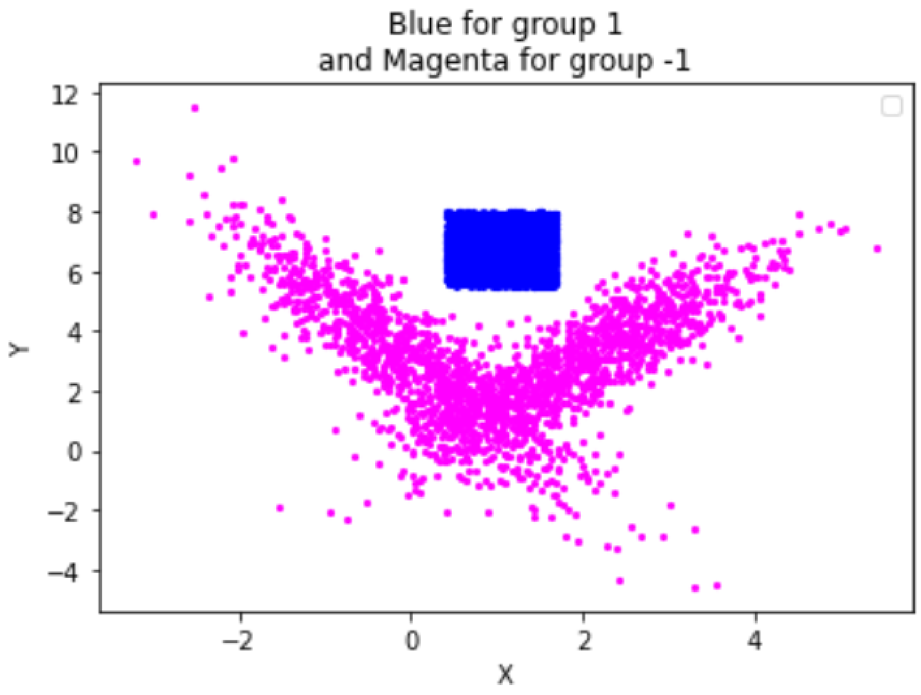
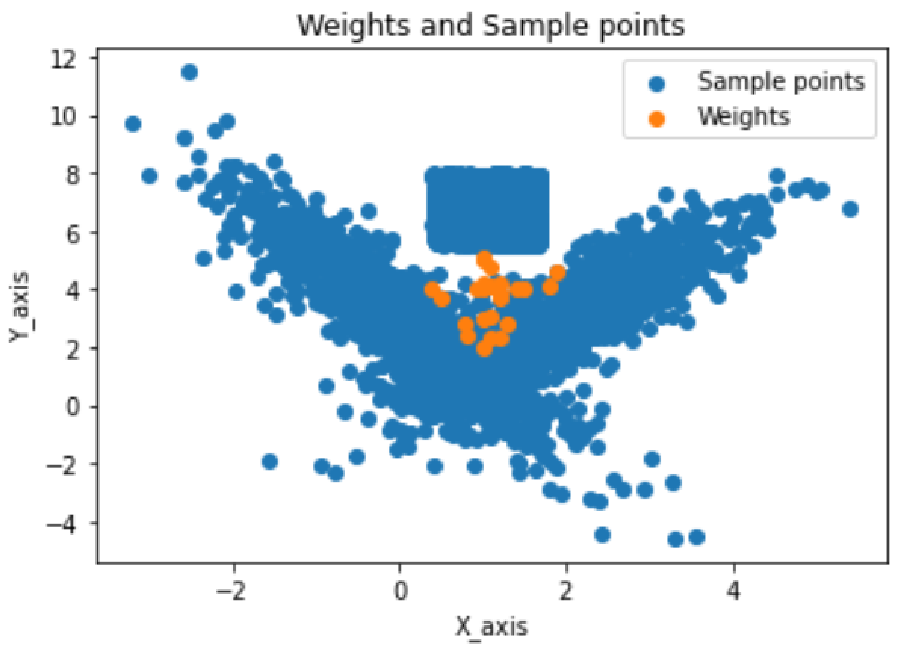
Homework 3:
Name: Brandon Eyongherok

1.)

A) Note: A 3 by 3 and a 5 by 5 lattice structure were used in the homework for the Kohonen Self Organizing Map.
Summarize that the 5 by 5 SOM provided a better representation for the sample points than the 3 by 3 lattice SOM because the 5 by 5 had more weights that could provide a better representation of the sample points. However, the 5 by 5 took longer to learn than the 3 by 3 SOM because it had more weights to train.

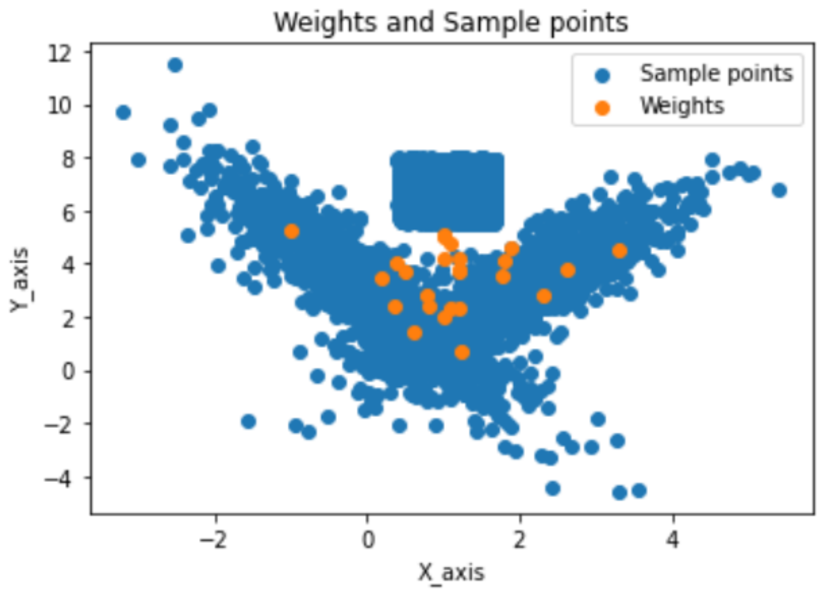


The homework started off with the graph above, that does a good job of encapsulating the density of the RBF dataset. Points where late chosen at the center between the two groups as illustrated in the group below. In the case below, a 5 by 5 square lattice. The 5 by 5 SOM took 280 epoch while the 3 by 3 SOM took 80 epoch to fit the sample points.

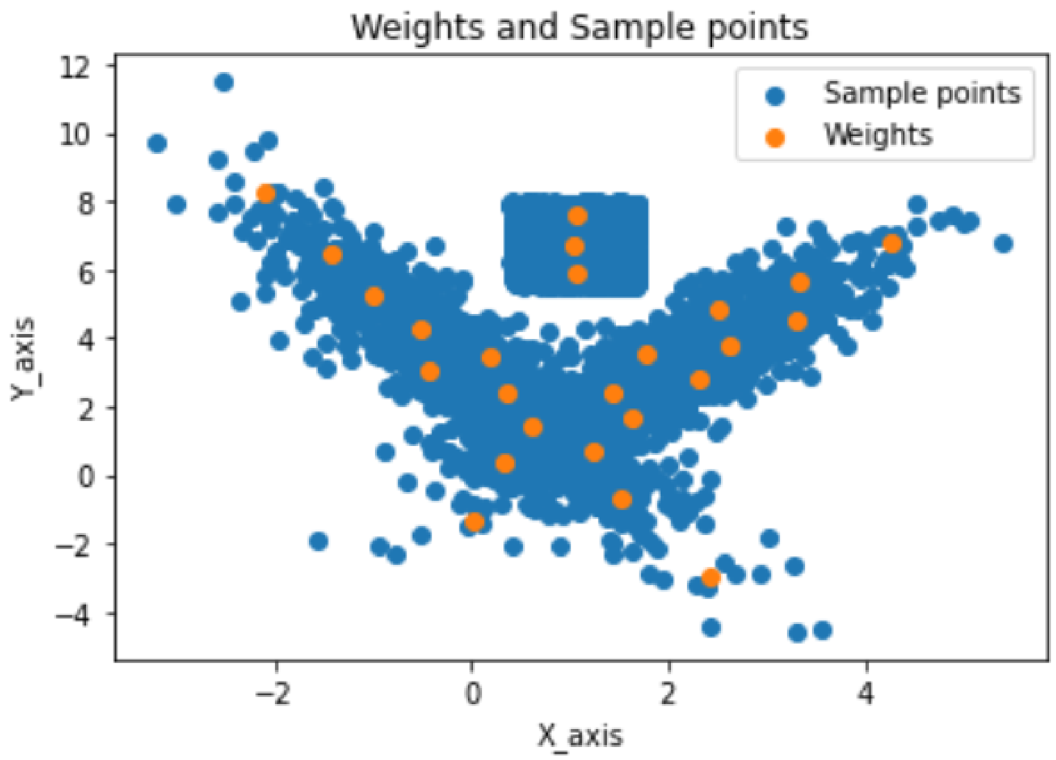


The points were chosen as shown above to ensure the weights would on average not travel too far to fit the sample points. The weights are later tracked during training to show how they move to fit the sample points.

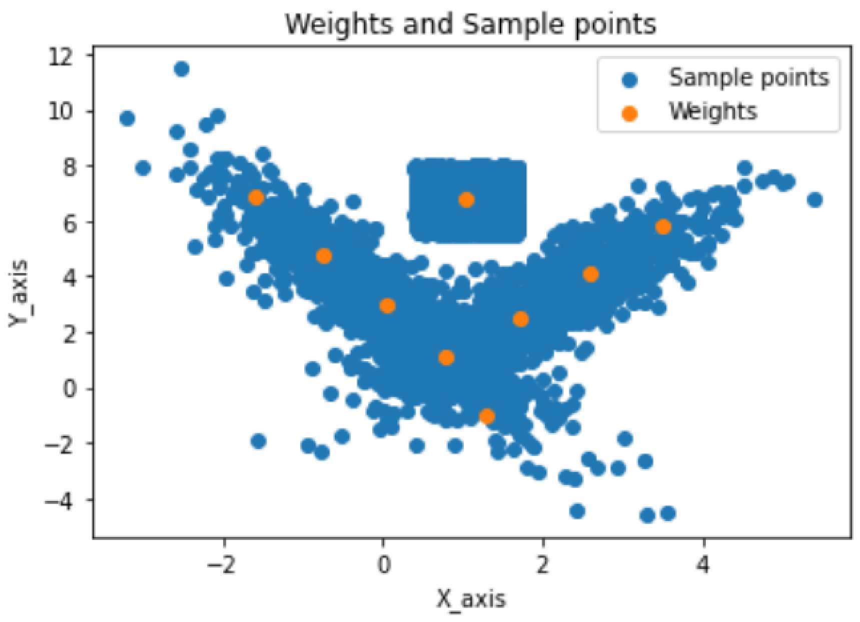
```
140
```



The graph above shows the weights movement at epoch 140 during training.

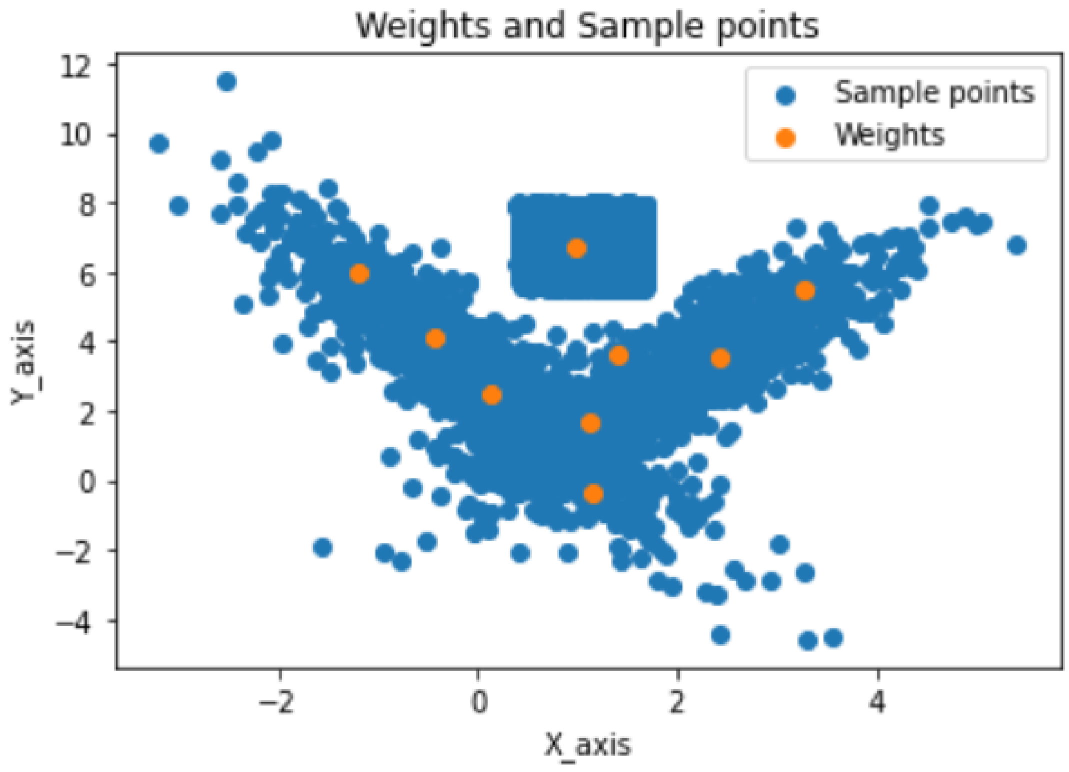


The weights and sample points fit the data better than its 3 by 3 square lattice counterpart below, because more weights means it can accurately represent all the structure of the sample points.



The 3 by 3 square lattice shown below at epoch 1.

1



After the very first epoch that weighs almost fits the entire sample points showing how quickly the smaller SOM learns.

B)

i) For the **3 by 3** which applies to the 5 by 5**:**

Calculation begins with the weight update formulate which has the learning rate and neighborhood function.

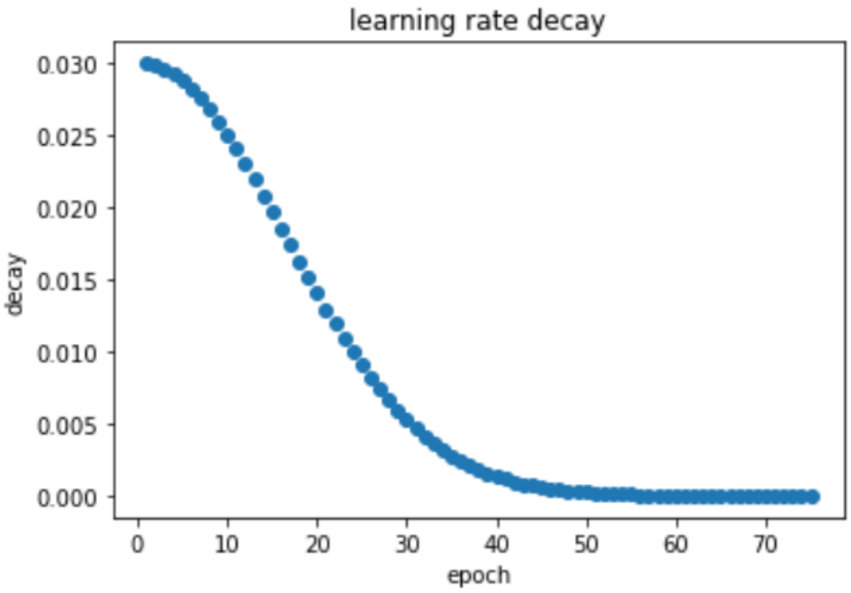
$$w_j(n+1) = w_j(n) + \eta(n)\underbrace{\left(h_{ji(x)}(n)\right)}_{\text{neighborhood function}}\left(x(n) - w_j(n)\right)$$

The learning rate: For the learning rate decay, different time constants were tried by which the learning rate decayed by corresponding with the epoch to get a decaying parameter that will result in the weights accurately fitting the sample points.
The learning rate decay equation:

$$\eta(n) = \eta_0 e^{-\left(\frac{n}{\tau}\right)}$$

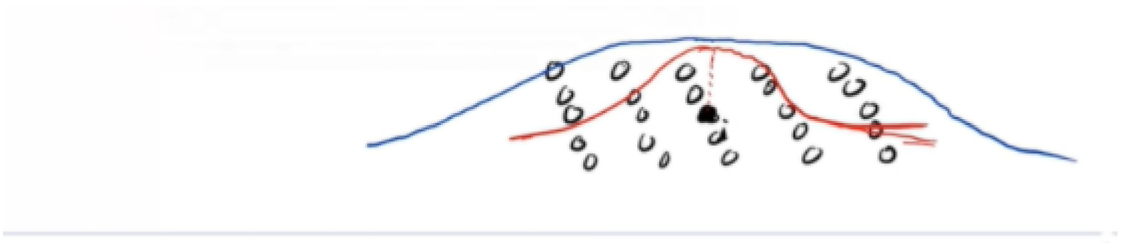
iteration or epoch #

time delay index user defined



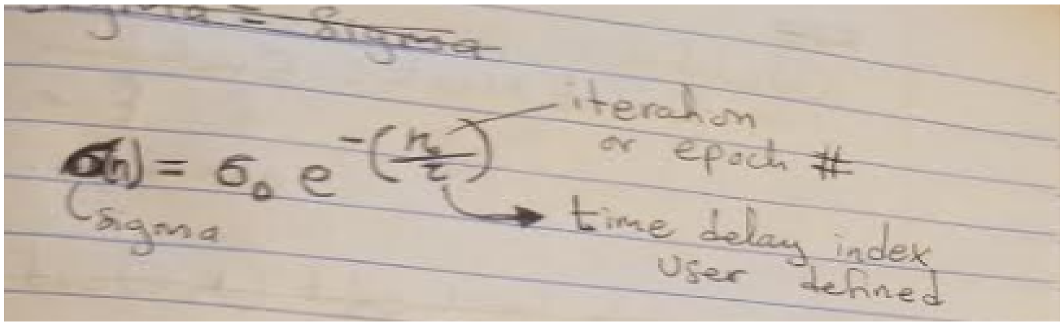
The learning rate decays to zero, shows that the weights at about the seventy epoch during training have stopped moving. As shown on the graph above for a 3 by 3 lattice. Note the 3 by 3 lattice learning rate and sigma decay occurs faster than the 5 by 5 lattice structure. This means it takes less time for the 3 by 3 lattice structure to fit the data although it does not have enough weight to more accurately represent the sample point like the 5 by 5 lattice. This means when the learning rate decays to zero, it has very little impact on the movement of the weights and the weights have arrived at their final destination.

Sigma decay: For the Sigma decay, different time constants were tried to give a sigma which led the weights to more accurately fit the sample points. The sigma function is tied to the neighborhood function in the weight update formula. The sigma function was designed to have a huge effect on the winning weight and its surrounding weights, but over time had diminishing effect on the surrounding node and in the end diminish effect on the winning node like a gaussian graph getting smaller in the example below.



The example illustrated above has the winning weight in black and the blue line shows how strong an impact in the beginning the winning weight has on its surrounding neighbor in the square grid, but the red line shows how as the epoch increases, the winning weight has less and less of an impact on its surrounding weight until it has no impact on the surrounding weight and finally, until it has no impact on the winning node. It is important to understand this is all due to the decay of the sigma in the neighborhood function.

Sigma decay equation:



~~sigma = sigma~~

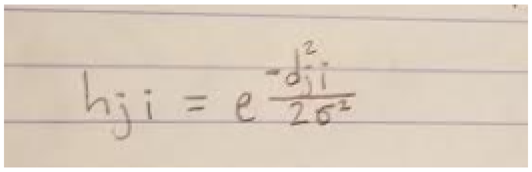
$$\sigma(n) = \sigma_0 e^{-\left(\frac{n}{\tau}\right)}$$

Sigma

iteration or epoch #

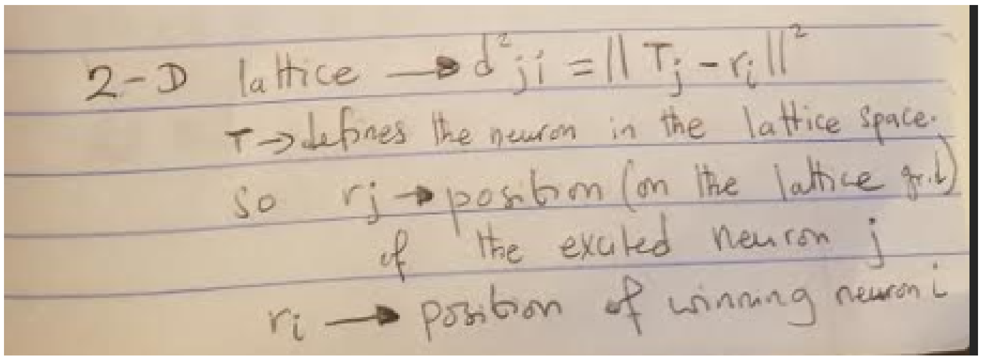
time delay index User defined

Equation for the neighborhood function:



$$h_{ji} = e^{\frac{-d_{ji}^2}{2\sigma^2}}$$

Explanation:

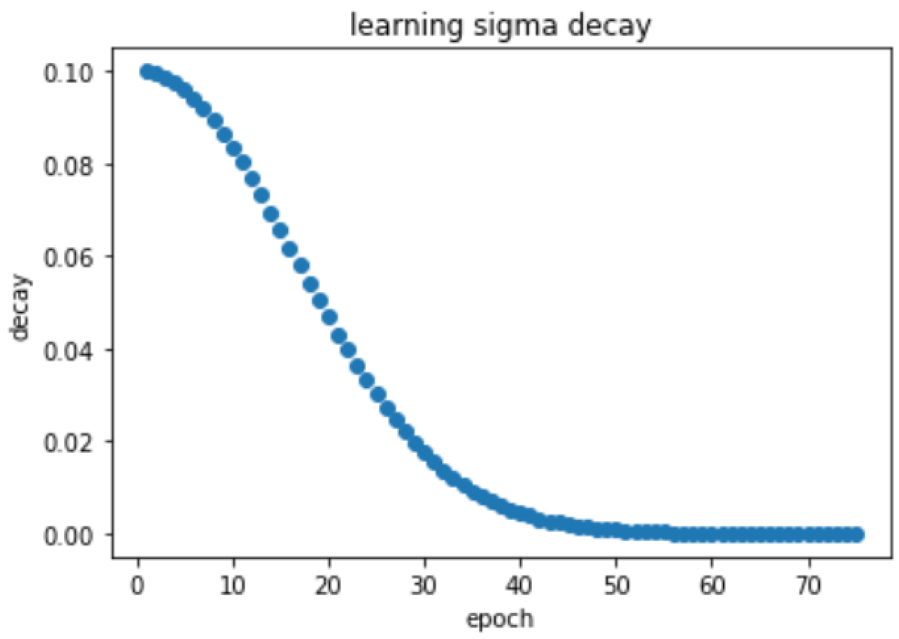


2-D lattice $\longrightarrow d^2_{ji} = \| T_j - r_i \|^2$

$T \rightarrow$ defines the neuron in the lattice space.

So $r_j \rightarrow$ position (on the lattice grid) of the excited neuron j

$r_i \longrightarrow$ position of winning neuron i



The sigma decreases to zero shows that the neighborhood function had a lot of impact on the movement of weights far from the winning weight, but overtime and less and less of an impact on the neighboring weights and in the end, had zero impact in the movement, even in the winning weight. As shown on the graph above for a 3 by 3 lattice.

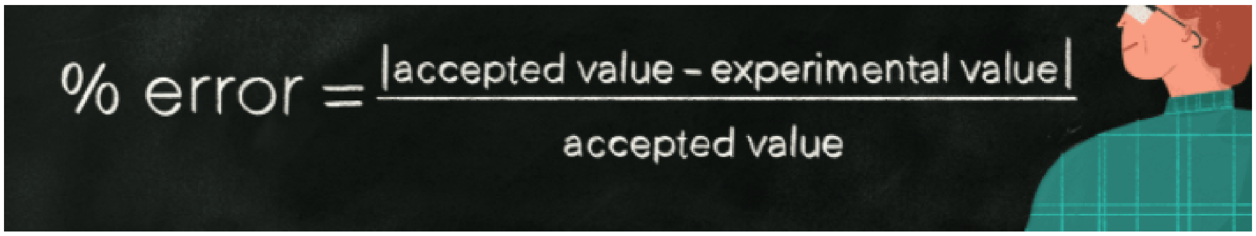
For the 3 by 3 lattice structure the epoch ends at 74 including zero:

The percent of error for each weight at the last epoch for the 3 by 3 lattice structure.

```
print the error for each weight
[array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.])]
```

Calculated by:

Derive from:

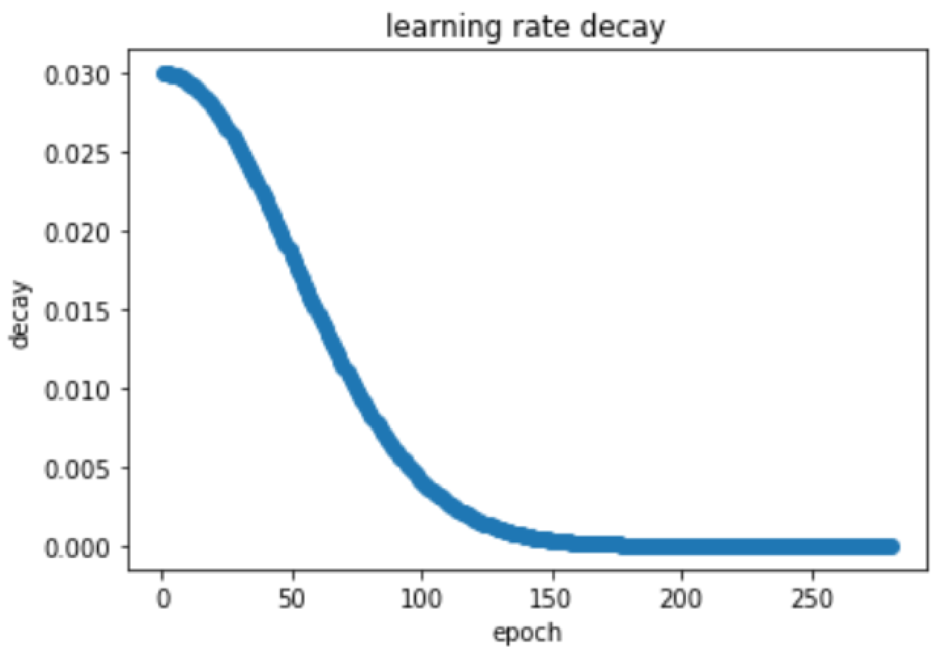


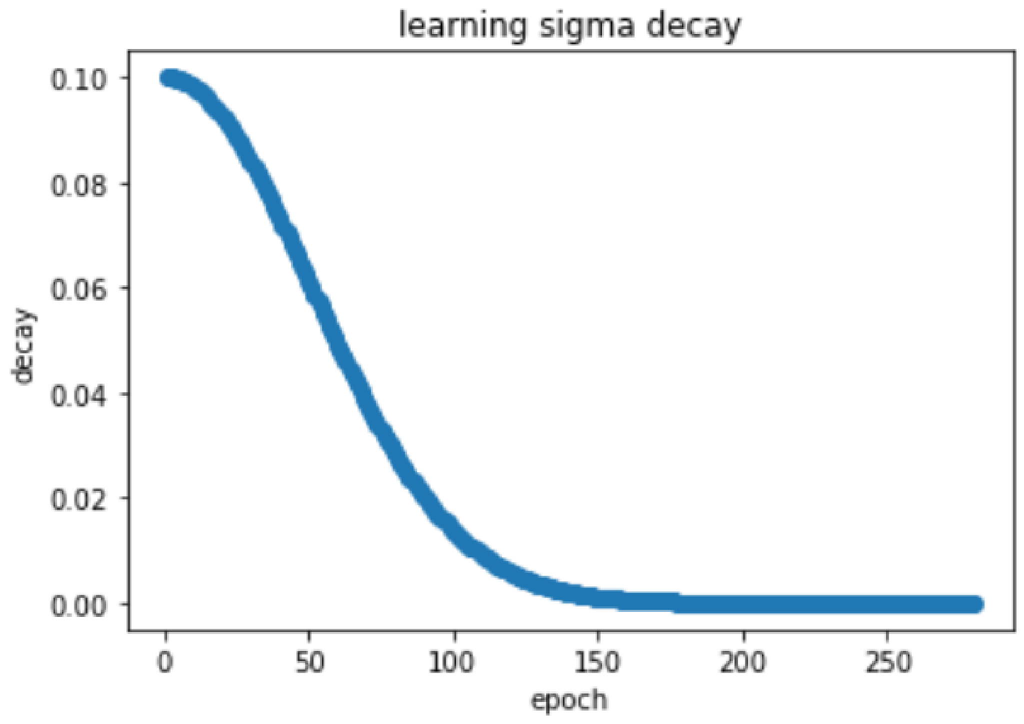
$$\% \text{ error} = \frac{|\text{accepted value} - \text{experimental value}|}{\text{accepted value}}$$

Where w[j] is accepted and oldw[j] is the experimental value.

Likewise for a **5 by 5 lattice structure:**

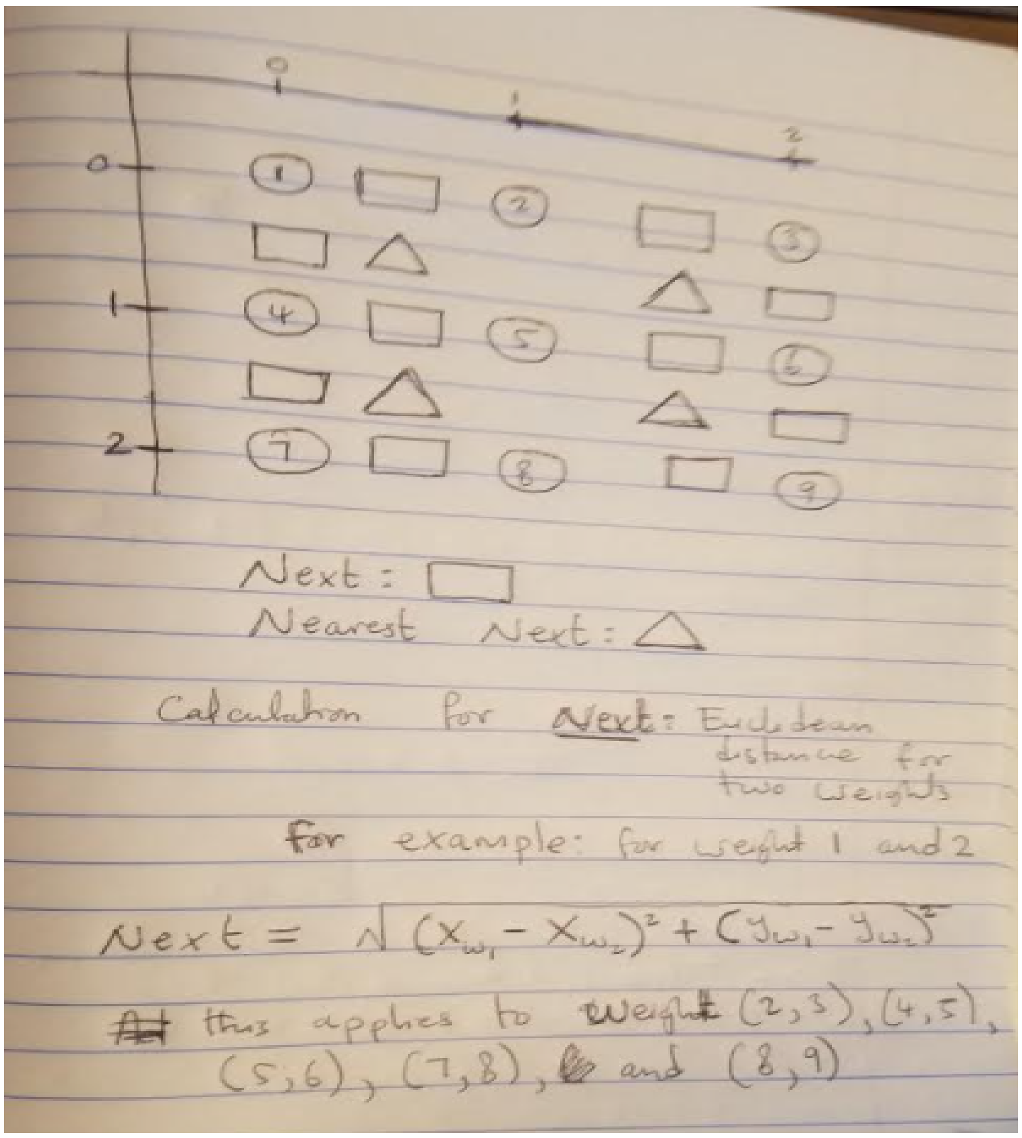
```
print the error for each weight
[array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array
([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.]), ar
ray([0.]), array([0.]), array([0.]), array([0.]), array([0.]), array([0.])]
```





ii) As shown in part A, the graphs of the two lattice structure track from the beginning to the final and reiterated: A 3 by 3 and a 5 by 5 lattice structure were used in the homework for the Kohonen Self Organizing Map.
Summarize that the 5 by 5 SOM provided a better representation for the sample points than the 3 by 3 lattice SOM because the 5 by 5 had more weights that could provide a better representation of the sample points. However, the 5 by 5 took longer to learn than the 3 by 3 SOM because it had more weights to train. In the case below, a 5 by 5 square lattice. The 5 by 5 SOM took 280 epoch while the 3 by 3 SOM took 80 epoch to fit the sample points.

iv) Calculation of the U-Matrix which the 3 by 3 lattice SOM is used as an example, but the 5 by 5 lattice SOM U-matrix and any square grid U-matrix can also be derived using the example.



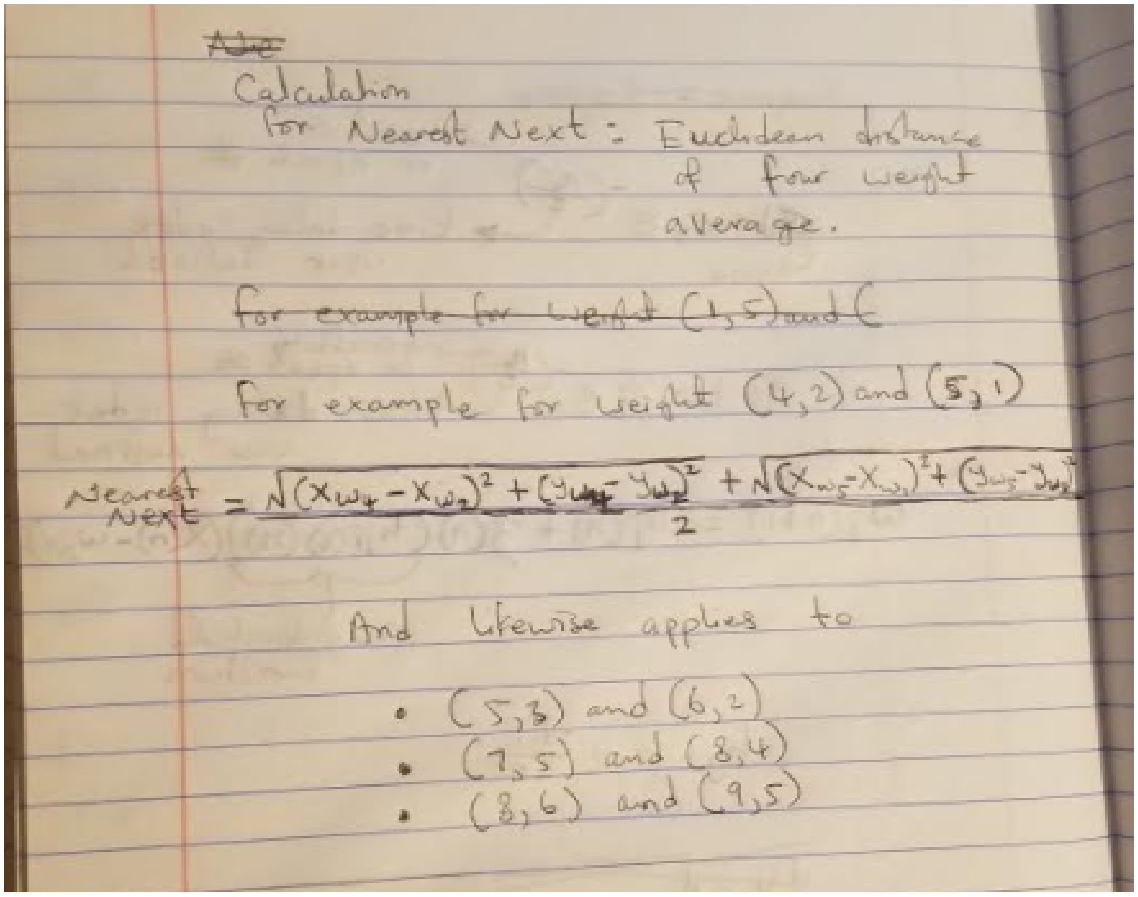
Next: □
Nearest Next: △

Calculation for ~~Next~~: Euclidean distance for two weights

For example: for weight 1 and 2

$$Next = \sqrt{(X_{w_1} - X_{w_2})^2 + (y_{w_1} - y_{w_2})^2}$$

~~At~~ this applies to weight (2,3), (4,5), (5,6), (7,8), and (8,9)



~~Add~~

Calculation
for Nearest Next : Euclidean distance of four weight average.

~~for example for weight (1,5) and (~~

for example for weight (4,2) and (5,1)

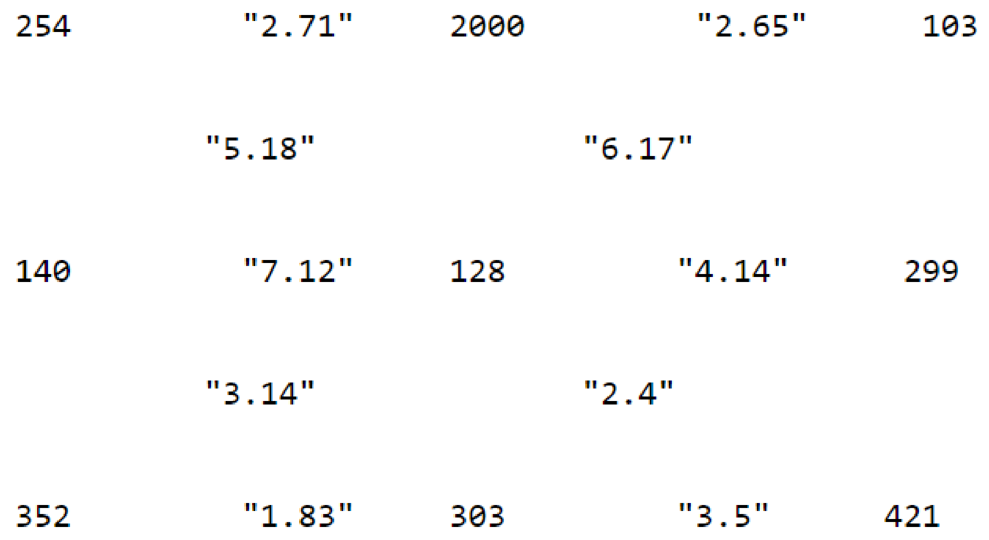
$$\text{Nearest Next} = \frac{\sqrt{(x_{w_4} - x_{w_2})^2 + (y_{w_4} - y_{w_2})^2} + \sqrt{(x_{w_5} - x_{w_1})^2 + (y_{w_5} - y_{w_1})^2}}{2}$$

And likewise applies to

- (5,3) and (6,2)
- (7,5) and (8,4)
- (8,6) and (9,5)

In the example above, the weight position symbolizes the winning for each weight at the last epoch and the rest of the numbers with quotes represent the lattice weights distance from each other in the U-Matrix.

3 by 3 U-Matrix:



```
254          "2.71"     2000         "2.65"      103

           "5.18"             "6.17"

140          "7.12"     128         "4.14"      299

           "3.14"             "2.4"

352          "1.83"     303         "3.5"      421
```

It is illustrated below that the second weight for the 3 by 3 U-matrix had the most wins and was relatively close to its neighboring weights. Weights 4 and 5 are the farthest from each other and also have the lowest wins. The nearest node for the upper weights are farther apart than those below.

5 by 5 lattice structure:

```
138        "1.28"    85        "1.98"    119       "4.3"    74    "3.86"      111

         "1.97"              "3.13"              "1.73"           "3.07"

177        "2.0"    94        "1.97"    100       "3.37"   104    "7.6"       26

         "2.81"              "2.19"              "2.97"           "2.7"

71        "4.94"    82        "4.27"    127       "2.99"   658    "0.82"      650

         "2.24"              "3.67"              "2.63"           "4.02"

33        "3.42"    119        "1.77"    133       "5.25"   692    "7.22"      106

         "3.19"              "3.33"              "3.98"           "3.59"

72        "1.62"    21        "4.38"    108       "2.82"   84    "5.79"       16
```

The weights at the right had the most wins, and weight 19 and 20 were far from each other. This is for the **5 by 5 lattice structure**.

```
658    "0.82"          650

     "4.02"

 692    "7.22"          106

     "3.59"

84    "5.79"          16
```

One cluster found based on how small the next and nearest next distances are, **still for a 5 by 5 lattice structure:**

```
138          "1.28"     85          "1.98"

           "1.97"                 "3.13"

177          "2.0"     94          "1.97"

           "2.81"                 "2.19"
```