AI Assignment #5 Report

Name: Sajjad Pakdaman Savoji   *SID: 810195517*

4. Testing Network:

For this part the corresponding script was used.

```
→  code_sajjad python3 neural_net_tester.py simple
----------------------------------------
Training on OR data
train:: 100%|████████████████████████████████████████|
 10000/10000 [00:02<00:00, 4036.88it/s]
weights: [w1A(11.17), w2A(11.17), wA(6.93)]
Trained weights:
Weight 'w1A': 11.172709
Weight 'w2A': 11.173043
Weight 'wA': 6.933418
Testing on OR test-data
test([0.1, 0.1, 0]) returned: 0.0090236407200403006 => 0 [correct]
test([0.1, 0.9, 1]) returned: 0.9857013164734841 => 1 [correct]
test([0.9, 0.1, 1]) returned: 0.9857873715894018 => 1 [correct]
test([0.9, 0.9, 1]) returned: 0.9999401077000629 => 1 [correct]
Accuracy: 1.000000
----------------------------------------
Training on AND data
train:: 100%|████████████████████████████████████████|
 10000/10000 [00:02<00:00, 4086.96it/s]
weights: [w1A(10.50), w2A(10.50), wA(14.37)]
Trained weights:
Weight 'w1A': 10.499844
Weight 'w2A': 10.499561
Weight 'wA': 14.366079
Testing on AND test-data
test([0.1, 0.1, 0]) returned: 4.704254617957318e-06 => 0 [correct]
test([0.1, 0.9, 0]) returned: 0.020484490369173127 => 0 [correct]
test([0.9, 0.1, 0]) returned: 0.020489030863720659 => 0 [correct]
test([0.9, 0.9, 1]) returned: 0.9893684979736843 => 1 [correct]
Accuracy: 1.000000
```

5. Finite difference:

For this part member function finite_difference(self) was implemented for class Network using which correctness of derivatives implementations was tested by simple test

```
→  code_sajjad python3 neural_net_tester.py simple
-----------------------------------------
Training on OR data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:02<00:00, 3805.44it/s]
weights: [w1A(11.17), w2A(11.17), wA(6.93)]
passed finite difference test with acc = 1e-08
Trained weights:
Weight 'w1A': 11.172709
Weight 'w2A': 11.173043
Weight 'wA': 6.933418
Testing on OR test-data
test((0.1, 0.1, 0)) returned: 0.008023648720036306 => 0 [correct]
test((0.1, 0.9, 1)) returned: 0.9857913164734841 => 1 [correct]
test((0.9, 0.1, 1)) returned: 0.9857075715094610 => 1 [correct]
test((0.9, 0.9, 1)) returned: 0.999998107780620 => 1 [correct]
Accuracy: 1.000000
-----------------------------------------
Training on AND data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:03<00:00, 3318.70it/s]
weights: [w1A(10.50), w2A(10.50), wA(14.37)]
passed finite difference test with acc = 1e-08
Trained weights:
Weight 'w1A': 10.499844
Weight 'w2A': 10.499561
Weight 'wA': 14.356979
Testing on AND test-data
test((0.1, 0.1, 0)) returned: 4.784254617057318e-06 => 0 [correct]
test((0.1, 0.9, 0)) returned: 0.020484400369173127 => 0 [correct]
test((0.9, 0.1, 0)) returned: 0.020489038037200559 => 0 [correct]
test((0.9, 0.9, 1)) returned: 0.9093604979790043 => 1 [correct]
Accuracy: 1.000000
```

6. Two Layer Neural Network:

Using API provided by neural_net.py the requested network was implemented.

OR :

```
-----------------------------------------
Training on OR data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:10<00:00, 992.25it/s]
weights: [w1A(-4.17), w2A(-4.15), wA(-2.08), w1B(-5.23), w2B(-5.25), wB(-2.84), wAC(-5.94), wBC(-8.29), wC(-6.16)]
Trained weights:
Weight 'w1A': -4.166633
Weight 'w2A': -4.149468
Weight 'wA': -2.081165
Weight 'w1B': -5.234393
Weight 'w2B': -5.249836
Weight 'wB': -2.835798
Weight 'wAC': -5.938956
Weight 'wBC': -8.290385
Weight 'wC': -6.155858
Testing on OR test-data
test((0.1, 0.1, 0)) returned: 0.008025079020622775 => 0 [correct]
test((0.1, 0.9, 1)) returned: 0.9919016393840399 => 1 [correct]
test((0.9, 0.1, 1)) returned: 0.9919040619425451 => 1 [correct]
test((0.9, 0.9, 1)) returned: 0.9978001861360183 => 1 [correct]
Accuracy: 1.000000
```

AND:

```
----------------------------------------
Training on AND data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:09<00:00, 1092.36it/s]
weights: [w1A(-3.57), w2A(-3.61), wA(-4.72), w1B(4.02), w2B(3.90), wB(5.31), wAC(-7.77), wBC(8.66), wC(0.96)]
Trained weights:
Weight 'w1A': -3.567641
Weight 'w2A': -3.608579
Weight 'wA': -4.719048
Weight 'w1B': 4.021667
Weight 'w2B': 3.901724
Weight 'wB': 5.306260
Weight 'wAC': -7.765078
Weight 'wBC': 8.655540
Weight 'wC': 0.962835
Testing on AND test-data
test((0.1, 0.1, 0)) returned: 0.0002045104268045321 => 0 [correct]
test((0.1, 0.9, 0)) returned: 0.006783813285854222 => 0 [correct]
test((0.9, 0.1, 0)) returned: 0.006780528452711041 => 0 [correct]
test((0.9, 0.9, 1)) returned: 0.995518965440981 => 1 [correct]
Accuracy: 1.000000
```

EQUAL:

```
----------------------------------------
Training on EQUAL data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:05<00:00, 1720.70it/s]
weights: [w1A(-6.60), w2A(-6.85), wA(-2.78), w1B(-5.01), w2B(-5.06), wB(-7.50), wAC(10.63), wBC(-10.51), wC(-5.01)]
Trained weights:
Weight 'w1A': -6.602529
Weight 'w2A': -6.845208
Weight 'wA': -2.782586
Weight 'w1B': -5.012516
Weight 'w2B': -5.056722
Weight 'wB': -7.501407
Weight 'wAC': 10.631200
Weight 'wBC': -10.505650
Weight 'wC': -5.008055
Testing on EQUAL test-data
test((0.1, 0.1, 1)) returned: 0.9572829757678457 => 1 [correct]
test((0.1, 0.9, 0)) returned: 0.011226265758113587 => 0 [correct]
test((0.9, 0.1, 0)) returned: 0.011358471362683439 => 0 [correct]
test((0.9, 0.9, 1)) returned: 0.9603879694444553 => 1 [correct]
Accuracy: 1.000000
```

HORIZONTAL BAND:

```
----------------------------------------
Training on horizontal-bands data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:24<00:00, 405.12it/s]
weights: [w1A(0.07), w2A(-4.67), wA(-11.38), w1B(0.22), w2B(-6.74), wB(-3.21), wAC(12.12), wBC(-11.04), wC(5.90)]
Trained weights:
Weight 'w1A': 0.074565
Weight 'w2A': -4.668671
Weight 'wA': -11.384945
Weight 'w1B': 0.210894
Weight 'w2B': -6.737621
Weight 'wB': -3.212910
Weight 'wAC': 12.123274
Weight 'wBC': -11.038896
Weight 'wC': 5.903178
Testing on horizontal-bands test-data
test((1, 1.5, 1)) returned: 0.9970851982897587 => 1 [correct]
test((2, 1.5, 1)) returned: 0.9970995487086657 => 1 [correct]
test((3, 1.5, 1)) returned: 0.9977101876957527 => 1 [correct]
test((0, 1.5, 1)) returned: 0.9976679543307334 => 1 [correct]
test((4, 0, 0)) returned: 0.0043903240060333963 => 0 [correct]
test((4, 4, 0)) returned: 0.002753737580151226 => 0 [correct]
test((-1, 0, 0)) returned: 0.0063416337956682344 => 0 [correct]
test((-1, 4, 0)) returned: 0.0027442307577797715 => 0 [correct]
Accuracy: 1.000000
```

VERTICAL BAND:

```
-----------------------------------------
Training on vertical-bands data
train:: 100%|████████████████████████████████████████████████████████████| 10000/10000 [00:24<00:00, 411.541t/s]
weights: [w1A(-4.63), w2A(0.07), wA(-11.29), w1B(-6.60), w2B(0.19), wB(-3.20), wAC(12.17), wBC(-11.88), wC(5.92)]
Trained weights:
Weight 'w1A': -4.631307
Weight 'w2A': 0.073528
Weight 'wA': -11.290081
Weight 'w1B': -6.602039
Weight 'w2B': 0.194034
Weight 'wB': -3.196709
Weight 'wAC': 12.171234
Weight 'wBC': -11.884988
Weight 'wC': 5.921322
Testing on vertical-bands test-data
test((0, 1, 0)) returned: 0.005230871970315181 => 0 [correct]
test((0, 2, 0)) returned: 0.004896216945513198 => 0 [correct]
test((0, 1.5, 0)) returned: 0.005053170168565381 => 0 [correct]
test((1.5, 2, 1)) returned: 0.9977478009786042 => 1 [correct]
test((1.5, 5, 1)) returned: 0.9977603402749306 => 1 [correct]
test((1.5, 1, 1)) returned: 0.9977333619340035 => 1 [correct]
test((3, 1, 0)) returned: 0.006541373237095254 => 0 [correct]
test((3, 1.5, 0)) returned: 0.006746413743060605 => 0 [correct]
test((3, 2, 0)) returned: 0.006964631731464771 => 0 [correct]
test((1, 1.5, 1)) returned: 0.9967653131579071 => 1 [correct]
test((1, -1.5, 1)) returned: 0.9973872903399298 => 1 [correct]
test((2, 1.5, 1)) returned: 0.9930786027614609 => 1 [correct]
test((2, -1.5, 1)) returned: 0.9909087004424382 => 1 [correct]
test((4, 0, 0)) returned: 0.002697970602214446 => 0 [correct]
test((4, 4, 0)) returned: 0.0027060530424612276 => 0 [correct]
test((-1, 0, 0)) returned: 0.003560037401315807 => 0 [correct]
test((-1, 4, 0)) returned: 0.0035507754630094517 => 0 [correct]
Accuracy: 1.000000
```

DIAGONAL BAND:

```
-----------------------------------------
Training on diagonal-band data
train:: 100%|████████████████████████████████████████████████████████████| 10000/10000 [00:25<00:00, 392.951t/s]
weights: [w1A(-4.37), w2A(4.70), wA(-4.05), w1B(4.76), w2B(-4.43), wB(-4.14), wAC(10.04), wBC(10.03), wC(15.12)]
Trained weights:
Weight 'w1A': -4.367353
Weight 'w2A': 4.701801
Weight 'wA': -4.049331
Weight 'w1B': 4.750030
Weight 'w2B': -4.434049
Weight 'wB': -4.141410
Weight 'wAC': 10.036737
Weight 'wBC': 10.033377
Weight 'wC': 15.122426
Testing on diagonal-band test-data
test((-1, -1, 1)) returned: 0.9889386357928035 => 1 [correct]
test((5, 5, 1)) returned: 0.9924861019778131 => 1 [correct]
test((-2, -2, 1)) returned: 0.9868919510788497 => 1 [correct]
test((6, 6, 1)) returned: 0.992619004389092 => 1 [correct]
test((3.5, 3.5, 1)) returned: 0.9921754795060245 => 1 [correct]
test((1.5, 1.5, 1)) returned: 0.9913055013937995 => 1 [correct]
test((4, 0, 0)) returned: 0.006126212455252016 => 0 [correct]
test((0, 4, 0)) returned: 0.006146656123367745 => 0 [correct]
Accuracy: 1.000000
```

INVERSE DIAGONAL BAND:

```
----------------------------------------
Training on inverse diagonal band data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:24<00:00, 412.30it/s]
weights: [w1A(-4.32), w2A(4.01), wA(3.95), w1B(4.02), w2B(-4.33), wB(3.96), wAC(10.80), wBC(10.80), wC(5.18)]
Trained weights:
Weight 'w1A': -4.323526
Weight 'w2A': 4.009195
Weight 'wA': 3.947959
Weight 'w1B': 4.016341
Weight 'w2B': -4.327635
Weight 'wB': 3.961009
Weight 'wAC': 10.796815
Weight 'wBC': 10.796226
Weight 'wC': 5.176297
Testing on inverse-diagonal-band test-data
test([-1, -1, 0]) returned: 0.009700016325943352 => 0 [correct]
test([5, 5, 0]) returned: 0.006119406340424041 => 0 [correct]
test([-2, -2, 0]) returned: 0.011770825265932533 => 0 [correct]
test([6, 6, 0]) returned: 0.005908108594455056 => 0 [correct]
test([3.5, 3.5, 0]) returned: 0.0064481057268764435 => 0 [correct]
test([1.5, 1.5, 0]) returned: 0.007242086015958109 => 0 [correct]
test([4, 0, 1]) returned: 0.9963879772177346 => 1 [correct]
test([0, 4, 1]) returned: 0.9963490094374257 => 1 [correct]
Accuracy: 1.000000
```

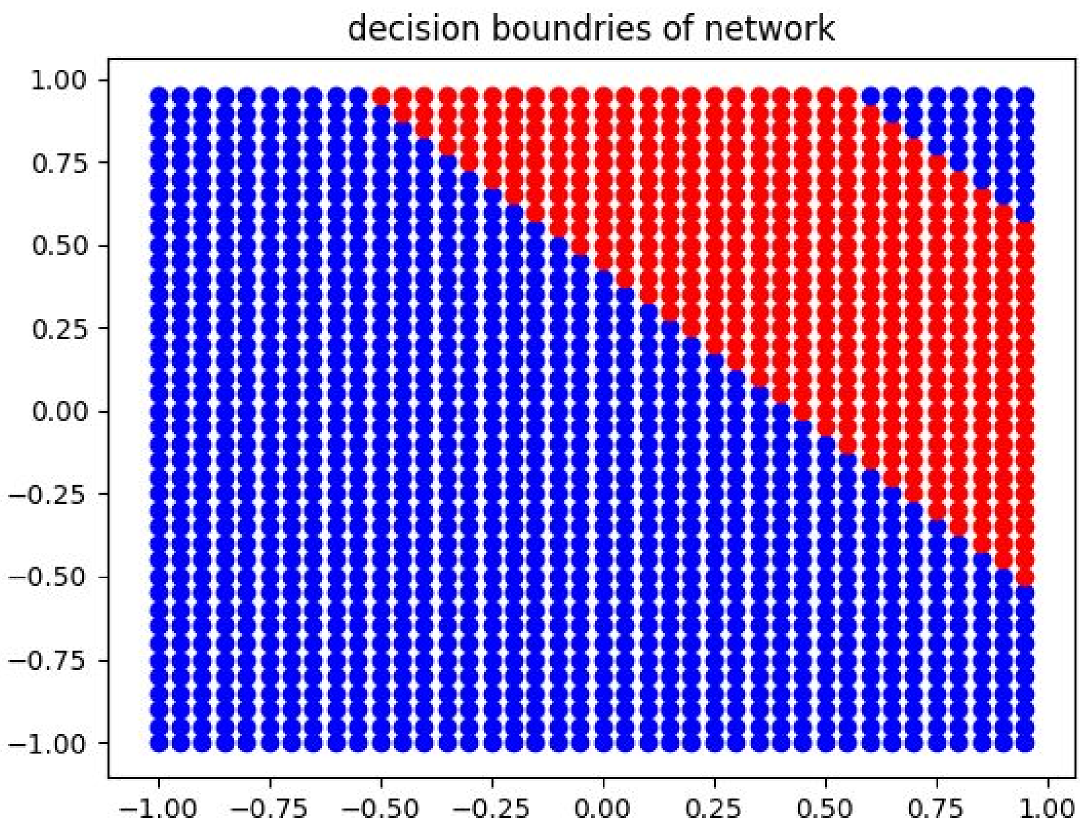
NOT EQUAL:

```
----------------------------------------
Training on NOT_EQUAL data
train:: 100%|████████████████████████████████████████| 10000/10000 [00:05<00:00, 1743.43it/s]
weights: [w1A(-6.58), w2A(6.59), wA(3.59), w1B(-5.97), w2B(6.23), wB(-2.97), wAC(10.22), wBC(-10.01), wC(-4.79)]
Trained weights:
Weight 'w1A': -6.582623
Weight 'w2A': 6.587322
Weight 'wA': 3.588473
Weight 'w1B': -5.969659
Weight 'w2B': 6.238543
Weight 'wB': -2.973468
Weight 'wAC': 10.221127
Weight 'wBC': -10.006440
Weight 'wC': -4.788837
Testing on NOT EQUAL test data
test([0.1, 0.1, 0]) returned: 0.011342586438581542 => 0 [correct]
test([0.1, 0.9, 1]) returned: 0.9878214787819529 => 1 [correct]
test([0.9, 0.1, 1]) returned: 0.9858289458612442 => 1 [correct]
test([0.9, 0.9, 0]) returned: 0.010428404981705212 => 0 [correct]
Accuracy: 1.000000
```

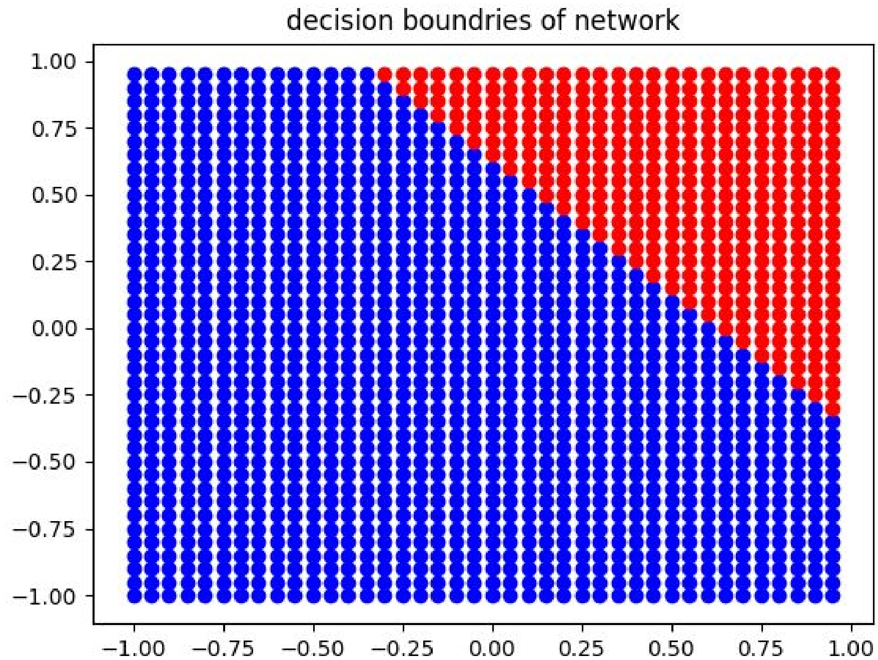
7. Plot Decision Boundary:

To plot the corresponding decision boundary plt.scatter was used.

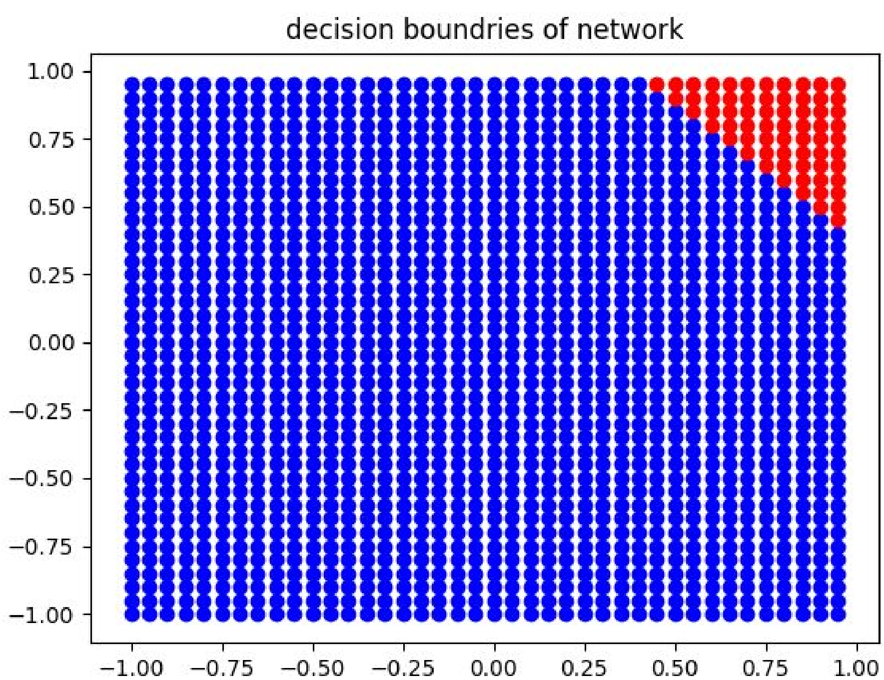
NOT EQ boundaries:



OR boundaries:



AND boundaries:

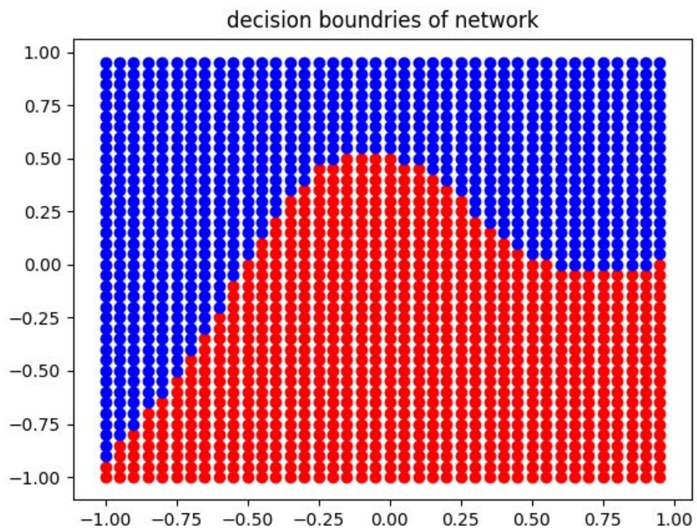


8. Overfitting and Regularization:

In short, Regularization in machine learning is the process of regularizing the parameters that constrain, regularizes, or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, avoiding the risk of Overfitting.

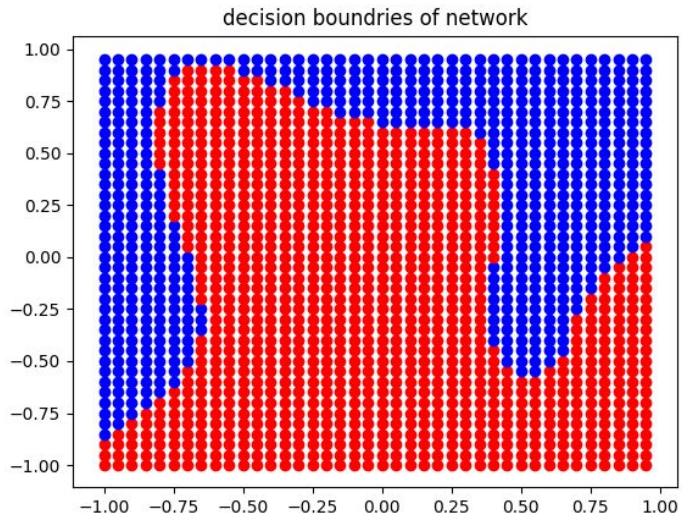
Iteration = 100

```
------------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 100/100 [00:25<00:00,  3.86it/s]
decision surface:: 16001it [00:07, 219.07it/s]
Accuracy: 0.990000
```



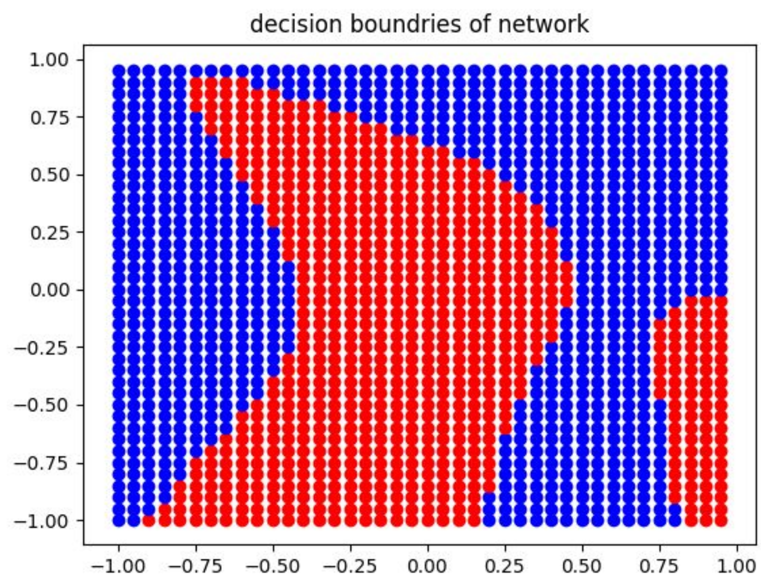
Iteration = 500

```
→ code_sajjad python3 neural_net_tester.py two_moons
------------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 500/500 [02:13<00:00,  3.74it/s]
decision surface:: 10001it [00:08, 178.69it/s]
Accuracy: 0.900000
```



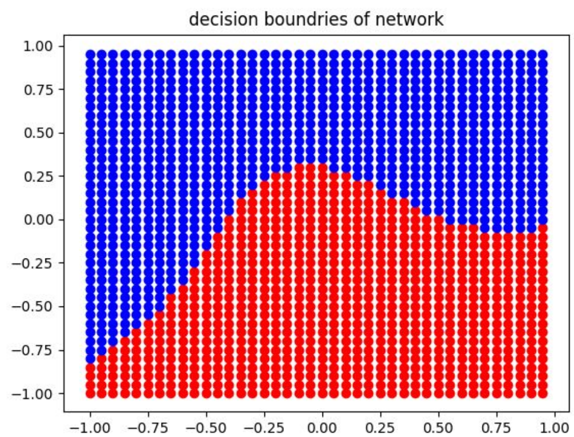
Iteration 1000:

```
----------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 1000/1000 [04:38<00:00,  3.59it/s]
decision surface:: 1600it [00:08, 182.38it/s]
Accuracy: 0.860000
```



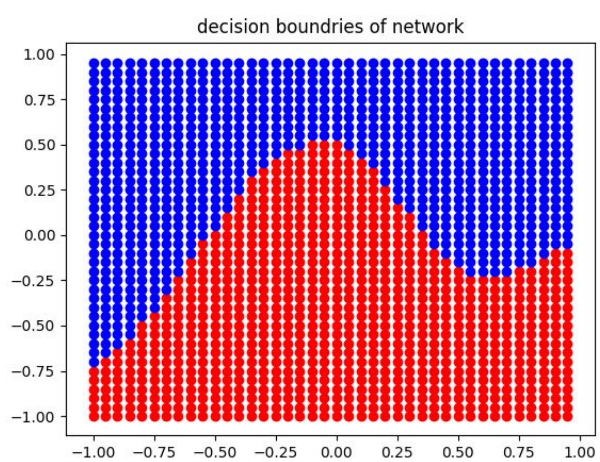
Iteration = 100 with regularization:

```
→ code_sajjad python3 neural_net_tester.py two_moons
----------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.59it/s]
decision surface:: 1600it [00:07, 227.48it/s]
Accuracy: 0.980000
```



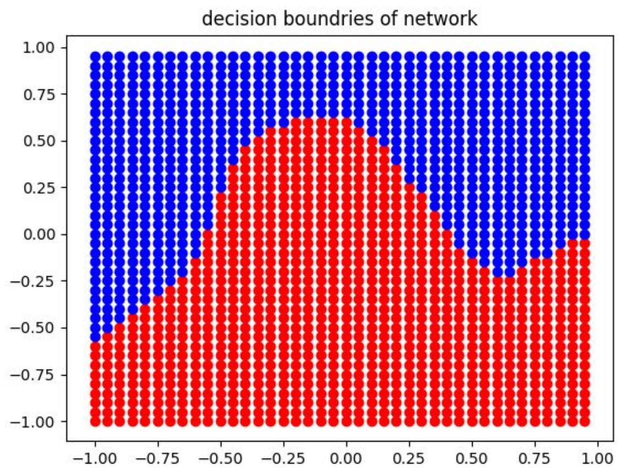
Iteration = 500 with regularization:

```
→ code_sajjad python3 neural_net_tester.py two_moons
----------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 500/500 [02:20<00:00,  3.57it/s]
decision surface:: 1600it [00:07, 227.70it/s]
Accuracy: 0.980000
```



Iteration = 1000 with regularization:

```
➜  code_sajjad python3 neural_net_tester.py two moons
----------------------------------------
Training on two-moons data
train:: 100%|████████████████████████████████████████| 1000/1000 [04:57<00:00,  3.36it/s]
decision surface:: 1600it [00:10, 157.11it/s]
Accuracy: 0.900000
```



In the first part as the number of iterations increased , model was overfitted on training data. It happens mainly for the extreme flexibility of neural networks. One of the signs of overfitting is complex decision boundaries which occurred in this part.

When regularization term is added , it makes parameters to constantly drift to 0 thus prevents overfitting by putting additional yet desirable constraints. As a result decisions boundaries would appear more smooth.

The problem of overfitting is obviously solved!