Homework 3

Rishab Goel

Neural Network Signal Processing

Problem 1:

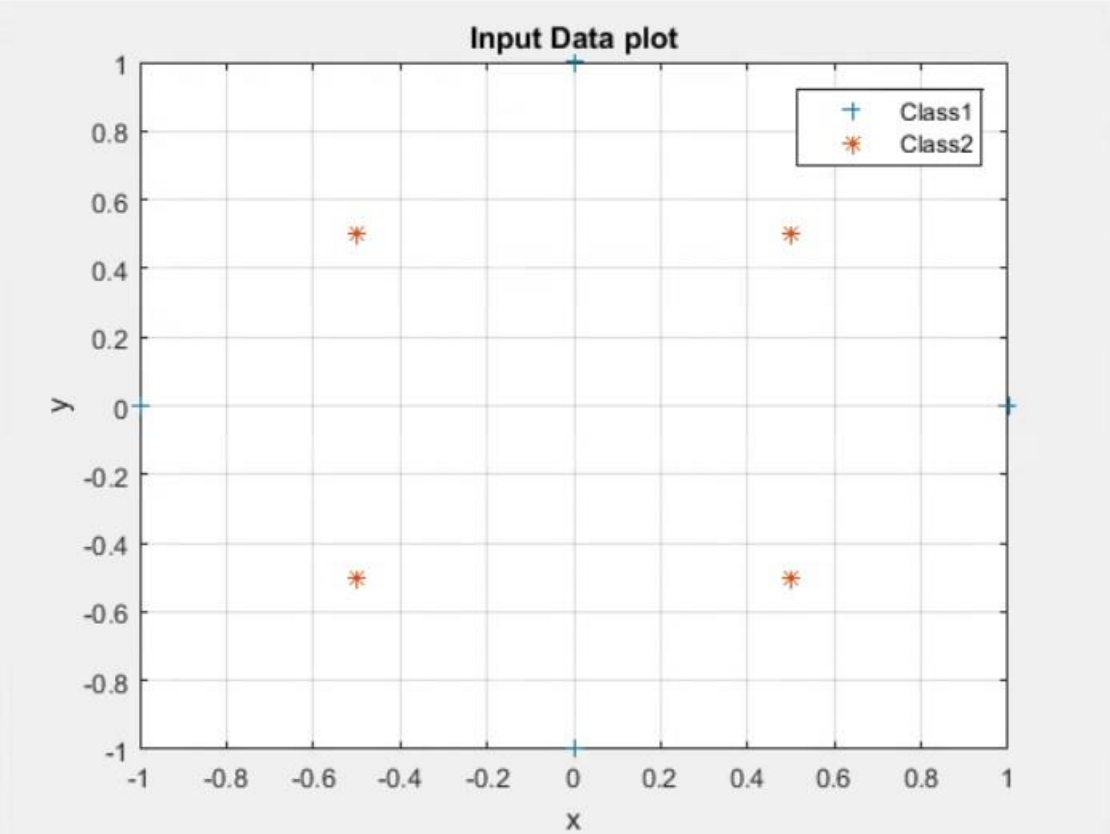
1- Code the backpropagation algorithm and test it in the following 2 class problem called the Star problem. You have to specify the size of the hidden layer and tell why you select that number of hidden PEs. x1 x2 d 1 0 1 0 1 1 -1 0 1 0 -1 1
0.5 0.5 0 -.5 0.5 0
0.5 -.5 0 -.5 -.5 0 I expect that the system learns this pattern exactly. You can think that points close to the x/y axes belong to one class and all the others belong to another class (hence the star). See how well your solution performs in points that do not belong to the training set, by selecting other points that follow the pattern (keep the points within the square of size 2). How could you improve the generalization accuracy of the solution?

Ans :

Initial Conditions for 8 given inputs:

Data given:

```
x= [1 0; 0 1; -1 0; 0 -1; 0.5 0.5; -0.5 0.5; 0.5 -0.5; -0.5 -0.5
d = [  1;   1;    1;    1;       0;         0;         0;       0];
```



From the input data plot it clearly appears

1) A Linear separation surface seems sufficient to classify the data points so number of hidden layers is only taken to be 1.
2) There need to be minimum 4 decision surfaces would be minimum to classify the input data points so number of PEs in the hidden layer are taken to be 4.
3) The shape of the minimal decision surfaces should be in from a square with two conescutive side having a slope of 0 and Inf .The distance from the origin of each side just larger than +ve or –ve 0.5.
4) Bias of all PEs is the last row the weights matrices. Weights and biases for each PE is given by each column of the weight matrices

Initial Conditons

Input to First Hidden Layer weights:

```
C = 0.5
w = [
     0   0    1   1;
     1   1    0   0;
    0.51 -0.51 0.51 -0.51;
    ];
    weights{1,1} = c.*w;
```

First Hidden layer to outputs weights:

```
    w=[1 1;
       1 1;
       1 1;
       1 1;
       1 1 ];
    weights{2,1} = w;
```

Learning Rate Value eta = 0.5 (Not varied or adaptive as this value gave satisfactory results)

Activation Function used was sigmoid and loss function MSE to keep it simple.

Output

- `Max_epoch = 15000 epochs = 3359  mse =    9.9978e-04`

Epochs and MSE are just to convey the stop criterion was mse < 0.001. Though for generalization it doesn't seem a valid stop criterion but for limited dataset it is satisfactory.

- Learned Weights

```
weights{1,1} =

   -6.9027   -6.1482   -5.8585    6.2348
   -6.8888    6.1999   -5.8675   -6.1789
   -3.7916   -3.4182    3.2902   -3.4488

weights{2,1} =

    7.6579   -7.6516
    7.7717   -7.7592
   -7.8904    7.8856
    7.7782   -7.7656
   -3.3070    3.2985
```

From the learned weights and biases it appears the separation surface form a rhombus or a kite with points closer to the origin belonging to one class and points outer the rhombus belonging to other class.

Test input 1 :

```
test_x = [ 0.3750    0.3750 ;  0   0.7500;   0.3750   -0.3750 ;  -0.3750
0.3750; -0.3750   -0.3750; -0.7500    0;   0   -0.7500 ;   0.7500  0]

test_d = [ 0     1; 1     0; 0     1; 0     1; 0     1; 1     0; 1     0; 1
0];

C_Mat =[ 4     0;
         0   4]
```

Accuracy= 100 %

Test Input 2

```
test_x= [0.75 0.75; 0.25 0.25; 0.25 -0.25; -0.25 -0.25 ;-0.25 0.25; 0.75 0.25
; 0.6 -0.6; 0.25 0.75];
test_d = [1;0;0;0;0;1;1;1];
```

C_Mat = [0 0;
4 4]

Accuracy= 50 %

Things could be done for Generalization Accuracy:

- For the test inputs 1 we saw full accuracy as (x,y) have similar pattern as training set while we could for test input2 were (x,y) don't have the similar pattern as training set. So for generalization we first and foremost require a larger dataset to learn from varied patterns.
- Failure of Test Input 2 could be considered a reason of overfitting to train data, so cross validation dataset could be helpful, but for that also more train data is required.
- More generalized decision surface to this problem would be a circle. Circle could be considered a shape with infinite number infinitesimally small straight lines, so increasing the number of neurons and in turn increasing the dimension of decision surface.
- For generalization the primary requirement is increasing dataset but adaptive learning rate and experimenting with softmax layer could give better results.

Problem 2 Classification of the Sleep data

2- The sleep datasets are larger, more involved classification problems where the goal is to design a MLP to classify the sleep data. Sleep is not an uniform physiologic state. When one sleeps our brain waves (EEG) go into a set of stages (awake to stage 5), but remember, no two brains are the same. The goal is to be able to classify each minute of sleep, by utilizing the information on a set of EEG descriptors (Alpha, Beta, Delta, Sigma, Theta, Artifact, REM 1 and REM2 counts per minute) that are automatically detected from the EEG. Therefore, sleep stages are your desired targets, and these descriptors are your inputs. You will find in the course website two data sets from two different patients. You train in patient one, and test on patient 2. There are four files, two with the inputs (EEG descriptors) and the other two with the manual sleep scoring. Summarize the results of the training and testing in a confusion table, and show the effect on performance of different stepsizes, and different initial conditions. How could you improve the generalization results?

Ans :

Explanation for the free variables and initial conditions taken in learning for Problem 2

Number of Classes are 6 and Number of Inputs were 8.

Number of Hidden Layer: We start simple with one hidden layer as theoretically as single hidden layer perceptron can classify any form of data. We vary the layers from 1 to 3.

Number of Neurons: Since the output classes are 6 it would at least require 5 separation surface for clearly separable classes. But we expected that the class categorizing different stages of sleep won't be as clearly separable to have the minimum separable surfaces possible. We started the number of neurons just one more than input i.e. 9 and got less than 50% accuracy and found it decreasing for lesser values. Therefore, for first hidden layer the minimum number of neurons were kept 9. The number of processing elements were varied and some results are shown in the table below.

Learning rate (Eta) was made adaptive using the formula eta = eta_0/ (1+n/N_0). The experiments for eta_0 started with 0.5 ranging to 0.9. The adaptive size was taken such that to avoid the increase the speed of convergence at initial epochs while reduce it slowly after some time so to avoid diverging and slow convergence.

Max Epoch 15000 and **N_0** 5000 were arrived to this value after multiple iterations to have least observed test data error and on the basis smoothest convergence for least square error observed. Epochs more than 15000 was rarely enhancing the Test Data accuracy, instead the time to generate results also increased. N_0 was narrowed to 5000 experimentally by test ran on one layer perceptron.

Initial Weights and Biases are taken as random values ranging from -1 to 1. So each run even with same learning rate would give varied results.

```
w = -1+ 2*rand(nodes_sizes(i)+1,nodes_sizes(i+1));
```

where nodes_sizes is the number of processing in each layer including input and output layer.
The weights were varied from -1 to 1 considering that the decision surfaces could have positive and negative slopes both. Giving such range of weights for the initial conditions may assist in the faster and more accurate convergence.
Note that the initial weights and biases of the best results is only displayed at the end to avoid too much data display.

Training Method : We tried online training initially instead of batch, but it was taking a lot time than batch, and online training updates weights is each sample which lead to highly vibrating weight tracks while batch training slightly smoothens out those weight tracks. Batch was changed from 1, 5 to 10. 1 is actually online and among 5 and 10, we narrowed down to 10 after multiple iterations of runs to obtain decent speed and good test error.

Activation function was kept sigmoid, with Least Square Error was the loss function. We have control to choose softmax activation function and cross entropy loss function (by putting loss = 1) but the cross entropy function was decreasing. Least Square error results were closer to satisfaction to we present results on the basis of MSE.

Stop Criterion was just the expiry of the Max epoch as the cross validation dataset error reduction gave worse results. MSE error > 0.01 was also the stop criteria, but that was to avoid overfitting to train data.

Confusion Matrix: The rows indicate the predicted class while columns indicate the actual class. The diagonal entries are the correct predictions.

Table to indicate some tested configurations for the Sleep data and their accuracy and predictions

Hidden Layers	# of PEs in each	Learning Rate(eta)	Confusion Matrix	Accuracy
1	[9]	0.5	46 0 0 19 6 35 8 34 0 104 0 0 0 10 0 0 0 0 4 16 11 84 0 3 0 1 0 2 1 9 0 0 0 0 1 3	42.32 %
1	[9]	0.7	49 0 0 5 4 38 0 24 1 23 0 0 0 2 1 0 0 0 7 35 9 170 0 5 2 0 0 1 1 1 0 0 0 10 3 6	63.22 %
1	[9]	0.8	41 0 0 25 5 40 0 30 0 0 0 0 0 9 2 2 0 0 15 17 9 178 1 3 2 5 0 4 2 4 0 0 0 0 0 3	64.48 %
1	[9]	0.85	55 0 0 28 7 42 0 55 2 6 1 0 0 1 1 34 0 0 3 5 8 136 0 2 0 0 0 5 0 2 0 0 0 0 0 4	63.2242%
1 *	[10]	0.8	54 0 0 21 3 39 0 40 0 9 2 0 0 2 1 0 0 0 3 19 10 174 0 1 1 0 0 2 2 3 0 0 0 3 1 7	70.025 %
1	[10]	0.9	51 0 0 22 4 29 1 42 1 51 3 0 0 3 2 3 0 0 5 15 8 128 0 4 1 1 0 1 1 12 0 0 0 4 0 5	57.682 %
2	[9 6]	0.8	47 3 0 9 2 25 0 37 1 4 0 0 0 9 1 63 0 0 5 10 9 129 1 6 6 2 0 4 5 13 0 0 0 0 0 6	56.675 %
2	[9 6]	0.7	55 0 0 30 4 40 0 34 0 3 0 0 0 4 1 0 0 0	68.76 %

			3 21 10 173 2 1 0 2 0 2 2 1 0 0 0 1 0 8	
2 **	[10 6]	0.8	53 0 0 17 5 34 0 46 0 2 0 0 0 4 2 4 0 0 4 10 9 180 0 6 0 1 0 5 1 6 1 0 0 1 2 4	72.04 %
3	[10 8 6]	0.9	52 0 0 11 6 40 0 32 0 0 0 0 0 8 2 2 0 0 5 21 9 191 0 3 1 0 0 5 2 2 0 0 0 0 0 5	71.53 %
3 ***	[10 8 6]	0.8	49 0 1 10 6 36 3 56 4 12 0 0 0 1 2 1 0 0 5 3 4 181 0 6 0 1 0 4 1 2 1 0 0 1 1 6	74.31 %

*Best Accuracy for Single Hidden layer

** Best Accuracy for Double Hidden layer

*** Best Accuracy for Triple Hidden layer

Initial and trained Weights of above three cases are tabulated at the end for information

Explanation for Table Observation:

- Firstly we get an accuracy 70% using only a single hidden layer for the neural network, which supports the fact that a single hidden layer perceptron is an optimal classifier.
- Adding the hidden layer to the neural network gives 2% accuracy increase for 2 hidden layer while 4% increase for 3 hidden layer case. This clearly shows that this test data could be classified with good accuracy with just using a linear decision surface.
- We observe the learning rate (eta_0) = 0.8 is the value which helps in faster and more accurate as the best results observed the learning. The learning rate value was adaptive with n_0 = 5000 so the eta value changes to 0.4 after 5000 epoch and then to 2.67 at 10000 epoch until it terminates at Max_epoch = 15000.
- We also observe for some cases is really low and it is observed to vary for same learning rare and hidden layer configuration. This could be attributed random initial weights and biases, and giving a more accurate stopping criterion may help like validation data set or different output layer activation or loss function may help. Though we plugged both the techniques were tried in our code, but best results were available without using them.
- We observe an accuracy of 74% maximum which indicates that sleep pattern data of one person can't partially predict the sleep pattern for the other person.

- We didn't increase the hidden layer further as no ran test for higher hidden layers had higher accuracy and with such minor increase in accuracy it doesn't seem to need to increase the memory and computation requirement of the neural network, thus, complicating the neural network.

Generalization Criteria

- For generalization training data we should have data from multiple persons sleep pattern rather than a single human sleep pattern which could increase the accuracy and reduce overfitting on the sleep pattern of one person.

Weights + Biases Data

1)

*h_l = 1 n_h_l = 10 eta0 = 0.8000

Initial Weights and Biases

weights{1,1} (Input with biases (9) to First hidden Layer (10)) [9x10]=

0.7685	-0.3914	-0.4604	0.9044	-0.8894	-0.4993	0.7919	0.0329	0.2255	-0.6750
0.4012	-0.4183	0.9795	0.0865	0.5076	-0.0232	-0.6199	-0.9850	-0.3984	-0.7729
-0.5163	-0.5150	-0.6326	-0.4972	-0.7361	0.4581	-0.9964	0.3779	0.5963	0.8258
0.5197	0.8734	0.7233	0.1571	-0.2882	-0.5948	0.4235	0.8920	0.5913	-0.0367
-0.4181	0.7204	-0.9347	0.8310	-0.2083	-0.5674	0.7355	0.7471	0.5622	0.7036
-0.4451	-0.2055	-0.3361	0.7912	0.7710	0.9527	-0.7634	-0.7734	-0.2978	0.6198
-0.9878	-0.0412	0.4975	-0.0350	-0.9575	0.1865	-0.9220	-0.2909	-0.8914	-0.6265
-0.2506	0.1300	0.2887	-0.1145	0.6882	-0.3912	0.1964	-0.5161	0.4174	-0.5056
-0.1261	-0.0208	-0.6615	-0.3765	-0.4239	0.9354	0.2086	0.1207	0.9859	-0.8916

weights{2,1} (First hidden Layer with biases (11) to Output(6)) [11x6]=

0.2179	-0.6930	0.4457	-0.0978	0.8517	0.0466
0.5545	-0.2864	0.3362	-0.9724	0.4817	-0.1402
0.0221	-0.7121	-0.6423	-0.0526	0.4753	-0.5857
-0.9445	0.7012	0.1010	0.9024	0.8938	-0.3532
0.9808	-0.3243	0.9198	-0.5021	0.0202	-0.7783
0.0019	-0.4496	0.1920	-0.2272	0.5838	-0.2496
-0.3360	-0.9880	0.6171	-0.1371	-0.0957	-0.3402

-0.6522 0.6038 0.9691 0.6618 0.6984 -0.3158

0.2513 -0.0052 0.7718 0.6493 -0.2191 0.6342

0.1503 0.0757 -0.5723 -0.0940 0.4768 0.0634

0.5020 0.7418 -0.9307 -0.2389 0.9529 0.0422

Trained Weights

weights{1,1} (Input with biases (9) to First hidden Layer (10)) [9x10]=

5.3714 0.7059 -4.9808 1.6312 -0.5645 -7.0310 5.7877 -3.8468 -0.8622 -2.0896

-3.3274 -4.6734 1.4531 4.2028 -0.7491 -4.6828 0.6782 -2.7304 -2.1326 2.6855

-10.3099 -9.8158 -1.9910 -2.3481 -1.0677 2.7808 -1.5077 0.6620 7.1965 14.6738

0.1949 0.4861 6.4188 2.3636 -0.2883 -0.6357 0.3835 3.8131 7.6332 -0.2160

-1.8155 0.0308 -1.8002 0.5617 -2.0667 -2.7762 2.0972 0.4699 -1.3432 -1.2546

3.0132 1.2721 0.5035 0.1310 0.2831 1.2047 -1.4054 -1.3541 -1.4835 -0.4720

0.8639 -2.4475 7.7741 0.6407 -5.1277 -2.4929 -3.7687 -3.9301 -2.0067 -3.5690

-4.9856 0.0501 0.5003 -2.7908 -1.3615 1.6795 6.4077 9.5810 1.9876 1.7084

3.0393 8.5580 -4.0314 12.7106 23.2350 -3.1862 16.0190 10.6674 10.2213 -1.3012

weights{2,1} (First hidden Layer with biases (11) to Output(6)) [11x6]=

12.2919 -2.8109 -2.1724 -11.0862 1.0830 3.8106

6.3440 -1.0595 -1.4350 -11.0772 3.0553 -6.9986

7.4784 6.3367 0.7060 6.8510 -10.4249 -3.5662

-6.8913 -2.4607 -13.8172 3.0755 3.1286 0.0992

-5.1880 -9.8184 7.1510 5.5021 6.1468 -5.8864

-1.4383 6.1082 2.9128 -10.8459 6.9229 -0.7577

-2.7338 -8.4423 7.0677 -1.4091 1.2339 0.6747

-5.4846 -7.0255 -3.7102 10.5721 -1.7988 2.6465

11.6259 -3.5707 1.7208 2.3883 -10.1077 -4.4452

-7.2357 4.3780 -0.7975 5.5273 7.6742 -8.1251

-6.9746 4.6046 -3.8529 -17.4496 -10.1821 -0.0196

2)

**h_l = 2 n_h_l = 10   6 eta0 =   0.8000
Initial Weights
weights{1,1} (Input with biases (9) to First hidden Layer (10)) [9x10]=

0.1010	-0.9724	-0.2389	-0.2191	-0.3402	-0.3308	0.7798	-0.2375	0.5500	-0.6836
0.9198	-0.0526	0.8517	0.4768	-0.3158	0.1492	0.0028	-0.8701	-0.6694	-0.4271
0.1920	0.9024	0.4817	0.9529	0.6342	0.7279	-0.4460	-0.2828	0.8244	0.3743
0.6171	-0.5021	0.4753	0.0466	0.0634	-0.6029	0.0679	-0.5315	-0.3616	-0.7177
0.9691	-0.2272	0.8938	-0.1402	0.0422	0.3449	0.1485	-0.5930	-0.3404	0.0242
0.7718	-0.1371	0.0202	-0.5857	0.5486	0.8037	-0.1744	0.6276	-0.5915	0.4427
-0.5723	0.6618	0.5838	-0.3532	-0.7595	-0.6017	-0.9705	-0.2131	0.5344	0.8577
-0.9307	0.6493	-0.0957	-0.7783	0.2509	-0.4034	0.4055	-0.8928	-0.8601	0.4642
-0.0978	-0.0940	0.6984	-0.2496	-0.3067	-0.0070	0.0135	-0.2499	0.9001	0.4997

weights{2,1} (First hidden Layer with biases (11) to Second hidden Layer (6)) [11x6]=

-0.1854	0.9737	0.0940	0.0152	-0.3460	0.6650
-0.5210	0.5399	-0.1941	0.5776	0.6081	0.9907
0.0417	0.6592	-0.7859	-0.0539	0.0765	0.2995
-0.5618	0.4122	0.4483	0.6576	-0.0734	0.4079
0.6848	0.1907	0.2274	-0.3550	0.6415	0.8646
0.3259	0.5057	0.5659	0.9523	0.9038	0.3753
0.6325	-0.0066	0.1332	-0.4436	-0.8475	0.1367
0.5878	0.7303	0.6226	-0.8543	0.4173	-0.2383
-0.0618	-0.8639	0.1536	0.5024	-0.5301	0.2692
-0.3810	0.9371	0.8881	0.6624	-0.2022	-0.2735
0.3752	-0.8025	0.7429	0.8447	-0.4638	-0.1848

weights{3,1} (Second hidden Layer with biases (7) to Output layer (6)) [6x6]=

-0.2626	-0.0261	0.1265	0.3217	-0.2774	-0.3633
-0.0632	-0.4756	0.0786	-0.2922	0.4833	0.1942
0.0068	0.1592	0.5361	-0.3056	0.4118	-0.4044
0.8211	0.7567	-0.5338	-0.4926	0.4018	-0.7500
-0.5871	-0.8781	0.1747	0.9051	-0.9875	-0.2233
-0.3228	-0.1182	-0.0821	-0.4036	-0.2513	0.6354
0.1483	-0.8315	0.7220	-0.6832	0.8030	0.9624

Trained Weights
weights{1,1} (Input with biases (9) to First hidden Layer (10)) [9x10]=

-0.5393	0.2133	0.8497	0.5018	-4.3178	-7.5694	3.8991	-2.5087	-0.5330	-3.7290
2.0455	-1.8226	1.1351	4.4204	2.1718	6.4690	2.9048	-1.8575	-6.7182	-4.6996
1.5624	4.9298	1.3832	4.8958	8.3299	15.0619	-11.1371	-0.3395	2.2606	2.4158
2.4612	0.6248	0.5963	-9.8194	-0.2296	-0.7070	-4.1689	-0.9852	1.8643	-1.6146
1.1998	-2.1388	0.8906	-1.4031	-4.3811	2.3428	1.2919	-2.4368	-4.3692	1.0112
0.4263	0.4281	0.2903	9.6628	-1.6625	-4.1158	0.1846	3.0266	3.1098	-0.9520
-0.8272	6.5056	0.7156	-0.5785	-2.8930	-3.0053	-1.1185	1.1861	-0.3795	-0.5535
-0.7357	12.7820	-0.0941	2.9242	0.4173	2.0374	-5.7269	-0.6308	-1.0793	1.8764
-0.2873	0.9745	1.3213	-10.7700	-0.2430	-1.1643	-4.8823	9.5625	2.6869	-3.0174

weights{2,1} (First hidden Layer with biases (11) to Second hidden Layer (6)) [11x6]=

-1.3498 2.0465 0.2644 0.4422 -0.5874 1.8853
-3.0121 5.2776 -13.1256 3.4719 3.5572 6.0652
3.1789 -2.2874 -3.2453 0.8587 0.1763 -1.5282
-4.6131 7.0945 6.6289 -5.4865 -5.6792 5.2321
3.8354 6.3150 6.8002 -2.9521 6.6731 0.9782
2.1789 2.9431 0.5171 4.3726 10.2625 -5.3571
-2.2305 1.1386 -3.0534 -2.3434 -9.3200 -4.2463
3.0711 -3.6898 -8.6996 -0.1979 7.1146 1.6076
2.0224 -4.1475 10.4963 3.0974 -10.0986 -1.7659
-6.1197 5.1974 2.3639 2.3410 -6.5629 4.9549
3.7076 -4.1406 -1.9564 1.9377 -0.4792 -1.9708
weights{3,1} (Second hidden Layer with biases (7) to Output layer (6)) [7x6]=

4.9610 -4.5795 -5.2607 2.7704 1.6501 -6.3956
-9.6833 1.0439 -5.0233 3.1185 6.1102 0.9880
-1.7367 12.7395 7.4533 -8.4328 -5.7335 -2.7482
0.1849 1.1374 1.5256 -2.7361 -3.8273 -4.3514
-6.3082 -5.5927 8.4239 9.5688 -4.3551 -2.8774
2.9166 -5.4315 3.4730 -4.1409 -7.7167 2.9851
1.4214 -4.3280 -8.4486 -5.3594 -2.3783 -0.0472
3)
*** h_l = 3 n_h_l = 10 8 6 eta0 = 0.8000
Initial weights
weights{1,1} (Input with biases (9) to First hidden Layer with biases (10)) [9x10]=

0.1001 -0.8552 0.5959 -0.8687 0.2743 -0.7371 0.9022 0.5994 -0.3938 -0.8883
-0.4476 0.8007 -0.9872 -0.5399 0.4930 -0.4886 0.2764 0.2605 -0.6639 -0.9383
0.3925 0.9386 0.8398 -0.7659 0.6105 -0.2434 0.0083 0.9656 -0.3202 -0.6268
0.7627 -0.2174 -0.9641 0.9795 -0.4619 0.9875 -0.2838 -0.6835 -0.8651 -0.4582
-0.9506 -0.3728 -0.9412 -0.8590 -0.1320 -0.3181 0.5369 0.2334 0.3070 -0.8675
-0.3176 0.1066 0.4229 -0.8800 -0.1964 0.7991 0.5687 -0.3826 -0.5114 -0.3330
-0.1572 0.5841 0.0768 0.3232 0.2163 -0.5250 -0.9422 -0.8298 0.5158 -0.9002
-0.8400 0.5966 0.0492 -0.3119 0.5402 -0.5600 -0.8951 0.7537 -0.4020 -0.7974
-0.8413 0.7265 0.0045 -0.7430 0.8822 0.9822 -0.3532 0.1814 -0.1624 -0.8563
weights{2,1} (First hidden Layer with biases (11) to Second hidden Layer (8)) [11x8]=

0.8090 -0.7375 0.1132 0.7030 -0.8371 -0.1561 0.1039 0.5483
-0.3764 -0.6550 0.7774 0.9693 0.2190 -0.2684 0.4845 -0.7044
0.1835 0.8772 0.6803 0.5804 0.0893 -0.8357 0.6493 -0.0289
-0.2447 -0.9310 -0.5119 -0.2986 0.7118 -0.8605 -0.2132 -0.8563
0.9031 0.7591 -0.1288 0.0038 0.2854 -0.7582 -0.7281 0.5749
0.4848 0.0447 0.6571 -0.4896 -0.4567 0.3557 -0.6085 -0.2125
0.7742 0.6165 -0.6946 0.0197 -0.7865 -0.5621 0.4406 -0.5277
0.1978 -0.1048 0.5159 -0.8428 -0.8790 -0.5254 0.3393 -0.8389
0.3793 -0.3069 0.7530 -0.8882 0.2438 -0.9827 -0.8803 -0.8420
-0.2178 0.5143 0.0042 0.2700 0.3981 0.4542 -0.2314 0.8549
-0.5535 -0.2796 0.1405 0.2338 -0.4501 0.1221 -0.0175 -0.5906
weights{3,1} (Second hidden Layer with biases (9) to Third hidden Layer (6)) [9x6]=

0.9132 0.5754 -0.7831 0.3662 0.9134 0.4086
-0.3205 -0.3046 0.7075 0.8224 -0.4560 -0.0060
0.9199 0.9443 0.8217 0.1410 0.0540 0.2192
-0.3137 0.6721 -0.3325 -0.4694 0.8224 -0.8443
-0.8732 0.4236 -0.0617 0.8569 -0.5706 -0.5171
0.2249 -0.5702 -0.9942 -0.8715 0.6536 -0.9484
0.1112 -0.9691 -0.3782 0.2526 -0.7233 0.6930
0.8129 -0.5799 -0.0296 0.8966 -0.1245 -0.9703
-0.5975 0.0729 -0.4757 0.0328 0.7506 0.1008

weights{4,1} (Third hidden Layer with biases (7) to Output layer (6)) [7x6]=

-0.7607 -0.7979 0.3241 -0.7827 0.7044 -0.7613
0.1759 -0.4378 0.5462 -0.8219 -0.5104 0.9571
0.2444 0.8150 0.8640 -0.7697 0.4543 0.0365
0.4940 -0.7778 -0.6462 -0.6641 -0.1984 0.0155
-0.3801 0.5796 -0.9592 0.9450 -0.3788 -0.3500
0.8330 -0.7954 0.6673 -0.4280 0.3076 -0.6055
0.5645 0.5359 -0.0536 0.1980 -0.2163 0.7258

Trained weights

weights{1,1} (Input with biases (9) to First hidden Layer with biases (11)) [9x11]=

0.2558 -4.1963 -1.7316 4.6049 -0.4113 -0.6254 2.1373 1.4217 5.8321 -1.0838
-3.1394 5.1226 -5.1258 -4.3543 1.8427 1.6632 1.8862 4.1966 -1.0742 -0.0833
1.9674 8.4777 3.1104 -10.2729 5.4241 -0.8199 -7.7382 4.0336 -1.8831 0.2148
5.7032 -0.1679 -5.2391 0.3957 -1.4064 0.6519 -1.0562 -11.2001 -1.0822 -0.5044
-5.4741 5.0964 -1.0286 -1.1041 -0.0246 -0.8305 1.8540 -2.1921 2.0775 -0.7416
1.0678 -0.8893 0.1618 1.9523 -0.0475 -0.8127 2.0836 1.2139 -1.2063 -0.3360
6.4856 -3.2688 -1.0657 3.9798 1.7788 0.1451 0.1522 1.3122 0.6128 -0.7478
3.8471 1.0943 3.4005 -0.6814 2.0432 2.9312 -5.9034 -3.0352 -0.0054 -0.8144
4.2356 -4.8775 -6.0966 0.2095 0.5095 15.2549 -11.3253 -2.0231 2.7097 1.2614

weights{2,1} (First hidden Layer with biases (11) to Second hidden Layer (8)) [11x8]=

4.8667 -3.8815 -5.6795 -1.1431 3.1900 -4.9947 3.8620 -1.6097
-2.9101 -4.4379 0.7210 5.4083 3.9882 0.0994 -3.9495 -0.7562
0.8952 7.1531 6.6517 2.9540 -2.3758 -3.1479 4.8181 -3.2849
-1.0619 2.3211 -0.8475 -0.9232 5.4515 1.3249 5.7268 2.6532
-5.4731 1.8478 0.3662 -1.4543 3.5287 -0.0675 -4.3489 5.9075
3.1893 -0.5822 -4.1476 1.2872 -3.6723 0.9737 -3.3879 -3.5122
1.5226 2.3594 -2.1555 -5.9747 1.2134 0.3372 8.6519 0.9491
1.6753 -2.0375 3.7298 -3.6412 -4.2038 3.8622 4.5275 0.1911
5.4591 -2.9186 2.8172 -4.7504 4.0505 2.6888 -4.3124 -4.6860
-0.0177 0.2277 -2.1327 0.4329 1.1226 0.4259 -0.6661 0.4769
-1.7818 -0.3801 1.9335 3.4165 -3.0899 1.1156 -0.3362 0.9316

weights{3,1} (Second hidden Layer with biases (9) to Third hidden Layer (6)) [9x6]=

6.0338 -3.4393 -7.6978 1.6368 0.7383 4.0577
6.9530 -1.4512 3.6913 -0.0208 0.8484 -2.8162
6.5434 -0.8656 0.3764 -2.1326 -7.6897 -2.6309
-0.4992 5.5092 4.0158 -6.1485 4.4523 -2.4929
-7.1246 2.7093 -1.3877 4.7948 -3.8345 -1.4298

-1.5925 1.1425 -4.9857 -4.7696 4.8079 -3.9400
0.4802 -10.1297 2.5782 4.2866 -2.0727 4.4114
-1.3092 -0.1103 5.7306 2.0385 -1.2729 -4.0471
-1.6678 -1.5660 -1.8561 -2.1131 2.3873 1.9656

weights{4,1} (Third hidden Layer with biases (7) to Output layer (6)) [7x6]=

-9.2567 -2.0676 5.3829 -6.3128 6.3827 -4.1880
-2.7704 -8.9297 5.2620 3.0291 -9.0995 -0.6508
-3.7545 7.5420 5.7142 -3.6470 -9.3255 -5.3547
1.9220 1.9944 -4.6972 -5.1978 -5.6915 5.7947
-0.4971 2.0571 -7.1520 7.3863 -3.5076 -6.9498
7.6552 -5.4520 -1.1039 -4.0835 2.7325 -5.4799
-3.8885 -4.8897 -7.9582 -1.5743 -1.2683 -0.6939