Machine Learning

Solving simple pattern recognition problem using ANN

Computer & Health Informatics Department

Outline

- Solving simple pattern recognition problem using ANN

Problem Statement

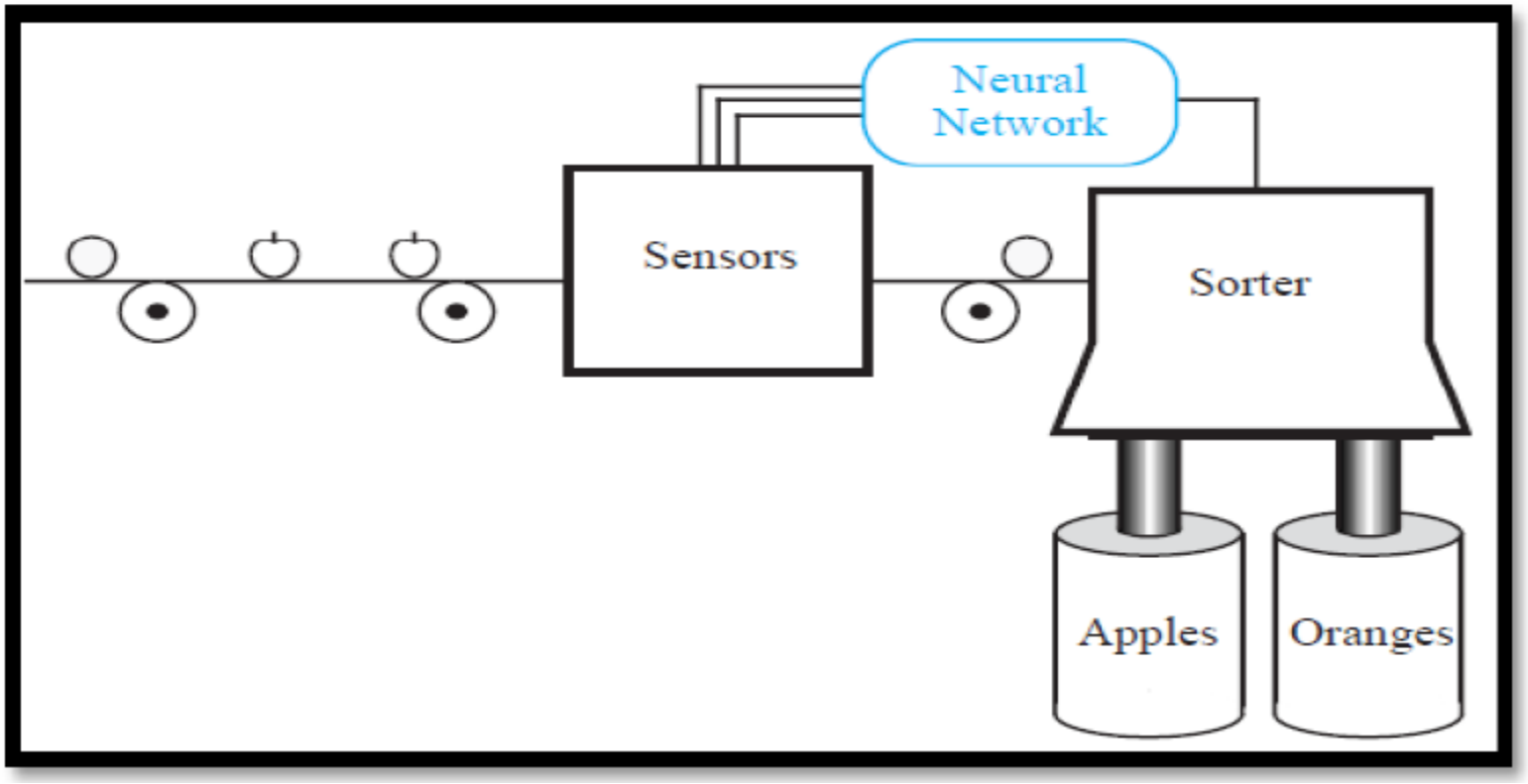
- Sorting the fruit according to type using a machine.
- Using sensors: shape, texture and weight.
  - The shape sensor will output a **1** if the fruit is approximately round and a **–1** if it is more elliptical.
  - The texture sensor will output a **1** if the surface of the fruit is smooth and a **–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.

The purpose of the network



The purpose of the network

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

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

The purpose of the network

- Orange would be represented by

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix},$$

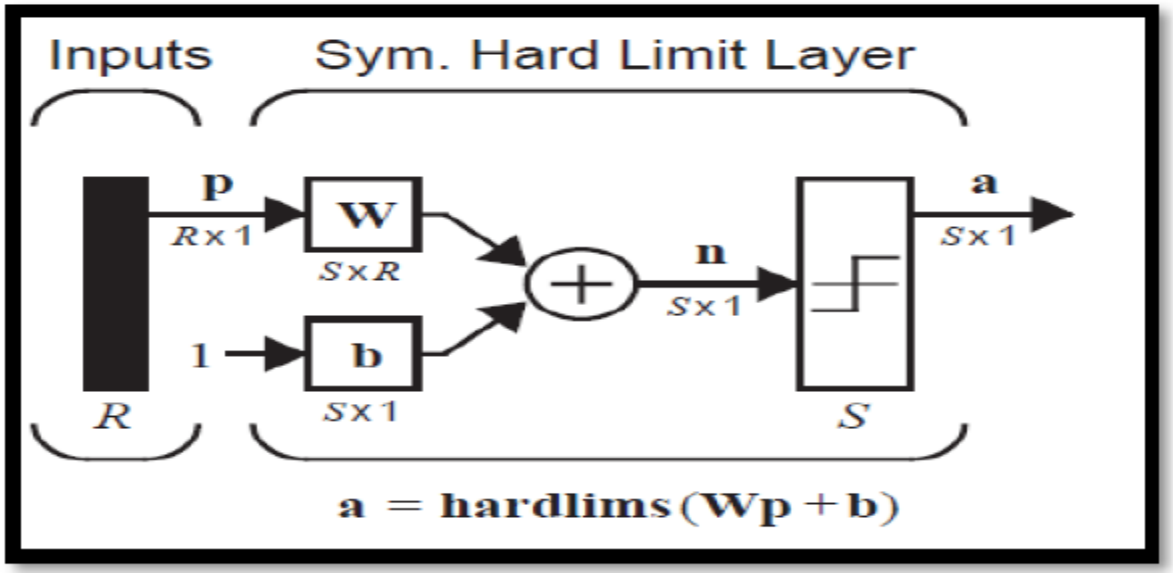
- Apple would be represented by

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

- 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 p_1 or an apple p_2.

Pattern recognition problem using Perceptron

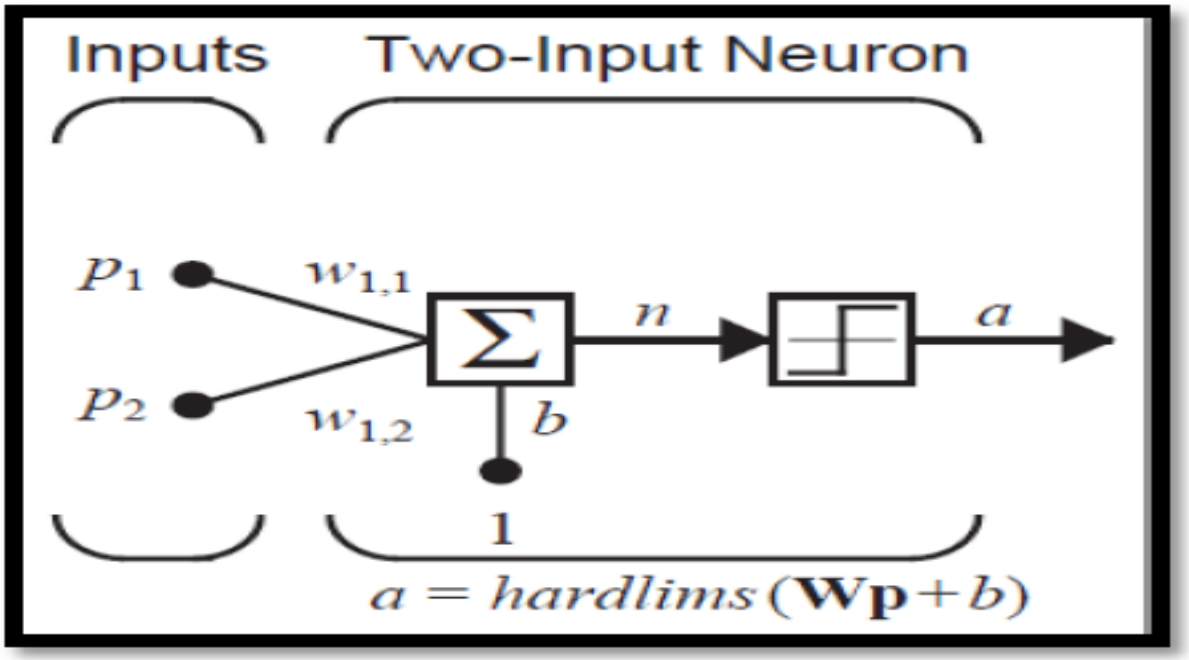
- Single-layer perceptron with a symmetric hard limit transfer function *hardlims*.



- **Remark**: Two-Input / Single-Neuron Perceptron Case

Pattern recognition problem using Perceptron

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



Pattern recognition problem using Perceptron

- 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.

OR

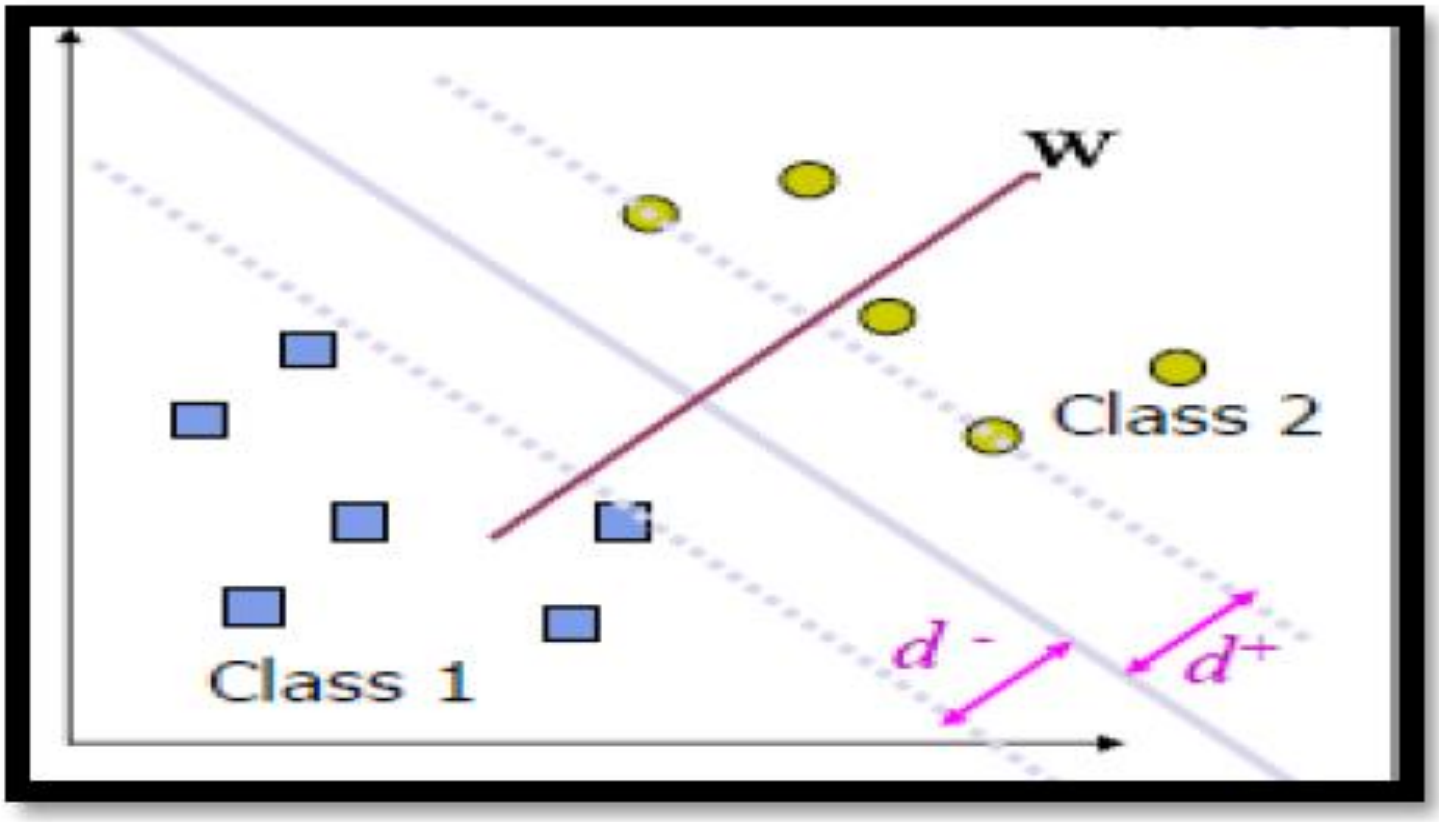
The decision boundary is determined by the input vectors for which the net Input n is zero:

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

Parameterizing decision boundary

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



Parameterizing decision boundary

- 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**

Parameterizing decision boundary

- **Decision Boundary**

$$n = \mathbf{w}^T \mathbf{p} + b = w_{1,1} p_1 + w_{1,2} p_2 + b = 0$$

- To make the example more concrete, let's assign the following values for the weights and bias:

$$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$$

- The decision boundary is then

Parameterizing decision boundary

$$n = \mathbf{w}^T \mathbf{p} + b = w_{1,1} p_1 + w_{1,2} p_2 + b = 0$$

- This defines a line in the input space. On one side of the line the network output will be 0; on the line and on the other side of the line the output will be 1. To draw the line, we can find the points where it intersects the p1 and p2 axes. To find the p2 intercept set p1 =0

Parameterizing decision boundary

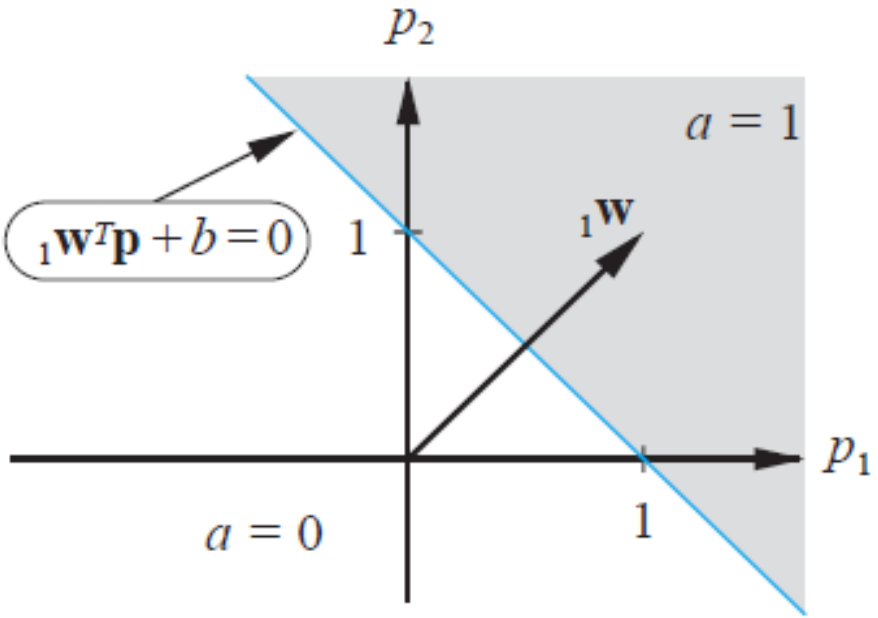
$$P_2 = -\frac{b}{W_{1,2}} = -\frac{-1}{1} = 1 \; if \; p_1 = 0.$$

To find the p_1 intercept, set p_2=0

$$P_1 = -\frac{b}{W_{1,1}} = -\frac{-1}{1} = 1 \; if \; p_2 = 0.$$

Parameterizing decision boundary

- The resulting decision boundary is illustrated in following Figure



Pattern Recognition Example

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

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

Pattern Recognition Example

- We may want the output of the perceptron to be **1 when an apple** is input and **-1 when an orange** is input.

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:

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

Test the operation of our perceptron pattern classifier

- 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

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

Test the operation of our perceptron pattern classifier

- The response of the network would be

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

Test the operation of our perceptron pattern classifier

- In fact, any input vector that is closer to the orange prototype vector than to the apple prototype vector 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?

Test the operation of our perceptron pattern classifier

- The multilayer networks are able to solve classification of complex problems.

The End

?