ANN - Lab 1

Tianxiao Zhao
Junxun Luo
Feiyang Liu

Jan 25th, 2017

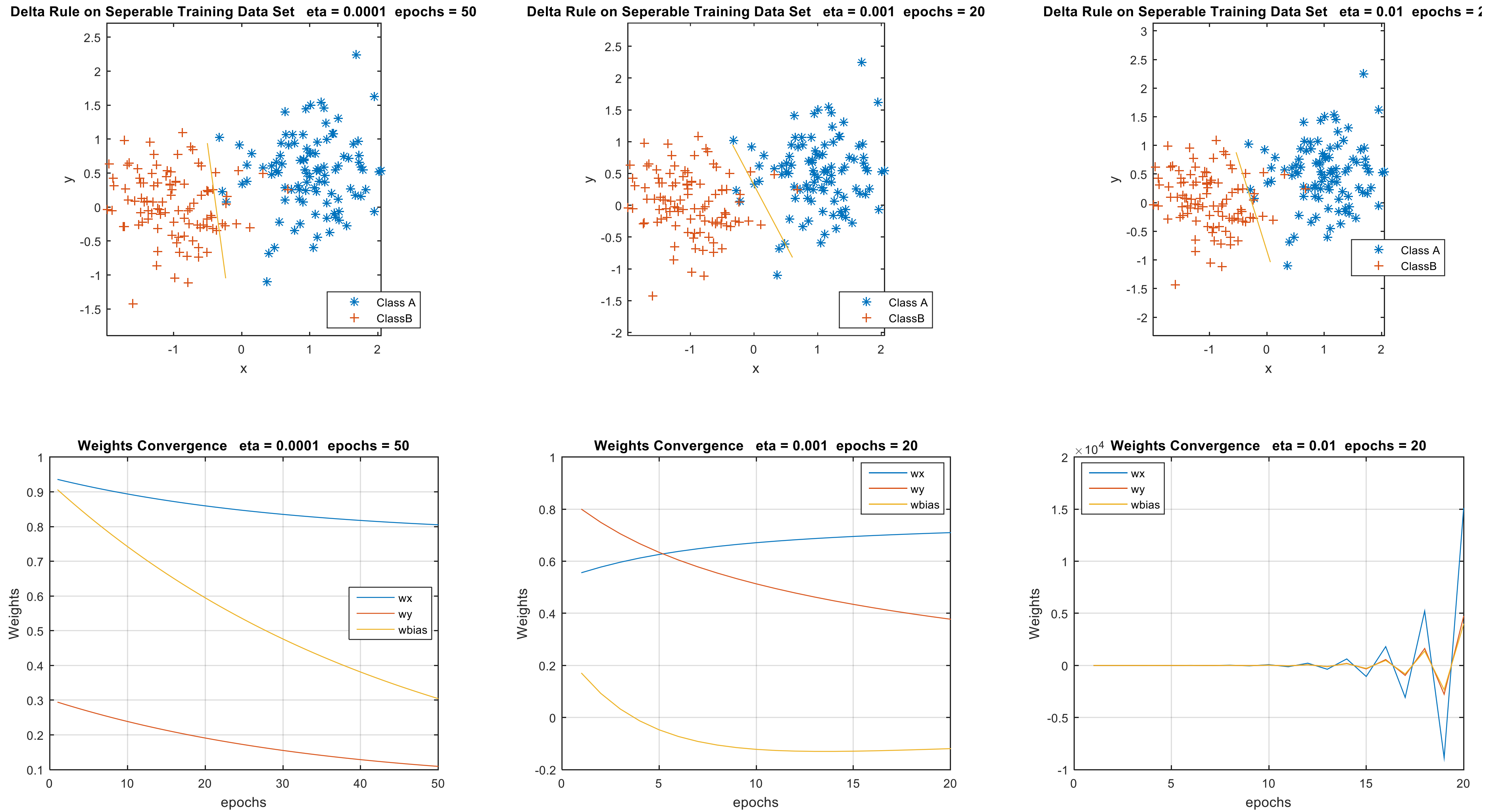
The Lab is about

- Construting feed forward networks
- Training with error based learning methods
- Its applications

Main Points

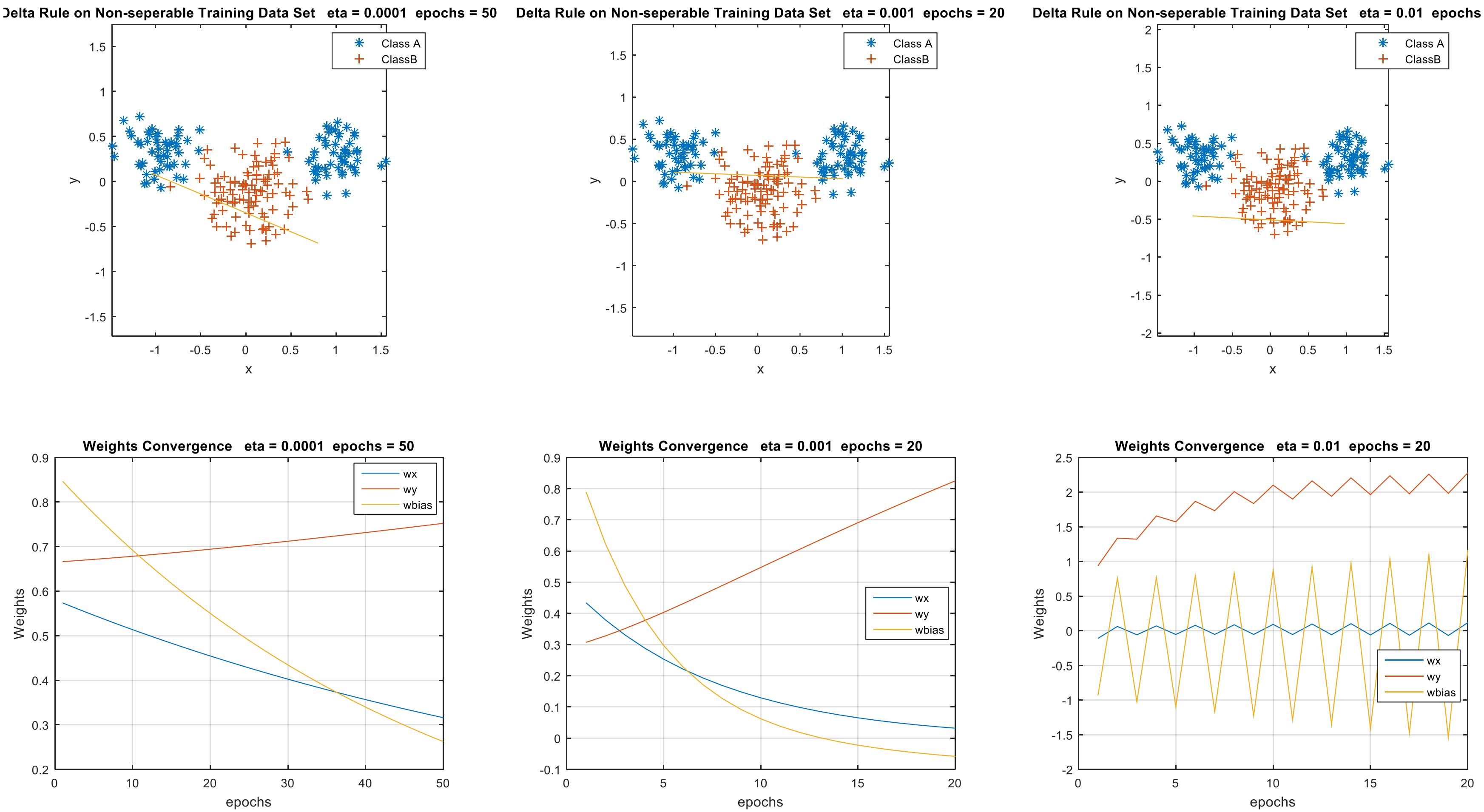
- Feed forward networks
- Classification
  - One layer perceptron
  - Two layer perceptron
- Function approximation
- Generalization

One Layer Delta-Rule Experiments on Seperable Training Data Set



- Eta too small ---> Slow convergence
- Eta too large ---> Weights fluctuation
- Need to try to find an appropriate eta by experiments

One Layer Delta-Rule Experiments on Non-seperable Training Data Set

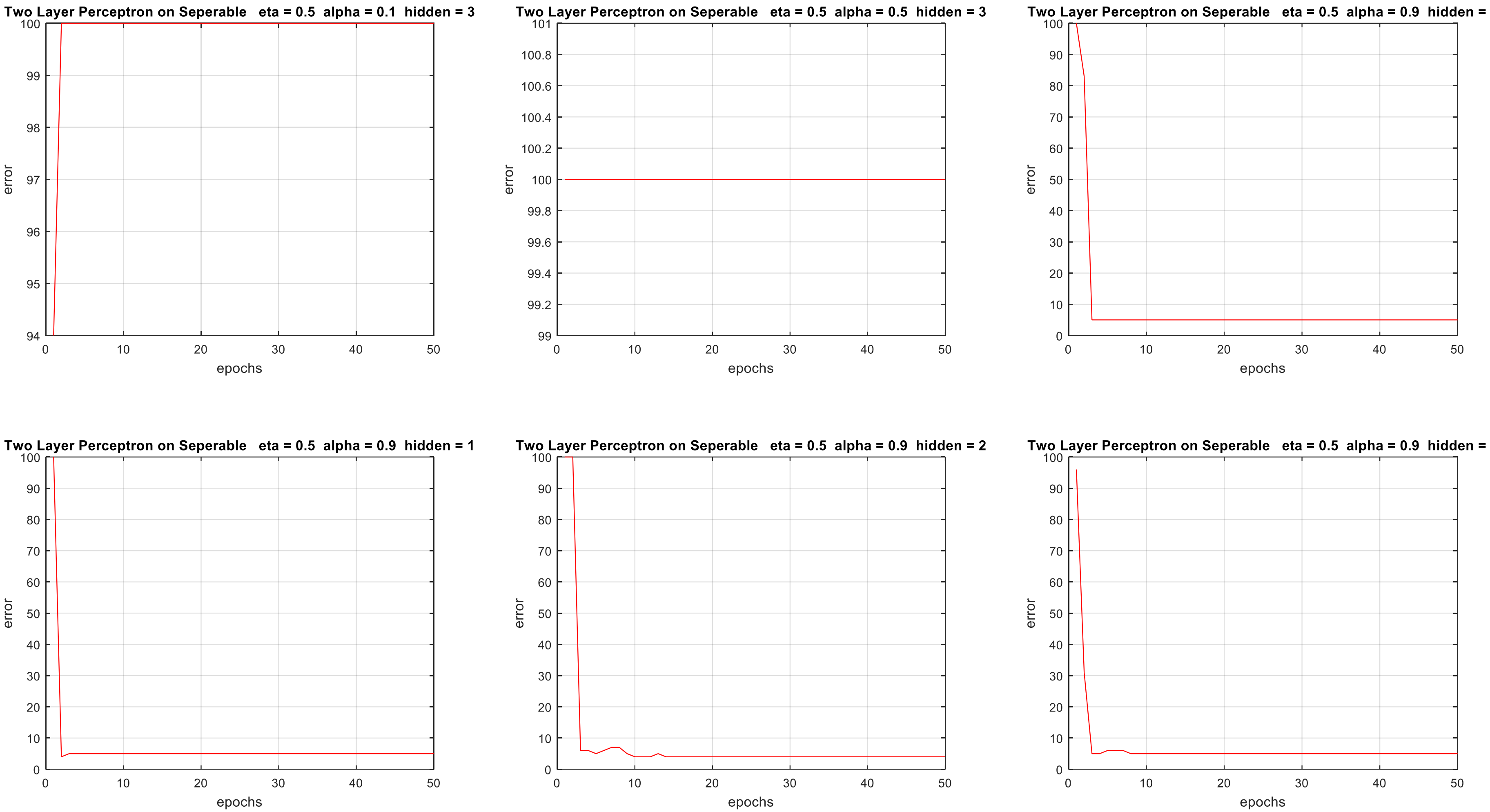


- Errors increase significantly compared to the one on seperable training data set

Again:

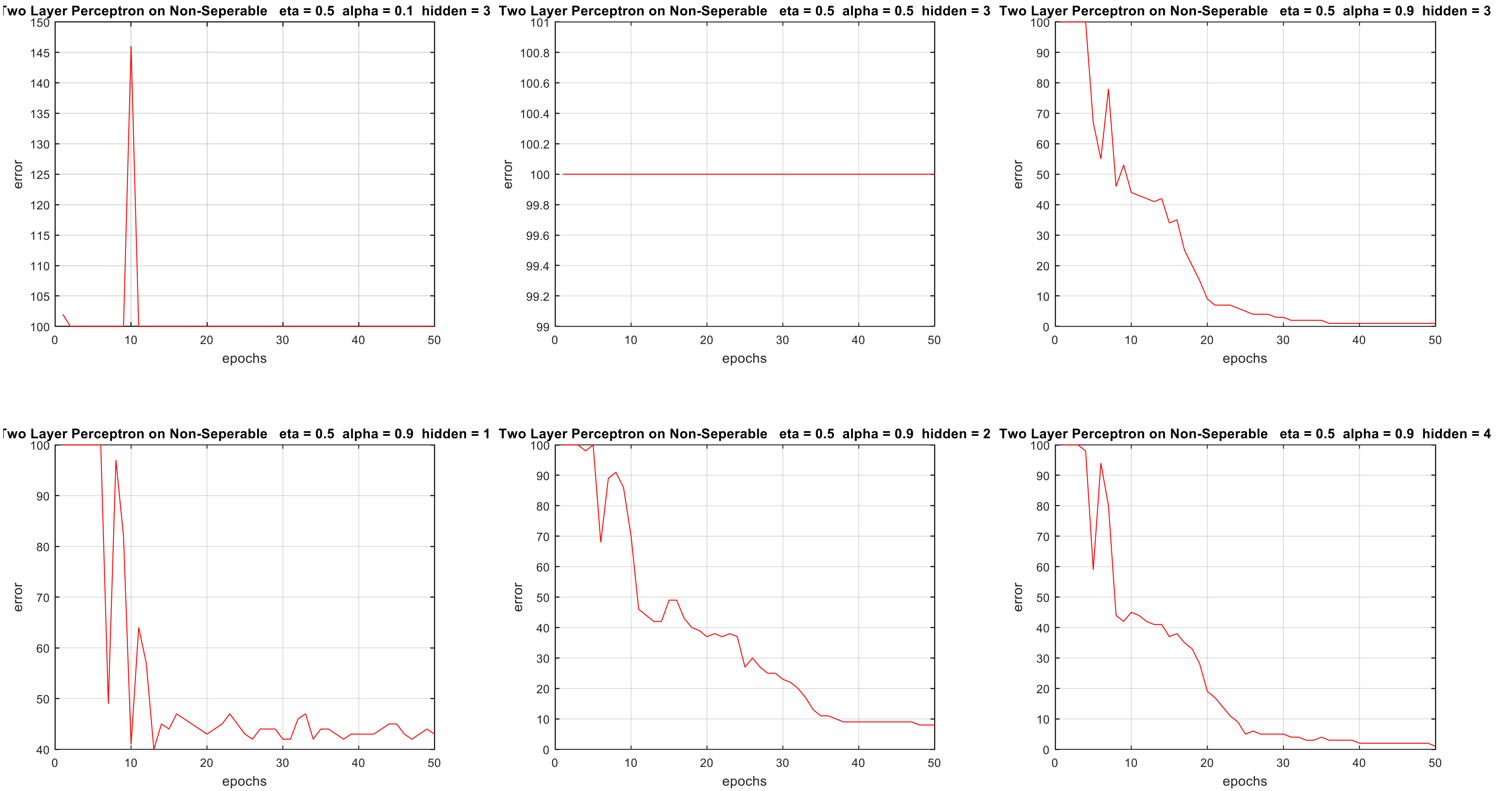
- Eta too small ---> Slow convergence
- Eta too large ---> Weights fluctuation
- Need to try to find an appropriate eta by experiments

Two Layer Delta-Rule Experiments on Seperable Training Data Set



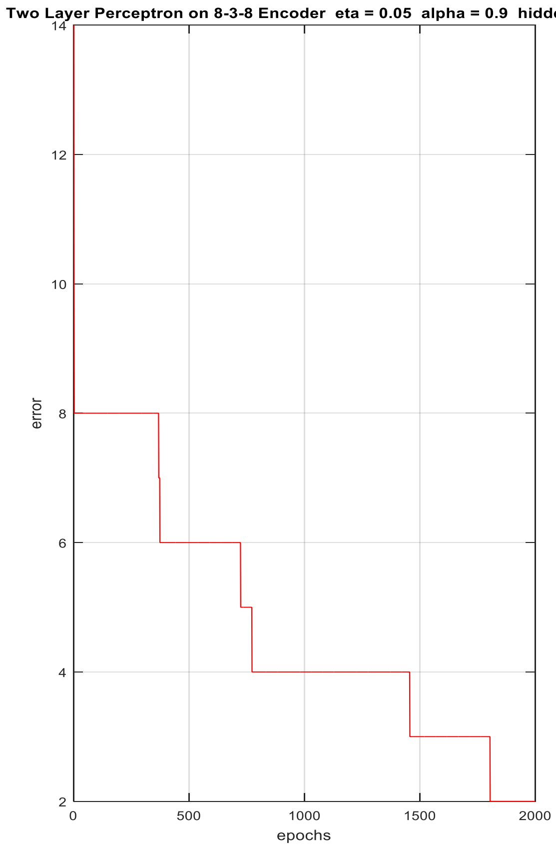
- Alpha increased ---> Convergence of higher probability since weights changes smoother
- To Seperable training data, only one node in the hidden layer performs well
- More than one node in the hidden layer does NOT help significantly

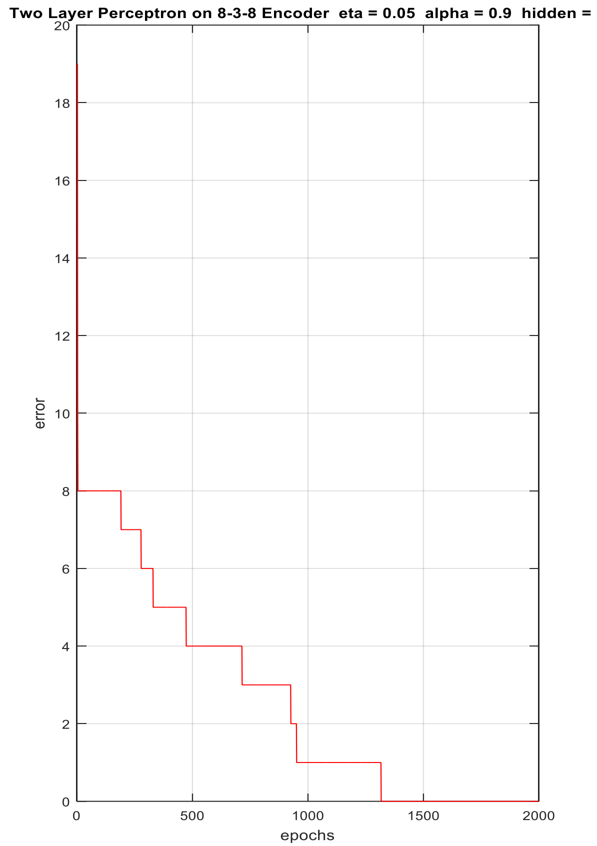
Two Layer Delta-Rule Experiments on Non-Seperable Training Data Set

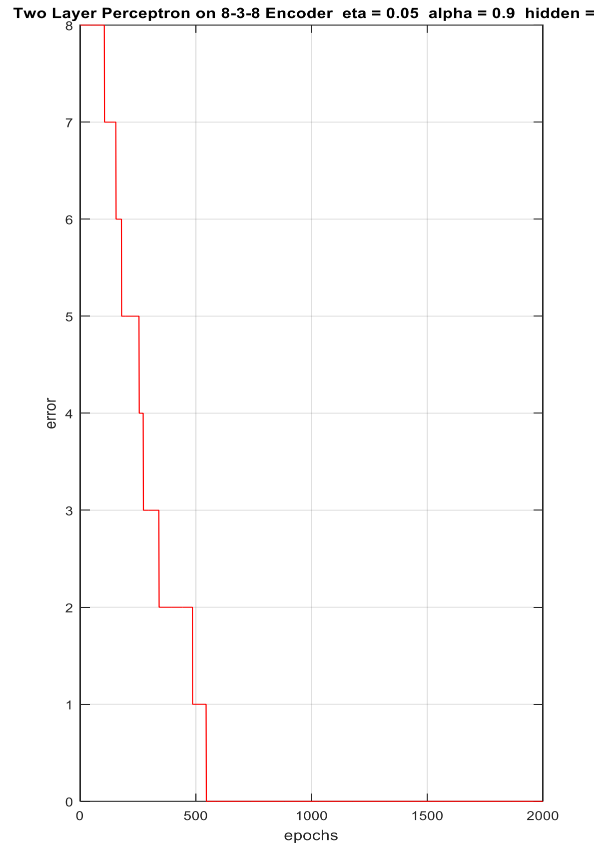


- Again: Alpha increased ---> Convergence of higher probability since weights changes smoother
- To Non-Seperable training data, only one node in the hidden layer performs bad because the training data set is NOT linearly seperable
- More than one node in the hidden layer helps significantly

Two Layer Delta-Rule Experiments on 8-3-8 Encoder



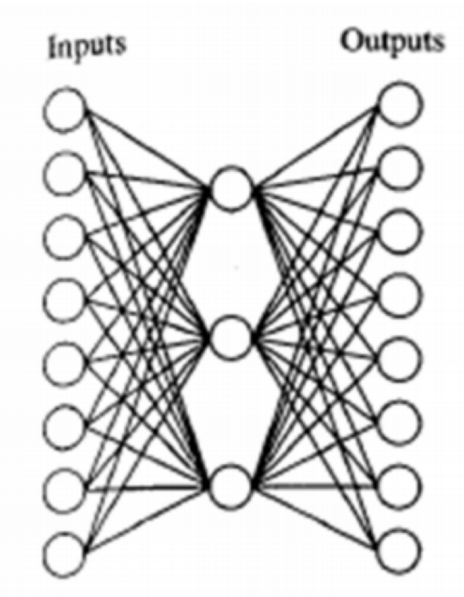




- 2 Hidden Layers ---> Not enough to reach 100% correctness
- 3 Hidden Layers ---> Can reach 100% correctness
- 4 Hidden Layers ---> Can reach 100% correctness, faster convergence than the 3 nodes in the hidden layer, but lower compression rate because of redundent encoding

Two Layer Delta-Rule Experiments on 8-3-8 Encoder

8-3-8 Binary Encoder -Decoder

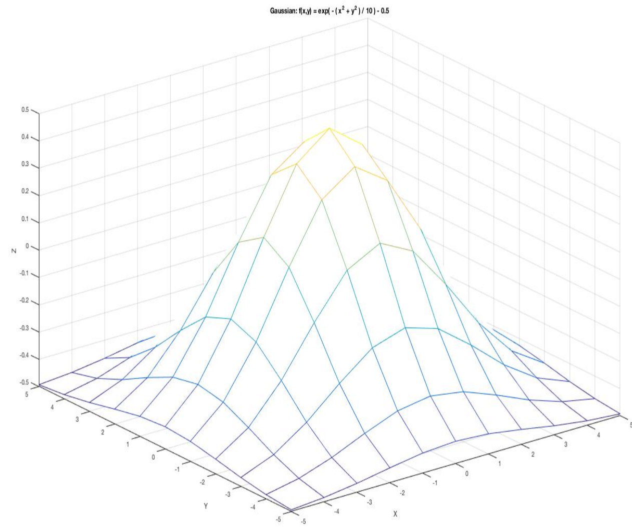


Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

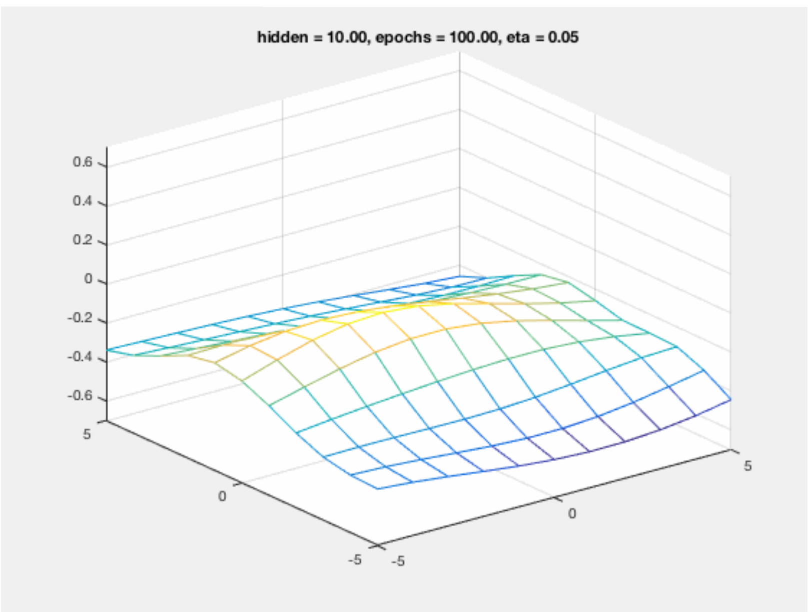
Taken from http://web.cs.hacettepe.edu.tr/~ilyas/Courses/BIL712/lec03-NeuralNetwork.pdf p34

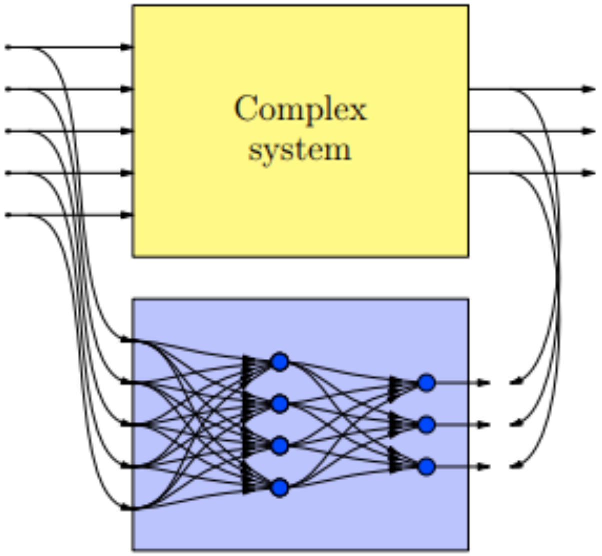
After 5000 training epochs, the three hidden unit values encode the eight distinct inputs using the encoding shown on the right.

Gaussian Function Approximation

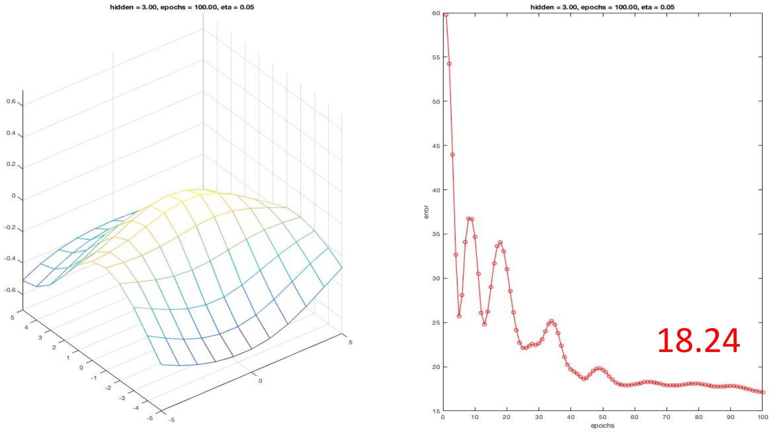


$$f(x, y) = e^{-(x^2+y^2)/10} - 0.5$$

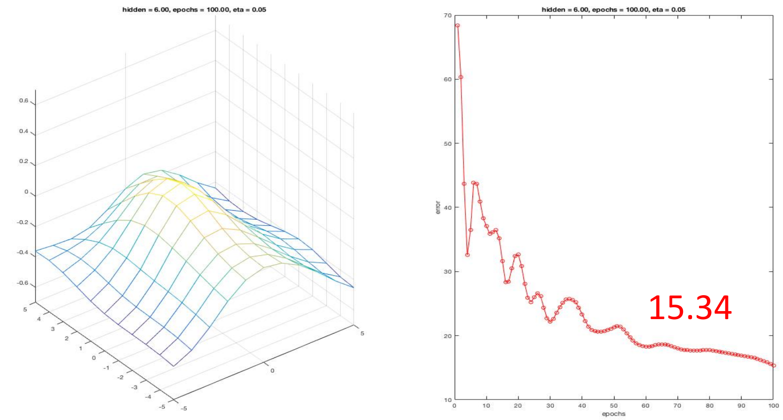




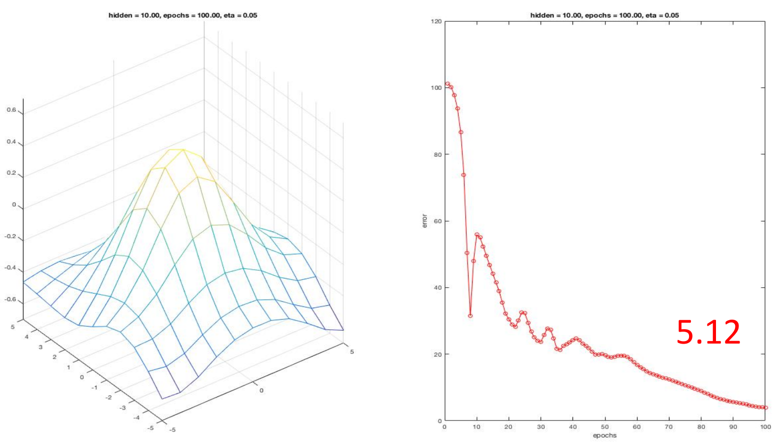
Gaussian Function Approximation



hidden = 3



hidden = 6



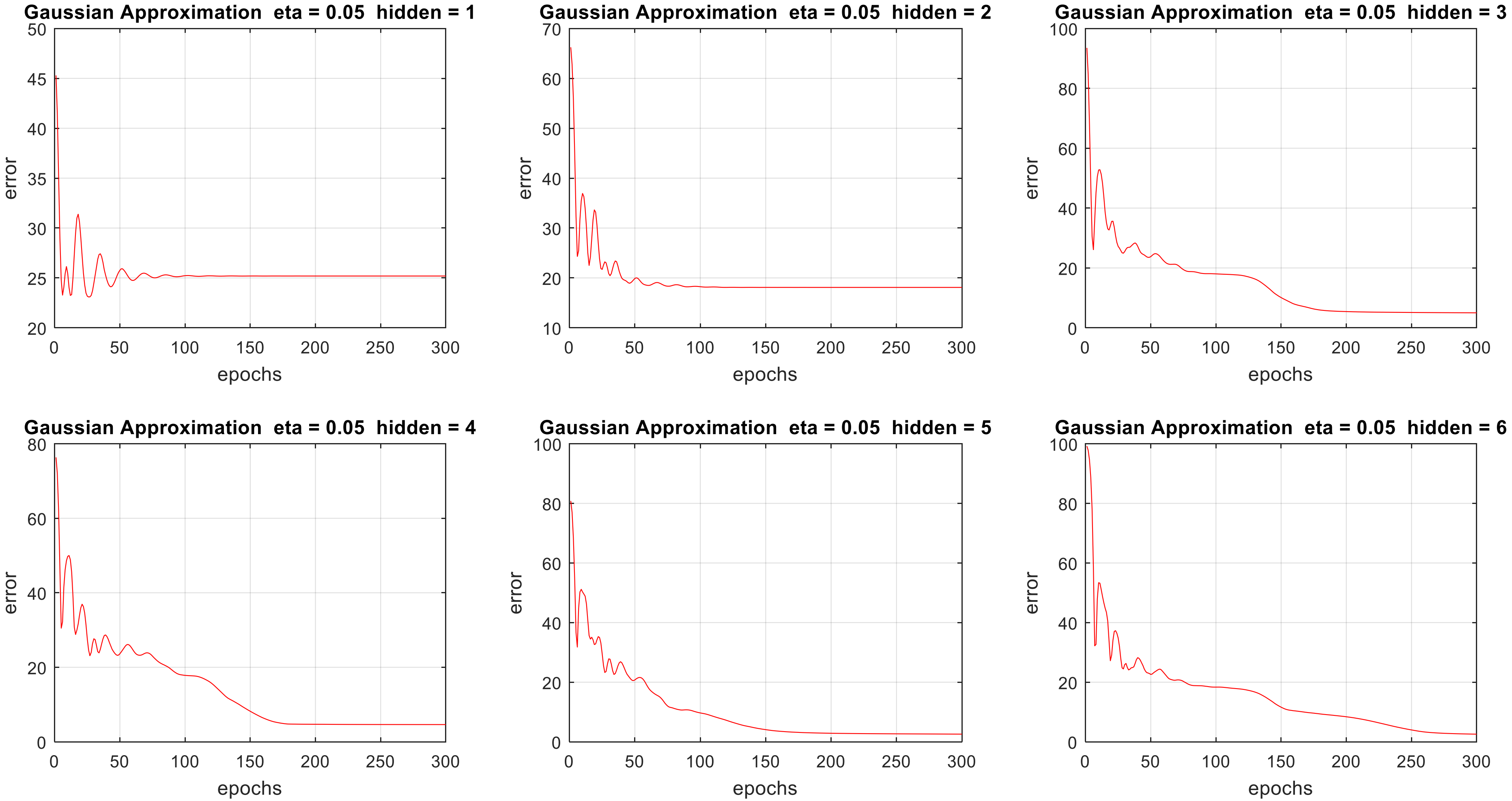
hidden = 10

Conclusion:

- More nodes in hidden layer, better performance on function approximation
- Too much nodes in hidden layer may cause overfitting problem when working on system identification

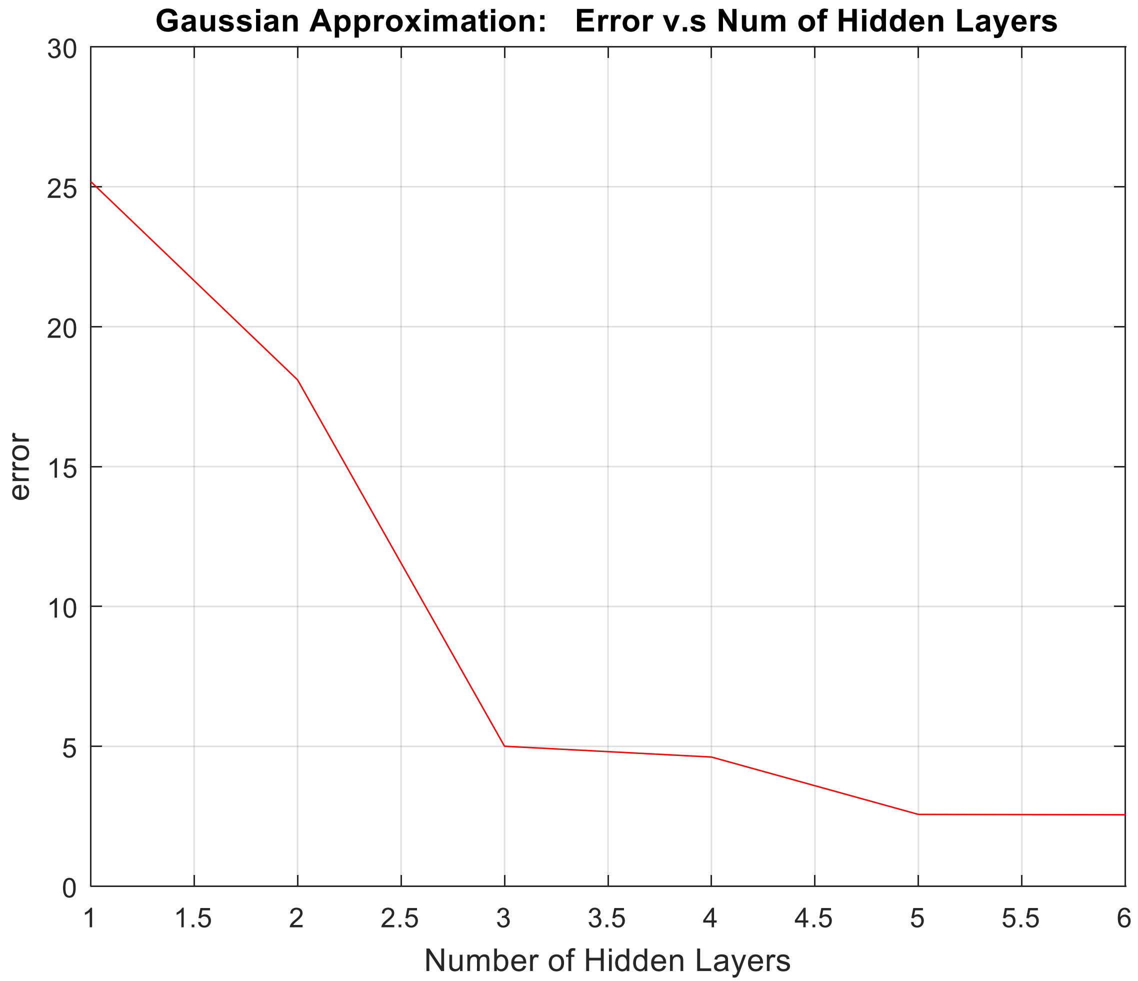
(epochs = 100, eta = 0.05)

Gaussian Function Approximation



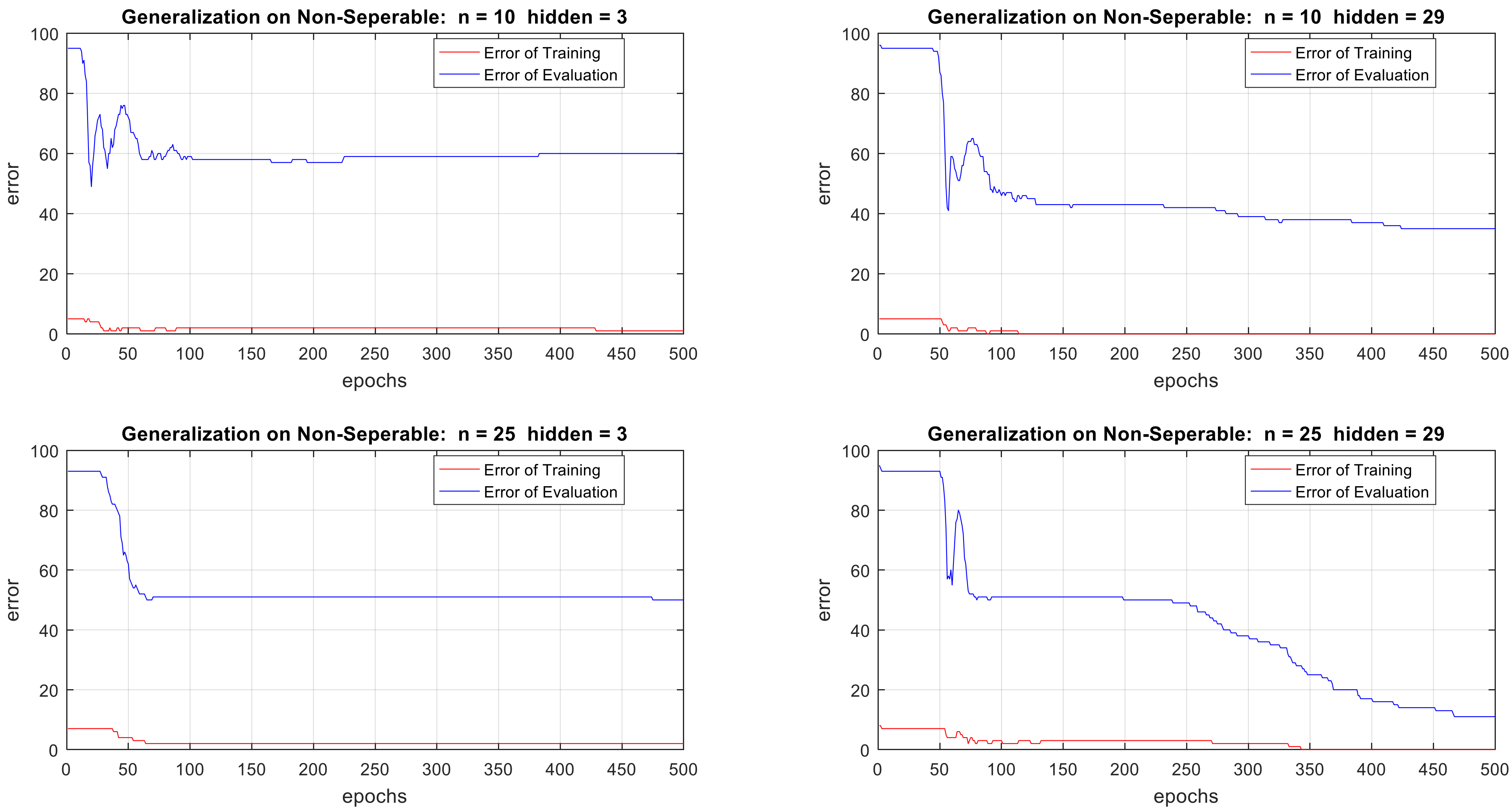
- Observation: More nodes in the hidden layer ---> Smaller the error

Gaussian Function Approximation



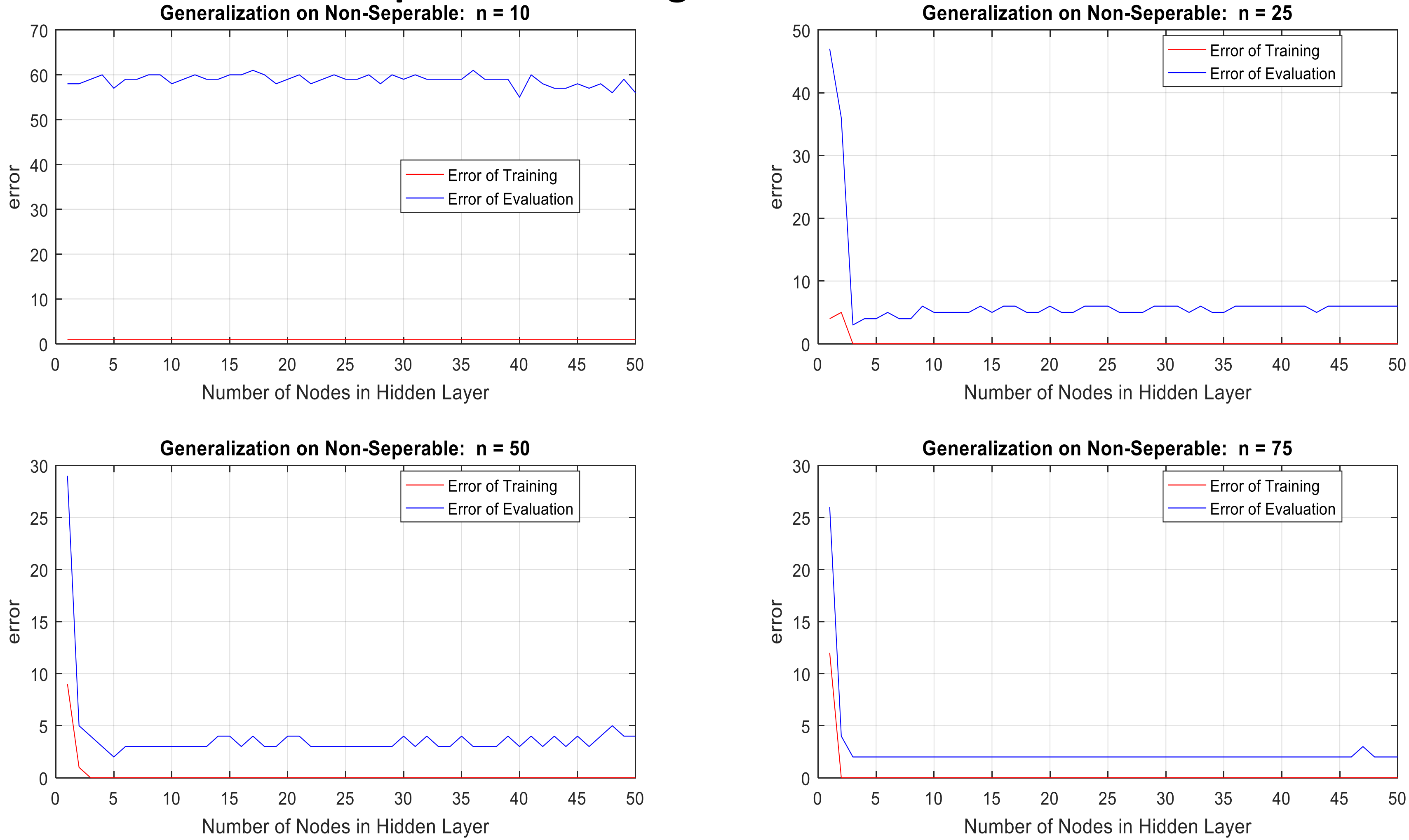
- As Hidden Layers increase beyond 3 ---> Error decreases NOT so significantly as before
- 3 Hidden Layers might be the best number for Gaussian function approximation

Generalization on Non-Seperable Training Data Set



- The figures above show the generalization process over epochs

Generalization on Non-Seperable Training Data Set



- More training data ---> More generalizaed
- More nodes in the hidden layer ---> Usually helps nothing when the number is larger than a paticular one based on our observations on the experiments ---> Too few nodes in the hidden layer cause under-fitting
- NOTE that over-fitting have NOT been observed in this example although it should happen theoretically as the number of nodes in the hidden layer increases