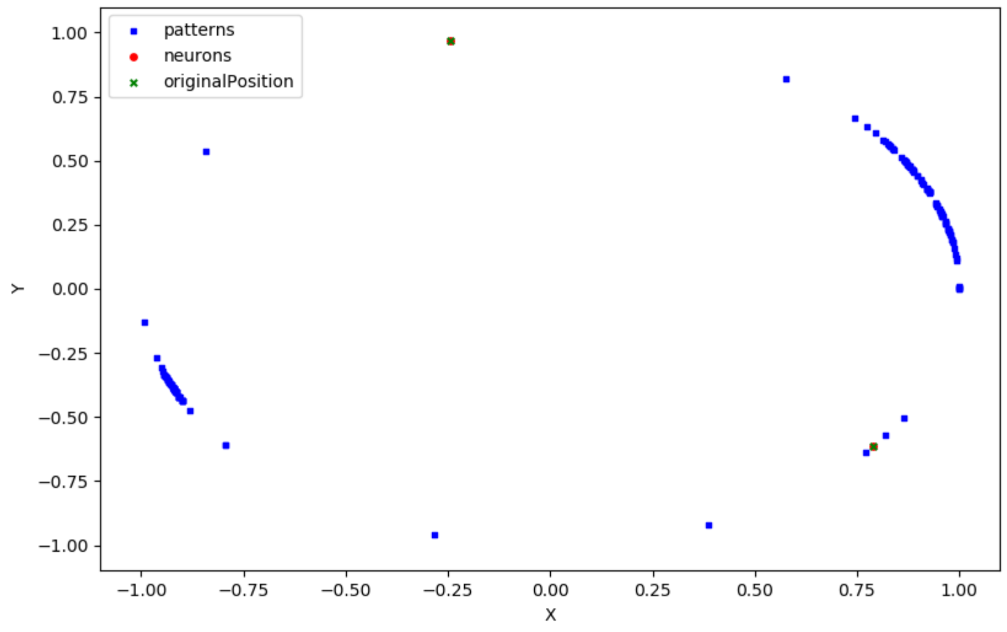
Kyle Hammerschmidt

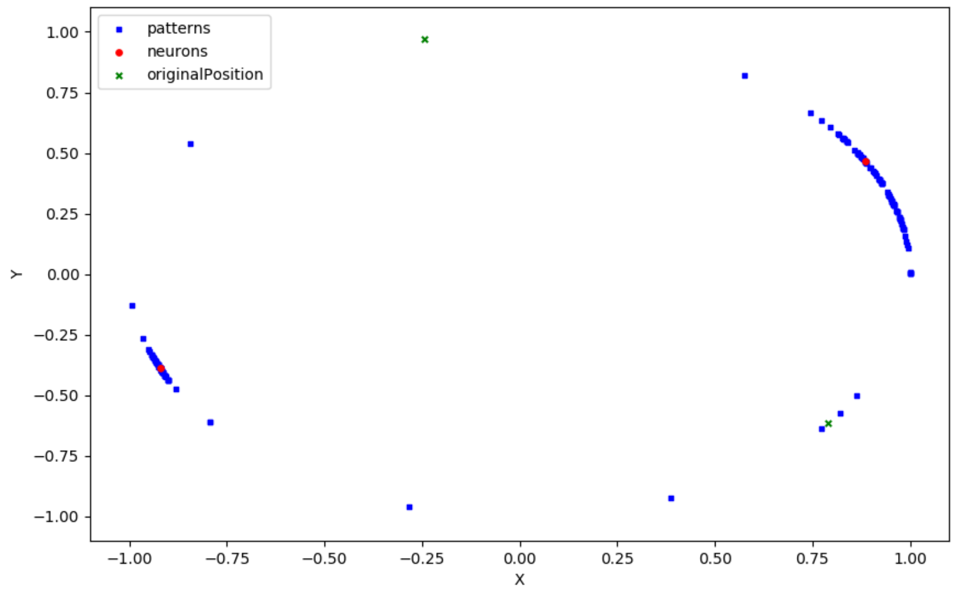
409 Final

Question 1.

Here is a plot of the graph with the data and random weights assigned to the neurons.

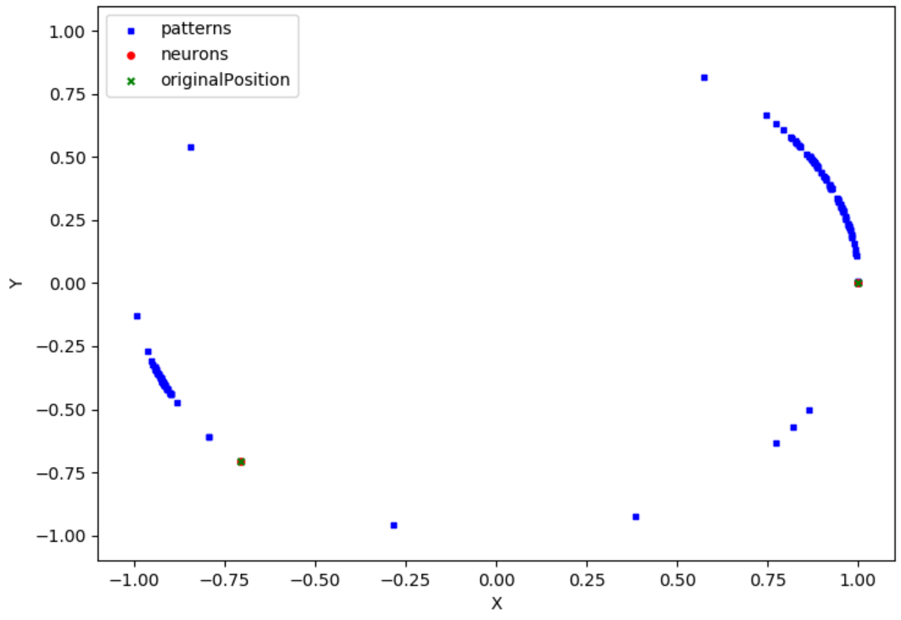


Here is a plot after the calculations are complete.

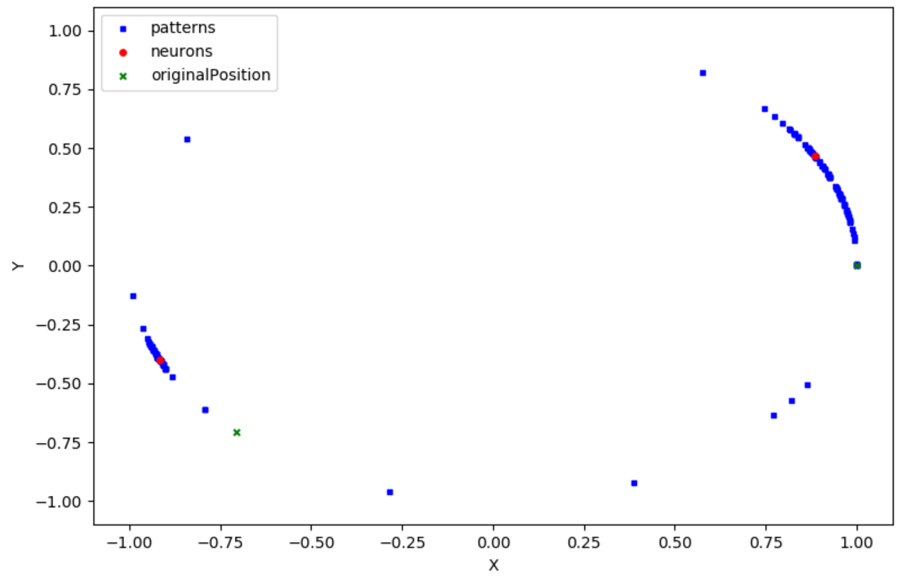


It appears that the neurons had no problems finding the clusters. This is with an alpha of 0.1 after 1 iteration. As the patterns are applied it appears one of the neurons will move for a few iterations and as it gets closer the other neuron will begin moving. So it appears there is a back and forth of each neuron moving. This due to the way the data is organized. If the order was changed then the neuron that would move would change.

Here is a plot of chosen weights.

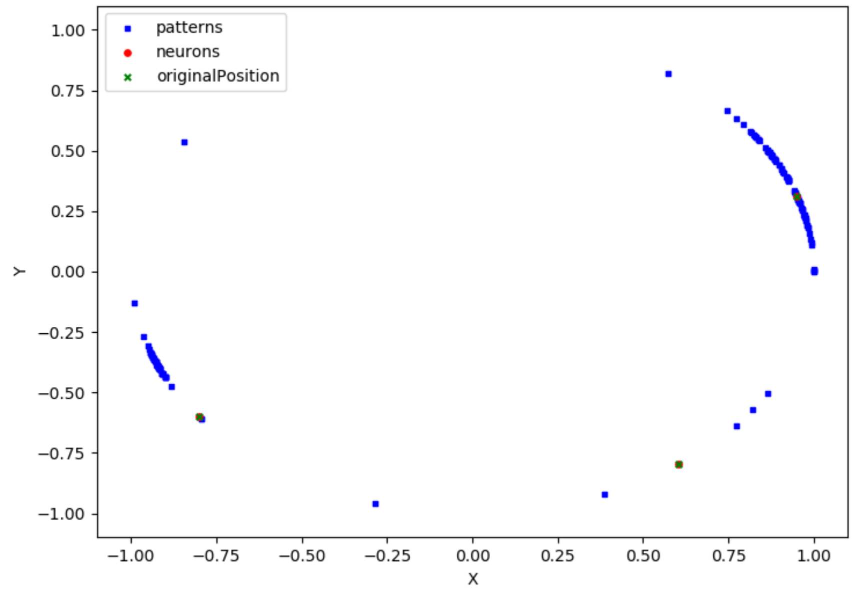


Here is a plot after they have converged.

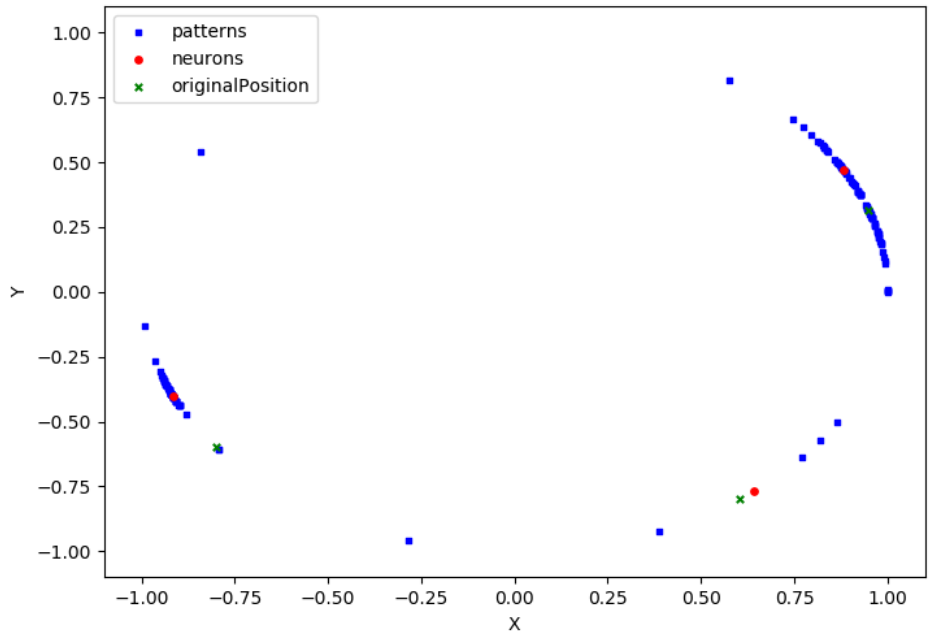


I decided to start the weights closer to the known clusters and it appeared they were able to find the clusters in one iteration. Watching the neurons move it appears one will move for about 10 steps then there will be a back and forth then one will stop moving and the other will take over. It seemed like just one iteration was enough for both the random weights and the assigned weights. Also having an alpha of 0.1 seemed to work perfectly so no adjustments were made.

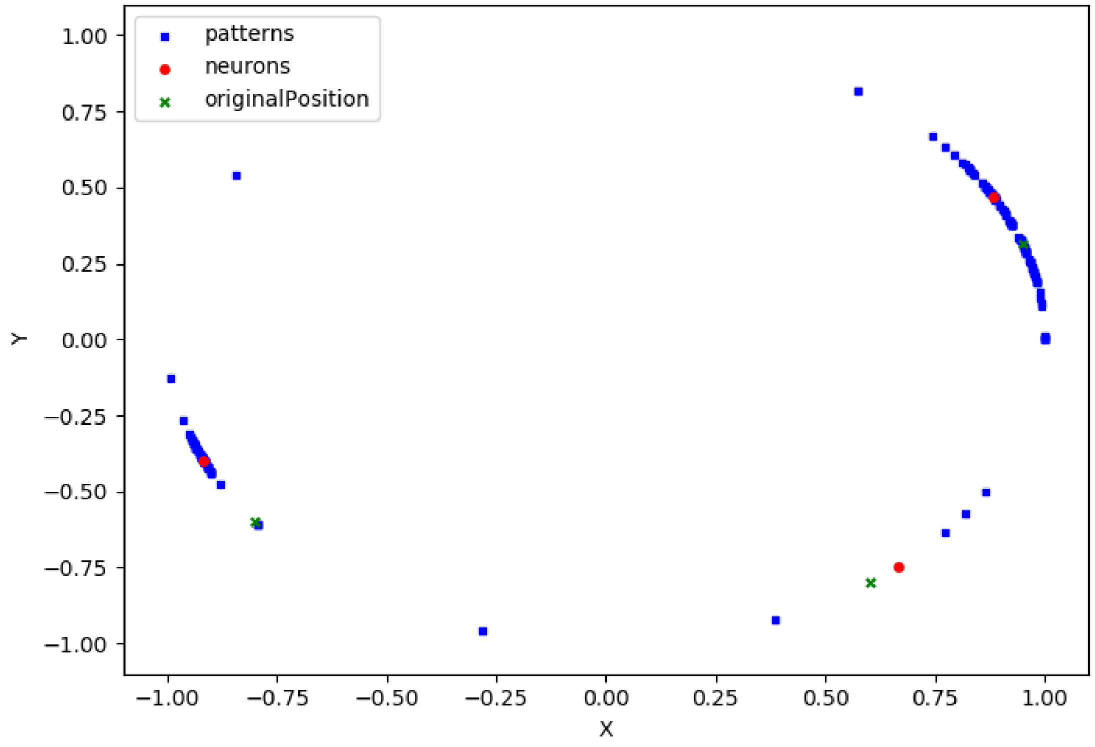
Here is a plot of 3 neurons with determined weights.



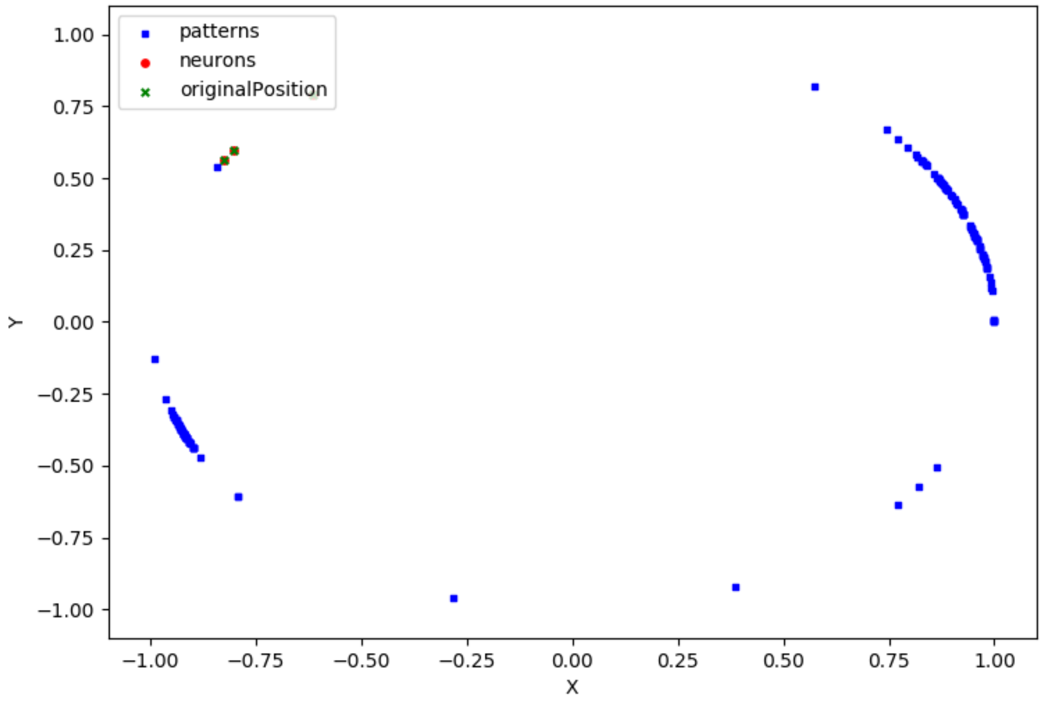
After 1 iteration.



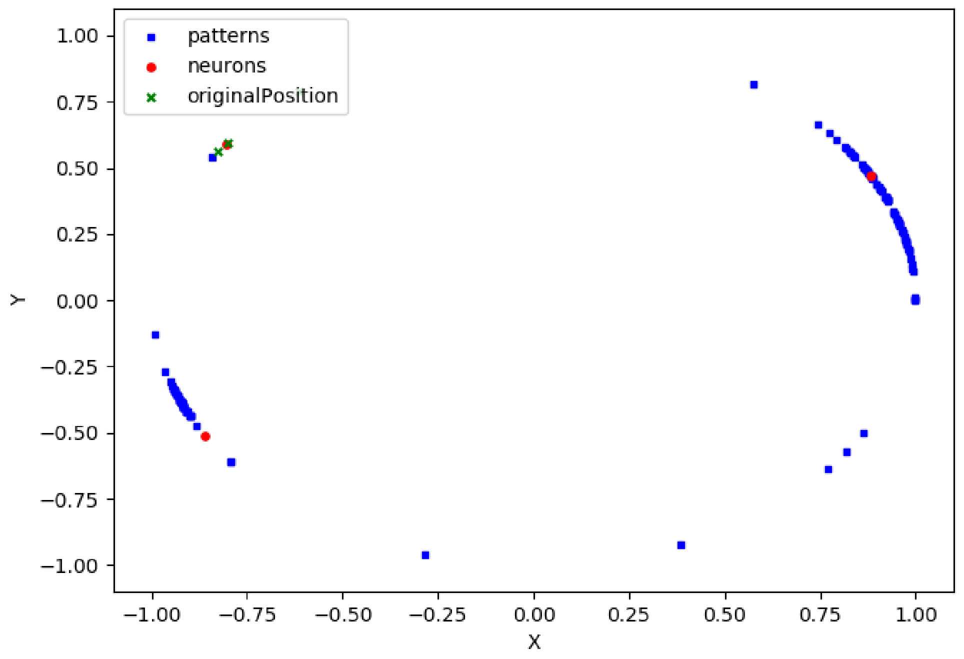
After 2 iterations.



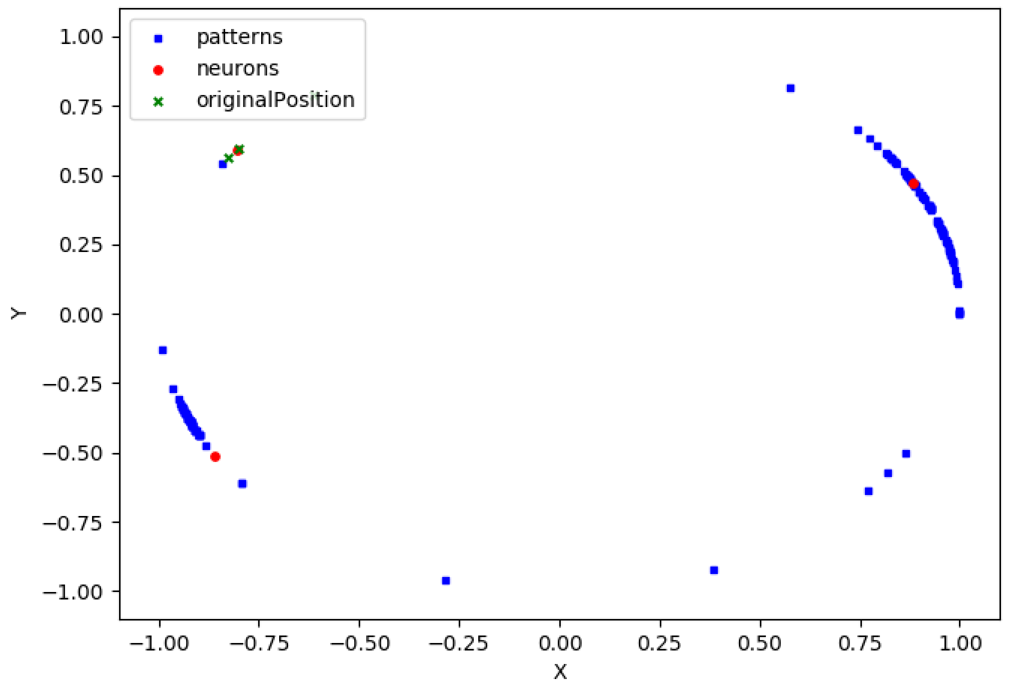
Here is a plot of 3 neurons with random determined weights.



After 1 iteration.



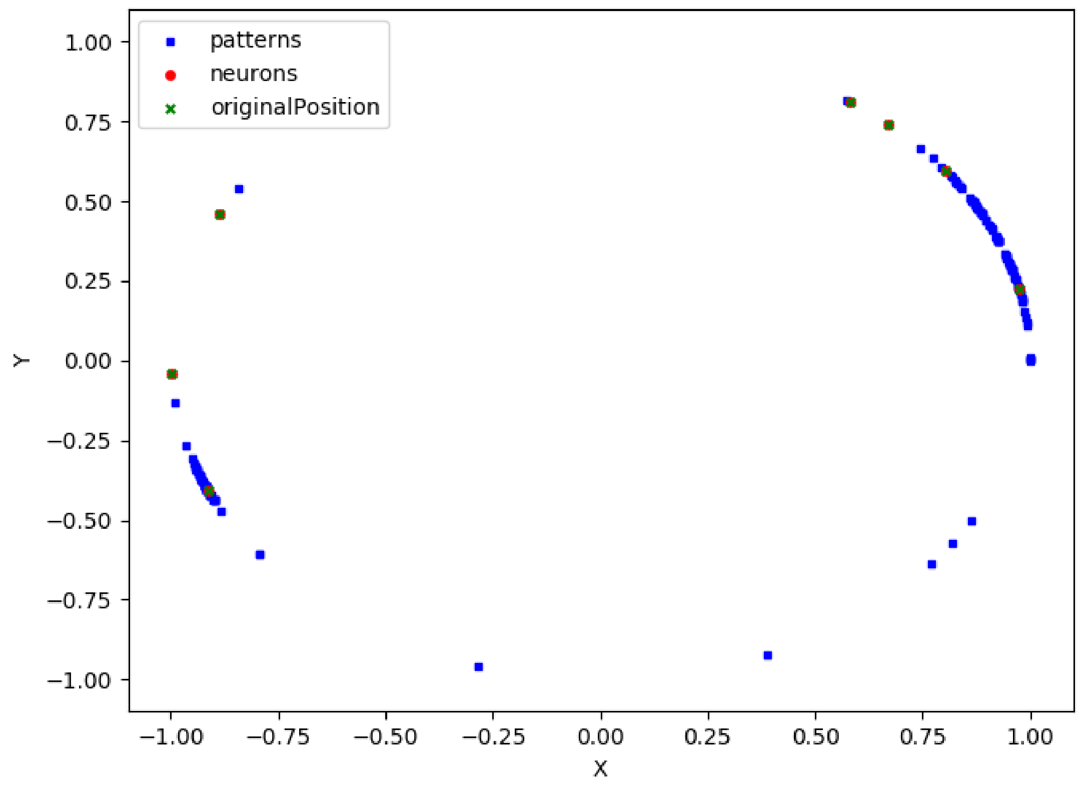
After 2 iterations.



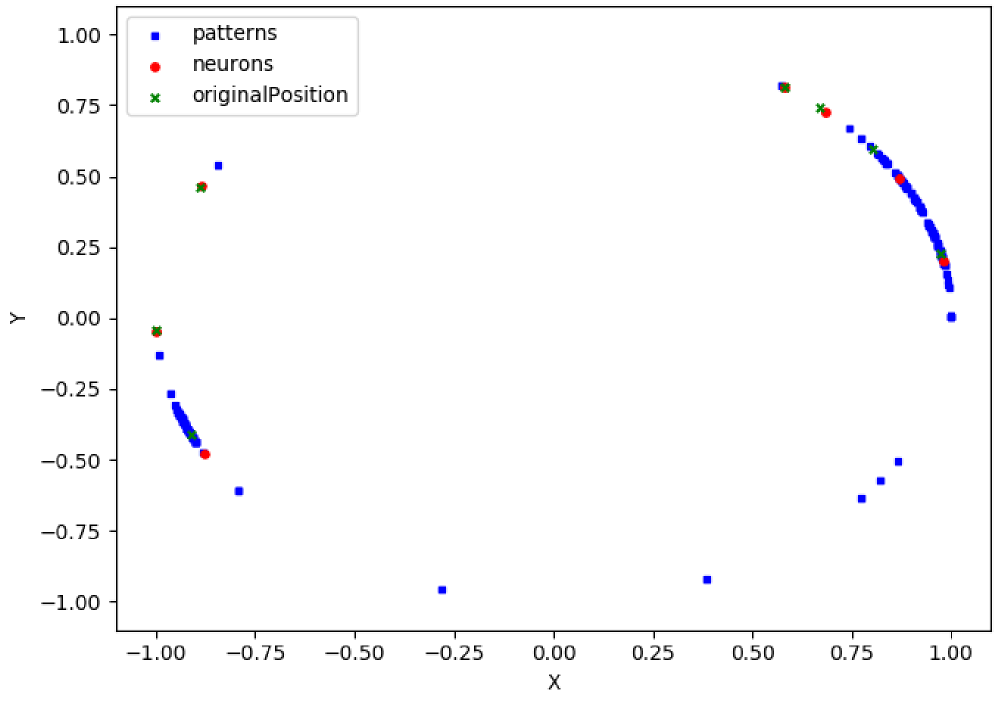
What appears to happen here is one of the three neurons will begin to starve. In the beginning one of the neurons will begin to move and get close to a cluster. Then a second will move towards another cluster. Then what happens is the original first will begin to move again. The third neuron will begin to gravitate towards one or two patterns that are far away from the other clusters. Sometimes a pair of neurons will share a cluster and be setup on either ends of it. For these neurons to be effective it took 2 iterations with an alpha of .01. It appeared it took a longer time for the neurons to latch onto a cluster compared to just the 2 neurons. Due to the competition.

Here is a plot with 7 neurons,

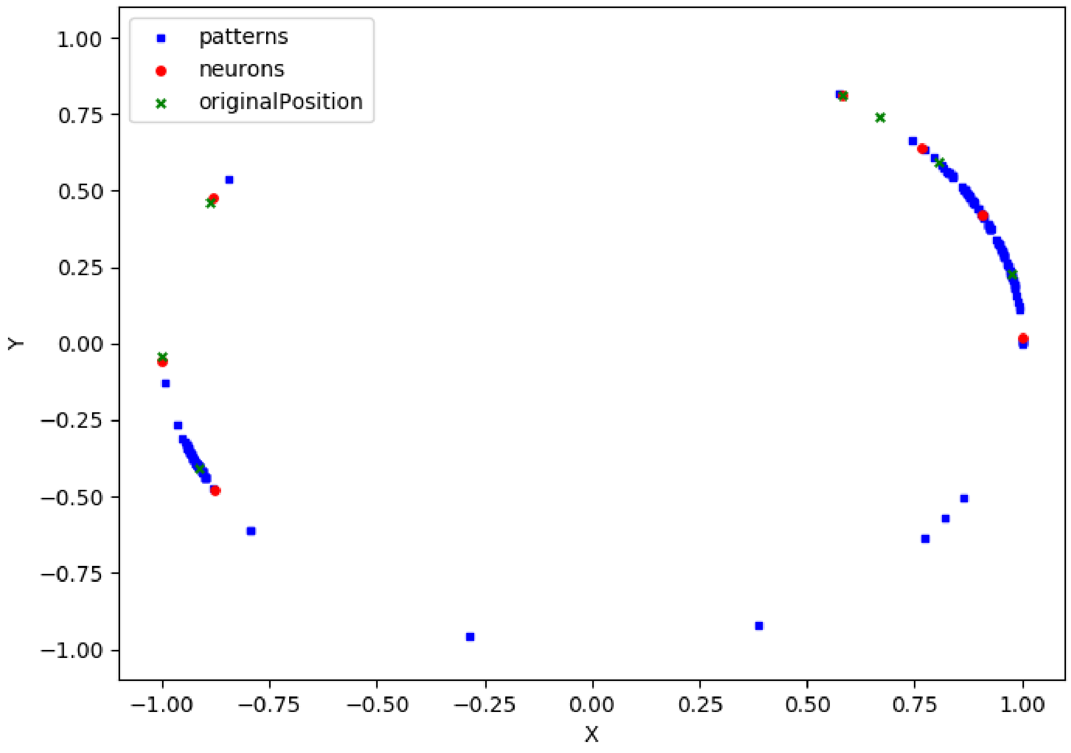
Here is after 1 iteration.



After 2.



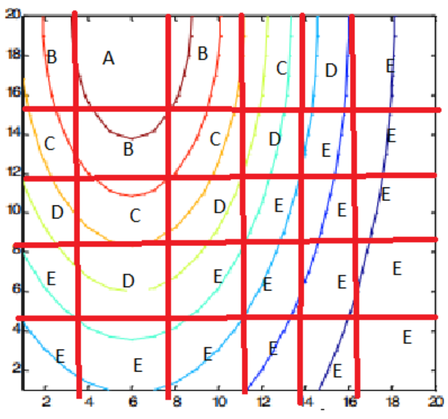
After 3.

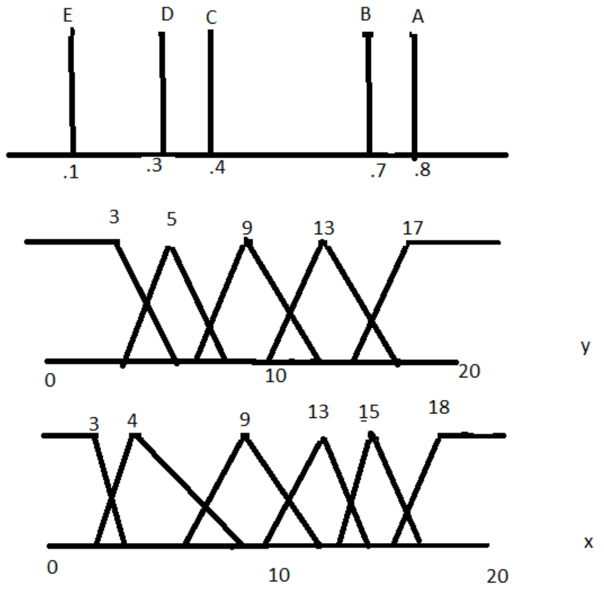


For the 7 neurons it didn't really matter if the weights were randomized or preselected since the data only has 2 large clusters. Whether random or predetermined behaved the same way. It appeared that one of the neurons would dominate in the beginning and get very close to one cluster. Then 2 or 3 of the neurons would begin to move and one of them would pull ahead and begin winning the net value. Then another set of neurons would compete and one of them would pull ahead. This continued over the course of 3 iterations. 3 or 4 iterations seemed optimal in making sure all of the neurons had a chance to move. This is still with an alpha of .01. Once the clusters had been claimed by a few of the neurons the rest would just gravitate towards a single pattern. Sometimes if neurons spawned on either end of a cluster they would appear to share it after a couple of iterations. This is clearly too many neurons for this many patterns and clusters. It seemed like 3 neurons with predetermined positions was optimal for this data set.

Question 3.

Here is my fuzzy controller.



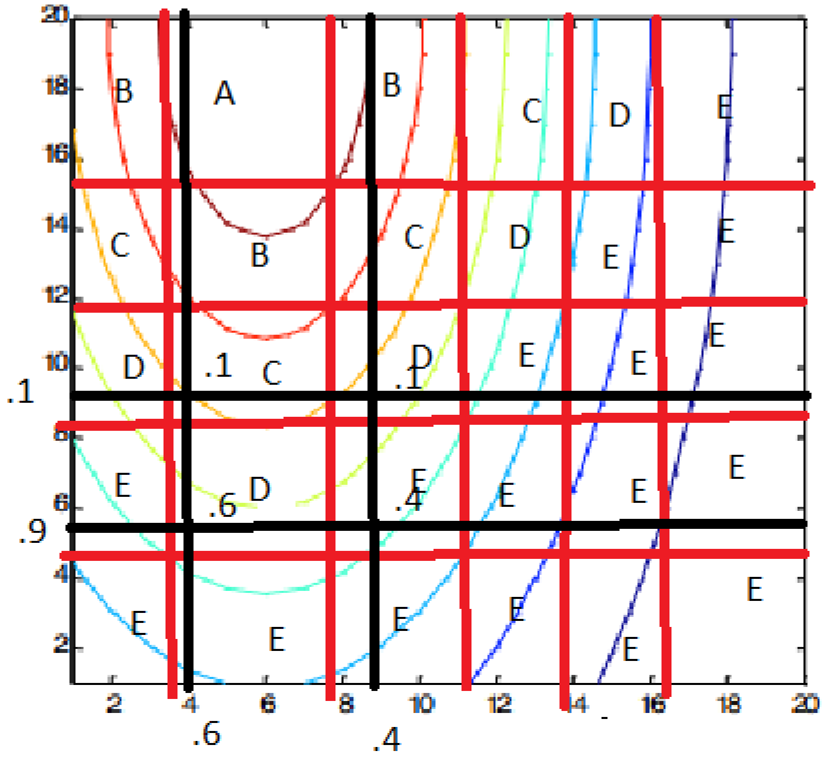


I chose the values based of my perception of the problem and how I thought the numbers should be assigned.

Point 1. 6,6

X = .6 in 4 and .4 in 9

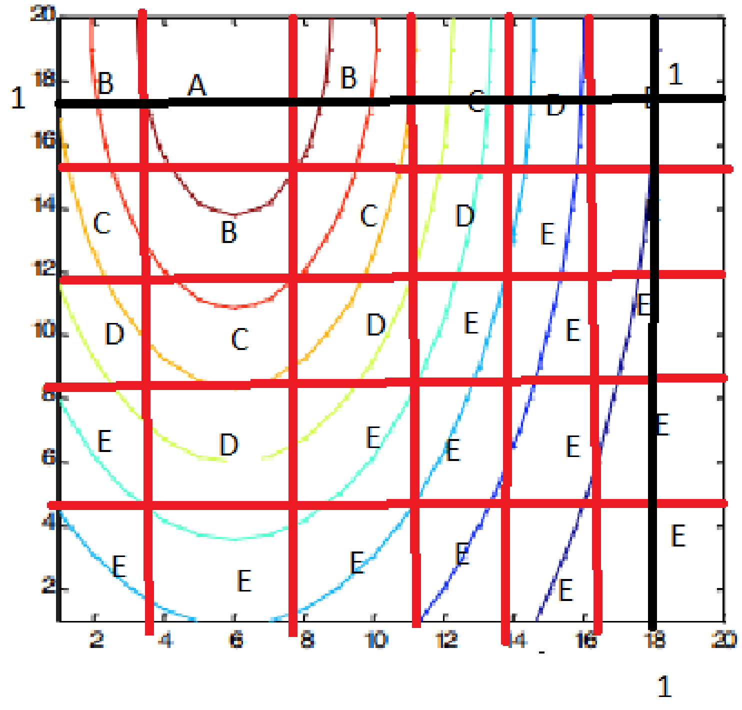
Y = .9 in 5 and .1 in 9



$$\frac{.6(D) + .4(E) + .1(C)}{.6 + .4 + .1} = 0.236$$

D = .3, E = .1, C = .4

Point 18,18

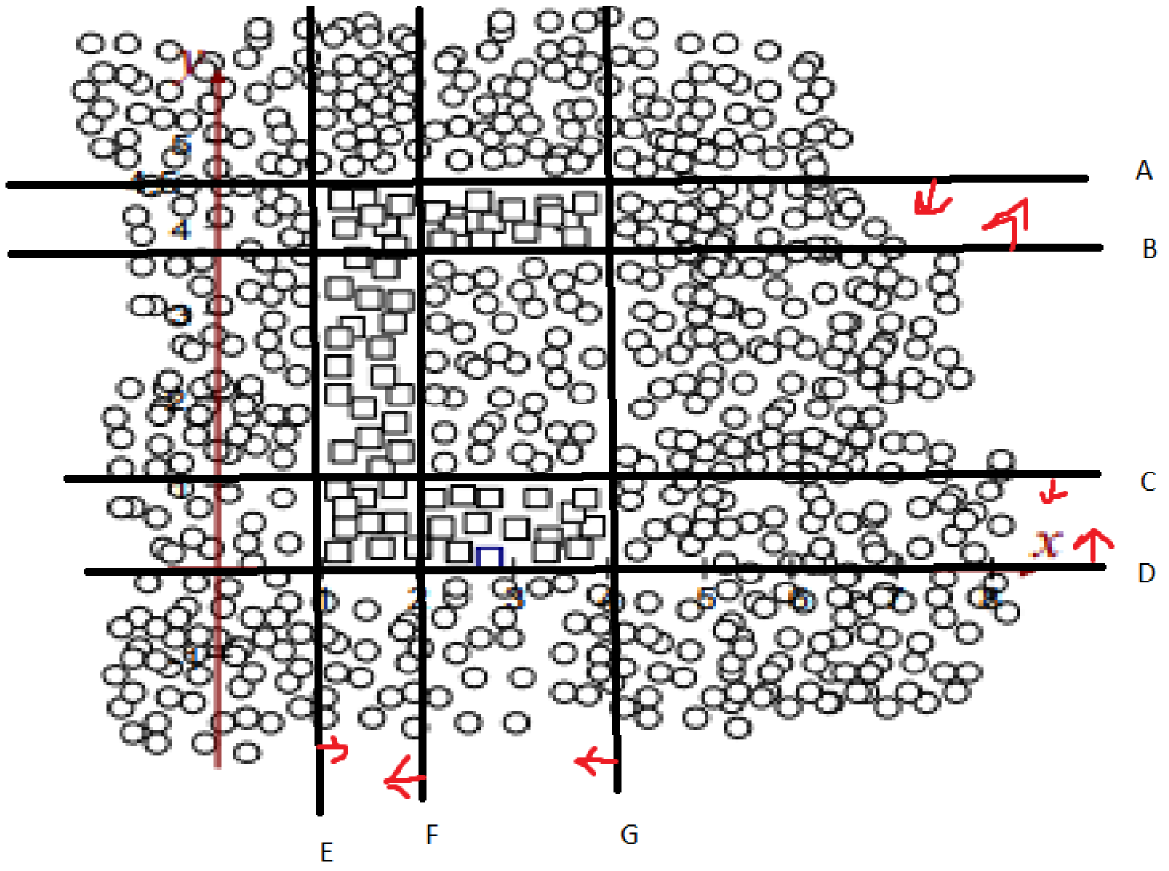


$$\frac{1(E)}{1} = .1$$

Since the only point it can exist in is E it is clearly part of E.

I am having trouble getting MATLAB working / anything else working for the purpose of this problem. But based off some examples I have found I can assume that my model will be relatively similar to the required control surface area. It appears that a triangle membership function is usually the best of both worlds regarding computation expense and similarity to the required. I can't comment on the error of my graph but I would believe it wouldn't be very large.

Question 4,



A = -1y+4.5 > 0

B = y - 3.5 > 0

C = -1y + 1 > 0

D = y > 0

E = x -1 > 0

F = -1x + 2 > 0

G = -1x +4 > 0

H = 1x + 1 y -1.5 > 0

N = 1x + 1y -1.5 > 0

J = 1x + 1y -1.5 > 0

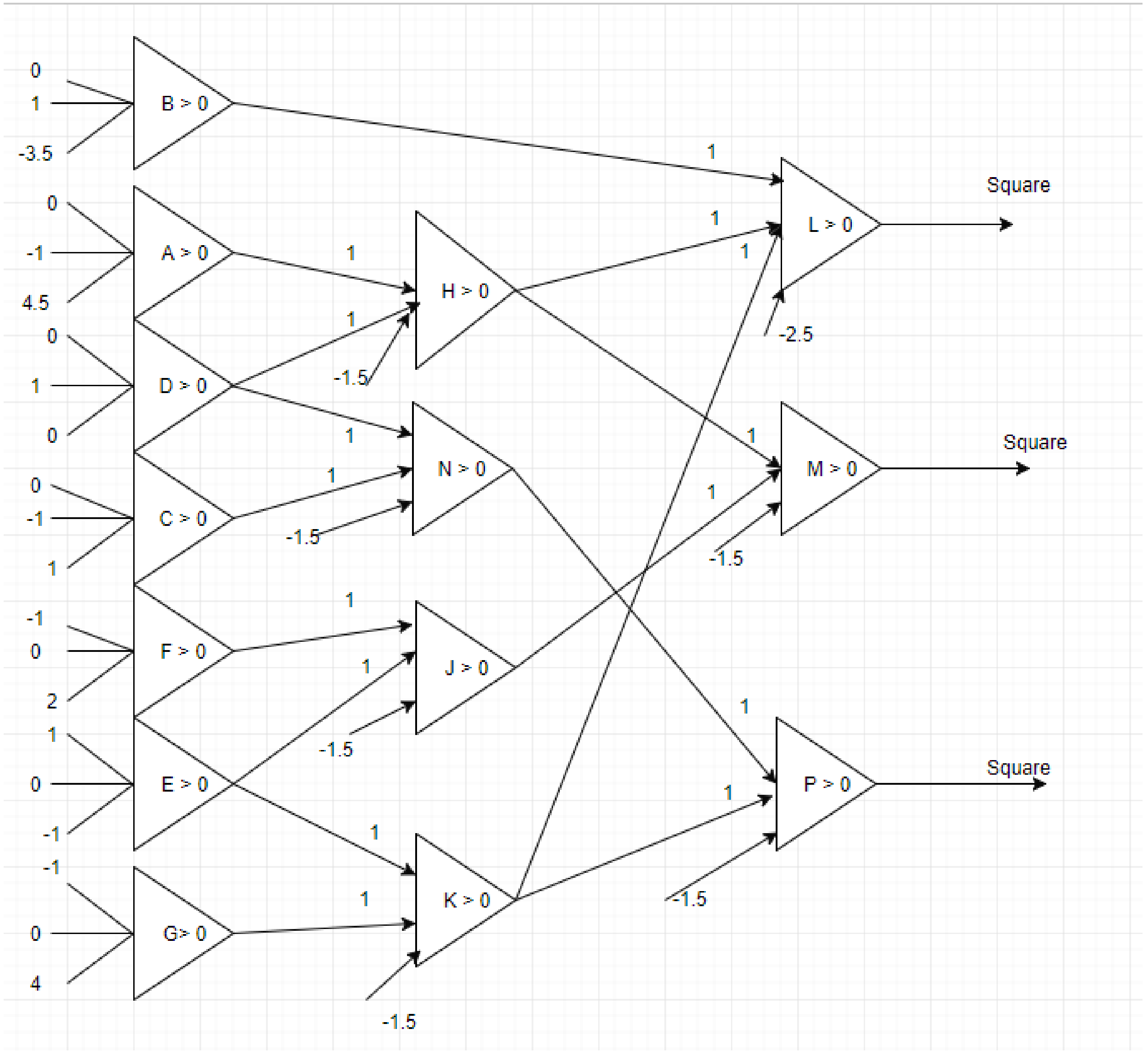
K = 1x + 1y -1.5 > 0

L = 1x + 1y + 1z -2.5 > 0

M = 1x + 1y -1.5 > 0

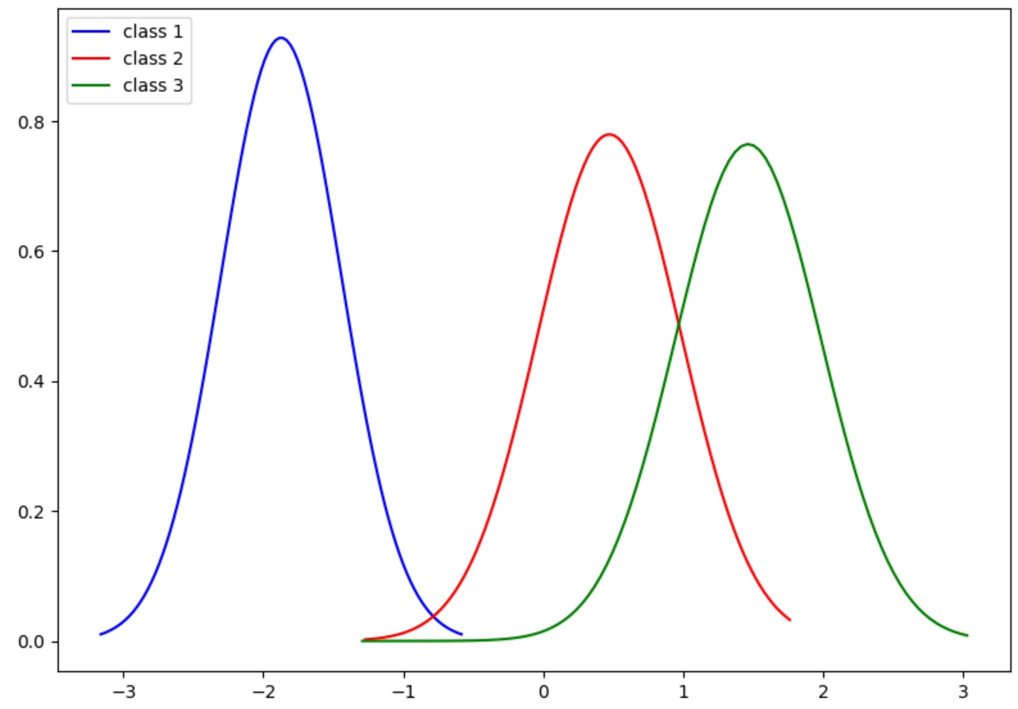
P = 1x +_1y -1.5 > 0

Hard Transfer Function



The network was a bit hard to design due to the fact of such tight data. Another issue was the area in the middle of the squares that are circles. So designing a network around that took a bit of thought. Once the neurons were drawn onto the graph it was easy to decide what to put where and how to draw the diagram. Also it is assumed that the input is x, y, bias for the diagram starting top to bottom on the inputs.

Bonus



This is the plot of the probability distributions. It looks like there is a lot of overlap between class 2 and 3. When I ran my program it actually had 100% accuracy and didn't misclassify anything despite that overlap. It is very interesting that such a simple algorithm has such a high accuracy. I was worried about underflow for some of the calculations but there didn't seem to be a problem. The Bayes algorithm is very simple yet very powerful.