Homework #5

Submitted by: Veera Venakata Sathya Bhargav Nunna

a.

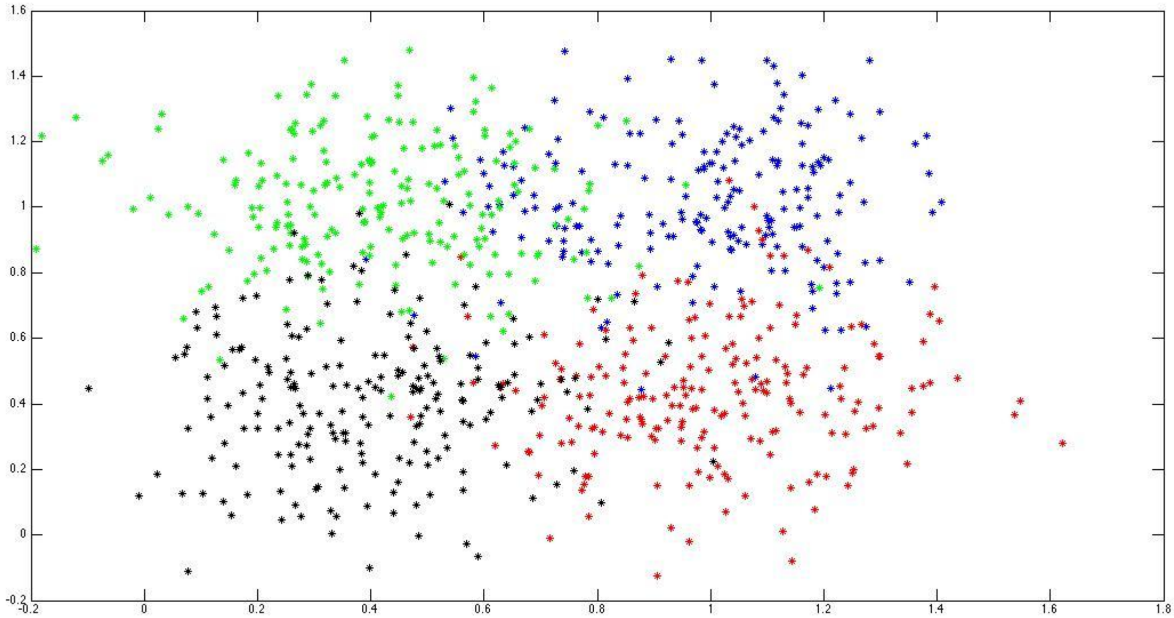

Fig. 1 Training dataset scatter plot Blue: Class1 Red: Class2 Green: Class3 Black: Class4

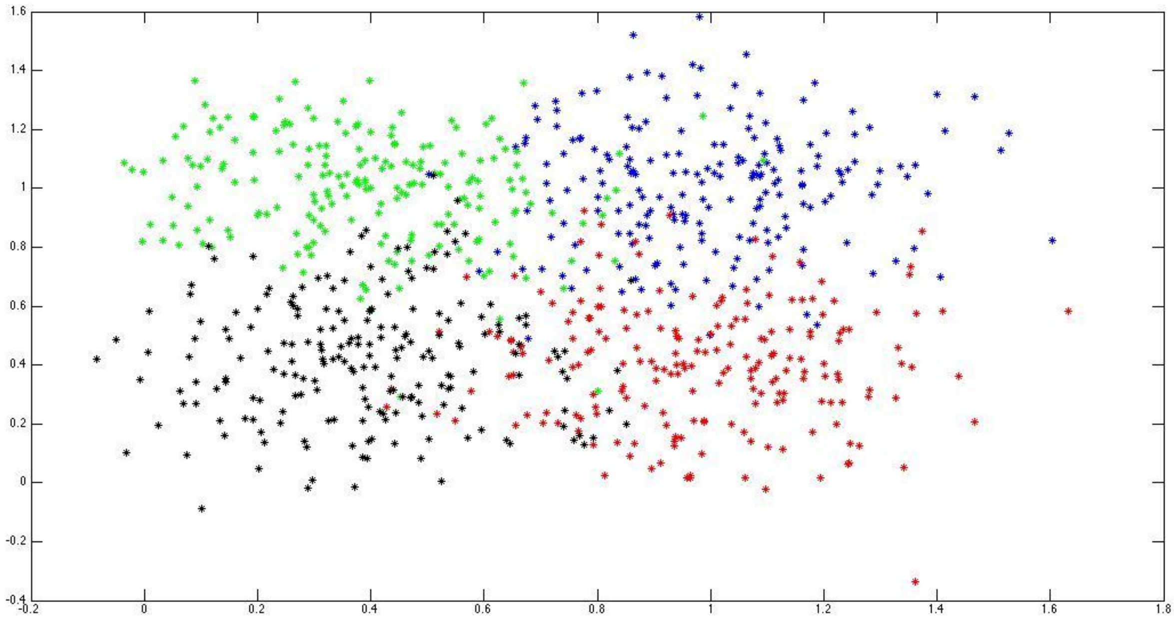

Fig. 2 Testing data scatter plot Blue: Class1 Red: Class2 Green: Class3 Black: Class4

The plot for the training data set suggests that the data can be divided into 4 clusters where in which more data points are concentrated. But there are some outliers of each cluster which go out into other clusters. It is the same for testing data set. In both cases the data is not linearly separable. Hence the number of wrongly classified data points (errors) can never be zero. We try to reduce the errors.

b.

Below is the table summarizing the number of neurons in the hidden layer:

Number of Neurons	EPOCH	Training Class1	Training Class2	Testing Class1	Testing Class2
10	10	19	33	21	41
20	10	21	33	29	52
30	11	26	22	38	31
40	22	21	26	27	32

Table1: Number of neurons in the hidden layer and the number of epoch took to reach the stopping criterion.

Below is the graph for number of neurons in the hidden layer vs number of epoch to reach stopping criterion

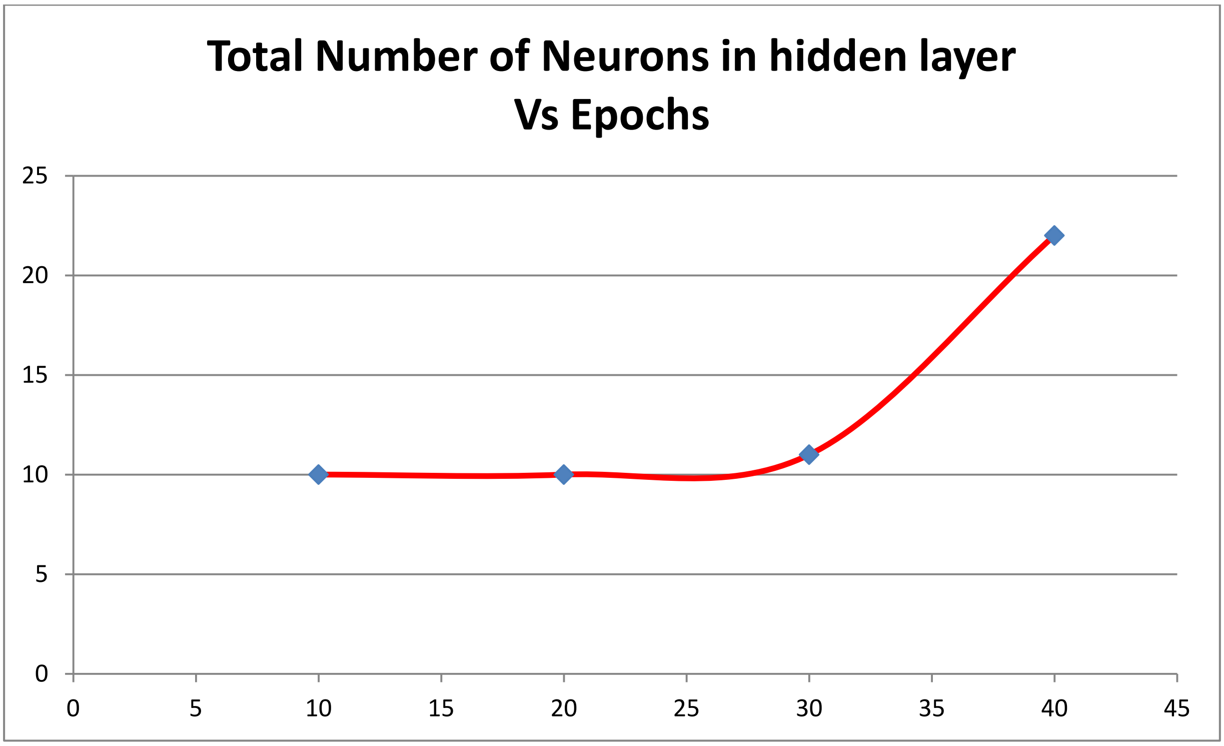

Fig. 3: X – axis: Total Number of neurons ; Y – axis: Number of epochs;

Below is the graph for number of epochs vs the number of neurons in the hidden layer

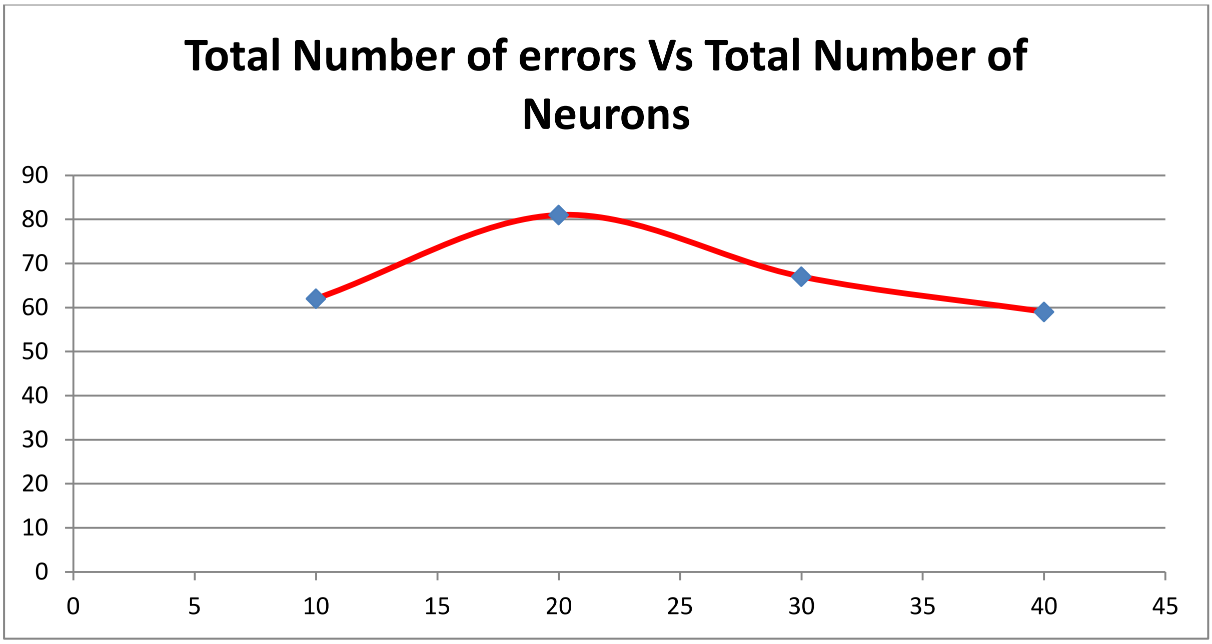

Fig. 4: X - axis Number of neurons; Y – Total Number of errors;

c.

For the learning with 40 neurons in the hidden layer the total number of errors for testing data seems to be lower than any other combination. Below is the graph for the gradient vs epochs with 40 neurons in the hidden layers.

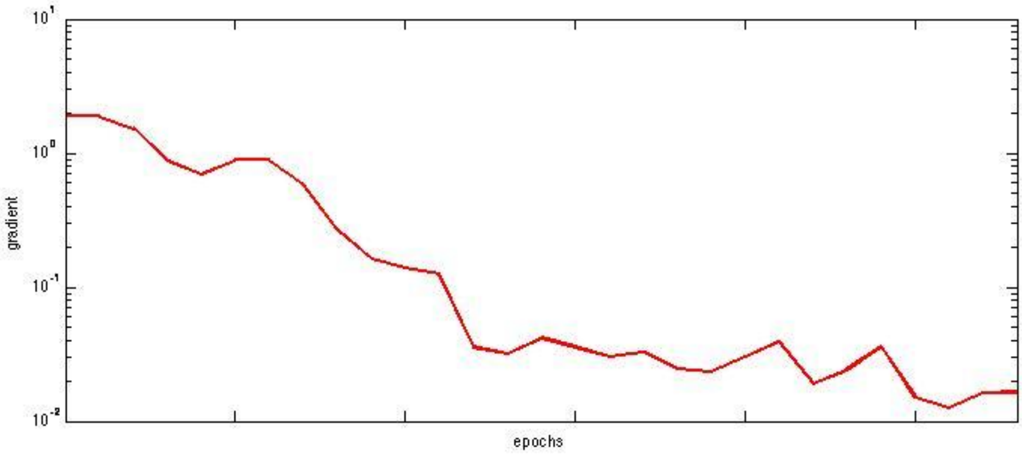

Fig. 5: Gradient vs epochs for 40 neurons in hidden layer

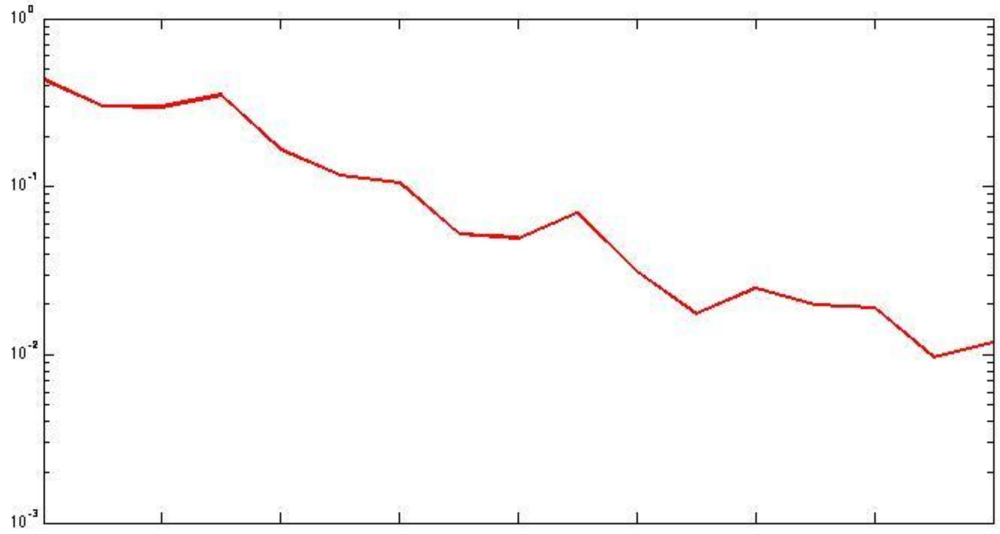

Fig. 6 Gradient Vs epochs with 10 neurons in hidden layer X-axis: Gradient; Y-axis: epochs

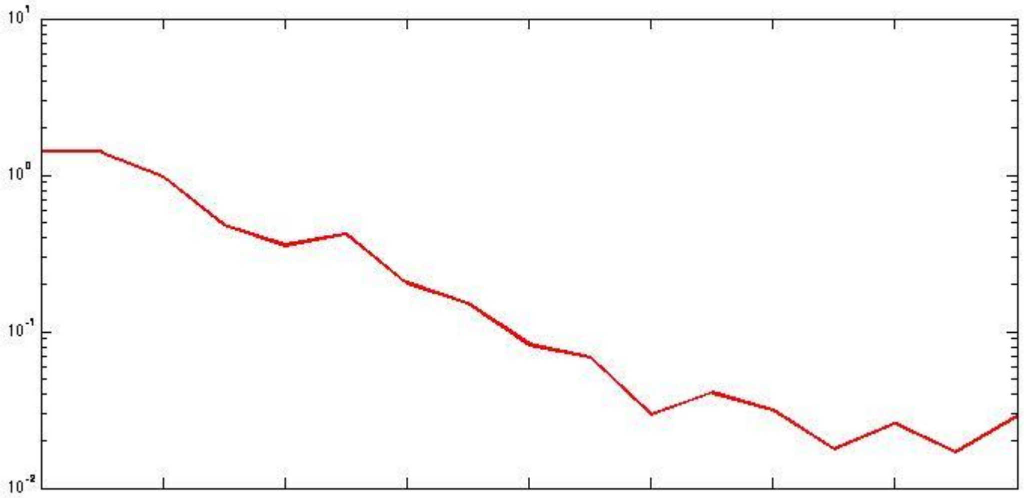

Fig. 7 Gradient Vs epochs with 20 neurons in hidden layer X-axis: Gradient; Y-axis: epochs

As the number of epochs increase the gradient seems to remain remains constant which is considered as the stopping criterion for the learning.

Appendix:

```
#include<iostream.h>
#include<math.h>



int main()
{

c1x1[200],c1x2[200],c2x1[200],c2x2[200],c3x1[200],c3x2[200],c4
x1[200],c4x2[200];
    double c1x1[800]={Training X1 data here };
    double c1x2[800]={Training X2 data here };
    double tc1x1[800]={Testing X1 data here };
    double tc1x2[800]={Testing X2data here };
    double ii[2],wi[2],oi[2],bi[2];
    double eh1[50],eh2[50],eo[4];
    double n=0.1;


    int l1,l2;
    cout<<”Enter number of neurons in Layer 1: ";
    cin>>l1;
    cout<<"Enter number of neurons in Layer 2: ";
    cin>>l2;
    double
h1w[l1][4],h1sum[l1],h1o[l1],h2w[l2][l1],h2sum[l2],h2o[l2],wo[
4][l2],oo[4];
    double err_o[4],err_h1[l1],err_h2[l2];
    double d[4]={1,0,0,0};;
   double sum;
    int itr=0;
    int er_count=0;
    int fl=0;
    //Initial random weights

    cout<<"Initializing weights..\n";
    for (int i=0;i<2;i++)
    {
        oi[i]=1;
    }
    for(int i=0;i<l1;i++)
    {
       for(int j=0;j<4;j++)
       {
           h1w[i][j]=1;
```

```
    }
 }

 for(int i=0;i<l2;i++)
 {
    for(int j=0;j<l1;j++)
    {
        h2w[i][j]=1;
    }
 }

 for(int i=0;i<4;i++)
 {
    for(int j=0;j<l2;j++)
    {
        wo[i][j]=1;
    }
 }
 for(int i=0;i<100;i++)
 {
         itr++;
          er_count=0;
// Input t neurons
 for(int i=0;i<200;i++)
 {
     if(fl==0)
     {
     ii[0]=c1x1[i];
     ii[1]=c1x2[i];
     fl=1;
     cout<<"Training...\n";
     }
    else if(fl==1)
    {
    ii[0]=tc1x1[i];
     ii[1]=tc1x2[i];
     fl=0;
     cout<<"Testing...\n";
    }
     oi[0]=ii[0]*wi[0]+1;
     oi[1]=ii[1]*wi[1]+1;
     oi[0]=tanh(oi[0]);
     oi[1]=tanh(oi[1]);
     //Input layer to hidden layer 1
     for(int i=0;i<l1;i++)
     {
       sum=0;
       for(int j=0;j<2;j++)
       {
          sum=(oi[j]*h1w[i][j])+sum;
       }
```

```
    sum=sum+1;
    h1sum[i]=sum;
    h1o[i]=tanh(sum);
  }

  //Hidden layer 1 to hidden layer 2

  for(int i=0;i<l2;i++)
  {
  //cout<<"hidden 1 to 2 I :"<<i<<"\n";
    sum=0;
    for(int j=0;j<l1;j++)
    {
       sum=(h1o[j]*h2w[i][j])+sum;
    }
    sum=sum+1;
    h2sum[i]=sum;
    h2o[i]=tanh(sum);
  }

  //Hidden layer 2 to output layer

  for(int i=0;i<4;i++)
  {
    sum=0;
    for(int j=0;j<l2;j++)
    {
       sum=(h2o[j]*wo[i][j])+sum;
    }
    sum=sum+1;
    oo[i]=tanh(sum);
  }
  //Feedforward complete

  if(d[0]!=oo[0]&&d[1]!=oo[1]&&d[2]!=oo[2]&&d[3]!=oo[3])
  {
 //   if(fl==0)
   // {
      er_count++;
     //}
  //if(fl==1)
  //{
//     cout<<"Error corec..\n";
  for(int i=0;i<4;i++) //output layer errors
  {

     err_o[i]=oo[i]*(1-oo[i])*(d[i]-oo[i]);
  }

  for(int i=0;i<4;i++) //weight changes in output layer
  {
```

```
            for(int j=0;j<l2;j++)
            {
              wo[i][j]= wo[i][j]+(n*err_o[i]*h2o[i]);
            }
          }

          for(int i=0;i<l2;i++) //hidden layer 2 errors
          {
            sum=0;
            for(int j=0;j<4;j++)
            {
               sum=err_o[j]*wo[j][i]+sum;
            }
            err_h2[i]=h2o[i]*(1-h2o[i])*sum;
             //err_h2[i]=h20[i]*(1-
h2o[i])*((err_o[0]*wo[i][0])+(err_o[1]*wo[i][1])+(err_o[2]*wo[
i][2])+(err_o[3]*wo[i][3]));
          }

          for(int i=0;i<l2;i++) //weight changes for layer 2
          {
             for(int j=0;j<l1;j++)
             {
                h2w[i][j]= h2w[i][j]+(n*err_h2[i]*h1o[i]);
             }
          }

          for(int i=0;i<l1;i++)    //errors for hidden layer 1
          {
            sum=0;
            for(int j=0;j<l2;j++)
            {
              sum=err_h2[j]*h2w[j][1]+sum;
            }
            err_h1[i]=h1o[i]*(1-h1o[i])*sum;
          }

          for(int i=0;i<l1;i++) //weight changes for hidden
layer 1
          {
             for(int j=0;j<2;j++)
             {
                h1w[i][j]= h1w[i][j]+(n*err_h1[i]*oi[j]);
             }
          }
        //} //if
        }// if

      }// for 200
      cout<<"ITR: "<<itr<<"\n";
      cout<<"Errors: "<<er_count<<"\n";
```

```
  }// for itr
    int dummy;
    cin>>dummy;


    return 0;
}
```