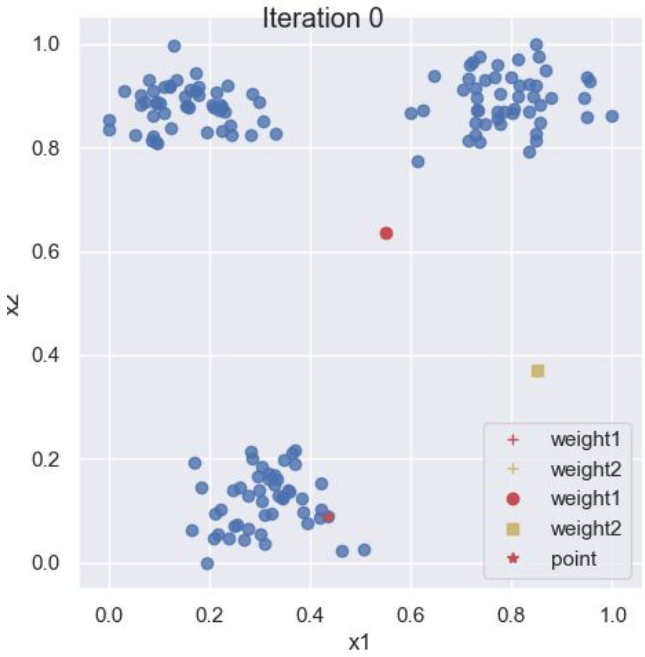
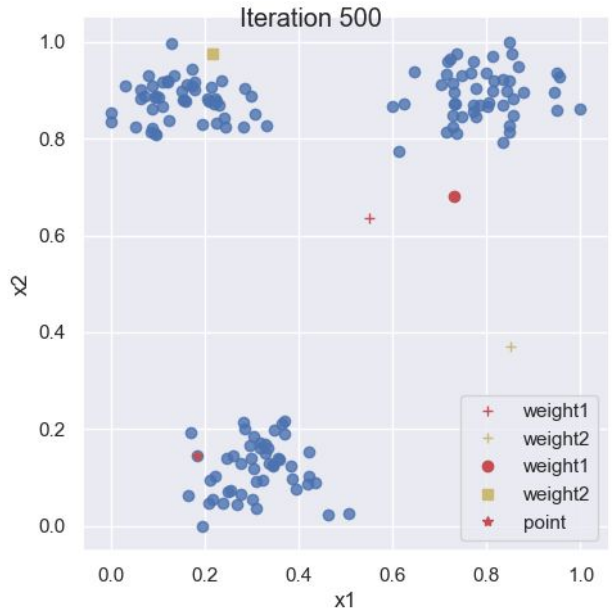
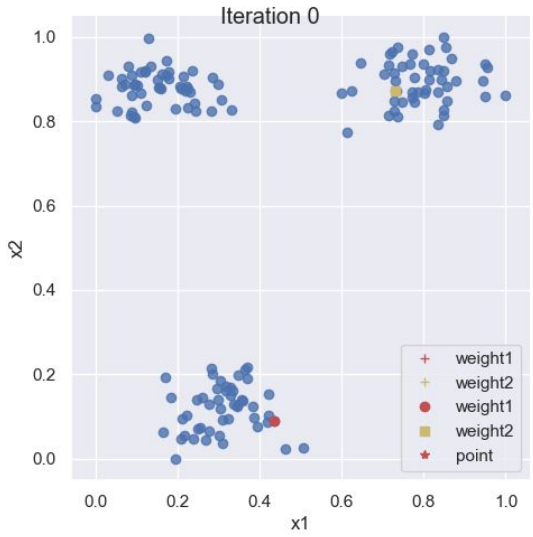
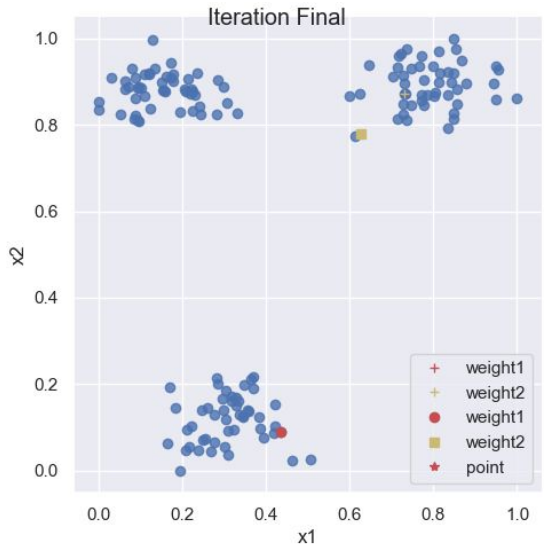
CMSC 409
Extra Credit
Chady Aboulhosn

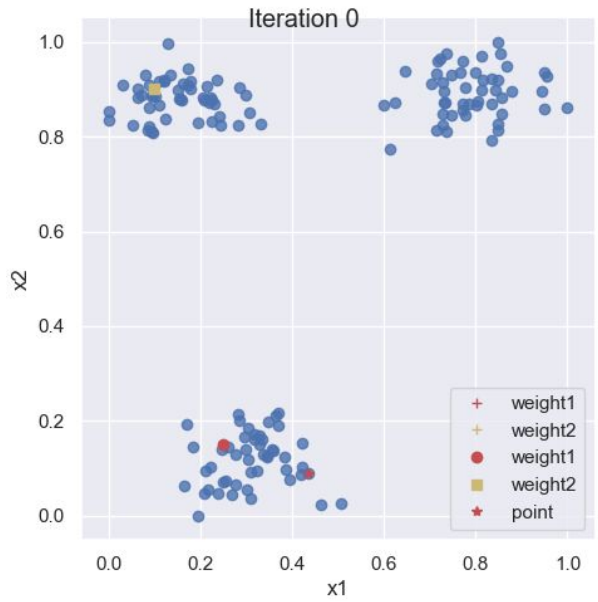
2 Neurons
Random weights

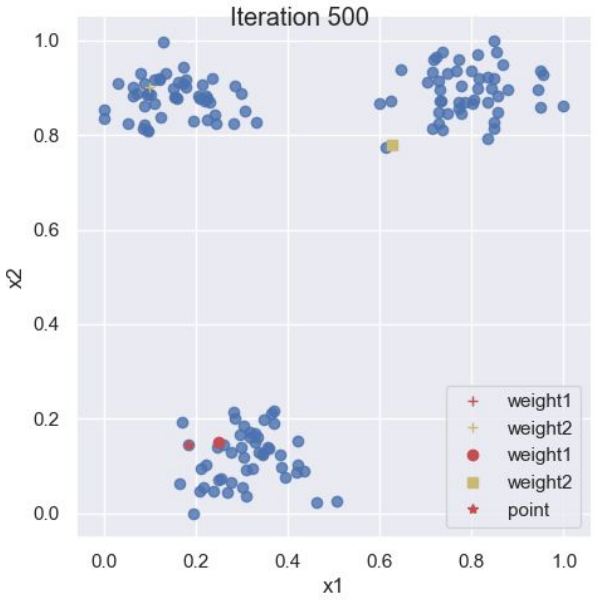





Iteration 0
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight1
weight2
point

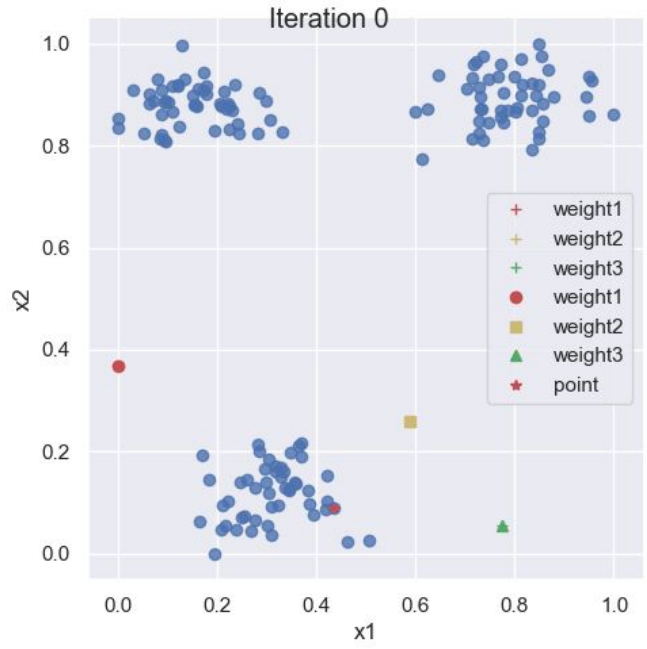

Iteration Final
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight1
weight2
point

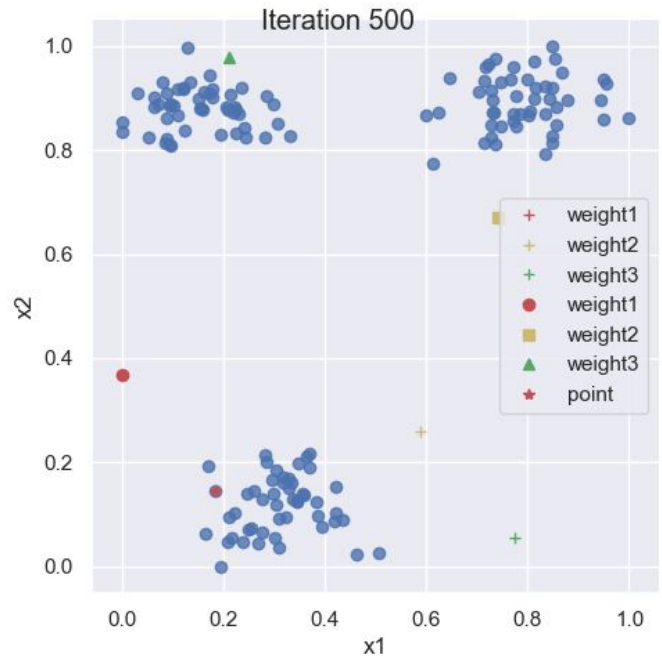


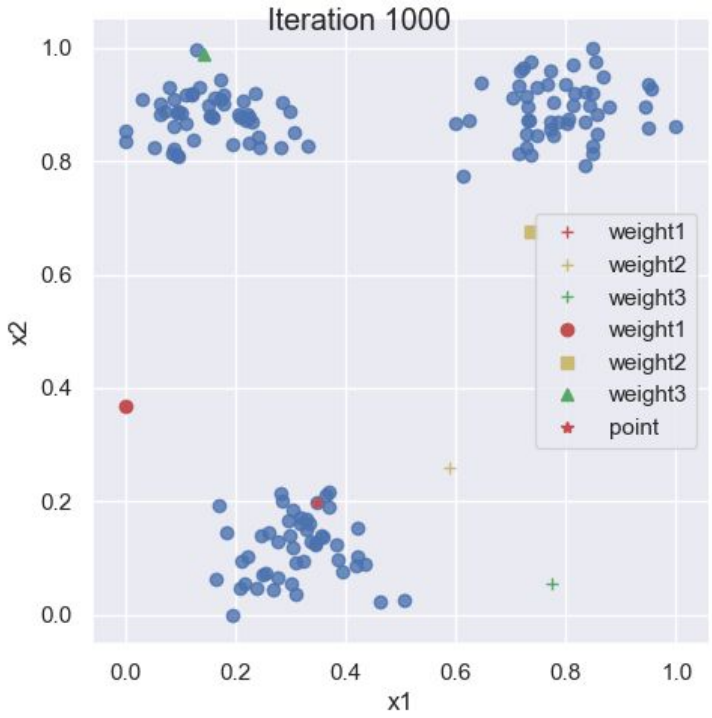


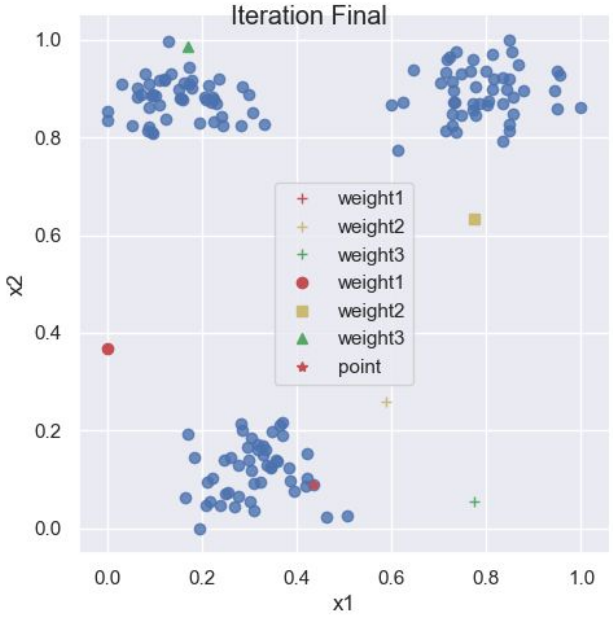
Chosen weights
3 Neurons

Random Weights
I ran each iteration 10 times the length of the data so in this case 1500 times


Iteration 0
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

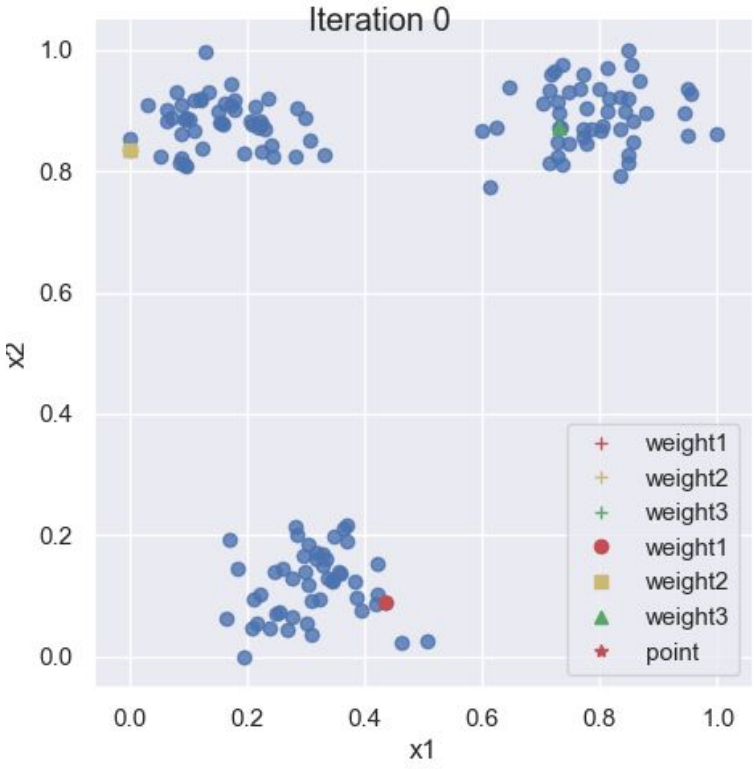

Iteration 500
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

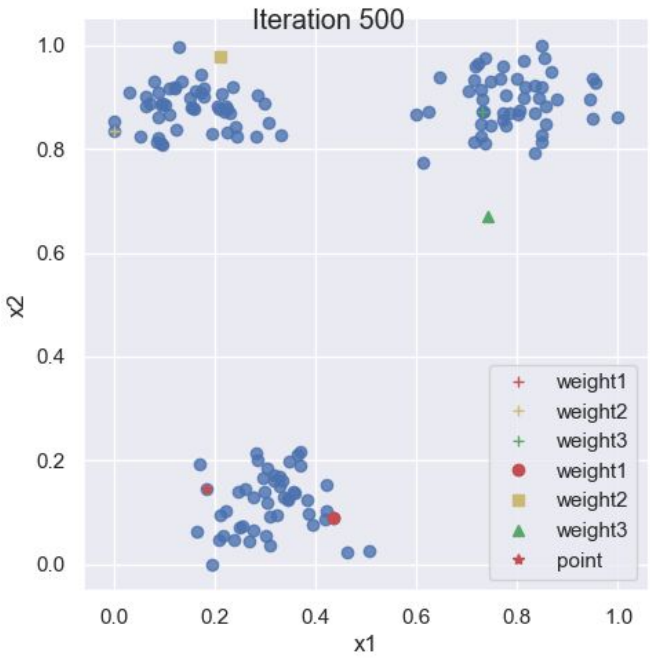

Iteration 1000
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

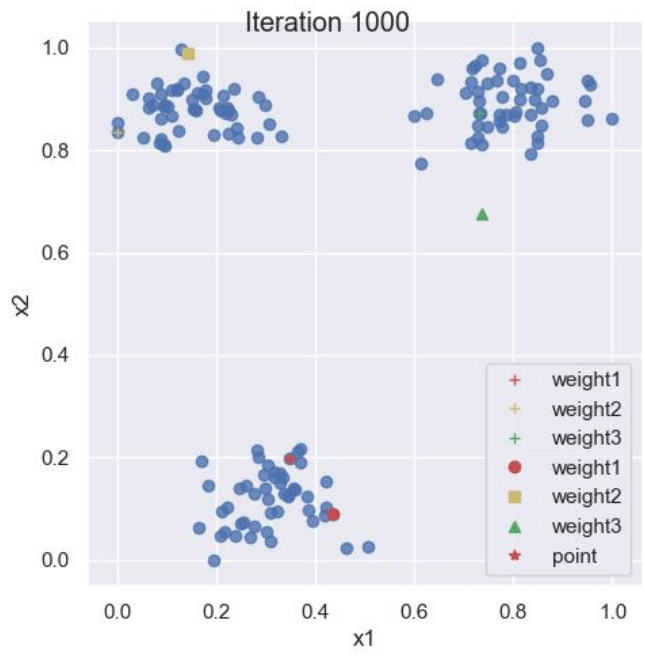

Iteration Final
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

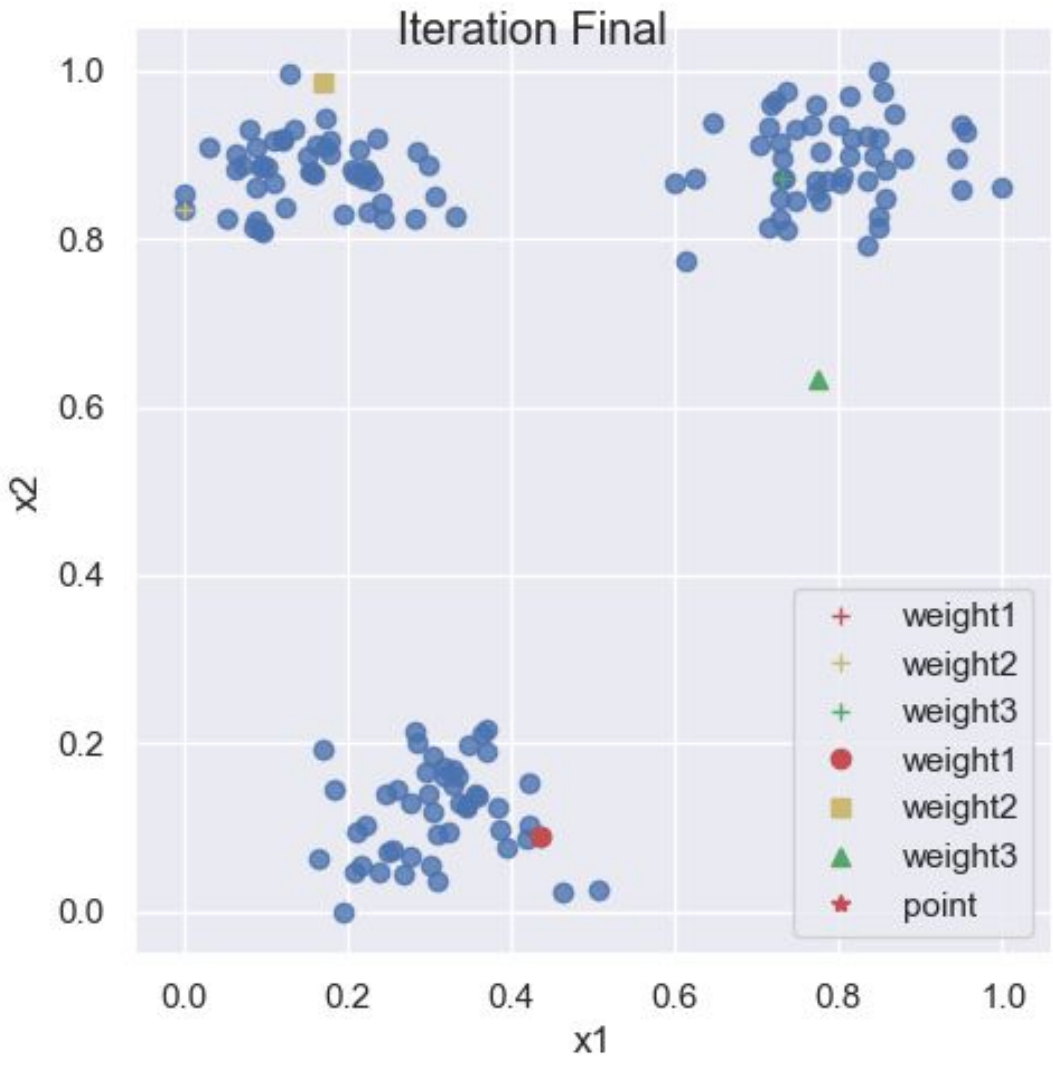
The problem with random weights is the loser neurons never updated so only the winning neuron will update in the end. While this is the point of the Winner takes all method, in this case random weights do not set it up for success.

In the next situation the weights are based on the inputs for x1 and x2 with the weights coming from the beginning middle and end of the data

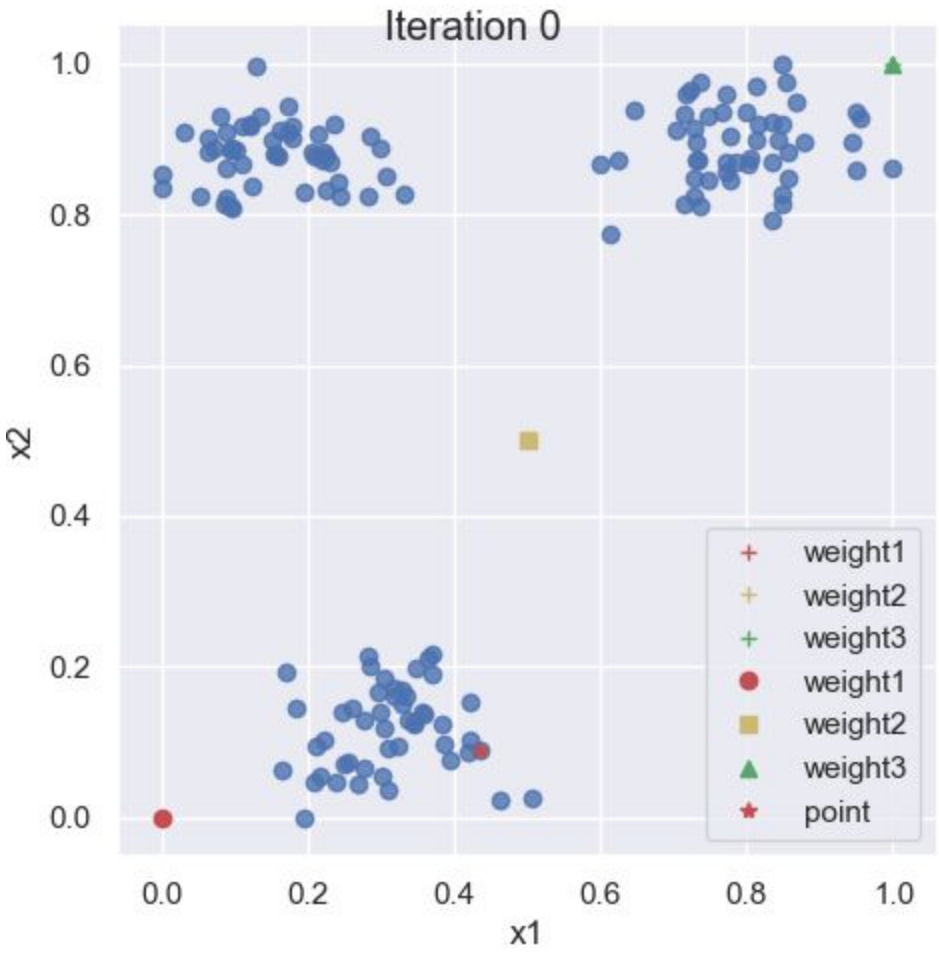


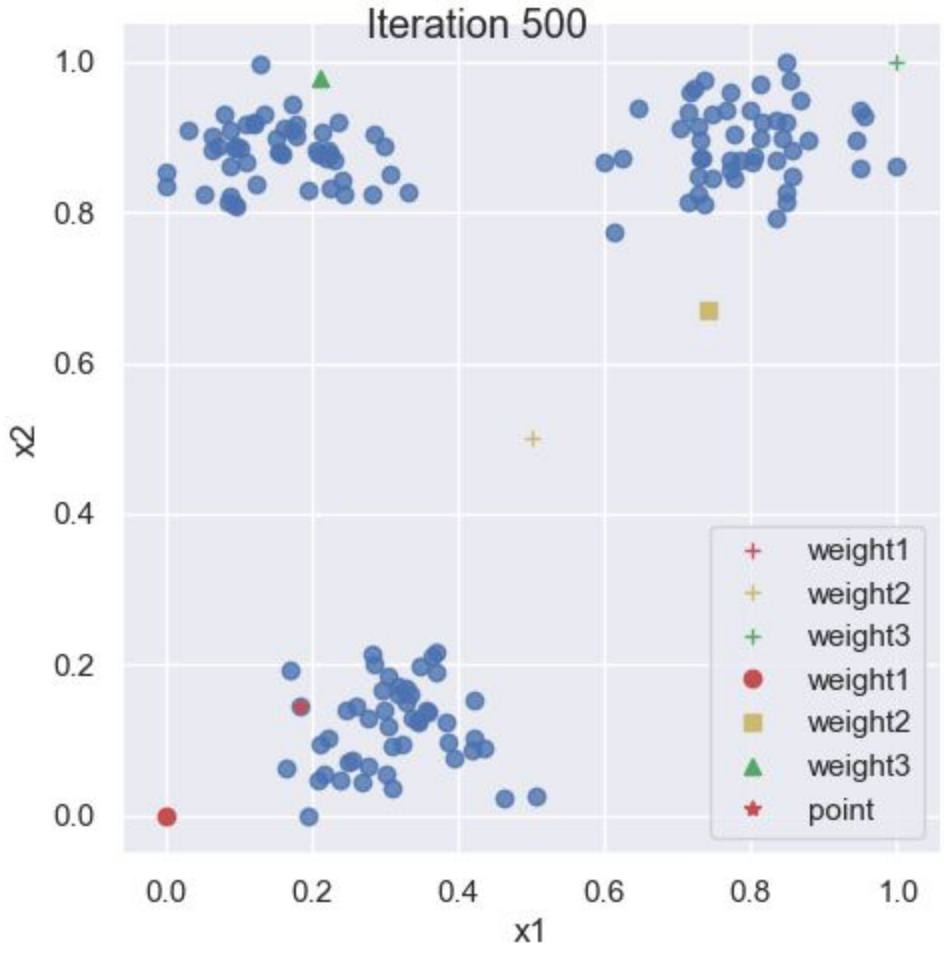

Iteration 500
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

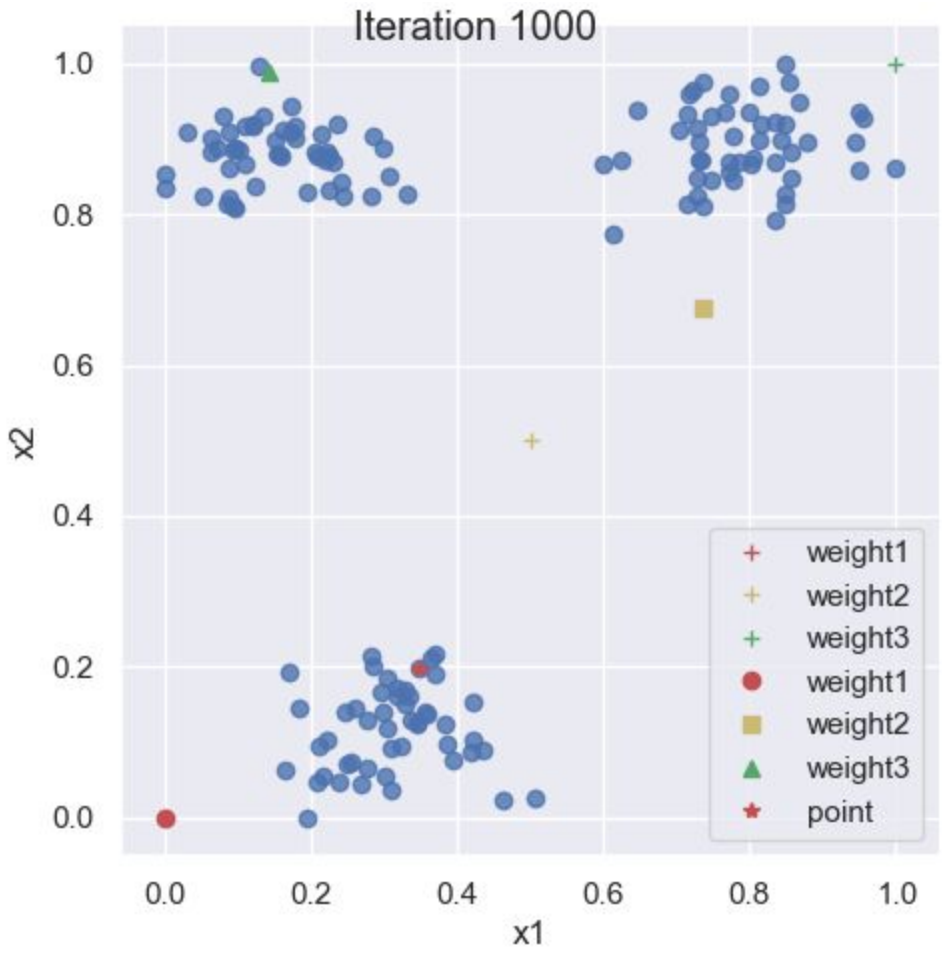

Iteration 1000
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

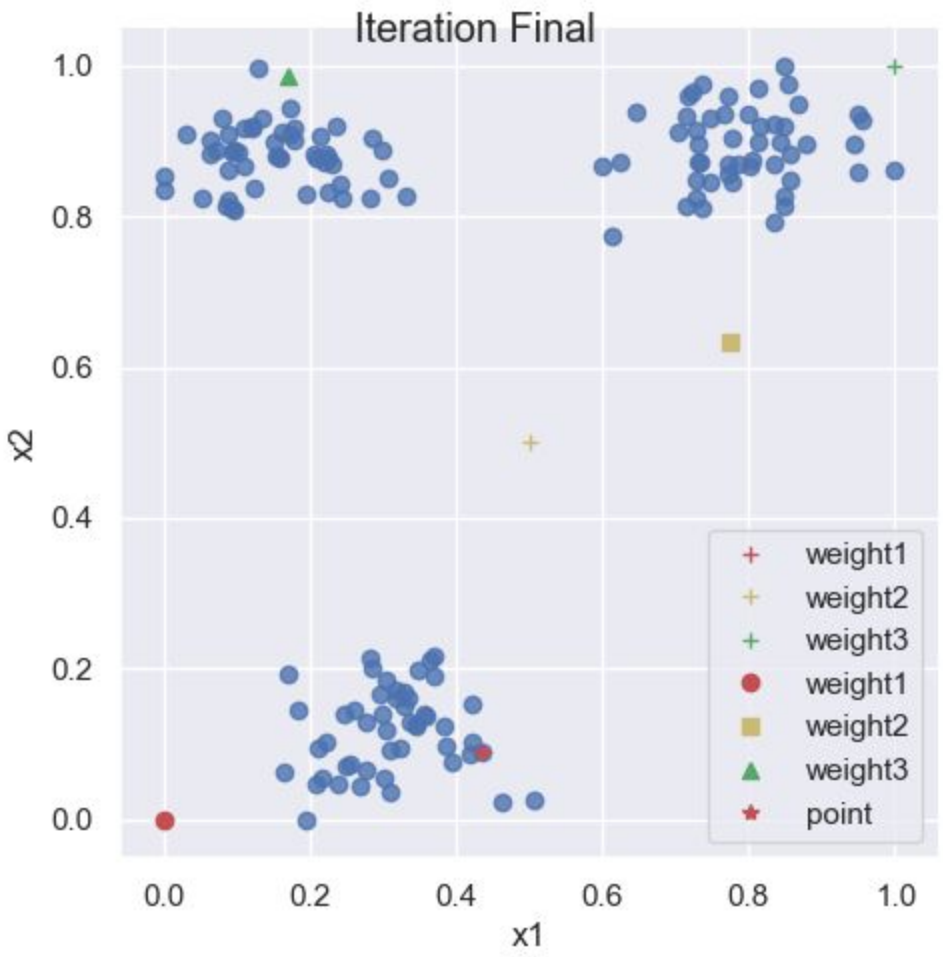


This situation the weights were close to the perfect beginning positions eventually they migrated away from a winning situation.

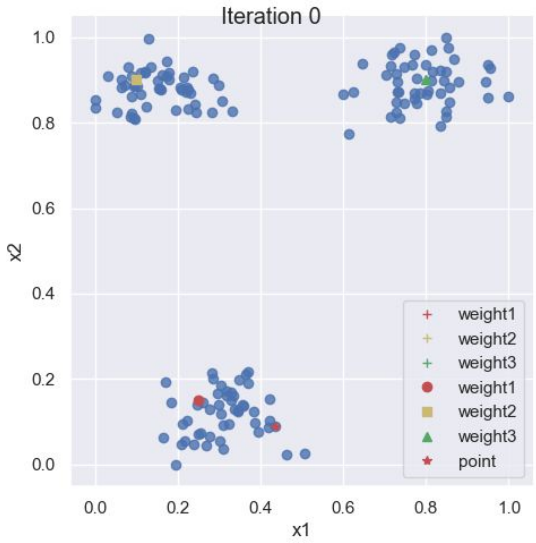

Iteration 0
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

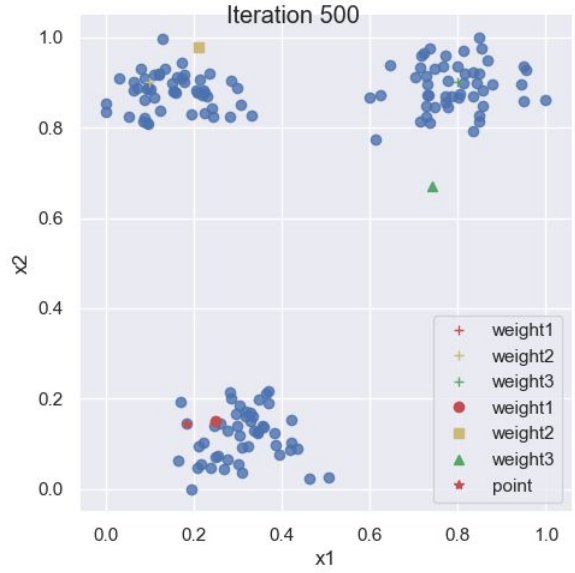

Iteration 500
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

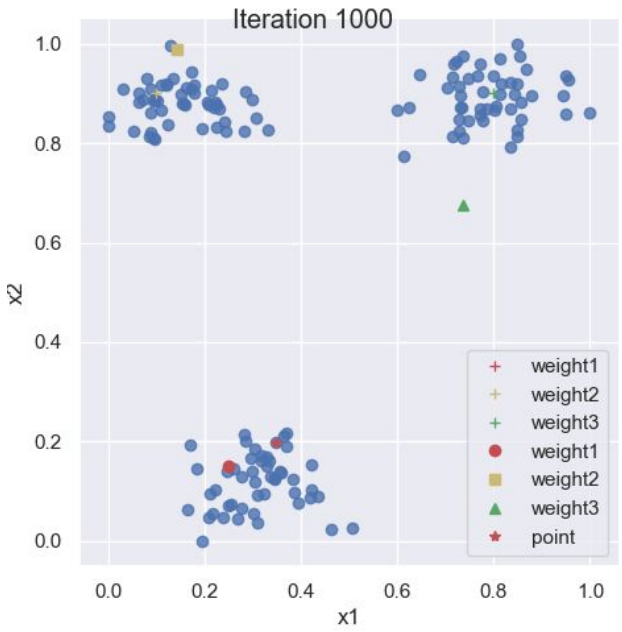

Iteration 1000
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

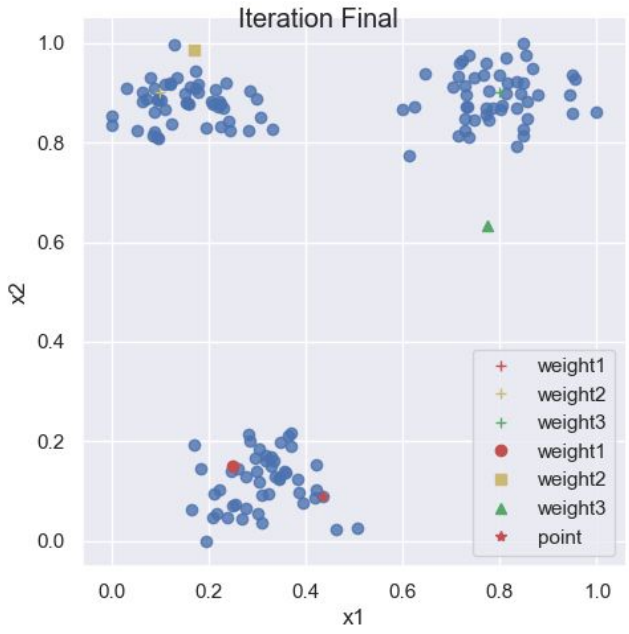


In the situation where the weights are based on the middle of the x values it is clear that the extra iterations did not help at all in a convergence towards the clusters. Once a winner was picked it would have the highest net even after training

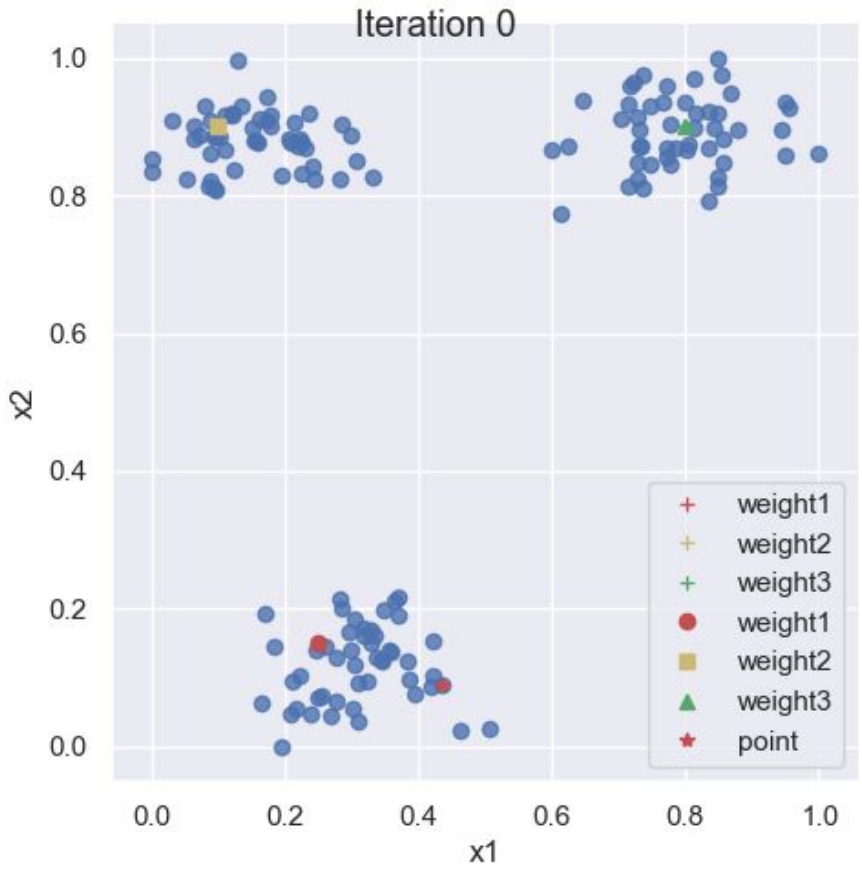

Iteration 0
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

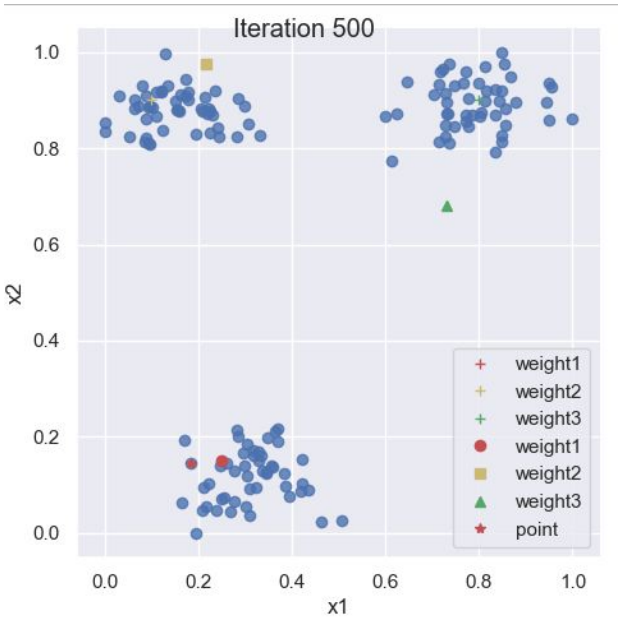

Iteration 500
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point





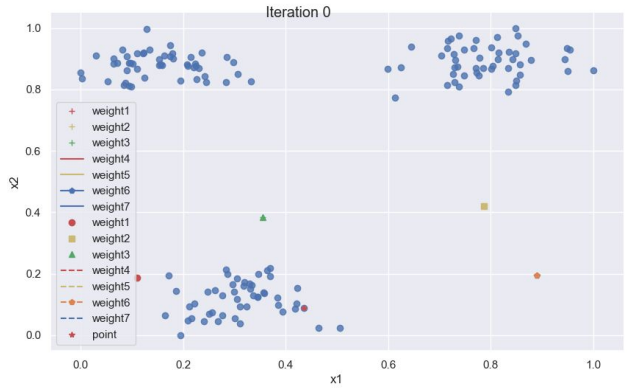
With manual imputation of weights with a different alpha of .5 and an iteration of 500

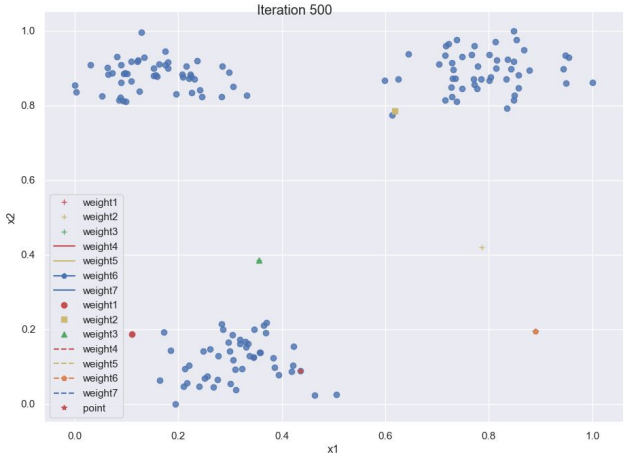

Iteration 0
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point


Iteration 500
1.0
0.8
0.6
0.4
0.2
0.0
x2
0.0
0.2
0.4
0.6
0.8
1.0
x1
weight1
weight2
weight3
weight1
weight2
weight3
point

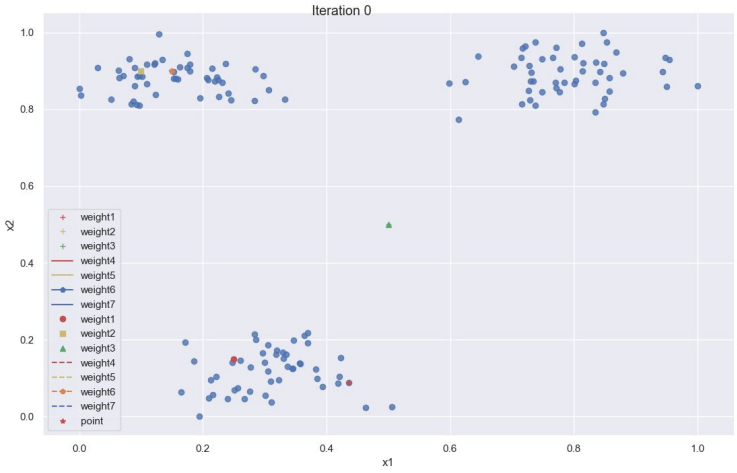
7 Neurons

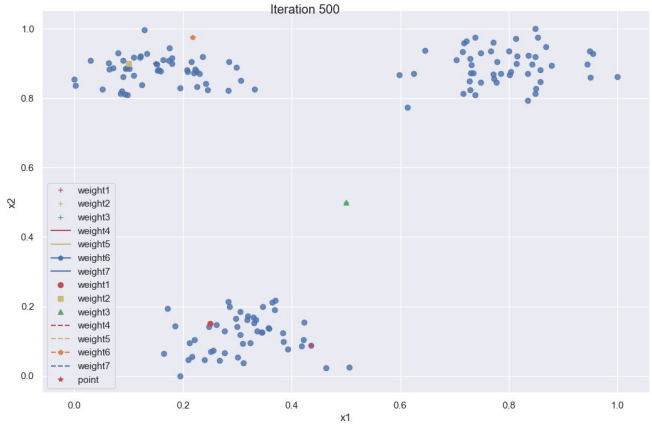
With 7 neutrons the weights don’t move much due to the fact that the net for all the weights is similar as there is more neurons than clusters. I placed a net in the middle manually to show that it will not move at all if it never wins vs the others.

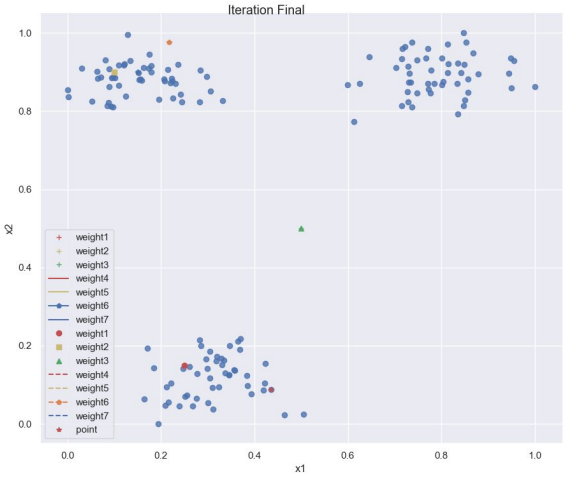




Manual weights




Iteration 500
x1
x2
weight1
weight2
weight3
weight4
weight5
weight6
weight7
weight1
weight2
weight3
weight4
weight5
weight6
weight7
point


Iteration Final
x1
x2
weight1
weight2
weight3
weight4
weight5
weight6
weight7
weight1
weight2
weight3
weight4
weight5
weight6
weight7
point