Chapter 13
Perceptron Learning Rule

13.1 Single Layer Perceptron

A single layer perceptron is a simplest form of neural network. This type of neural network is used for pattern classifications that are linearly separable. Single layer perceptron consists of one input layer with one or many input units and one output layer with one or many output units. The neurons are connected to each other by weights and bias (Gkanogiannis and Kalamboukis (2009, 2010)).

13.2 Architecture of Single Layer Perceptron

The structure or architecture of single layer perceptron is shown in Fig. 13.1. In Fig. 13.1, $X_1, X_2, \ldots, X_n$ are the input neurons of the input layer. There exists a common bias b with a value 1. $w_1, w_2, \ldots, w_n$ are the weights connected from input node to the output node. Y is the output neuron in the output layer (Zurada 1994).

13.3 Algorithm of Single Layer Perceptron

The step-by-step algorithm is given below (Sivanandam 2006):

Step1: Initialize all weights and bias to zero, i.e., $w_i = 0$ for $i = 1$–n, $b = 0$. Here, n is the number of input neurons. Let us take $\alpha = (0$–$1)$ as the learning rate.
Step2: For each input training vector and target output pair, $\mathbf{S} : t$, do steps 2–5.
Step3: Set activation for input units : $x_i = \mathbf{S}_i$, $i = 1, \ldots, n$.

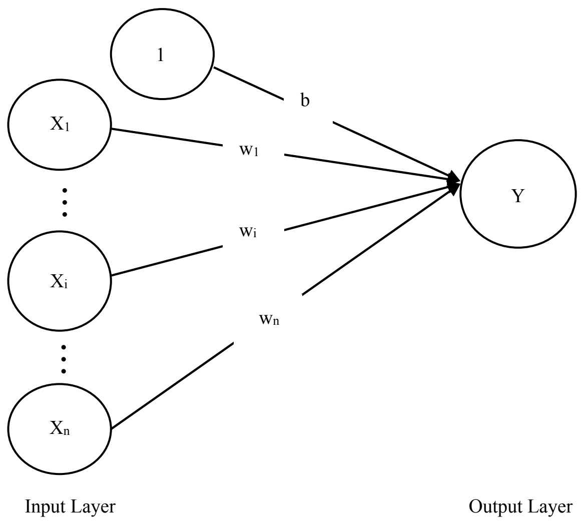
Fig. 13.1 Single layer perceptron

Step 4: Compute the output unit as $y_{in} = b + \sum_{i=1}^{n} x_i w_i$.
The activation function is used as

$$Y = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > \theta \\ 0, & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < \theta \end{cases}.$$

Step5: The weights and bias are updated if the target is not equal to the output response.
If $t \neq y$ and the value of $x_i \neq 0$
then $w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$ for $i = 1, \ldots, n$

$$b(\text{new}) = b(\text{old}) + \alpha t$$

else $w_i(\text{new}) = w_i(\text{old})$

$$b(\text{new}) = b(\text{old})$$

The stopping conditions may be the change in weights.

13.4 Matlab Programs Using Perceptron Learning Rule

13.4.1 Write a Matlab Program for and Function with Bipolar Inputs and Targets Using Perceptron Learning Algorithm

Program:

```
% Perceptron 1

clear all
clc
x1=[1 -1 1 -1];
x2=[1 1 -1 -1];
t=[1 -1 -1 -1];
v=[1 1 1 1];
th=input('enter the value of theta: ');
a=input('enter the value of alpha: ');
w1=0;
w2=0;
b=0;
dw1=a.*x1.*t;
dw2=a.*x2.*t;
db=a.*t;
for i=1:4
   yin=w1*x1(i)+w2*x2(i)+b;
   if yin>th
      y=1;
   elseif -th<=yin&&yin<=th
      y=0;
   else
      y=-1;
   end
   if y~=t(i)
      w1=w1+dw1(i);
      w2=w2+dw2(i);
      b=b+db(i);
   end
   w1_n(i)=w1;
   w2_n(i)=w2;
   b_n(i)=b;
end
```

```
disp('        Input     Target   Weight Change        Weights');
disp('   x1    x2    b     Y    dw1   dw2    db    w1    w2   B');
H=[x1' x2' v' t' dw1' dw2' db' w1_n' w2_n' b_n'];
disp(H)
```

Output:
enter the value of theta: 0
enter the value of alpha: 1

```
  Input       Target   Weight Change        Weights

 x1   x2    b      Y       dw1   dw2    db      w1    w2    B
  1    1    1      1        1     1      1       1     1    1
 –1    1    1     –1        1    –1     –1       2     0    0
  1   –1    1     –1       –1     1     –1       1     1   –1
 –1   –1    1     –1        1     1     –1       1     1   –1
```

13.4.2 Write a Matlab Program for AND Function with Binary Inputs and Bipolar Targets Without Bias up to 2 Epochs Using Perceptron Learning Algorithm

(a) With (0, 0) and without bias

Program:

```
% Perceptron 2

clear all
clc
x1=[1 1 0 0];
x2=[1 0 1 0];
t=[1 -1 -1 -1];
th=input('enter the value of theta: ');
a=input('enter the value of alpha: ');
dw1=a.*x1.*t;
dw2=a.*x2.*t;
for l=4:-1:3
   w1=0;
   w2=0;
   for j=1:2
```

```
        for i=1:l
            yin=x1(i)*w1+x2(i)*w2;
            if yin>th
                y=1;
            elseif -th<=yin&&yin<=th
                y=0;
            else
                y=-1;
            end
           if y~=t(i)
               w1=w1+dw1(i);
               w2=w2+dw2(i);
           end
           w1_n(i)=w1;
           w2_n(i)=w2;
       end
   end
   disp('  Weights');
   disp('  w1    w2');
   H=[w1_n' w2_n'];
   disp(H)
   clear w1_n;
   clear w2_n;
end
```

Output:
enter the value of theta: 0
enter the value of alpha: 1

```
 Input        Target   Weights

x1   x2          Y        w1    w2
 1    1          1        1      1

  0    1       –1          0     1

  1    0       –1          0     0

  0    0       –1          0     0
```

(b) Without bias and (0, 0)

Input		Target	Weights	
x1	**x2**	**Y**	**w1**	**w2**
1	**1**	**1**	**1**	**1**
0	**1**	**–1**	**0**	**1**
1	**0**	**–1**	**0**	**0**

References

J. Macek Zurada, *Introduction to Artificial Neural Systems* (Jaico Publishing House, 1994)

S.N. Sivanandam, S. Sumathi, S.N. Deepa, *Introduction to Neural Networks using Matlab 6.0* (McGraw Hill Education (India) Private Ltd., 2006)

A. Gkanogiannis, T. Kalamboukis, *A modified and fast Perceptron learning rule and its use for Tag Recommendations in Social Bookmarking Systems*. ECML PKDD Discovery Challenge 2009 (DC09). **497**(71), 1–13 (2009)

A. Gkanogiannis, T. Kalamboukis, A Perceptron-Like Linear Supervised Algorithm for Text Classification. In: L. Cao, Y. Feng, J. Zhong (eds) *Advanced Data Mining and Applications*. ADMA 2010. Lect. Notes Comp. Sci. 6440. Springer, Berlin, Heidelberg (2010)