# OBJECT OF THE ASSIGNMENT:

To understand how the perceptron learning rule learns the weight and bias values for multiple-neuron perceptrons.

# PROBLEM:

Solve a four-category classification problem having two/three components in the input.

- Implement the perceptron learning rule for a two-neuron perceptron.
- Implement the perceptron learning rule for a four-neuron perceptron.

# INPUT OF THE PROBLEM:

Training dataset/testing dataset

# OUTPUT OF THE PROBLEM:

- Display weight/bias values of proper decision boundaries and the number of epochs when the perceptron learning rule converges; otherwise display the maximum number of epochs if proper decision boundaries are not found.
- Predict the classes (target vectors) of testing examples.

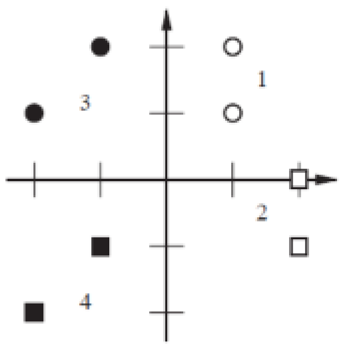
# DATASETS:

Dataset 1

This is a four-class classification problem described in Problem P4.3 in the textbook.

- Training Data

| Training Pattern $i$ | | $\mathbf{p}_1$ | $\mathbf{p}_2$ | $\mathbf{p}_3$ | $\mathbf{p}_4$ | $\mathbf{p}_5$ | $\mathbf{p}_6$ | $\mathbf{p}_7$ | $\mathbf{p}_8$ |
|---|---|---|---|---|---|---|---|---|---|
| Neural Inputs | $p_1$ | 1 | 1 | 2 | 2 | –1 | –2 | –1 | –2 |
| | $p_2$ | 1 | 2 | –1 | 0 | 2 | 1 | –1 | –2 |
| Target Values | $t_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | $t_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |




  - -

- Testing Data

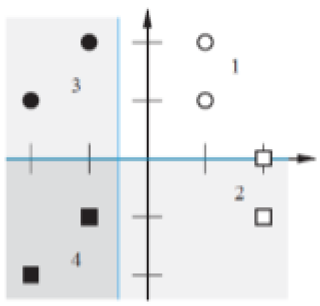
- $$\mathbf{P}_9 = \begin{bmatrix} 5 \\ 2 \end{bmatrix}, \mathbf{P}_{10} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \mathbf{P}_{11} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{P}_{12} = \begin{bmatrix} -3 \\ -4 \end{bmatrix}.$$

Dataset 2

Suppose that we want to distinguish between watermelons, bananas, pineapples, and oranges, based on three sensor measurements (shape, texture, and weight). Your TA will provide you a dataset containing 1000 training examples and another dataset containing 40 testing examples.

## EXPERIMENTS:

Two-neuron perceptron :

- Dataset 1
  - Begin your algorithm with the following initial weights and biases to train your perceptron network:
  - $\mathrm{W} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathrm{b} = [1 \quad 1]$
  - Expected output
  - The weight/bias values of the final decision boundaries are
  - $\mathrm{W} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}, \mathrm{b} = [-1 \quad 0]$
  - 

- Dataset 2 – Use the first two components. That is, shape and texture.
- Dataset 2 – Use the three components. That is, shape, texture, and weight.

Four-neuron perceptron

- Run your four-neuron perceptron on Dataset 1. Suppose that we change the target vectors to have four components. For example, the following table shows a possible mapping between two-neuron and four-neuron target vectors.

| Target vectors of two-neuron | $[0\ 0]^T$ | $[0\ 1]^T$ | $[1\ 0]^T$ | $[1\ 1]^T$ |
|---|---|---|---|---|
| Target vectors | $[1\ 0\ 0\ 0]^T$ | $[0\ 1\ 0\ 0]^T$ | $[0\ 0\ 1\ 0]^T$ | $[0\ 0\ 0\ 1]^T$ |

| | | | | |
|---|---|---|---|---|
| of four-neuron | | | | |

- Run your four-neuron perceptron on Dataset 2 and use the first two components.
- Run your four-neuron perceptron on Dataset 2 and use all of the three components.

## REMARK: One variation of the perceptron learning rule is

$W^{new} = W^{old} + \alpha \mathrm{ep}^{\mathrm{T}}$ and $\mathrm{b}^{\{\mathrm{new}\}} = \mathrm{b}^{\{\mathrm{old}\}} + \alpha e$ where $\mathrm{e} = (\mathrm{t} - \mathrm{a})$ and $\alpha$ is called the learning rate (small positive value, e.g., $0.1$).

## DISCUSSION:

Summarize your results and discuss what your observations. For example:

- Compare the performance of your two-neuron perceptron and four-neuron perceptron.
- Compare the performance of different initial weights/biases.
- Compare the performance of different learning rates.
- Everything else you consider it is important.