly classify the training examples
- May lead to difficulties when
  - there is noise in the data or
  - the number of training examples are too small
- The algorithm can produce trees that overfit the training examples
- Hypothesis overfits the training examples if
  - some other hypothesis that fits the training examples less well
  - But actually performs better over the entire distribution of instances
  - Better for instances beyond the training set as well

IMPACT OF OVERFITTING



- Accuracy of the tree over training examples increases monotonically
- accuracy over independent test samples first increases then decreases

APPROACHES TO AVOID OVERFITTING

PRUNING:

- reduces the size of decision trees
- removes sections of the tree that provide little power to classify instances
- reduces the complexity of the final classifier
- improves predictive accuracy by the reduction of overfitting
- Pruning can be applied:
  - Earlier to overfitting
  - After overfitting occurs

Reduced error pruning:

- each of the decision trees to be a candidate for pruning
- Pruning a decision node involves:
  - removing the subtree rooted at that node
  - making it a leaf node
  - assigning it the most common classification

- Nodes are removed only if the resulting pruned tree performs no worse than the original
- Nodes are pruned iteratively:
  - choosing the node whose removal most increases the accuracy
- Pruning of nodes is continued until further pruning decreases the accuracy

PROBLEM OF MISSING ATTRIBUTES

Case	Temperature	Headache	Nausea	Decision (Flue)
1	high	?	no	yes
2	very high	yes	no	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Methods used to handle the problem of missing attributes:

- Deleting cases with missing attribute values
- Replacing a missing attribute value by the most common value of that attribute
- Assigning all possible values to the missing attribute value
- Replacing a missing attribute value by the mean for numerical attributes
- Assigning the corresponding value taken from the closest t cases
- Replacing a missing attribute value by a new value

NEURAL NETWORKS

Artificial Neural Network (ANN)

- Models the relationship between a set of input signals and an output signal
- Model based on how a biological brain responds to stimuli from sensory input
- Uses a network of artificial neurons or nodes to solve learning problems

BIOLOGICAL MOTIVATION

- Biochemical process
  - Accumulating the incoming signals, a threshold is reached
  - Cell fires and the output signal is transmitted
- Dendrites - incoming signals are received
- Axon – output signals fired through
- Synapse

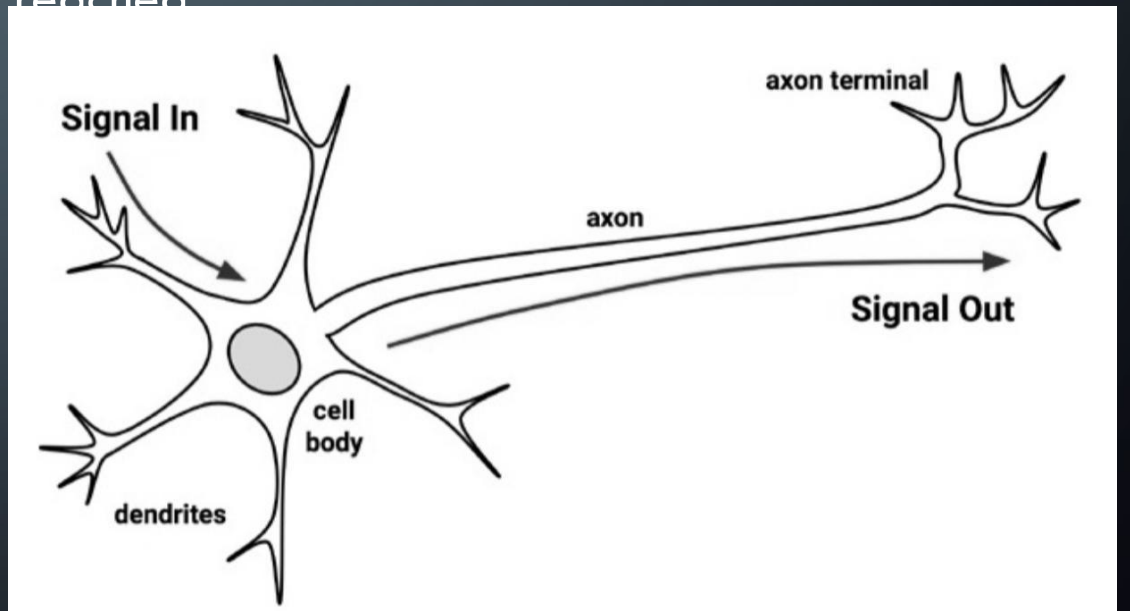


Figure 9.2: Flow of signals in a biological neuron

- Neuron switching speeds are much slower than computer switching speeds
- Humans take complex decisions relatively quickly
- Very fast information processing capabilities
- Huge number of **parallel processes** distributed over many neurons

ARTIFICIAL NEURON

- Mathematical function conceived as a model of biological neurons
- Elementary units in an artificial neural network
- Artificial Neuron
  - Receives one or more inputs
  - Each input is separately weighted
  - Sums them to produce an output
  - Sum is passed through a function (activation function or transfer function)

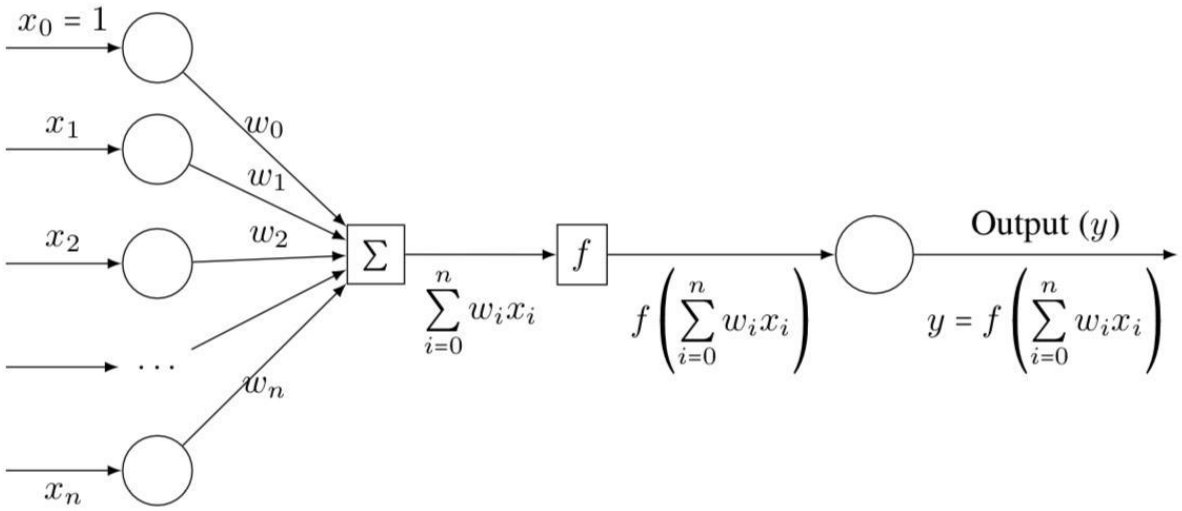


Figure 9.3: Schematic representation of an artificial neuron

$x_1, x_2, \ldots x_n$: input signals

$w_1, w_2, \ldots w_n$: weights associated with input signals

x_0 : input signal taking the constant value 1

w_0 : weight associated with x_0 (called bias)

$\sum$: indicates summation of input signals

f : function which produces the output

y : output signal

The function f can be expressed in the following form:

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right)$$

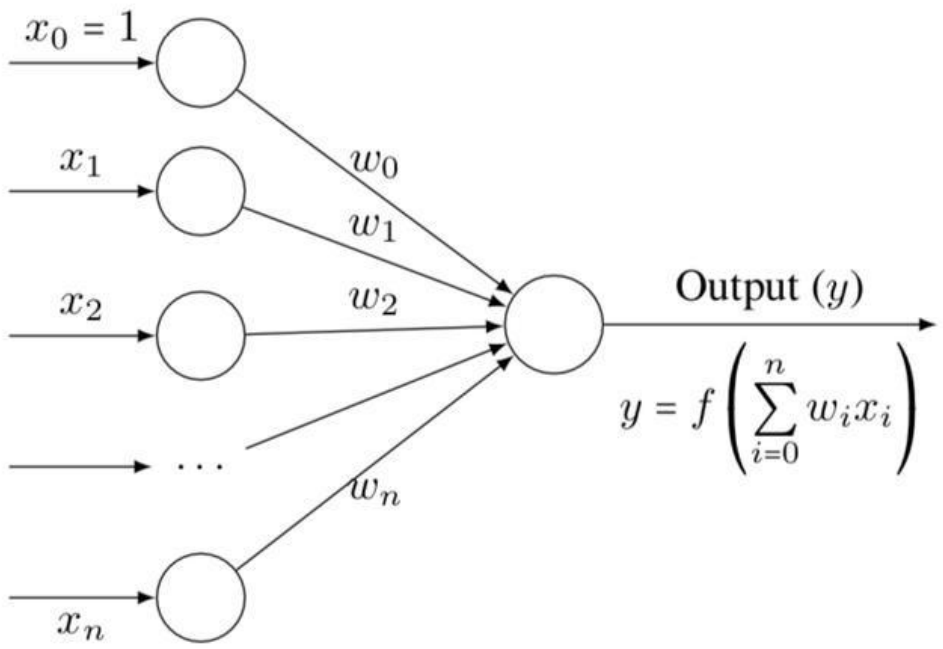


Figure 9.4: Simplified representation of an artificial neuron

ACTIVATION FUNCTION

- Function which takes the incoming signals as input & produces the output signal
- Threshold activation function-

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$$

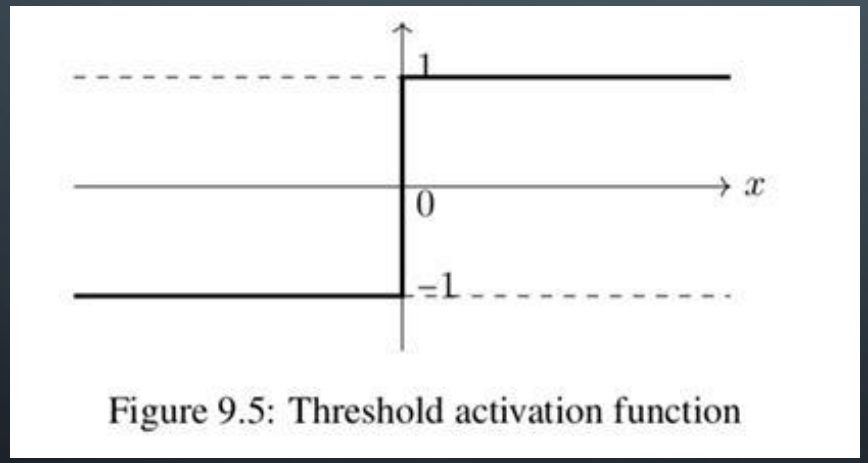


Figure 9.5: Threshold activation function

- Unit step functions -

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

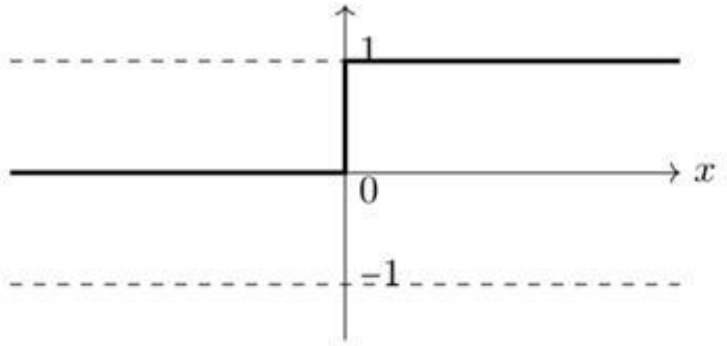


Figure 9.6: Unit step activation function

- Sigmoid activation function (logistic function) -

$$f(x) = \frac{1}{1 + e^{-x}}$$

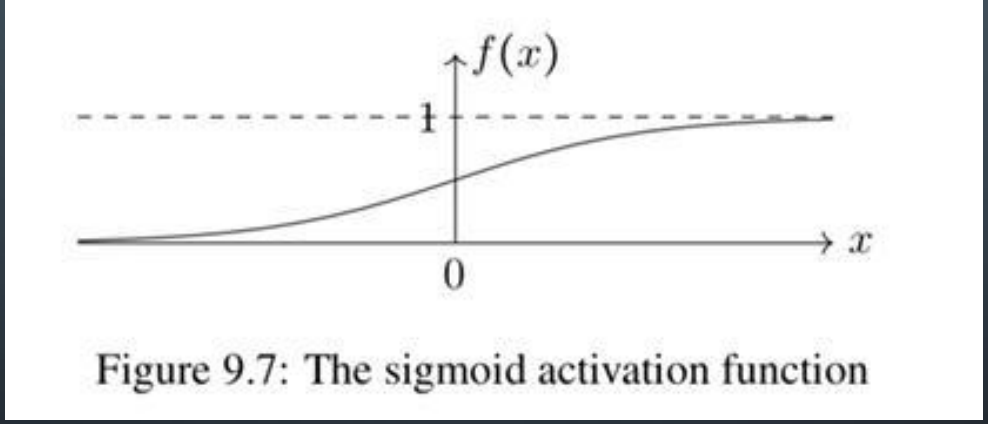


Figure 9.7: The sigmoid activation function

- Linear activation function $F(x) = mx + c$

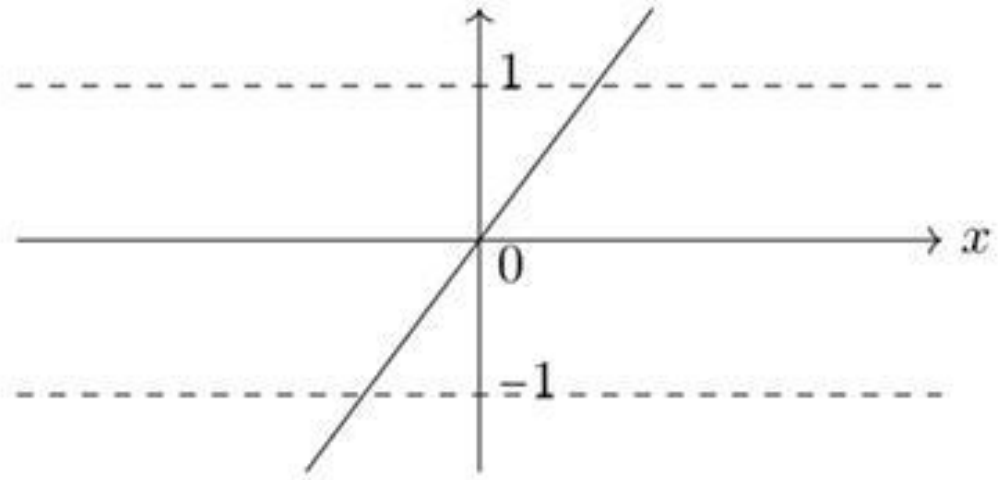


Figure 9.8: Linear activation function

- Piecewise (or, saturated) linear activation function

$$f(x) = \begin{cases} -1 & \text{if } x < x_{\min} \\ mx + c & \text{if } x_{\min} \le x \le x_{\max} \\ 1 & \text{if } x > x_{\max} \end{cases}$$

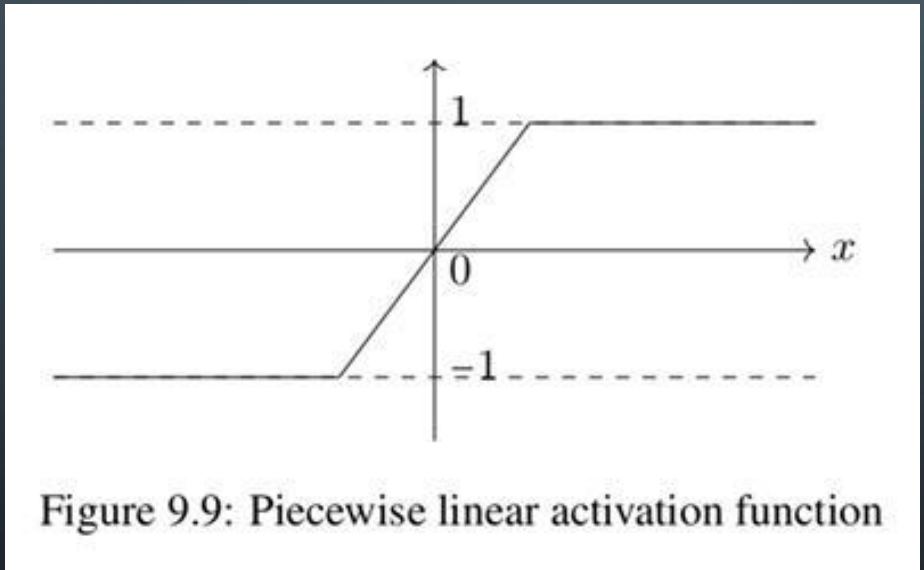


Figure 9.9: Piecewise linear activation function

- Gaussian activation function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

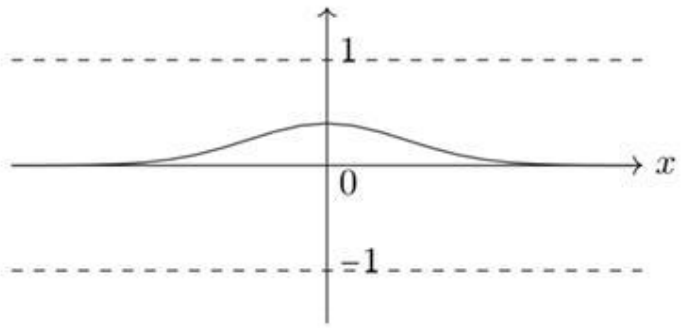


Figure 9.10: Gaussian activation function

- Hyperbolic tangential activation function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

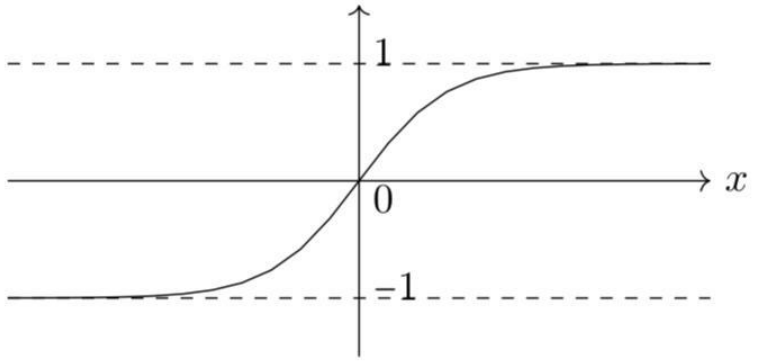


Figure 9.11: Hyperbolic tangent activation function

PERCEPTRON

- Special type of artificial neuron - activation function is the threshold function

Consider a neuron with

- x1, x2, …. , xn as the input signals
- w1, w2, …. , wn as the associated weights
- w0 be some constant

- Neuron is called a perceptron if the output of the neuron is given by:

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n \leq 0 \end{cases}$$

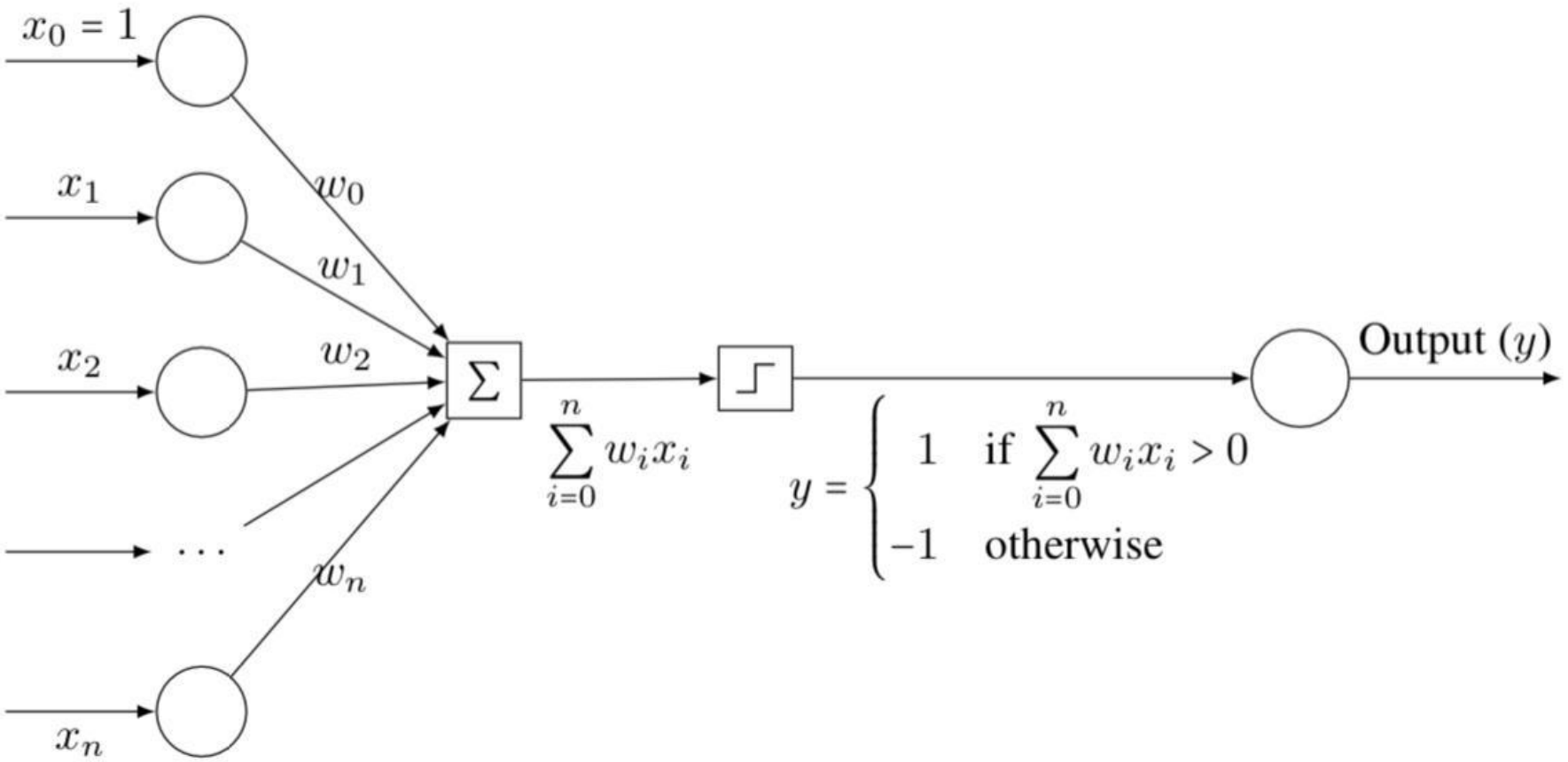


Figure 9.12: Schematic representation of a perceptrn

REPRESENTATIONS OF BOOLEAN FUNCTIONS BY PERCEPTRONS

x1 AND x2

x_1	x_2	x_1 AND x_2
−1	−1	−1
−1	1	−1
1	−1	−1
1	1	1

Table 9.1: The boolean function x_1 AND x_2

x_1 AND x_2.

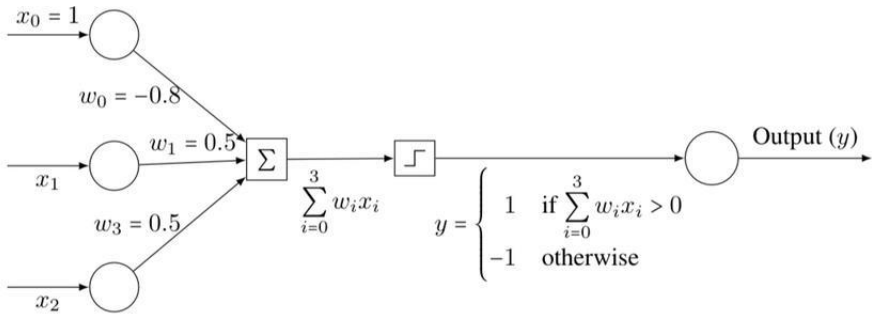


Figure 9.13: Representation of x_1 AND x_2 by a perceptron

In the perceptron shown in Figure 9.13, the output is given by

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{3} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1 & \text{if } -0.8 + 0.5x_1 + 0.5x_2 > 0 \\ -1 & \text{otherwise} \end{cases}$$

Boolean function	w_0	w_1	w_2
x_1 AND x_2	-0.8	0.5	0.5
x_1 OR x_2	-0.3	0.5	0.5
x_1 NAND x_2	0.8	-0.5	-0.5
x_1 NOR x_2	0.3	-0.5	-0.5

Table 9.2: Representations of boolean functions by perceptrons

- Learning a perceptron:
  - Assigning values to weights and threshold such that the perceptron produces correct o/p

- Two algorithms to solve this learning problem-
  - Perceptron learning algorithm
  - ANN

PERCEPTRON LEARNING ALGORITHM

Algorithm in pdf page 118-119

ARTIFICIAL NEURAL NETWORKS

- Collection of connected units called artificial neurons
- The artificial neuron- receives the signal, processes it and then signal AN connected to it
- Each connection between artificial neurons has a weight attached to it
- Weight gets adjusted as learning proceeds
- Artificial Neuron:
  - AN may have a threshold -only if the aggregate signal crosses that threshold the signal is sent
  - Artificial neurons are organized in layers
  - Different layers - different kinds of transformations on their inputs

- Signals travel from input layer to output layer, possibly after traversing the layers multiple times

CHARACTERISTICS OF AN ANN

- The activation function
  - Neuron's combined input signals are transformed into a single output signal to be broadcasted
- The network topology (or architecture)
  - Number of neurons in the model and the number of layers and manner in which they are connected
- The training algorithm
  - How connection weights are set to inhibit or excite neurons in proportion to the input signal

NETWORK TOPOLOGY

- Patterns and structures in the collection of interconnected nodes
- Determines the complexity of tasks that can be learned by the network
- Different forms of network architecture can be differentiated by:
  - The number of layers
  - Whether information in the network is allowed to travel backward
  - The number of nodes within each layer of the network

Number of Layers:

- Input nodes
- Output nodes
- Hidden nodes

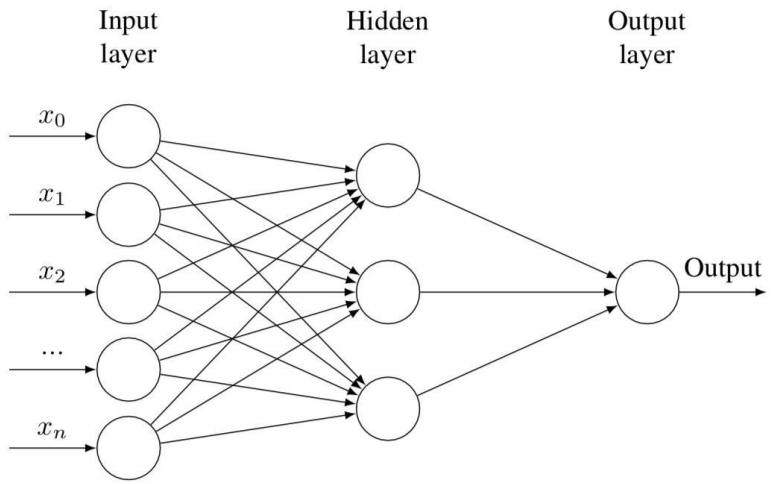
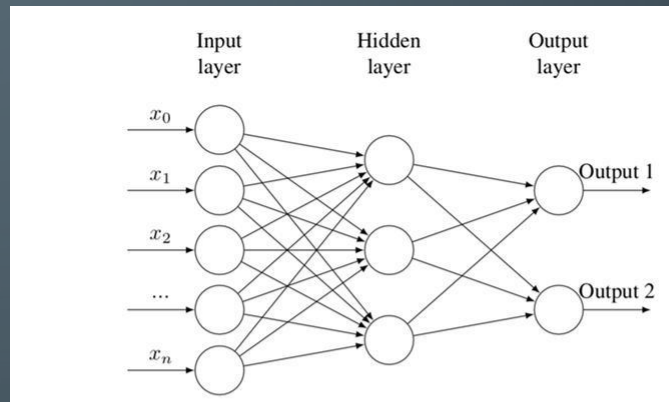
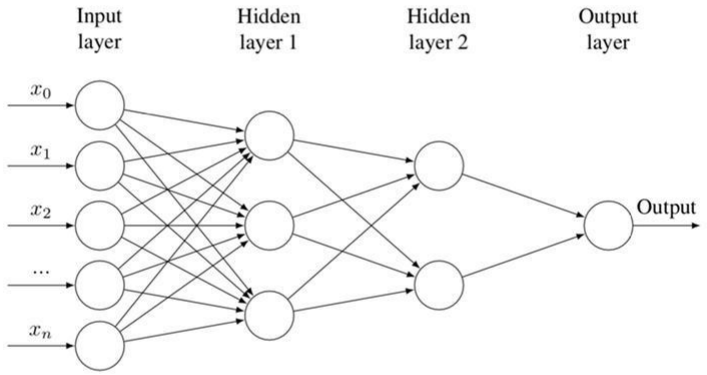


Figure 9.15: An ANN with two layers



(a) Network with one hidden layer and two output nodes



(b) Network with two hidden layers

Figure 9.16: Examples of different topologies of networks

Direction of Information Travel

- Feedforward networks
  - extensively applied to real-world problems
- Recurrent networks (or, feedback networks)
  - still largely theoretical and are rarely used in practice

- Multilayer feedforward n/w, sometimes called Multilayer Perceptron (MLP), is the de facto standard ANN topology

The number of nodes in each layer:

- Input nodes is predetermined by the number of features in the input data
- Number of output nodes is predetermined by
  - number of outcomes to be modeled or
  - number of class levels
- The number of hidden nodes is left to the user to decide prior to training the model
- Some factors it depends on
  - the number of input nodes
  - The amount of training data
  - the amount of noisy data
  - the complexity of the learning task

TRAINING ALGORITHM

- Two commonly used algorithms for learning a single perceptron:
  - Perceptron rule - used when the training data set is linearly separable
  - Delta rule - used when the training data set is not linearly separable
- Commonly used algo is ANN (or backpropagation algo)

COST FUNCTION

- Measures how well the algorithm maps the target function
- Determines how well the algorithm performs in an optimization problem
- Also called: loss function, objective function, scoring function, error function
- Examples:
  - Sum of squares of the differences between the predicted and actual values of y

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

  - Mean of the sum of squares of the differences between predicted and actual values of y

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

BACKPROPAGATION

- Initially the weights are assigned at random
- Algorithm iterates through many cycles of two processes until a stopping criterion is reached
- Each cycle is known as an epoch. Each epoch includes
  - A forward phase
  - A backward phase
- Determine how much a weight should be changed using gradient descent method

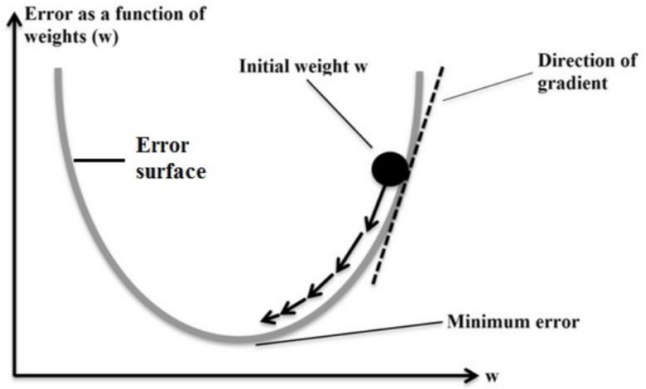


Figure 9.17: A simplified model of the error surface showing the direction of gradient

BACKPROPAGATION EXAMPLE

Step 1: Initialise the connection weights to small random values

Sample	Input 1 i_1	Input 2 i_2	Output target 1 T_1	Output target 2 T_2
1	0.05	0.10	0.01	0.99
2	0.25	0.18	0.23	0.79

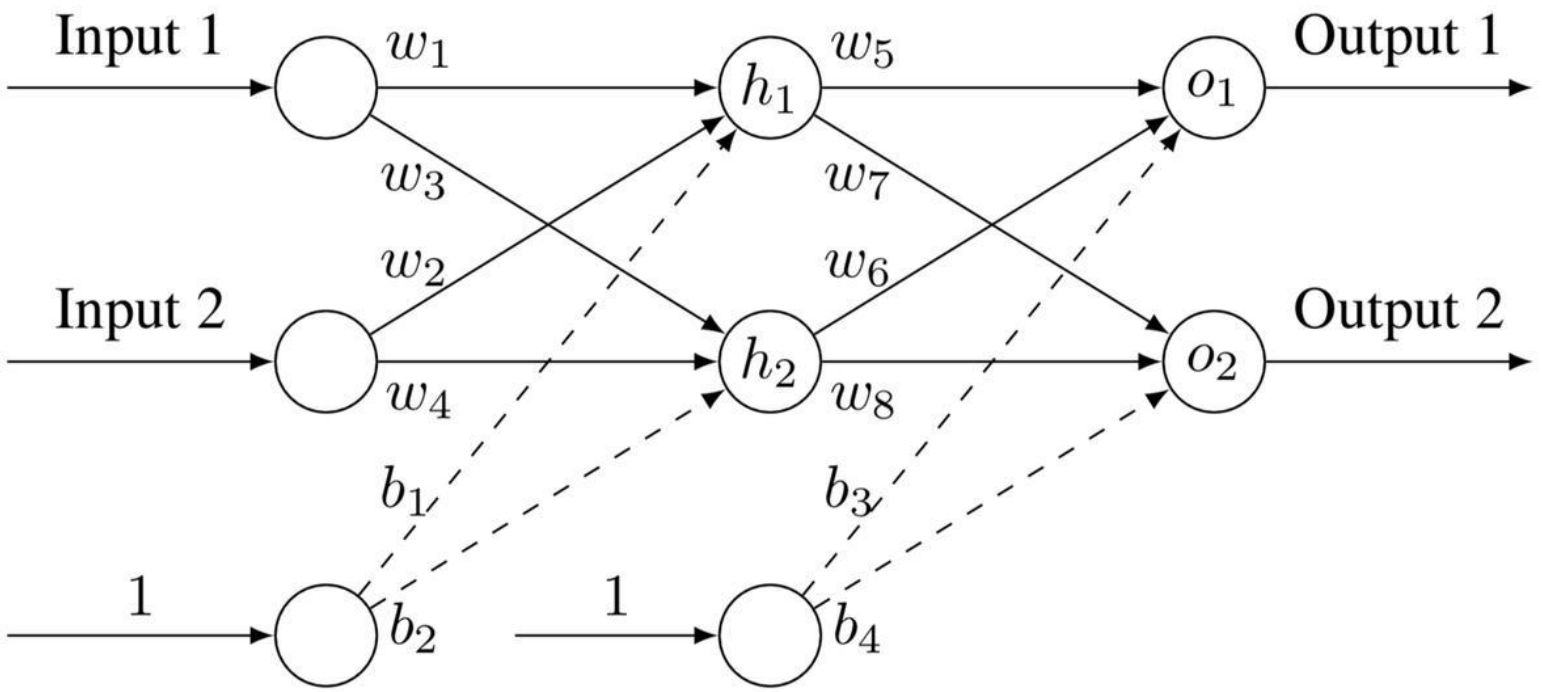


Figure 9.18: ANN for illustrating backpropagation algorithm

Step 2: Present the first sample inputs and the corresponding output targets to the network

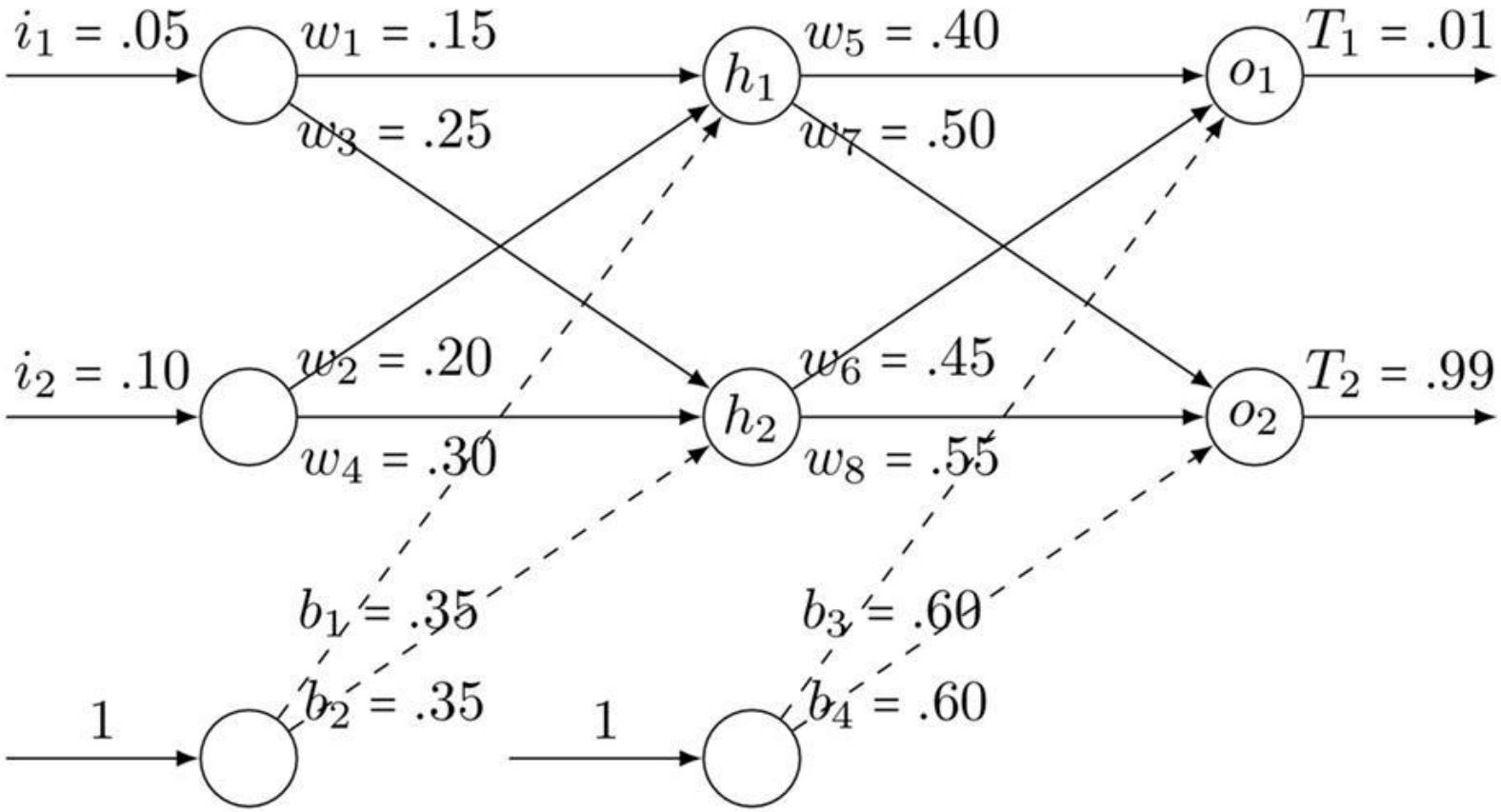


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

Step 3: Pass the input values to the first layer

Step 4: calculate the outputs from h1 and h2

Logistic activation function=> $f(x) = \frac{1}{1+e^{-x}}$

$$
\begin{aligned}
\text{out}_{h_1} &= f(w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1) \\
&= f(0.15 \times 0.05 + 0.20 \times 0.10 + 0.35 \times 1) \\
&= f(0.3775) \\
&= \frac{1}{1+e^{-0.3775}} \\
&= 0.59327 \\
\text{out}_{h_2} &= f(w_3 \times i_1 + w_4 \times i_2 + b_2 \times 1) \\
&= f(0.25 \times 0.05 + 0.30 \times 0.10 + 0.35 \times 1) \\
&= f(0.3925) \\
&= \frac{1}{1+e^{-0.3925}} \\
&= 0.59689
\end{aligned}
$$

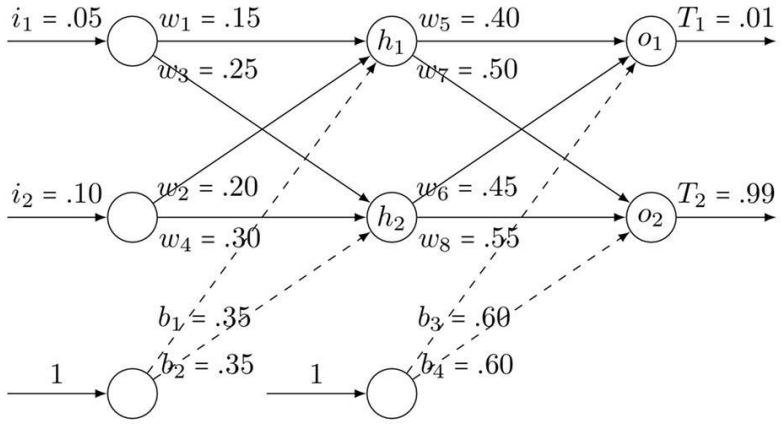


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

Step 5: Repeat this process for every layer

$$
\begin{aligned}
\text{out}_{o_1} &= f(w_5 \times \text{out}_{h_1} + w_6 \times \text{out}_{h_2} + b_3 \times 1) \\
&= f(0.40 \times 0.59327 + 0.45 \times 0.59689 + 0.60 \times 1) \\
&= f(1.10591) \\
&= \frac{1}{1 + e^{-1.10591}} \\
&= 0.75137 \\
\text{out}_{o_2} &= f(w_7 \times \text{out}_{h_1} + w_8 \times \text{out}_{h_2} + b_4 \times 1) \\
&= f(0.50 \times 0.59327 + 0.55 \times 0.59689 + 0.60 \times 1) \\
&= f(1.22492) \\
&= \frac{1}{1 + e^{-1.22492}} \\
&= 0.77293
\end{aligned}
$$

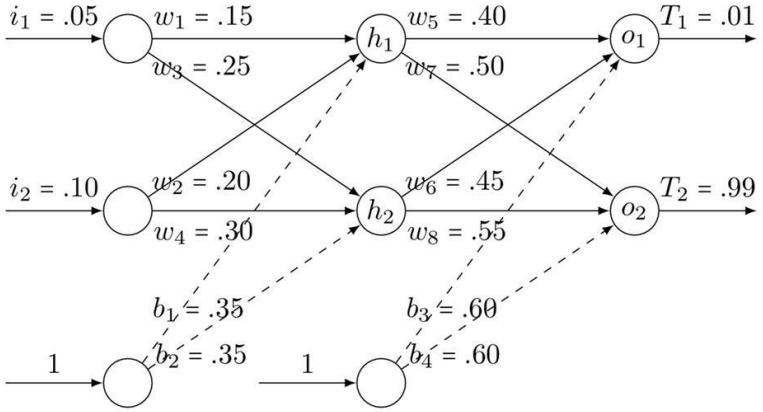


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

ERROR CALCULATION

Sample	Input 1 i_1	Input 2 i_2	Output target 1 T_1	Output target 2 T_2
1	0.05	0.10	0.01	0.99
2	0.25	0.18	0.23	0.79

$$
\begin{aligned}
E &= \frac{1}{2}(T_1 - \text{out}_{o_1})^2 + \frac{1}{2}(T_2 - \text{out}_{o_2})^2 \\
&= (0.01 - 0.75137)^2 + (0.99 - 0.77293)^2 \\
&= 0.298371
\end{aligned}
$$

Step 6: Adjust weights backwards:

- Learning rate $\eta = 0.5$

$$
\begin{aligned}
\delta_{o_1} &= (T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1}) \\
&= (0.01 - 0.75137) \times 0.75137 \times (1 - 0.75137) \\
&= -0.13850 \\
w_5^+ &= w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1} \\
&= 0.40 + 0.5 \times (-0.13850) \times 0.59327 \\
&= 0.35892 \\
w_6^+ &= w_6 + \eta \times \delta_{o_1} \times \text{out}_{h_2} \\
&= 0.45 + 0.5 \times (-0.13850) \times 0.59689 \\
&= 0.40867 \\
b_3^+ &= b_3 + \eta \times \delta_{o_1} \times 1 \\
&= 0.60 + 0.5 \times (-0.13850) \times 1 \\
&= 0.53075
\end{aligned}
$$

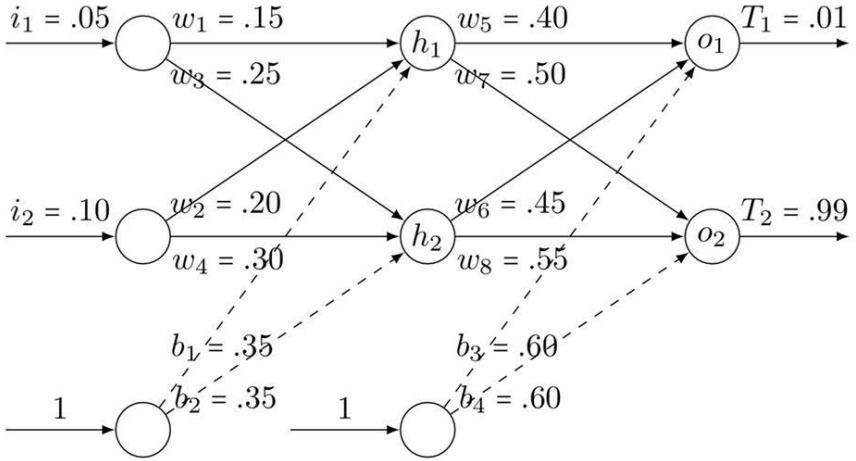


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

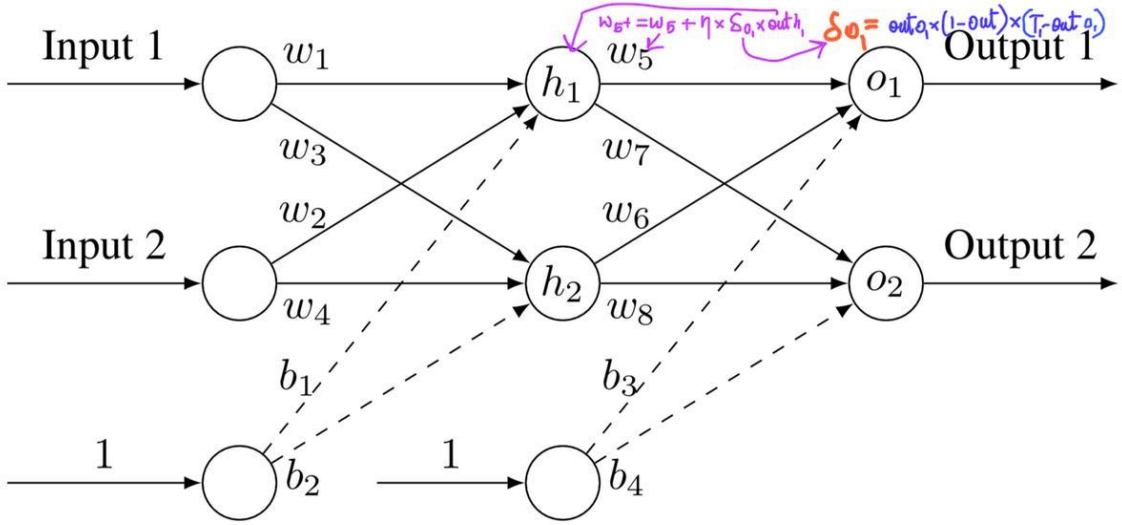


Figure 9.18: ANN for illustrating backpropagation algorithm

$$\delta_{o_2} = (T_2 - \text{out}_{o_2}) \times \text{out}_{o_2} \times (1 - \text{out}_{o_2})$$
$$= (0.99 - 0.77293) \times 0.77293 \times (1 - 0.77293)$$
$$= 0.03810$$
$$w_7^+ = w_7 + \eta \times \delta_{o_2} \times \text{out}_{h_1}$$
$$= 0.50 + 0.5 \times 0.03810 \times 0.59327$$

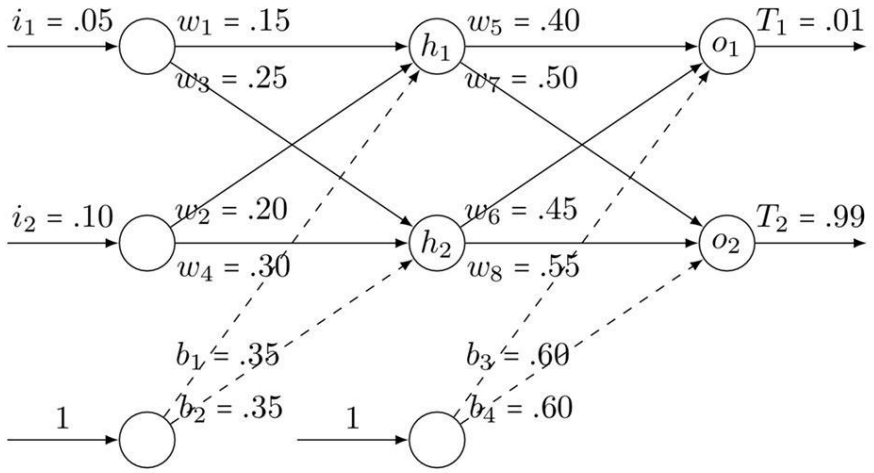


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

$$= 0.51130$$
$$w_8^+ = w_8 + \eta \times \delta_{o_2} \times \text{out}_{h_2}$$
$$= 0.55 + 0.5 \times 0.03810 \times 0.59689$$
$$= 0.56137$$
$$b_4^+ = b_4 + \eta \times \delta_{o_2} \times 1$$
$$= 0.60 + 0.5 \times 0.03810 \times 1$$
$$= 0.61905$$

$$
\begin{aligned}
\delta_{h_1} &= (\delta_{o_1} \times w_5 + \delta_{o_2} \times w_7) \times \text{out}_{h_1} \times (1 - \text{out}_{h_1}) \\
&= (-0.13850 \times 0.40 + 0.03810 \times 0.50) \times 0.59327 \times (1 - 0.59327) \\
&= -0.00877 \\
w_1^+ &= w_1 + \eta \times \delta_{h_1} \times i_1 \\
&= 0.15 + 0.5 \times (-0.00877) \times 0.05 \\
&= 0.14978 \\
w_2^+ &= w_2 + \eta \times \delta_{h_1} \times i_2 \\
&= 0.20 + 0.5 \times (-0.00877) \times 0.10 \\
&= 0.19956 \\
b_1^+ &= b_1 + \eta \times \delta_{h_1} \times 1 \\
&= 0.35 + 0.5 \times (-0.00877) \times 1 \\
&= 0.34562
\end{aligned}
$$

$$
\begin{aligned}
\delta_{h_2} &= (\delta_{o_1} \times w_6 + \delta_{o_2} \times w_8) \times \text{out}_{h_2} \times (1 - \text{out}_{h_2}) \\
&= ((-0.13850) \times 0.45 + 0.03810 \times 0.55) \times 0.59689 \times (1 - 0.59689) \\
&= -0.00995 \\
w_3^+ &= w_3 + \eta \times \delta_{h_2} \times i_1 \\
&= 0.25 + 0.5 \times (-0.00995) \times 0.05 \\
&= 0.24975 \\
w_4^+ &= w_4 + \eta \times \delta_{h_2} \times i_2 \\
&= 0.30 + 0.5 \times (-0.00995) \times 0.10 \\
&= 0.29950 \\
b_2^+ &= b_2 + \eta \times \delta_{h_2} \times 1 \\
&= 0.35 + 0.5 \times (-0.00995) \times 1 \\
&= 0.34503
\end{aligned}
$$

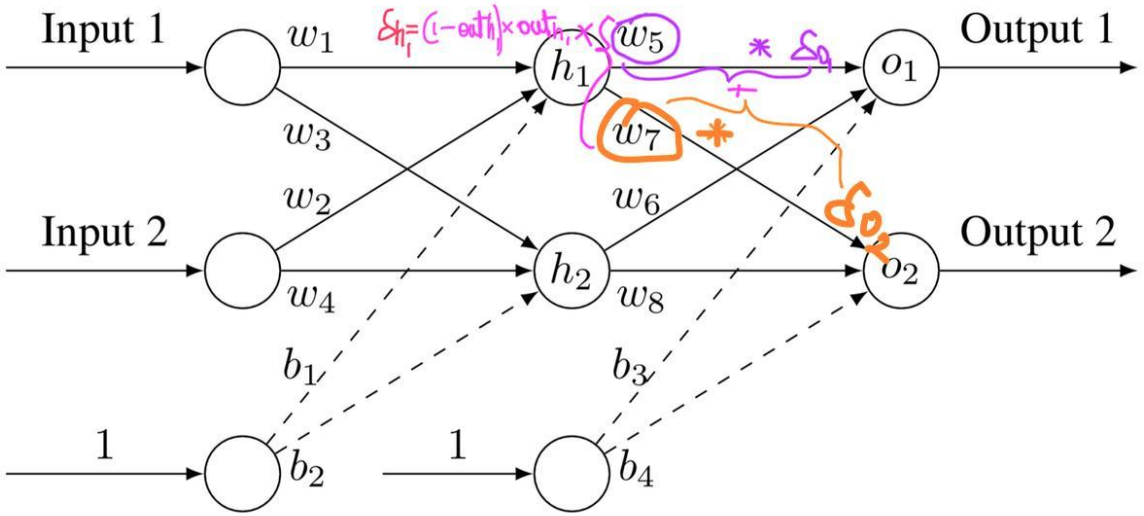


Figure 9.18: ANN for illustrating backpropagation algorithm

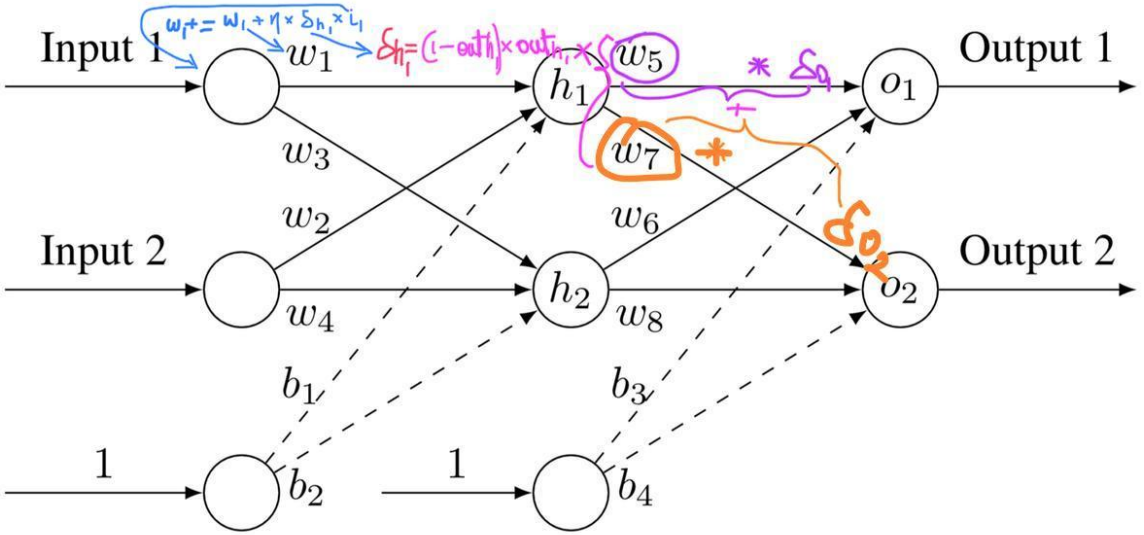


Figure 9.18: ANN for illustrating backpropagation algorithm

Step 7 : Adjust the weights

$$w_1 = w_1^+, \quad w_2 = w_2^+, \quad w_3 = w_3^+, \quad w_4 = w_4^+$$
$$w_5 = w_5^+, \quad w_6 = w_6^+, \quad w_7 = w_7^+, \quad w_8 = w_8^+$$
$$b_1 = b_1^+, \quad b_2 = b_2^+, \quad b_3 = b_3^+, \quad b_4 = b_4^+$$

BACKPROPAGATION ALGORITHM

- Notes pdf pages 142 -144

Remarks

1. The constant $\frac{1}{2}$ is included in the expression for E so that the exponent is cancelled when we differentiate it. The result has been multiplied by a learning rate $\eta = 0.5$ and so it doesnâĂŹt matter that we introduce the constant $\frac{1}{2}$ in E.

2. In the above computations, the method used to calculate the adjusted weights is known as the *delta rule*.

3. The rule for computing the adjusted weights can be succinctly stated as follows. Let w be a weight and w^+ its adjusted weight. Let E be the the total sum of squares of errors. Then w^+ is computed by

$$w^+ = w - \eta \frac{\partial E}{\partial w}.$$

Here $\frac{\partial E}{\partial w}$ is the gradient of E with respect to w; that is, the rate at which E is changing with respect to w. (The set of all such gradients specifies the direction in which E is decreasing the most rapidly, that is, the direction of quickest descent.) For example, it can be shown that

$$\begin{aligned} \frac{\partial E}{\partial w_5} &= -(T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1}) \times \text{out}_{h_1} \\ &= -\delta_{o_1} \times \text{out}_{h_1} \end{aligned}$$

and so

$$\begin{aligned} w_5^+ &= w_5 - \eta \frac{\partial E}{\partial w_5} \\ &= w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1} \end{aligned}$$