Dr. Qadri Hamarsheh

Neural Networks and Fuzzy Logic (630514)

Lecture 4

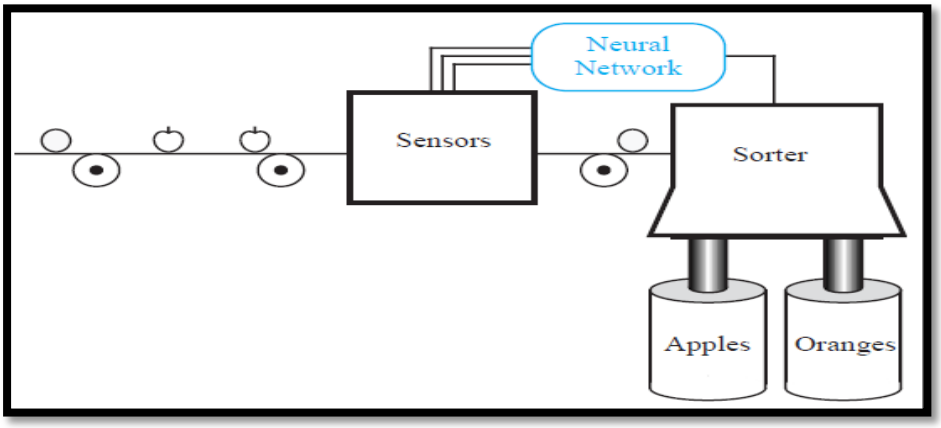
Outline: Solving simple pattern recognition problem using ANN

Problem Statement:

- Sorting the fruit according to type using a machine.
- Using sensors: shape, texture and weight.
  - The shape sensor will output a **1** if the fruit is approximately round and a **–1** if it is more elliptical.
  - The texture sensor will output a **1** if the surface of the fruit is smooth and a **–1** if it is rough.
  - The weight sensor will output a **1** if the fruit is more than one pound and a **–1** if it is less than one pound.
- The outputs of the sensors will then be input to a neural network.

The purpose of the network: decide which kind of fruit is on the conveyor, so that the fruit can be directed to the correct storage bin.

- Two kinds of fruit: apples and oranges.



- Fruit can be represented by a three dimensional vector:
  - First element represents shape.
  - Second element represents texture.
  - Third element represents weight.

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix}$$

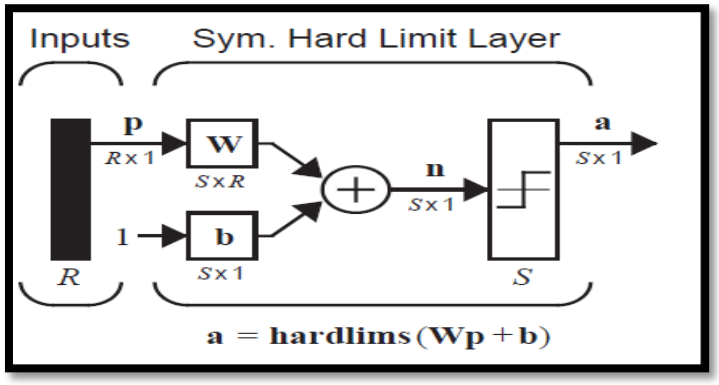
Orange would be represented by $\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$,

Apple would be represented by $\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$

- The neural network will receive one three-dimensional input vector for each fruit on the conveyer and must make a decision as to whether the fruit is an orange $\mathbf{p}_1$ or an apple $\mathbf{p}_2$.

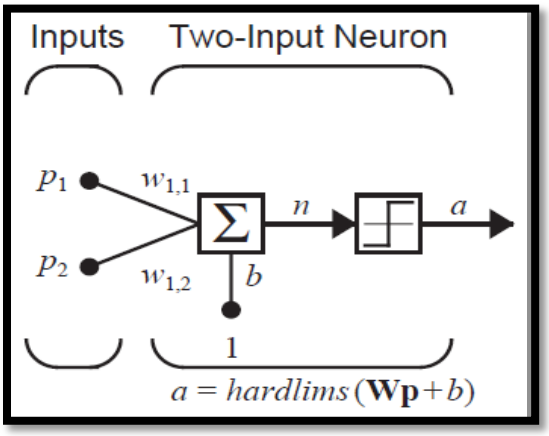
Pattern recognition problem using **Perceptron**

Single-layer perceptron with a symmetric hard limit transfer function *hardlims*.



Remark: Two-Input /Single-Neuron Perceptron Case

- Investigate the capabilities of a two-input/single-neuron perceptron (R=2), which can be easily analyzed graphically:

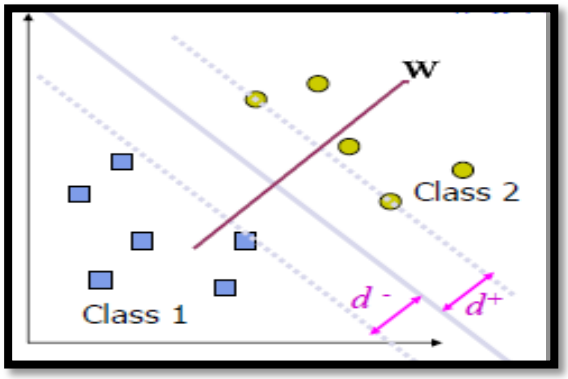


- What is a **decision boundary**?
  A decision boundary is the region of a problem space in which the output label of a classifier is ambiguous.

Parameterizing decision boundary

- Let **w** denote a vector orthogonal to the decision boundary, and **b** denote a scalar "**offset**" term, then we can write the decision boundary as

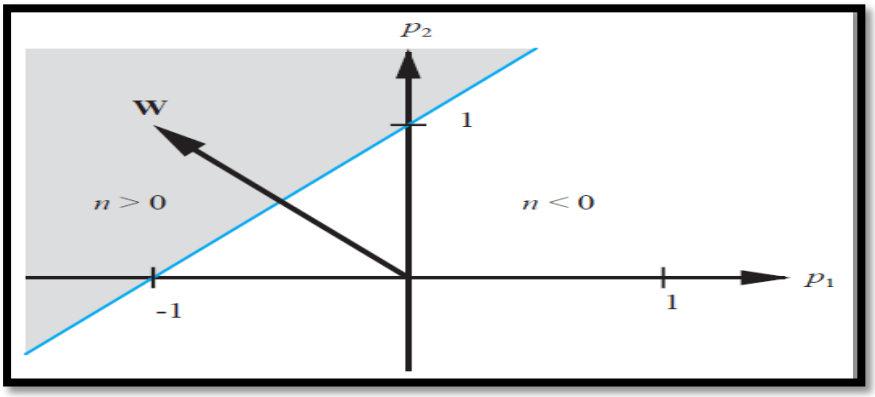
$$w^T x + b = 0$$



- Single-neuron perceptrons can classify input vectors into two categories: This divides the input space (the range of all possible input vectors) into two parts, Notice that this decision boundary will always be orthogonal to the weight matrix, and the position of the boundary can be shifted by changing **b**
- For example, if $w_{1,1} = -1$, **and** $w_{1,2} = 1$ then

$$a = hardlims(n) = hardlims([-1\ 1]\ p) + b)$$

Therefore, if the inner product of the weight matrix (a single row vector in this case) with the input vector is greater than or equal to **–b** the output will be **1**. If the inner product of the weight vector and the input is less than **–b**, the output will be **-1**, the following figure illustrates this for the case where **b = –1**



Perceptron Decision Boundary

- The shaded region contains all input vectors for which the output of the network will be **1**. The output will be **-1** for all other input vectors.
- The decision boundary between the categories is determined by the equation

$$\mathbf{W}\mathbf{p} + b = 0$$

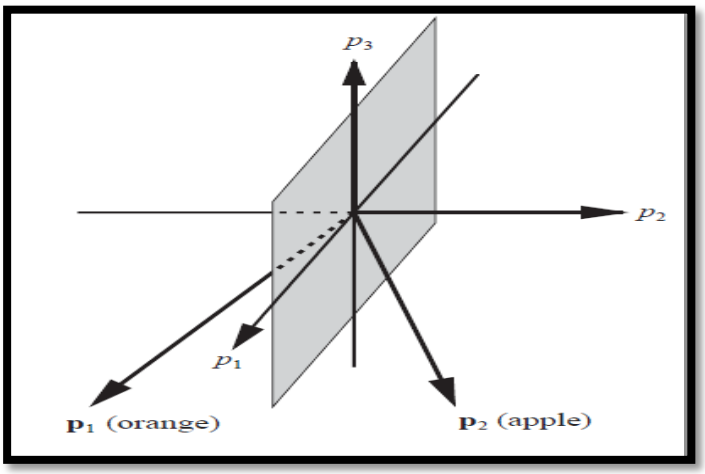
- The single-layer perceptron can only be used to recognize patterns that are linearly separable (can be separated by a linear boundary).

Pattern Recognition Example

- We can use a single-neuron perceptron with three inputs, because there are only two categories (**apple or orange**).

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right)$$

- We may want the output of the perceptron to be **1 when an apple** is input and **-1 when an orange** is input.
- The two prototype vectors are shown in the following figure, from this figure we can see that the linear boundary that divides these two vectors symmetrically is the p_1 and p_3 plane.



- The p_1 and p_3 plane, which will be our decision boundary, can be described by the equation

$$p_2 = 0$$

Or

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0.$$

Therefore the weight matrix and bias will be

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \; b = 0.$$

- The weight matrix is **orthogonal** to the decision boundary and points toward the region that contains the prototype pattern $\mathbf{P_2}$ (apple), for which we want the perceptron to produce an output of **1**.
- The bias is **0** because the decision boundary passes through the origin.

Test the operation of our perceptron pattern classifier

Orange:

$$a = hardlims\left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1 (orange)$$

Apple:

$$a = hardlims\left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1 (apple)$$

- But what happens if we put a not-so-perfect orange into the classifier (noisy data)? Let's say that an orange with an elliptical shape is passed through the sensors. The input vector would then be

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

The response of the network would be

$$a = hardlims\left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1(orange)$$

- In fact, any input vector that is closer to the orange prototype vector than to the apple prototype vector (in Euclidean distance) will be classified as an orange (and vice versa).
- The key characteristic of the single-layer perceptron is that it creates linear decision boundaries to separate categories of input vector.
- What if we have categories that cannot be separated by linear boundaries? **The multilayer networks are able to solve classification problems of arbitrary complexity**.