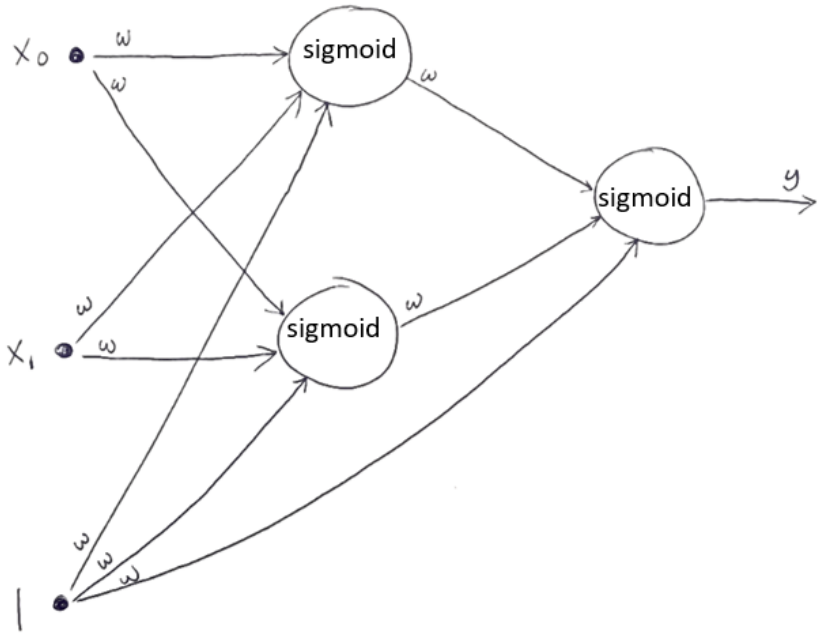
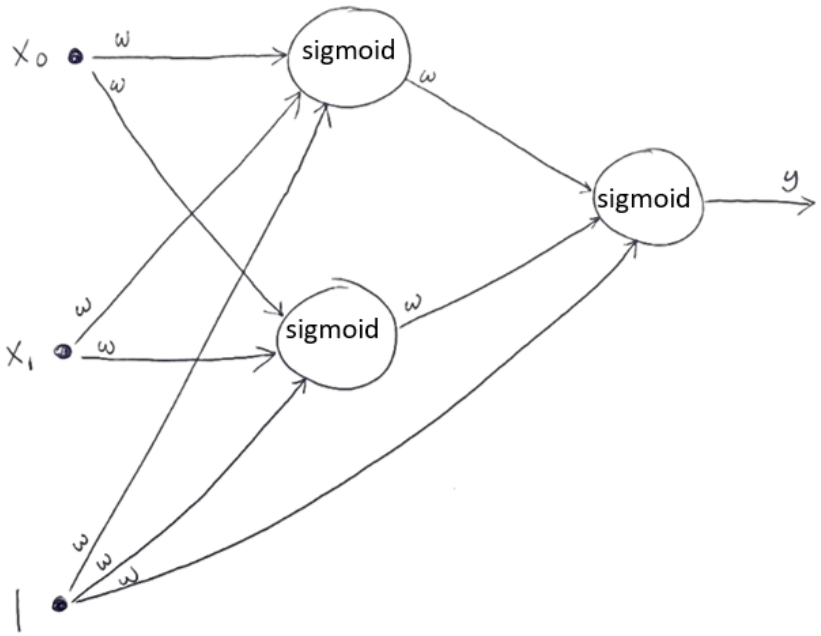
F25 **HW3:** Training a Dense FF neural network using ES(1+1) with 1/5 rule algorithm



LTU CS CJ Chung

HW3.1 – Complete Missing Lines

- ES1115_XOR_optimW_incomp.ipynb file is provided on Canvas
- First, complete the program with 5 missing lines to find 9 weight values for a swallow dense feed forward neural net with one hidden layer with 2 hidden neurons.
- Sigmoid activation function is used for the 2 hidden and one output neurons
- One *example* of the expected output is shown on 2 next slides:



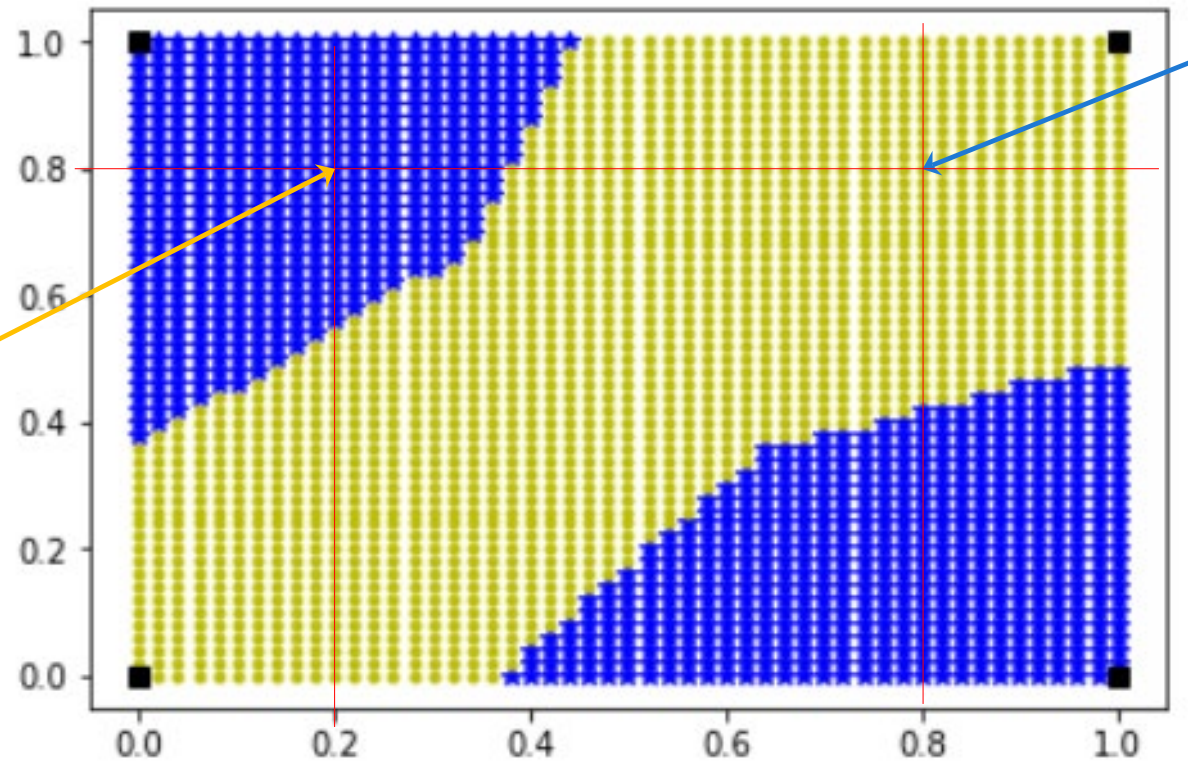
```
Trial = 1,      0.500016582712444
Trial = 2, Acceptable solution found at generation 189
         w[0]: 11.454774622577283
         w[1]: 9.068140495268418
         w[2]: -18.186258672670228
         w[3]: -19.12830567064856
         w[4]: -15.029279648883044
         w[5]: 6.17253772212217
         w[6]: -12.755123082394112
         w[7]: -11.58573626283945
         w[8]: 5.88125940246944
        SSE = 3.741815632036595e-05
        ***** To verify if the w vector above is correct *****
           Test Input: [0.01 0.00 1.00]         Output: 0.0
           Test Input: [0.00 0.95 1.00]         Output: 1.0
           Test Input: [1.00 0.05 1.00]         Output: 1.0
           Test Input: [0.90 1.00 1.00]         Output: 0.0
Trial = 3,      0.6666743437686713
Trial = 4, Acceptable solution found at generation 4719
         w[0]: -14.349789197134792
         w[1]: 15.109427079555285
         w[2]: 8.234163543104792
         w[3]: 4.317633017051261
         w[4]: -4.250361931573438
         w[5]: 2.260668347024739
         w[6]: -12.684631560160984
         w[7]: -14.422345890105095
         w[8]: 20.0
        SSE = 4.744493598805533e-05
        ***** To verify if the w vector above is correct *****
           Test Input: [0.01 0.00 1.00]         Output: 0.0
           Test Input: [0.00 0.95 1.00]         Output: 1.0
           Test Input: [1.00 0.05 1.00]         Output: 1.0
           Test Input: [0.90 1.00 1.00]         Output: 0.0
Trial = 5,      0.6666775429083995
Trial = 6,      0.5000655008528831
Trial = 7,      0.6667629773265705
Trial = 8,      0.5000381463558929
Trial = 9,      0.6666717816379472
```

```
Trial = 10,     0.5000248547400437
Trial = 11,     0.6666813840199102
Trial = 12, Acceptable solution found at generation 197
         w[0]: 8.374680290283202
         w[1]: 11.828941182484098
         w[2]: -16.117525760915942
         w[3]: 13.487976371485459
         w[4]: 4.94643819642753
         w[5]: -2.502401629381464
         w[6]: -14.4420421425789
         w[7]: 12.925978986040704
         w[8]: -6.121516284884849
        SSE = 4.954426604647078e-05
        ***** To verify if the w vector above is correct *****
           Test Input: [0.01 0.00 1.00]         Output: 0.0
           Test Input: [0.00 0.95 1.00]         Output: 1.0
           Test Input: [1.00 0.05 1.00]         Output: 1.0
           Test Input: [0.90 1.00 1.00]         Output: 0.0
Trial = 13,     0.6667641410893582
Trial = 14, Acceptable solution found at generation 446
         w[0]: 7.791702242656202
         w[1]: -6.043700689814957
         w[2]: -3.4683239283909915
         w[3]: 18.60478735464511
         w[4]: -12.170752812244343
         w[5]: 5.117128287872833
         w[6]: 15.232168340795129
         w[7]: -16.3177303686176
         w[8]: 6.838868971546033
        SSE = 1.6663952241764803e-05
        ***** To verify if the w vector above is correct *****
           Test Input: [0.01 0.00 1.00]         Output: 0.0
           Test Input: [0.00 0.95 1.00]         Output: 1.0
           Test Input: [1.00 0.05 1.00]         Output: 1.0
           Test Input: [0.90 1.00 1.00]         Output: 0.0
Trial = 15,     0.66669743094126
Trial = 16,     0.6668500890718511
Trial = 17,     0.500023845176707
Trial = 18,     0.5000316220034776
```

```
Trial = 19,     0.5000186433828077
Trial = 20,     0.6667475407715873
Trial = 21,     0.6666713467949291
Trial = 22,     0.500110639462309
Trial = 23, Acceptable solution found at generation 5184
         w[0]: 6.4111351195385495
         w[1]: -14.365417266441993
         w[2]: 12.258778170869011
         w[3]: 7.02556409601966
         w[4]: -15.296590842667904
         w[5]: -2.7030993887264443
         w[6]: -19.296124937590147
         w[7]: 11.718378784365665
         w[8]: 13.1823547395996
        SSE = 4.7526789826157214e-05
        ***** To verify if the w vector above is correct *****
           Test Input: [0.01 0.00 1.00]         Output: 0.0
           Test Input: [0.00 0.95 1.00]         Output: 1.0
           Test Input: [1.00 0.05 1.00]         Output: 1.0
           Test Input: [0.90 1.00 1.00]         Output: 0.0
Trial = 24,     0.6668854260117105
Trial = 25,     0.666485153722645
System Success: 20.0%
```

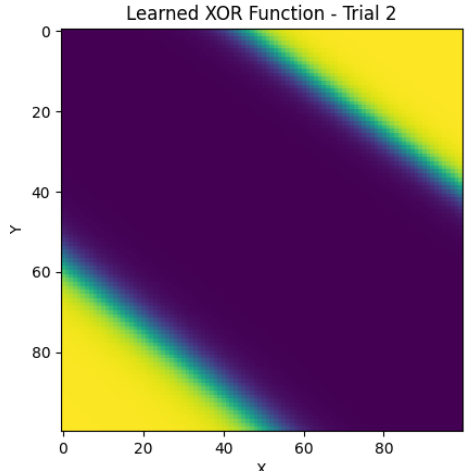
HW3.2 – Plot a Decision Boundary (DB) graph to show the XOR learning for each Trial, when acceptable solution is found. For Overfit/Underfit/Goodfit analysis.

An example of the DB graph using mesh for the XOR function

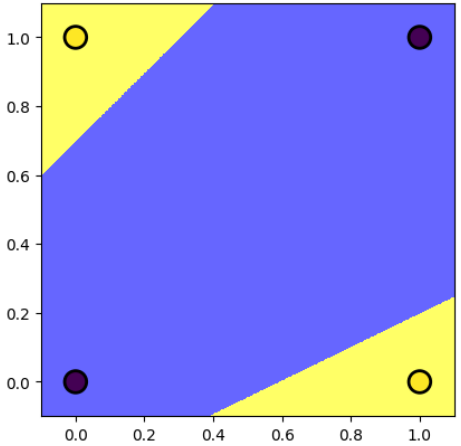


Input [0.2, 0.8]. Inferenced NN output value belongs to blue (1) class

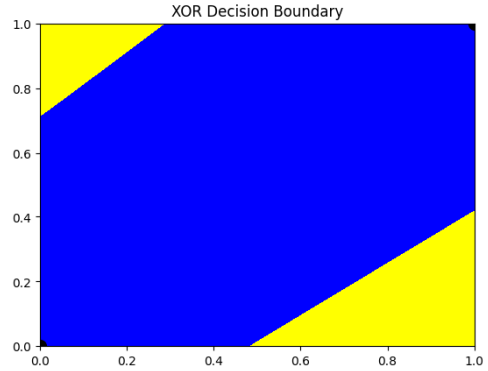
Input [0.8, 0.8]. Inferenced NN output value belongs to yellow (0) class



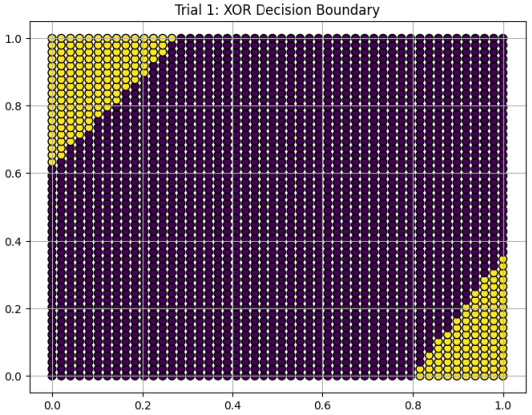
plt.imshow(Z_learned, cmap='viridis', interpolation='nearest')



ListedColormap



Using contour



Using mesh

HW3 Checklist (4/4)

- Coding standards including your name, LTU honor code
- ffp() Python function is completed & working well?
- Your submitted .ipynb file includes the run result of at least one acceptable solution with
  - Test input verification
  - a correct DB graph

Future HW:
Refine the code to handle *n* hidden neurons

- The current code is fixed to handle only 2 hidden neurons. Generalize the ffp() function code to handle *n* hidden neurons
- Re-run your refined code with "4" hidden neurons using sigmoid activation function