

Introduction:

1. This is a pattern recognition program that recognizes Characters using Neural Network
2. Back propagation learning method is used in this program.

Program Components:

1. Interface

The result of this component is Input array o1 which is two dimension array of size[12][35](Where 12 = 4learning patterns + 8 test patterns),\ Also keep number of test patterns you enter.

2. Neural

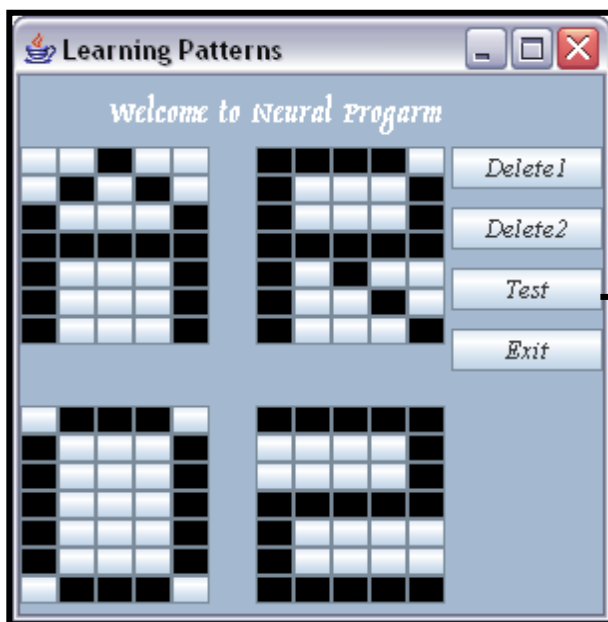
The result of this component is array node_max which is array of one dimension of size[12]

3. Result

The final result is showed here after classification.

Program Steps:

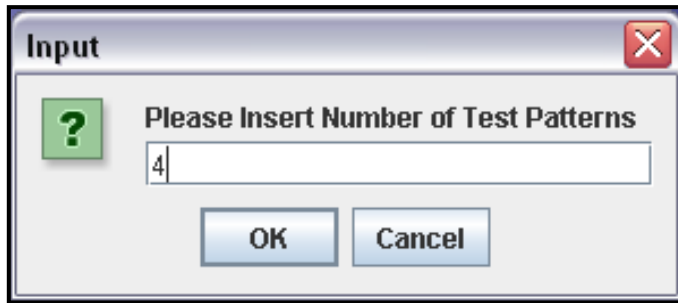
- When we run **interface** we enter our 4 patterns, then we test



Go to Test
Patterns Frame

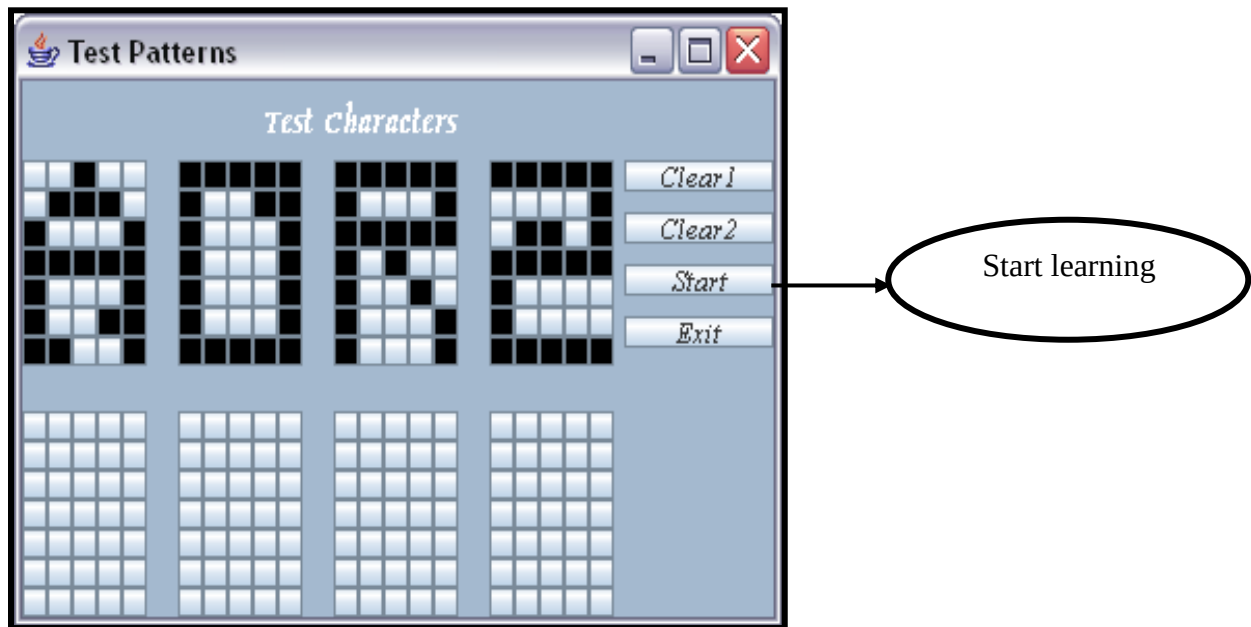
Learning Patterns

- Get number of test characters to check:



In this example we will test 4 Patterns

- Insert test characters to check and try to recognize it :



Maximum allowed test patterns is 8

In Neural Component

As we use back propagation Algorithm we implement it as follow:-

Step1:

Feed forward training

In feed forward training we initialize all weights and biases random by the same way we use in perceptron learning and perform the same computation for each learning pattern.

until 4 learning patterns are used then we compute $MSE = [\sum (t-o)^2]/2$ if the MSE is greater than the error we assume in our project (value very small like 0.07) then we start the second step in back propagation

Step2-Step3

Backtracking errors and update weights

- **Take first learning pattern** and go the forward path calculation as you previously did, use the weights you generate random, but only with these learning pattern
- Then once you get output in output layer start to compute delta rule for output neurons as follow:-

$$\delta_3 = (t - y)f'(o_3)$$

$$f'(o_3) = o_3(1-o_3)$$

- Then use delta rule of output neurons to update weights from hidden to output layer as follow:-

$$w(\text{new}) = w(\text{old}) + \Delta w$$

Where

$$\Delta w = \eta \delta_j x_i$$

- Compute delta rule for hidden neurons as follow:-

$$\delta_j = \sum_{k=1}^m \delta_k w_{jk}$$

For each **j** of hidden neurons connect with each **k** of output neuron
And then update weights from input to hidden using the same rule as above.

- After you calculate delta rule for both output and hidden neurons and update weights use the new weight with the second learning pattern and go through the forward path as before until you get the output , then repeat the same procedure as above(calculate delta and update weights).
- After 4 learning patterns compute MSE again and compare it with the error again and continue until the MSE is less than or equal to the error you assume.
- Now your Neural Net has learnt the 4 patterns start to test.
- Now with the last values of weights take every row of input array o1 (start from learning ,then test) and compute what so called node_max it is array of one dimension of size 12(guess why?!) this array contains index of max value of each output you got from forward training you perform on learning patterns and test patterns using final weights(go forward path only).

node_max will look like = [0 1 2 3 2 1 0 3 2 1]

In Result Component

- Use node_max to classify each of test patterns as follow:-

```

For (i=4, j=0; j< #Test_pattern ; i++,j++)
If(node_max[i]==0){
    For(k=0 ; k<35; k++)
        If(o1[0][k]==1)
            Matrix[j][k].setBackground(Color.black);
}
If(node_max[i]==1){
    For(k=0 ; k<35; k++)
        If(o1[1][k]==1)
            Matrix[j][k].setBackground(Color.black);
}
If(node_max[i]==2){
    For(k=0 ; k<35; k++)
        If(o1[2][k]==1)
            Matrix[j][k].setBackground(Color.black);
}

```

```
}  
If(node_max[i]==3){  
  For(k=0 ; k<35; k++)  
    If(o1[3][k]==1)  
      Matrix[j][k].setBackground(Color.black);  
  
}
```

- That is All thnx.

GOOL Luck