**Project Name**: Perceptron – Hamming – Hebbian – Competitive - Minimization

**Team members:**

1. محمد عبدالله خلف الله السيد
2. أمانى حمدى
3. أيه محمد مهدى
4. ريهام رأفت
5. حنين خيرى
6. ايمان محمود

**Perceptron:**This network consists of single layer. First ask user to put the number of P and the number of elements in each P, then ask him if he has initial weight or not, the bias is required or not and if required is him has initial bias, then ask about the number of elements in each target, then enter your data and click on finish to beginning the learning with the perceptron rule until the output equal the target then show the finial weight and bias if require.

**Hamming:**

The Hamming network consists of two layers:

1- feed-forward layer

2- Recurrent layer

The first layer uses a linear transfer function and the output of this Layer is => a1 =W1p+b1

The second layer uses a positive linear transfer function, In this layer

We use a delay function => a2(t+1) = poslin(W2 a2(t))

The initial condition: a2(0)=a1

and the weight matrix => w2 = (1 - ε) ε is number less than 1/(S-1)

(-ε 1) , and S number of neurons

the aim of the hamming network is to decide which prototype vector is closest to the input vector. This decision is determined by the output of the recurrent layer.

**Competitive**

The competitive networks are closely to the Hamming network, the only different in the second layer.

In hamming: are recurrent which performs a competition to determine

Which of the prototype vectors is closest to the input vector?

In competitive: we will define a transfer function that does the job of a recurrent competitive layer

**a** = **compet**(**n**)

**Hebbian:**

It is rule that used to generate the corresponding weight for the given prototypes and the bias if it is required but the prototype must be orthonormal, if not it will use Pseudoinverse rule

Hebbian rule: W = T PT

Pseudoinverse rule: W = T P+ , P+ = (PT p)-1 pT

**Minimization:**

It convert multi linear-layer network to single linear-layer network using the following rules

Wnew = WM WM-1 … W2 W1

bnew = [WM WM-1 … W2] b1 + [WM WM-1 … W3] b2 + ….. + bM

Where M is the number of layers