

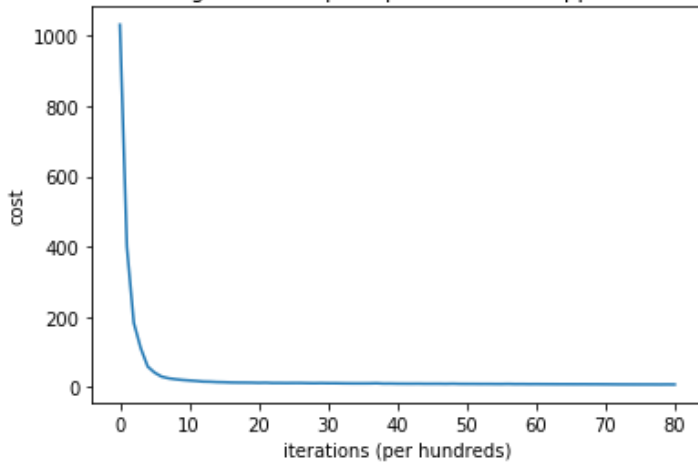
Programming Assignment 1

MLFNN with 2 hidden layers

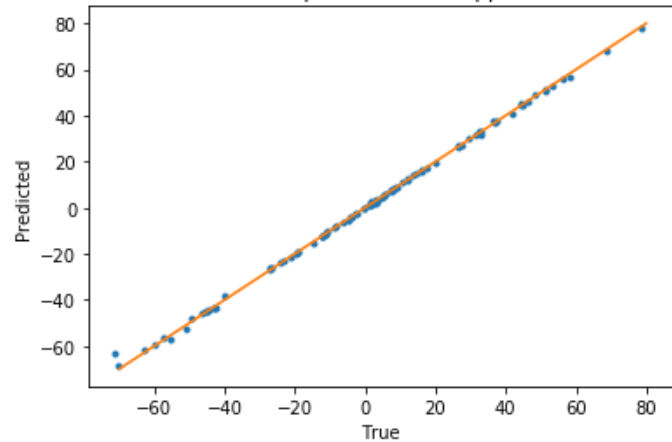
Function approximation task

Plots :

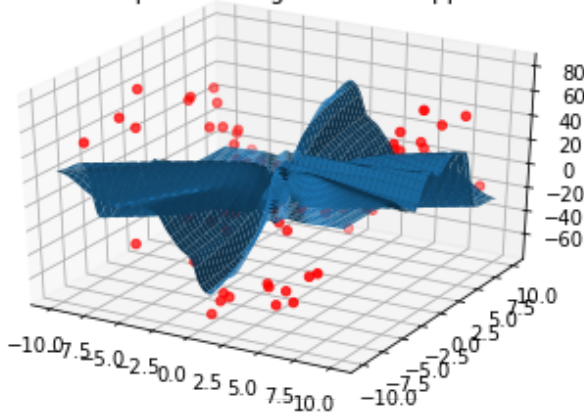
Avg. error vs Epoch plot - Function approx.



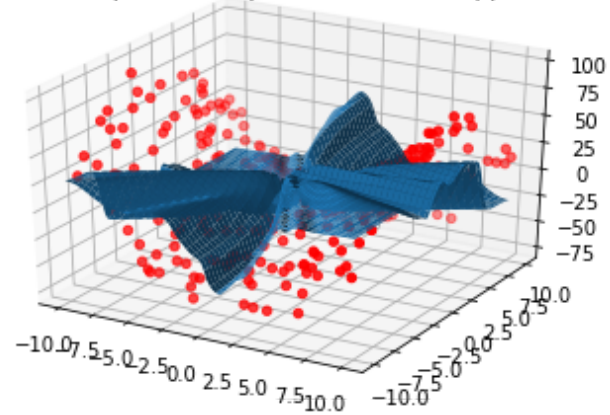
Scatter plot - Function approx.



Surface plot Training - Function approx.



Surface plot Development - Function approx.



Inference :

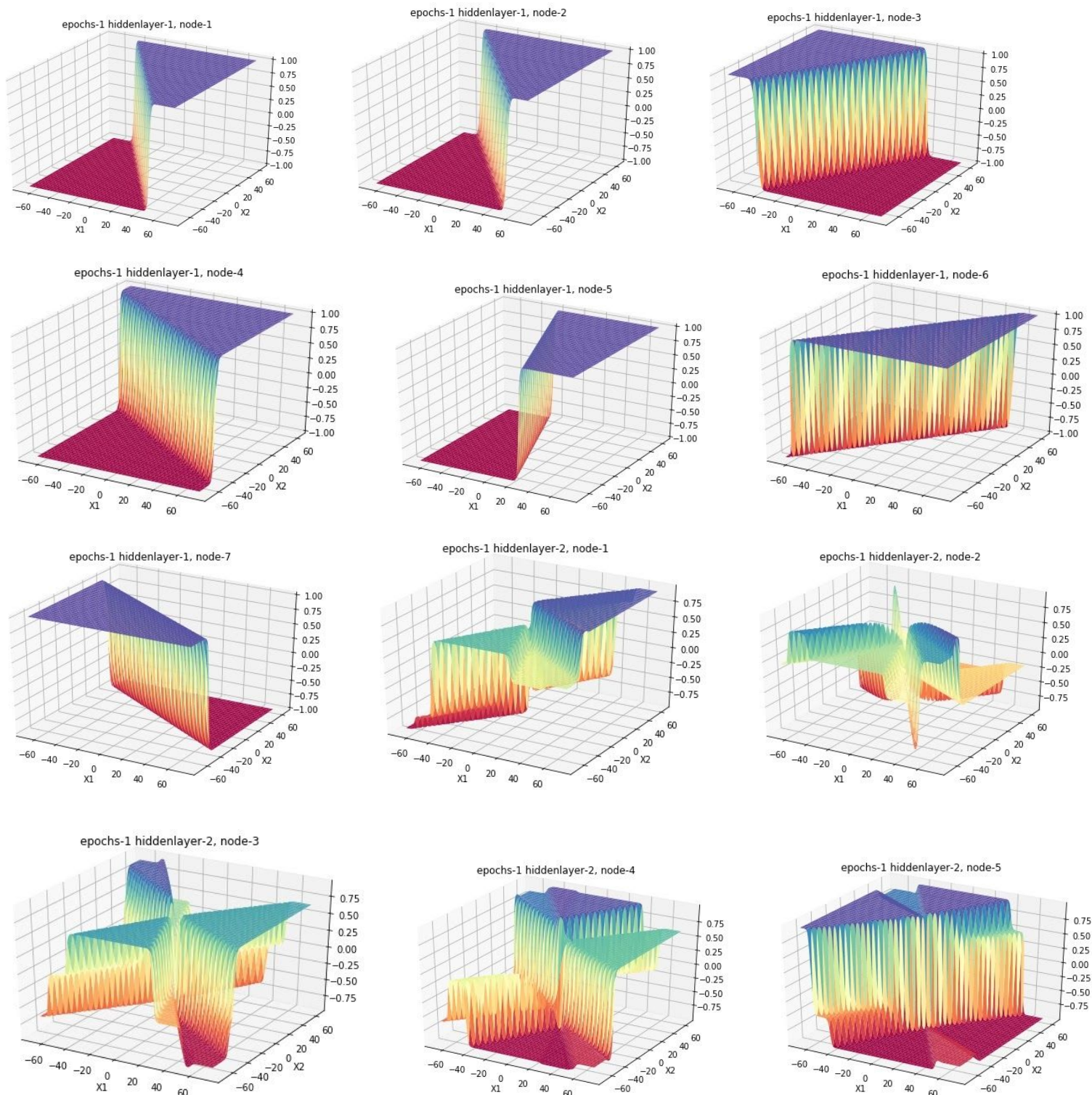
- Learning rate = 0.01, Delta cost = 10^{-6}
- Nodes in first hidden layer = 30, second hidden layer = 10

- RMS error for training is 1.01 , testing data is 11.82

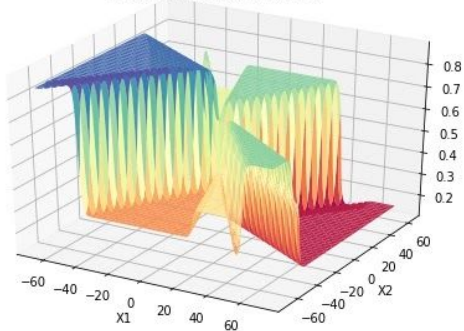
Classification task for 2-d data

Plots :

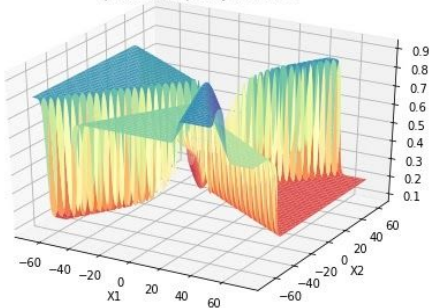
Epoch1



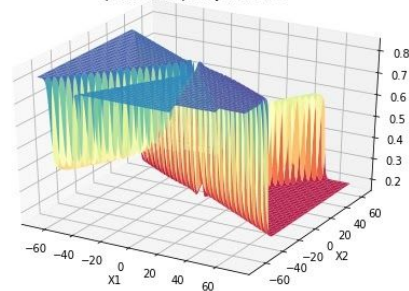
epochs-1 outputlayer, node-1



epochs-1 outputlayer, node-2

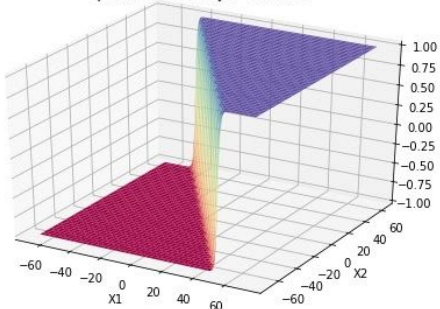


epochs-1 outputlayer, node-3

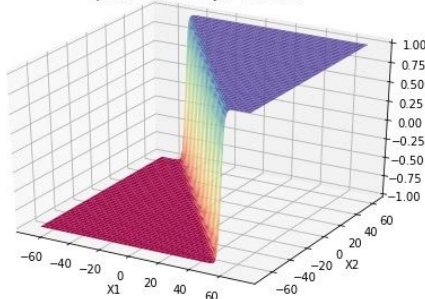


Epoch 2

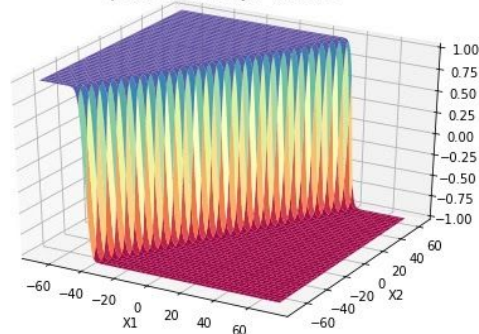
epochs-2 hiddenlayer-1, node-1



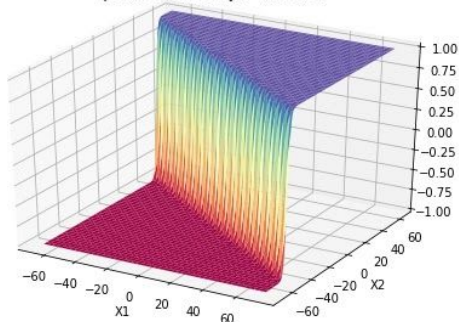
epochs-2 hiddenlayer-1, node-2



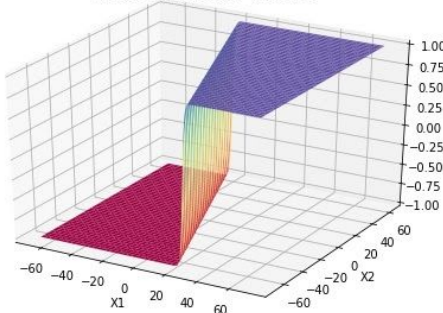
epochs-2 hiddenlayer-1, node-3



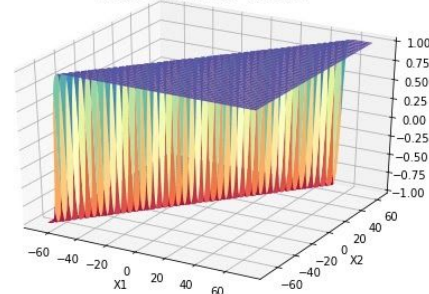
epochs-2 hiddenlayer-1, node-4



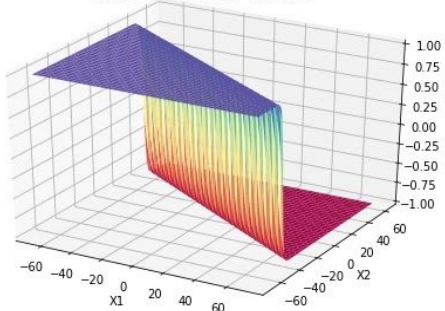
epochs-2 hiddenlayer-1, node-5



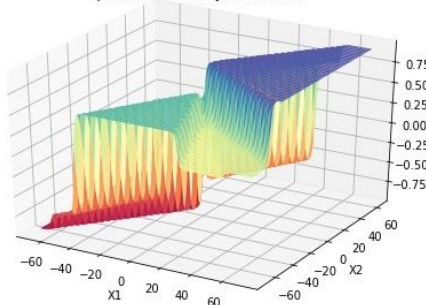
epochs-2 hiddenlayer-1, node-6



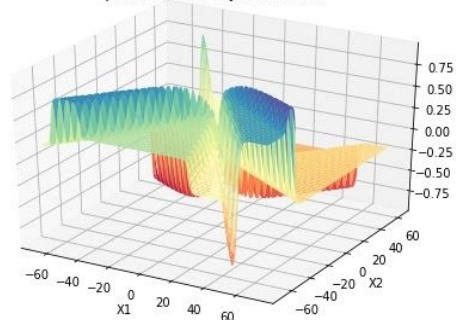
epochs-2 hiddenlayer-1, node-7



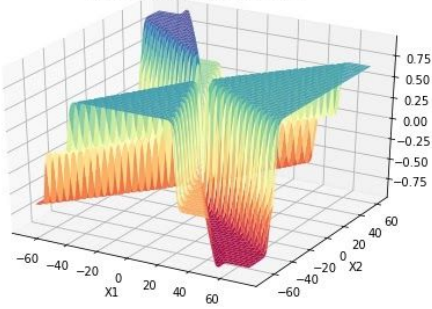
epochs-2 hiddenlayer-2, node-1



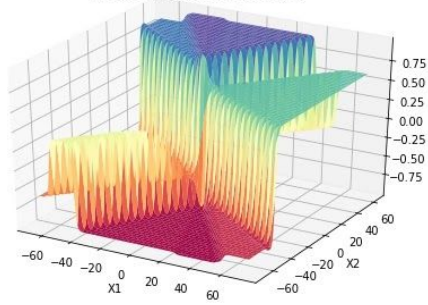
epochs-2 hiddenlayer-2, node-2



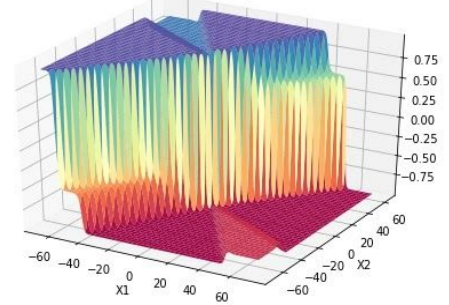
epochs-2 hiddenlayer-2, node-3



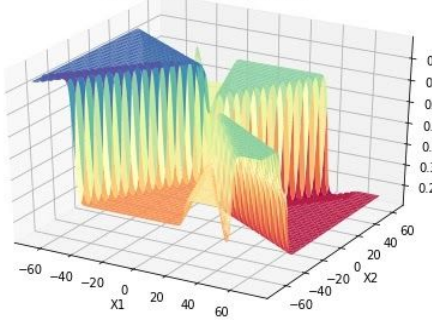
epochs-2 hiddenlayer-2, node-4



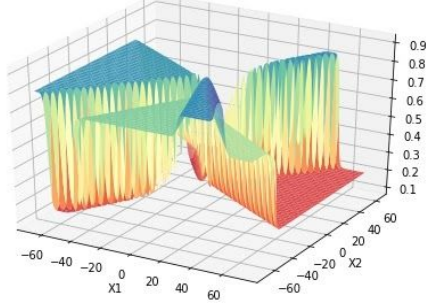
epochs-2 hiddenlayer-2, node-5



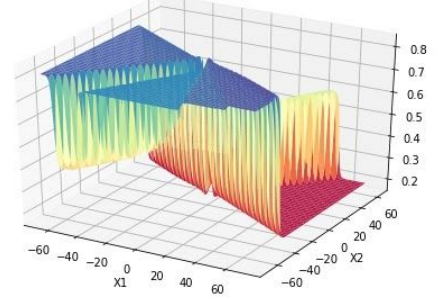
epochs-2 outputlayer, node-1



epochs-2 outputlayer, node-2

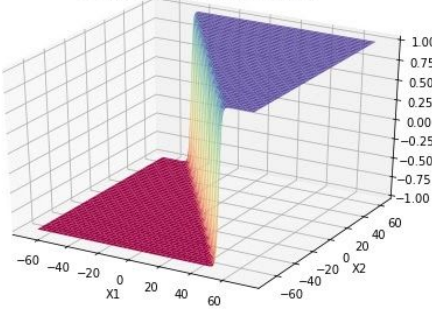


epochs-2 outputlayer, node-3

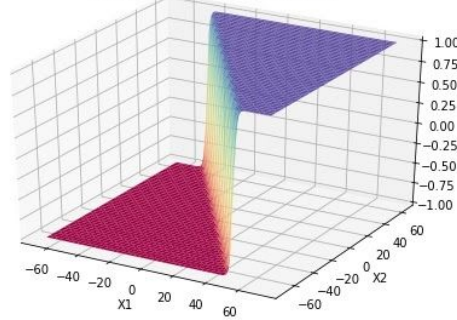


Epoch 10

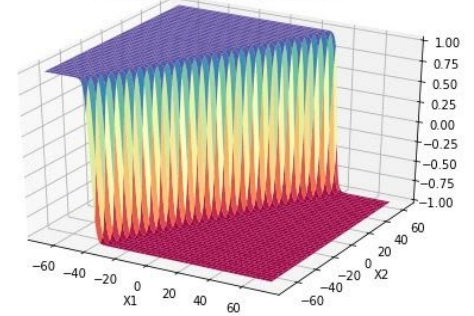
epochs-10 hiddenlayer-1, node-1



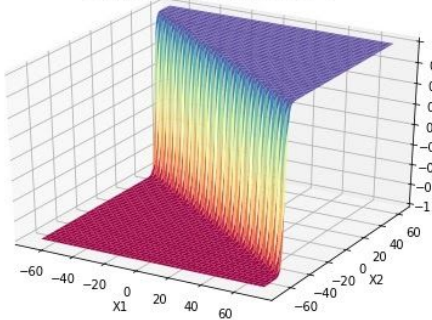
epochs-10 hiddenlayer-1, node-2



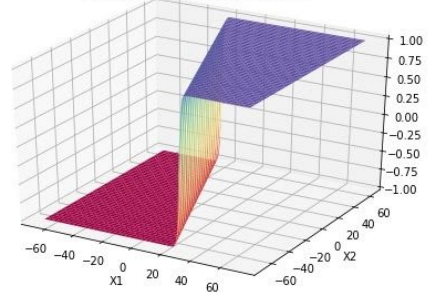
epochs-10 hiddenlayer-1, node-3



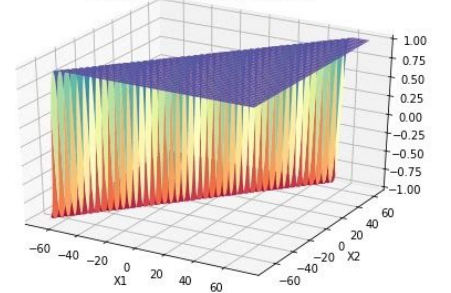
epochs-10 hiddenlayer-1, node-4



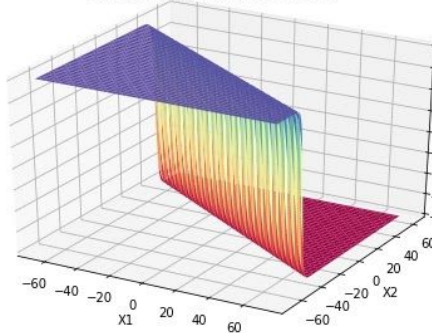
epochs-10 hiddenlayer-1, node-5



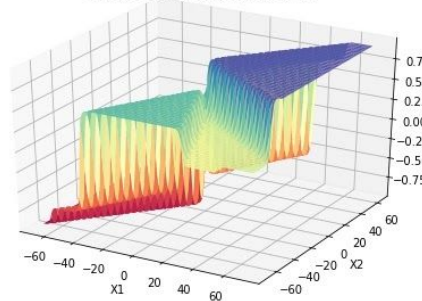
epochs-10 hiddenlayer-1, node-6



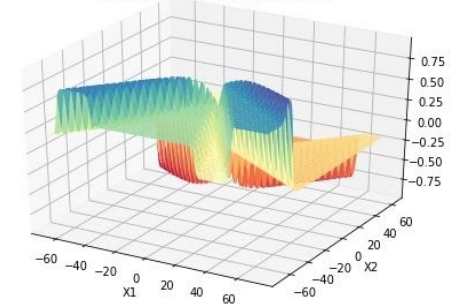
epochs-10 hiddenlayer-1, node-7



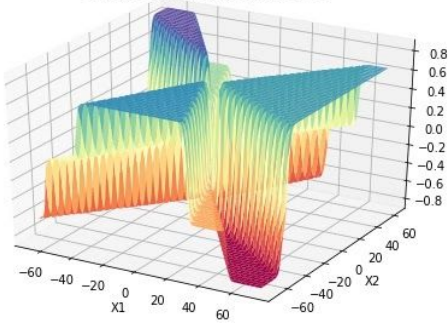
epochs-10 hiddenlayer-2, node-1



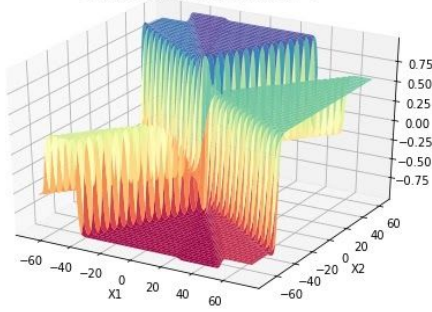
epochs-10 hiddenlayer-2, node-2



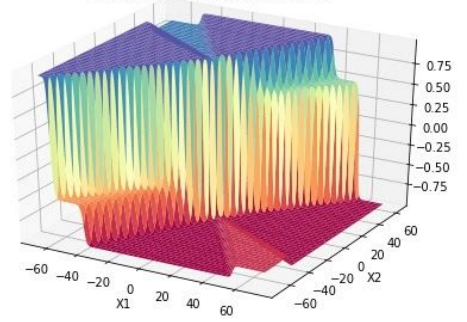
epochs-10 hiddenlayer-2, node-3



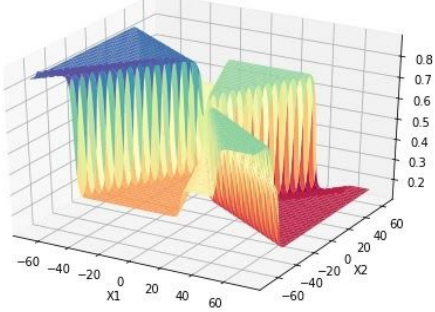
epochs-10 hiddenlayer-2, node-4



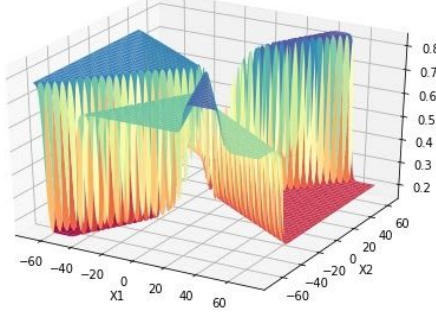
epochs-10 hiddenlayer-2, node-5



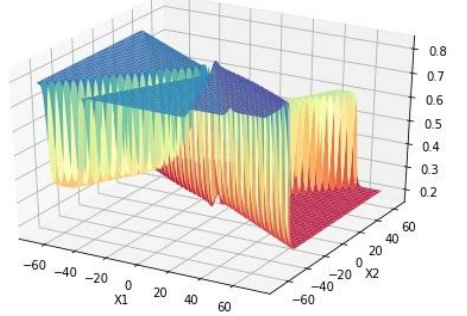
epochs-10 outputlayer, node-1



epochs-10 outputlayer, node-2

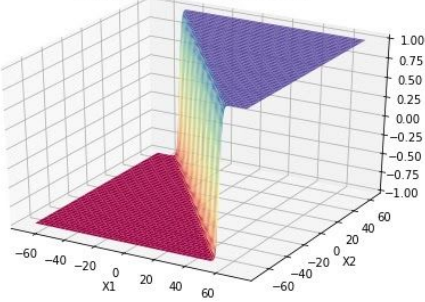


epochs-10 outputlayer, node-3

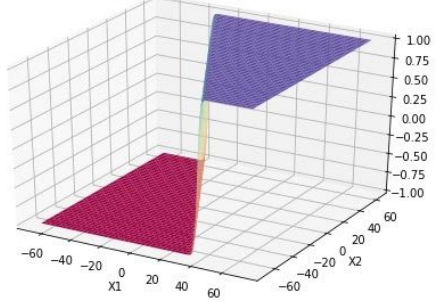


Epoch 50

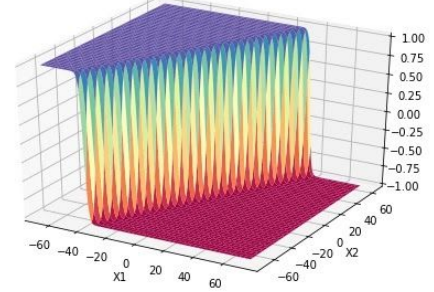
epochs-50 hiddenlayer-1, node-1



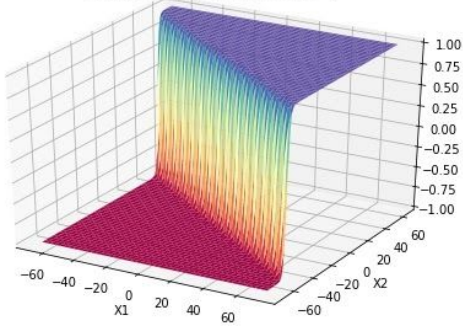
epochs-50 hiddenlayer-1, node-2



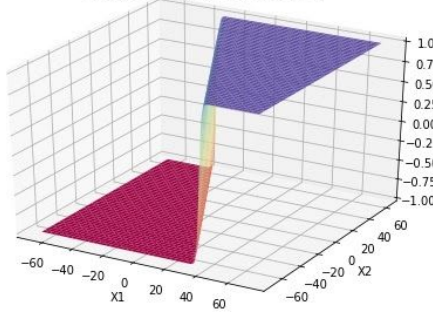
epochs-50 hiddenlayer-1, node-3



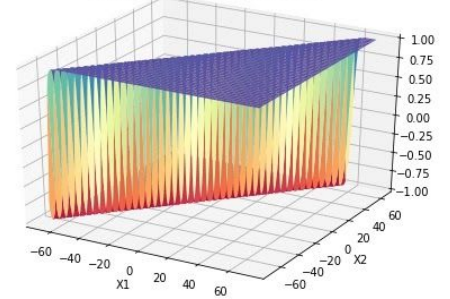
epochs-50 hiddenlayer-1, node-4



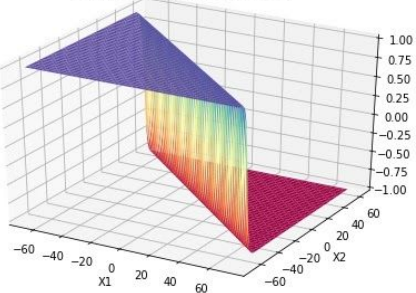
epochs-50 hiddenlayer-1, node-5



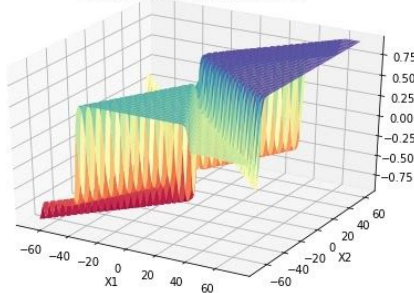
epochs-50 hiddenlayer-1, node-6



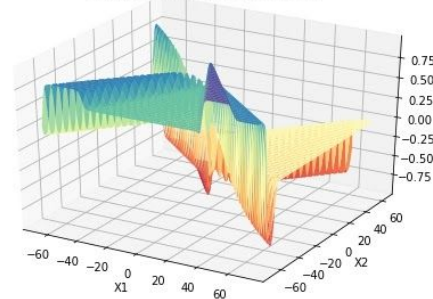
epochs-50 hiddenlayer-1, node-7



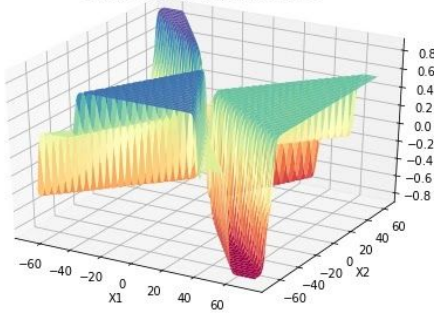
epochs-50 hiddenlayer-2, node-1



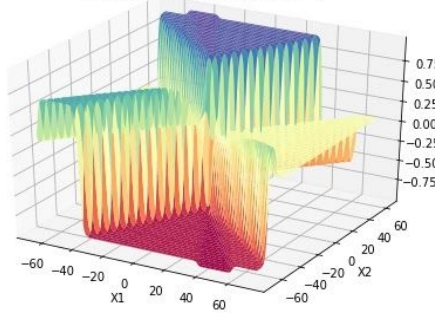
epochs-50 hiddenlayer-2, node-2



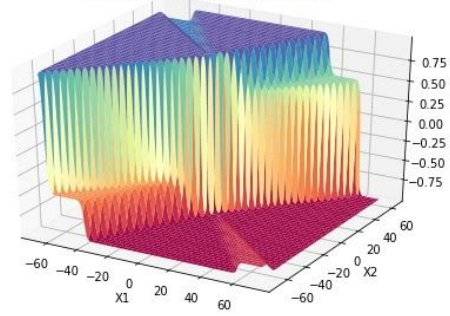
epochs-50 hiddenlayer-2, node-3



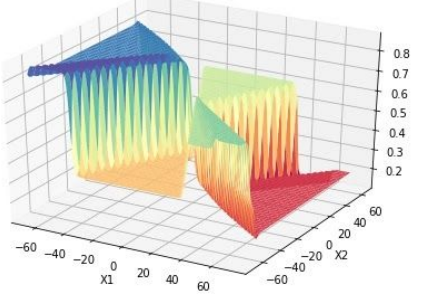
epochs-50 hiddenlayer-2, node-4



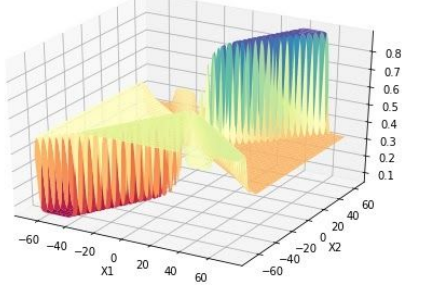
epochs-50 hiddenlayer-2, node-5



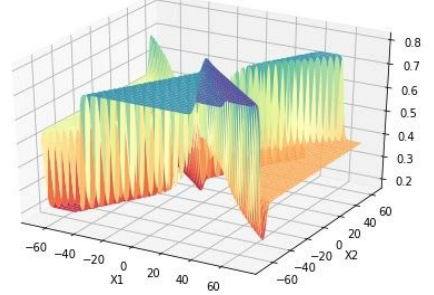
epochs-50 outputlayer, node-1



epochs-50 outputlayer, node-2

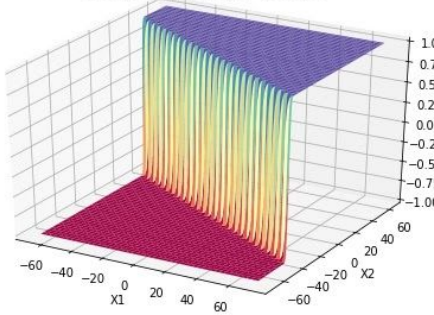


epochs-50 outputlayer, node-3

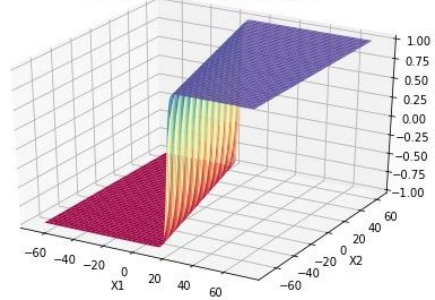


Termination

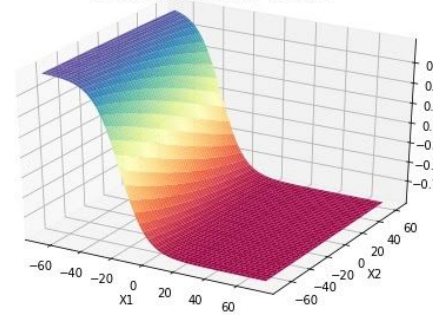
epochs-end hiddenlayer-1, node-1



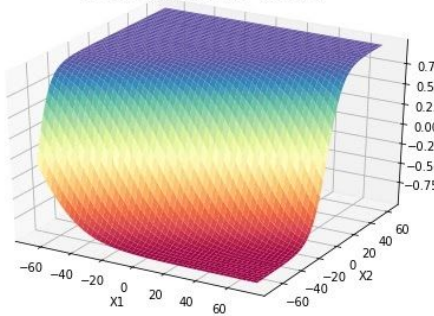
epochs-end hiddenlayer-1, node-2



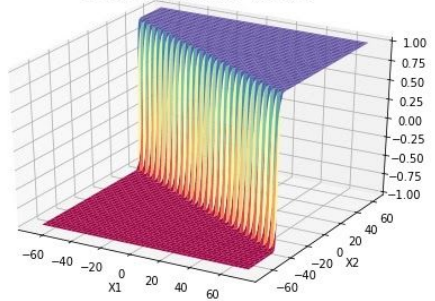
epochs-end hiddenlayer-1, node-3



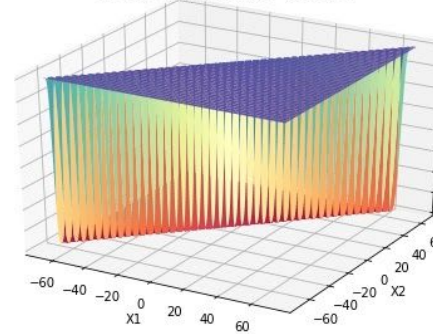
epochs-end hiddenlayer-1, node-4



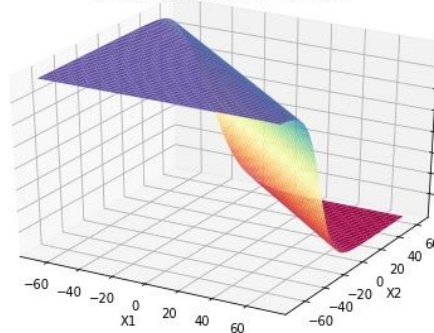
epochs-end hiddenlayer-1, node-5



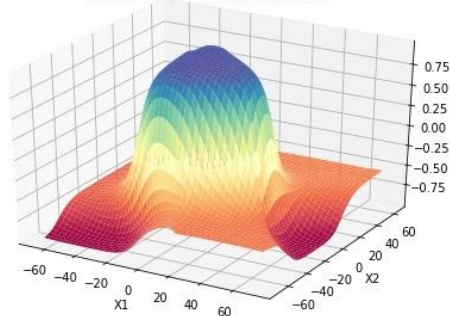
epochs-end hiddenlayer-1, node-6



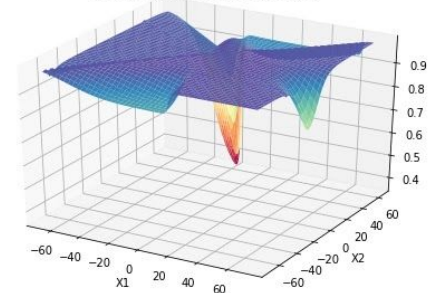
epochs-end hiddenlayer-1, node-7

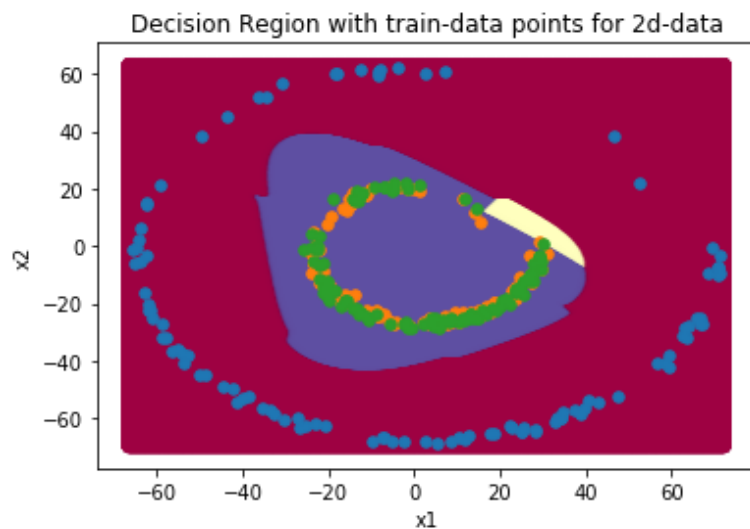
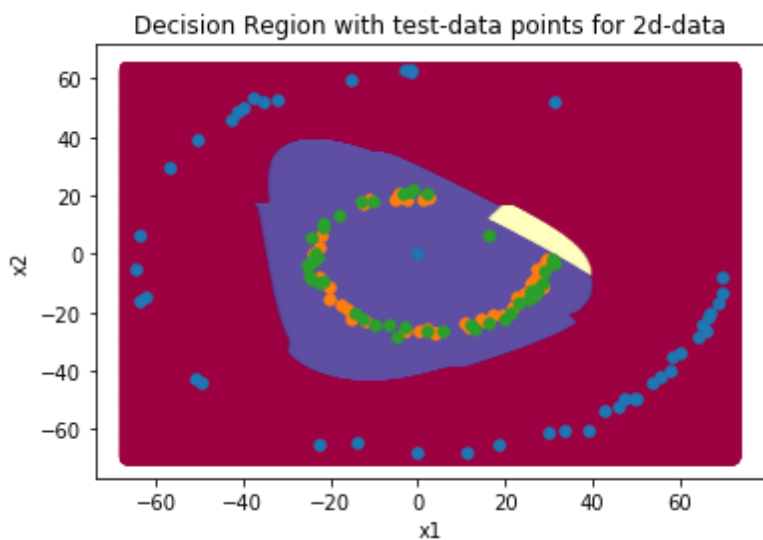
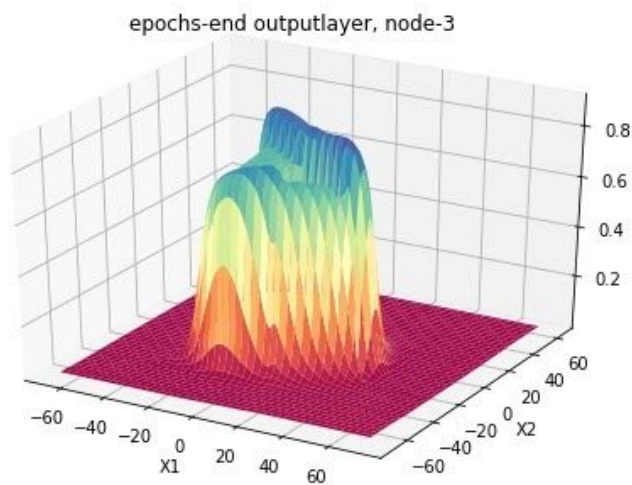
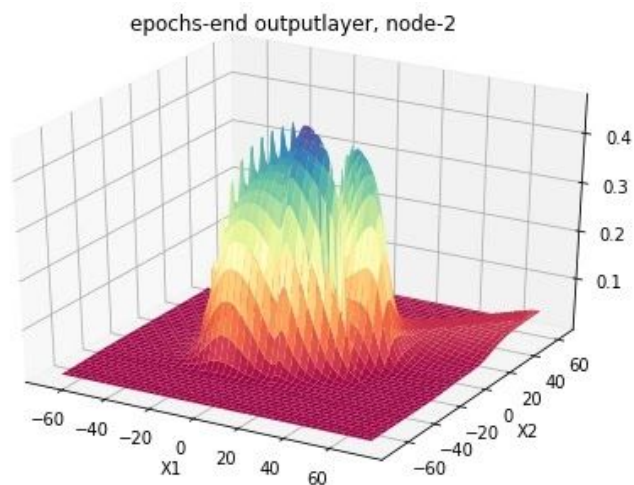
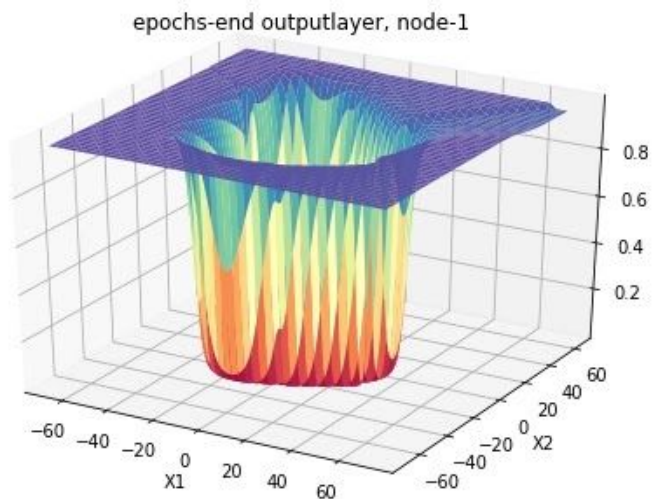
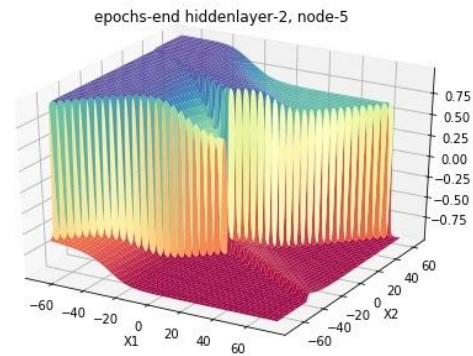
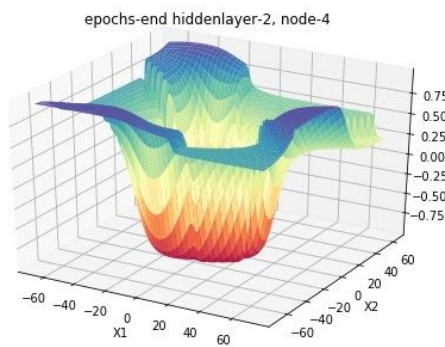
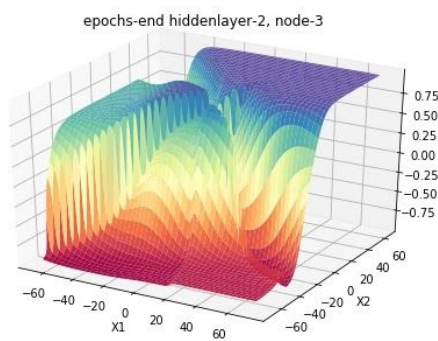


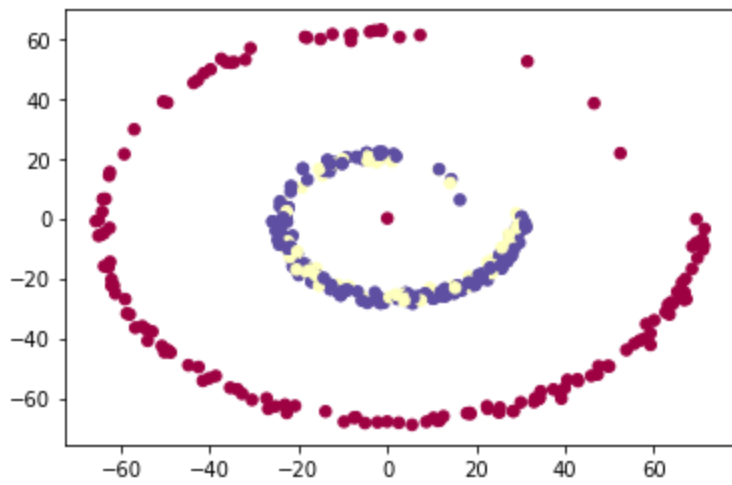
epochs-end hiddenlayer-2, node-1



epochs-end hiddenlayer-2, node-2







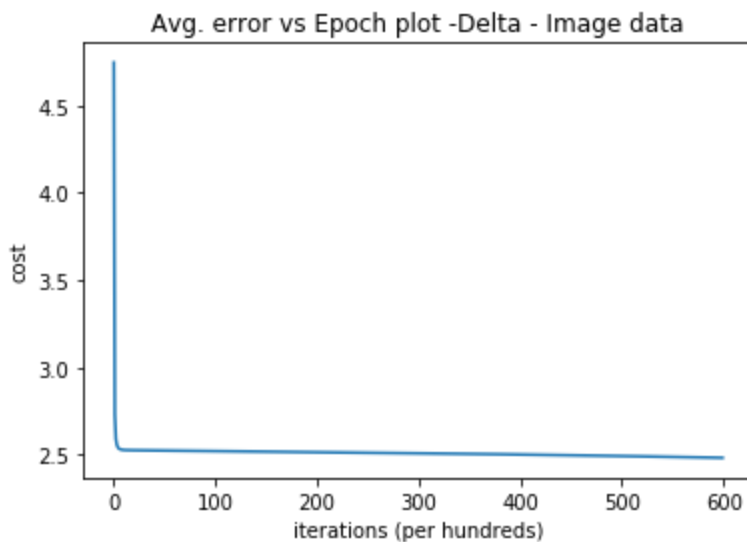
Inference :

- Learning parameter = 0.01 and delta cost = 10^{-4}
- Nodes in first hidden layer = 7 and second hidden layer = 5
- In given data 2 classes blue and yellow require more complex boundaries to separate them. A deep network will classify them more accurately.
- The accuracy is 83% on training data and 80% on test data.
- On increasing the number of nodes, the curve is getting over-fitted.

Classification task for image data

Delta :

Plots :



Confusion matrix for Training - Delta - Image Data

Predicted \ Actual	class A	class B	class C	class D	class E	sum_col
class A	37 3.08%	35 2.92%	53 4.42%	52 4.33%	59 4.92%	236 15.68% 84.32%
class B	20 1.67%	80 6.67%	70 5.83%	17 1.42%	53 4.42%	240 33.33% 66.67%
class C	26 2.17%	36 3.00%	110 9.17%	21 1.75%	44 3.67%	237 46.41% 53.59%
class D	26 2.17%	32 2.67%	35 2.92%	71 5.92%	76 6.33%	240 29.58% 70.42%
class E	10 0.83%	39 3.25%	37 3.08%	57 4.75%	104 8.67%	247 42.11% 57.89%
sum_col	119 31.09% 68.91%	222 36.04% 63.96%	305 36.07% 63.93%	218 32.57% 67.43%	336 30.95% 69.05%	1200 33.50% 66.50%

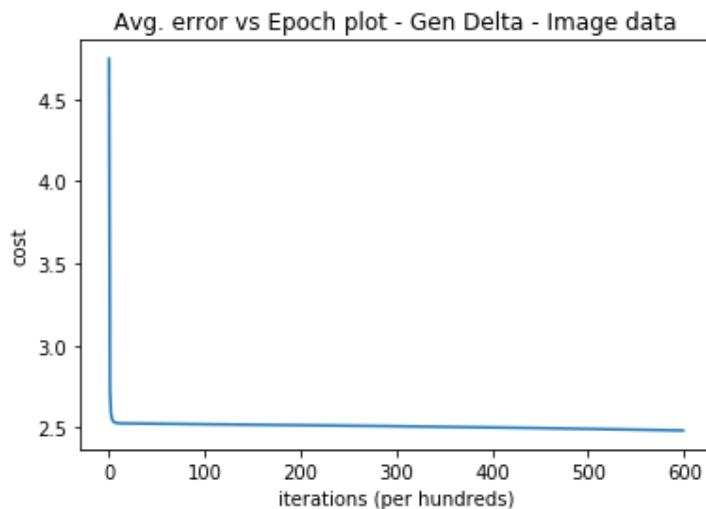
Confusion matrix for Development - Delta - Image Data

Predicted \ Actual	class A	class B	class C	class D	class E	sum_col
class A	5 1.67%	9 3.00%	8 2.67%	12 4.00%	30 10.00%	64 7.81% 92.19%
class B	6 2.00%	19 6.33%	13 4.33%	8 2.67%	14 4.67%	60 31.67% 68.33%
class C	4 1.33%	16 5.33%	23 7.67%	10 3.33%	10 3.33%	63 36.51% 63.49%
class D	8 2.67%	7 2.33%	14 4.67%	13 4.33%	18 6.00%	60 21.67% 78.33%
class E	2 0.67%	5 1.67%	6 2.00%	15 5.00%	25 8.33%	53 47.17% 52.83%
sum_col	25 20.00% 80.00%	56 33.93% 66.07%	64 35.94% 64.06%	58 22.41% 77.59%	97 25.77% 74.23%	300 28.33% 71.67%

Inference :

- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^{-8}
- Beta = 0.9 and gamma = 0.99
- On decreasing learning rate, we are ending up at minima which has greater avg cost.
- On increasing the learning rate, we are going up and down around the second minima.
- Here it's reaching local minima we can infer this from much lower cost obtained from adam.

Generalised Delta or classical momentum method :



Plots :

Confusion matrix for Training - gen-Delta - Image Data

Predicted	class A	37 3.08%	35 2.92%	53 4.42%	52 4.33%	59 4.92%	236 15.68% 84.32%
	class B	21 1.75%	80 6.67%	69 5.75%	17 1.42%	53 4.42%	240 33.33% 66.67%
	class C	26 2.17%	37 3.08%	109 9.08%	21 1.75%	44 3.67%	237 45.99% 54.01%
	class D	26 2.17%	32 2.67%	35 2.92%	71 5.92%	76 6.33%	240 29.58% 70.42%
	class E	10 0.83%	39 3.25%	36 3.00%	57 4.75%	105 8.75%	247 42.51% 57.49%
	sum_col	120 30.83% 69.17%	223 35.87% 64.13%	302 36.09% 63.91%	218 32.57% 67.43%	337 31.16% 68.84%	1200 33.50% 66.50%
		Actual					sum_lin

Confusion matrix for Development - gen-Delta - Image Data

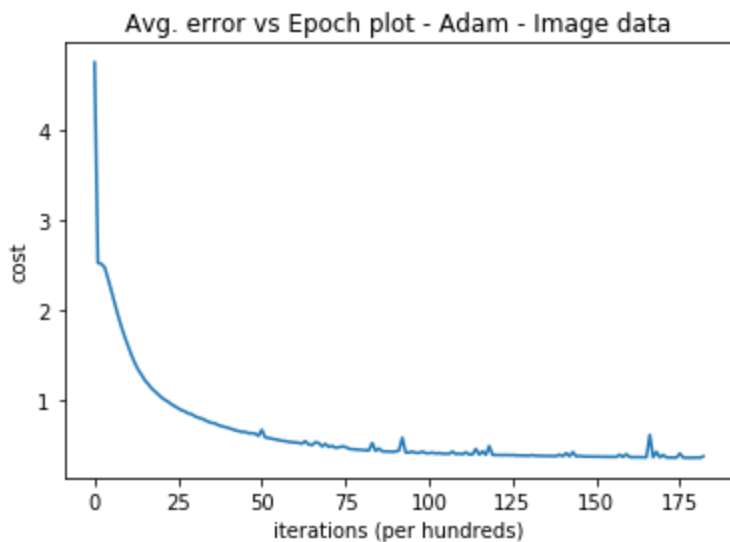
Predicted	class A	5 1.67%	9 3.00%	8 2.67%	12 4.00%	30 10.00%	64 7.81% 92.19%
	class B	6 2.00%	19 6.33%	13 4.33%	8 2.67%	14 4.67%	60 31.67% 68.33%
	class C	4 1.33%	16 5.33%	23 7.67%	10 3.33%	10 3.33%	63 36.51% 63.49%
	class D	8 2.67%	7 2.33%	14 4.67%	13 4.33%	18 6.00%	60 21.67% 78.33%
	class E	2 0.67%	5 1.67%	6 2.00%	15 5.00%	25 8.33%	53 47.17% 52.83%
	sum_col	25 20.00% 80.00%	56 33.93% 66.07%	64 35.94% 64.06%	58 22.41% 77.59%	97 25.77% 74.23%	300 28.33% 71.67%
		Actual					sum_lin

Inference :

- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^{-8}
- Similar to above we are reaching a local minimum.

Adam :

Plots :



Confusion matrix for Training - Adam - Image Data

Predicted	class A	class B	class C	class D	class E	sum_col
class A	236 19.67%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	236 100% 0.00%
class B	0 0.0%	240 20.00%	0 0.0%	0 0.0%	0 0.0%	240 100% 0.00%
class C	0 0.0%	0 0.0%	237 19.75%	0 0.0%	0 0.0%	237 100% 0.00%
class D	0 0.0%	0 0.0%	0 0.0%	240 20.00%	0 0.0%	240 100% 0.00%
class E	0 0.0%	0 0.0%	0 0.0%	0 0.0%	247 20.58%	247 100% 0.00%
sum_col	236 100% 0.00%	240 100% 0.00%	237 100% 0.00%	240 100% 0.00%	247 100% 0.00%	1200 100% 0.00%
	class A	class B	class C	class D	class E	sum_lin

Actual

Confusion matrix for Development - Adam - Image Data

Predicted	class A	class B	class C	class D	class E	sum_col
class A	10 3.33%	16 5.33%	10 3.33%	13 4.33%	15 5.00%	64 15.62% 84.38%
class B	13 4.33%	11 3.67%	10 3.33%	12 4.00%	14 4.67%	60 18.33% 81.67%
class C	14 4.67%	20 6.67%	12 4.00%	11 3.67%	6 2.00%	63 19.05% 80.95%
class D	6 2.00%	14 4.67%	18 6.00%	12 4.00%	10 3.33%	60 20.00% 80.00%
class E	10 3.33%	6 2.00%	5 1.67%	13 4.33%	19 6.33%	53 35.85% 64.15%
sum_col	53 18.87% 81.13%	67 16.42% 83.58%	55 21.82% 78.18%	61 19.67% 80.33%	64 29.69% 70.31%	300 21.33% 78.67%
	class A	class B	class C	class D	class E	sum_lin

Actual

Inference :

- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^{-8}
- This algorithm achieves good results faster than the above ones.
- We can observe 100 percent accuracy on training data. This is the case of overfitting and that is why we are observing lower development results.
- We know that above algorithms converged at a local minimum of 2.5 referring to that we can observe a dip in the graph.