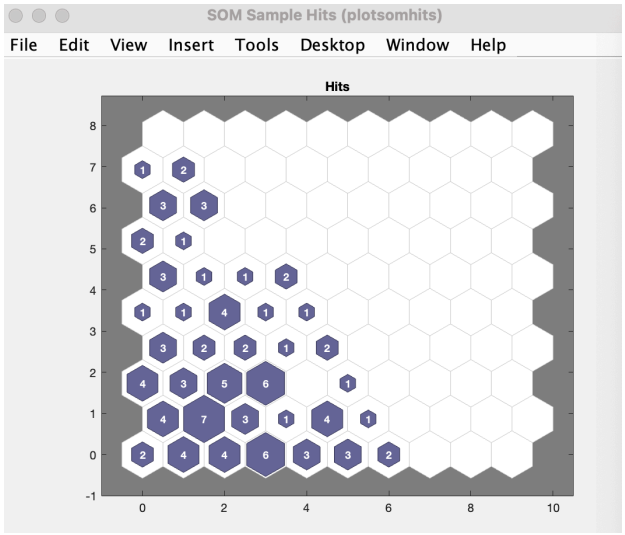
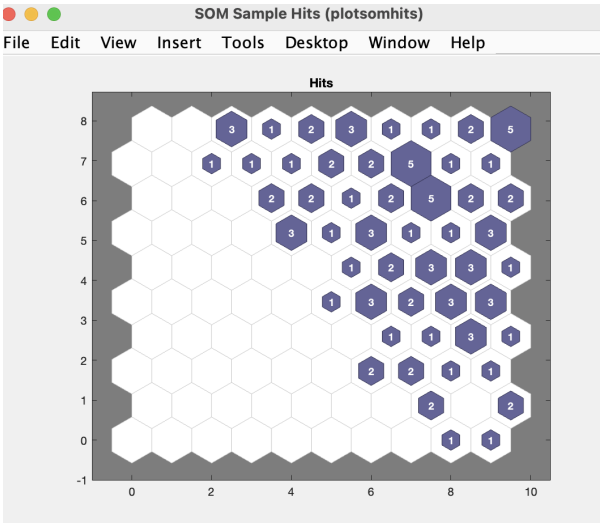
**Task 1**

**Question 1: Use plotsomhits to study the overlap between the winning nodes for the two clusters, as above. What amount of overlap do you see for each different data set? (Use som_Px0 for the Px0 data set, x ∈ {1, 2, 3}.)**

P10

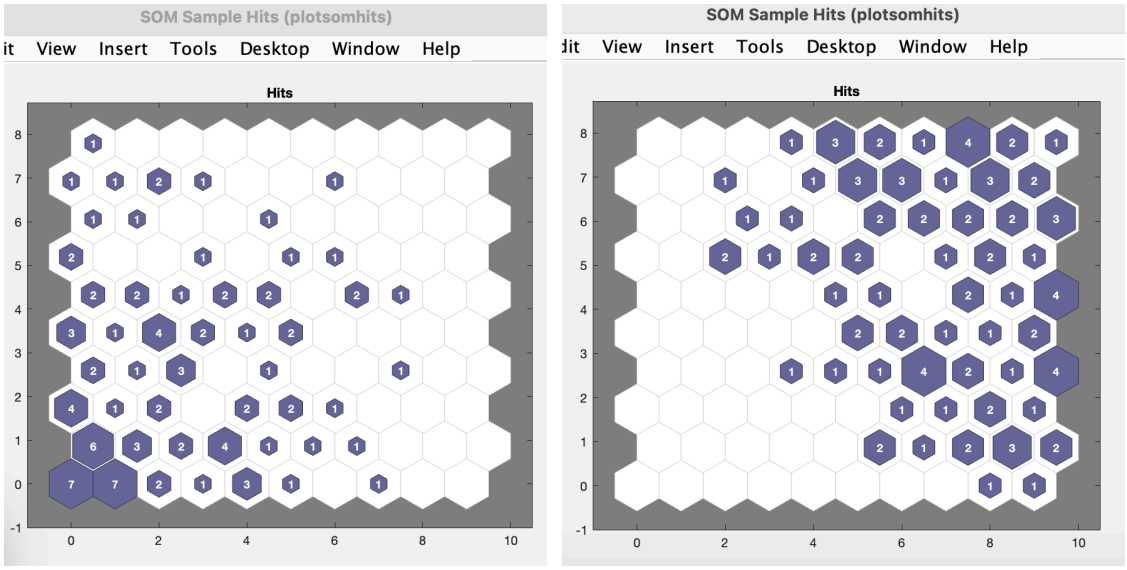



plotsomhits(som_P10, P10(:,1:100)) plotsomhits(som_P10, P10(:,101:200))

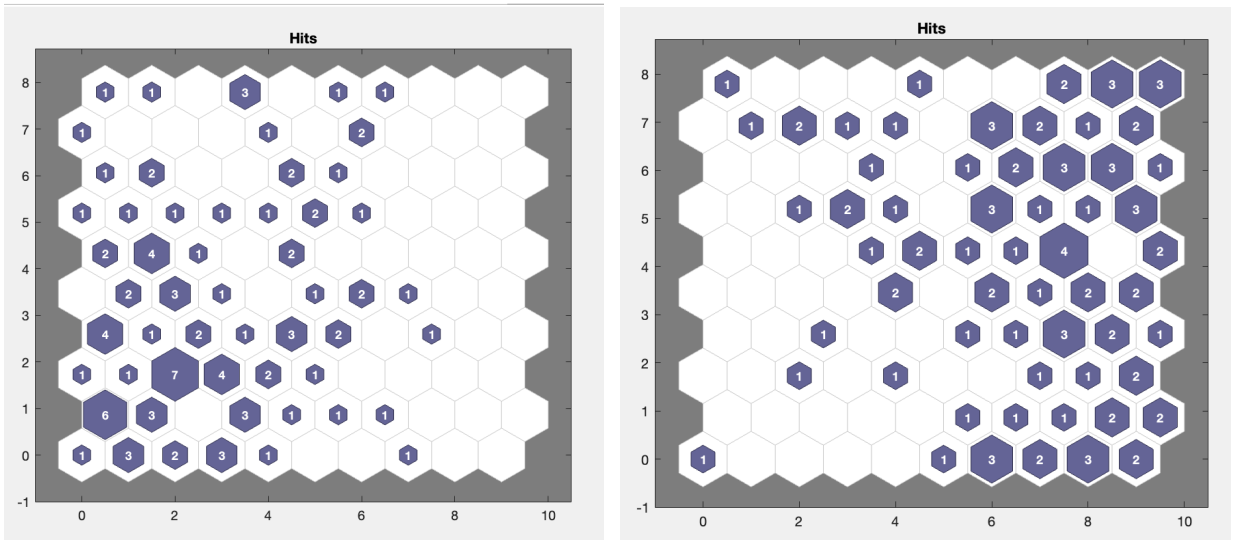
**Answer:** No overlaps. Our goal is basically to cluster the input data points. So, when we see the plots provided in Figure 1 on the pdf, P10 is not as spread out compared to P20 or P30. As it is not so spread out, the clusters are very dense and have no overlaps. On the other hand, the clusters are have no overlaps for P20 and P30 (see below).

P20:




Overlaps between 2-8 on the x-axis and 1-7 on the y-axis.

P30:

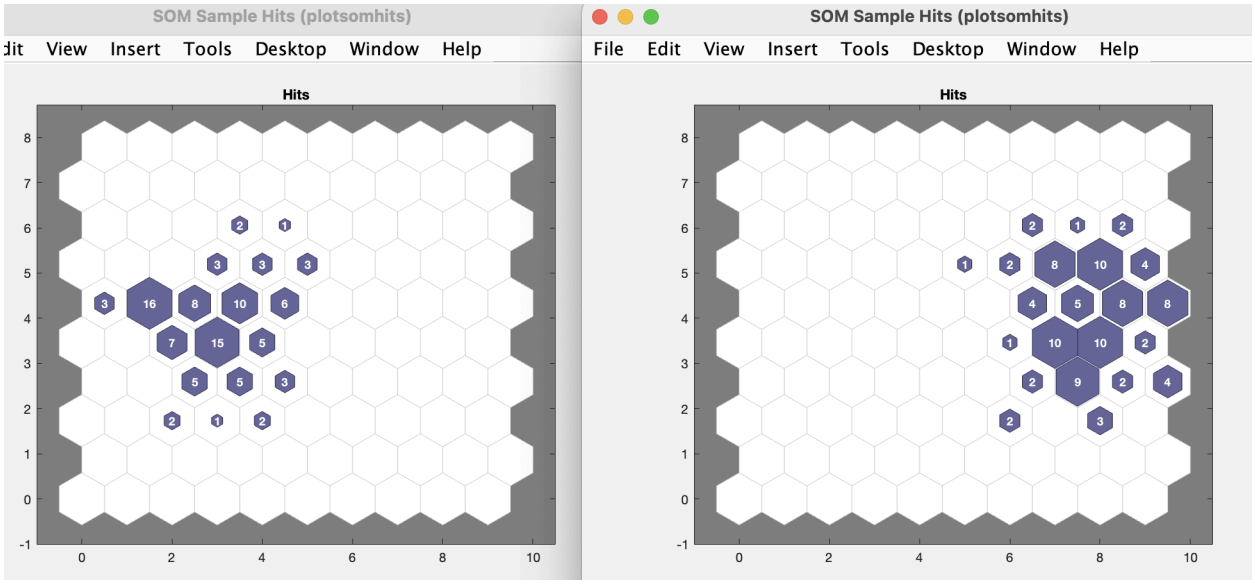



Overlaps from 0-8 on the x-axis and 0-8 on the y-axis.

## Question 2 ⋆: Explain why the two plotsomhits figures that you created for Plot 1 look the way that they do.

Plot 1: Include the plotsomhits figures for cluster F1, both when using P10 with som_P30 and when using P30 with som_P10.

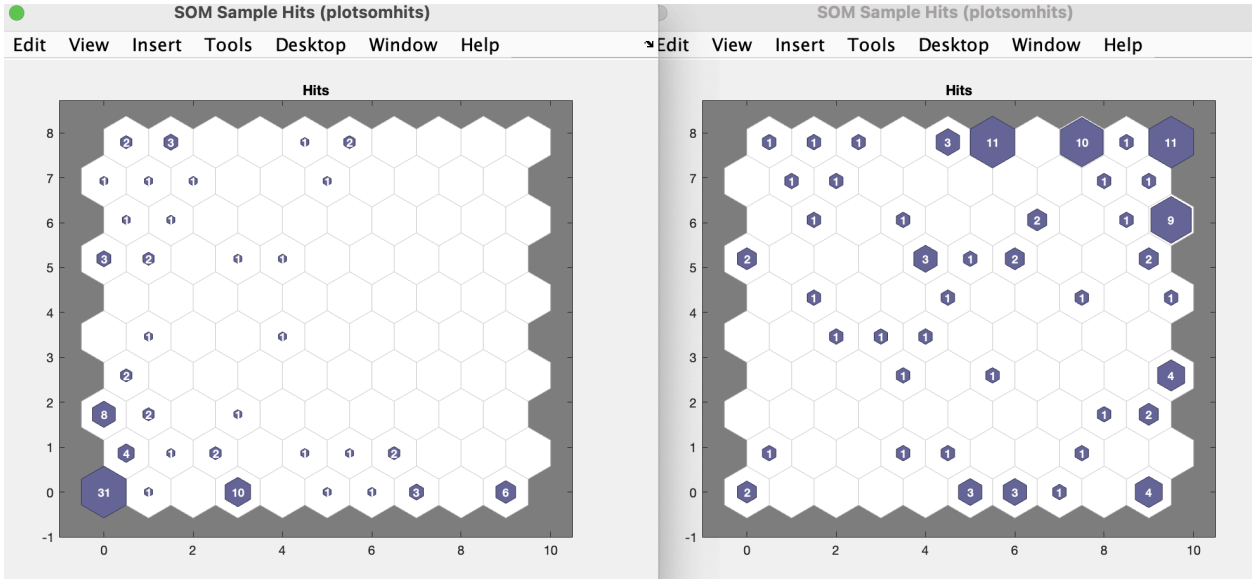
```
plotsomhits(som_P30, P10(:,1:100));          plotsomhits(som_P30,P10(:,101:200));
```




Since the som_P30 was trained on a dataset with more variance, plotting the sample hits for P10 shows smaller clusters as P30 has fewer inputs in the same region as P10.

```
plotsomhits(som_P10, P30(:,1:100));
```

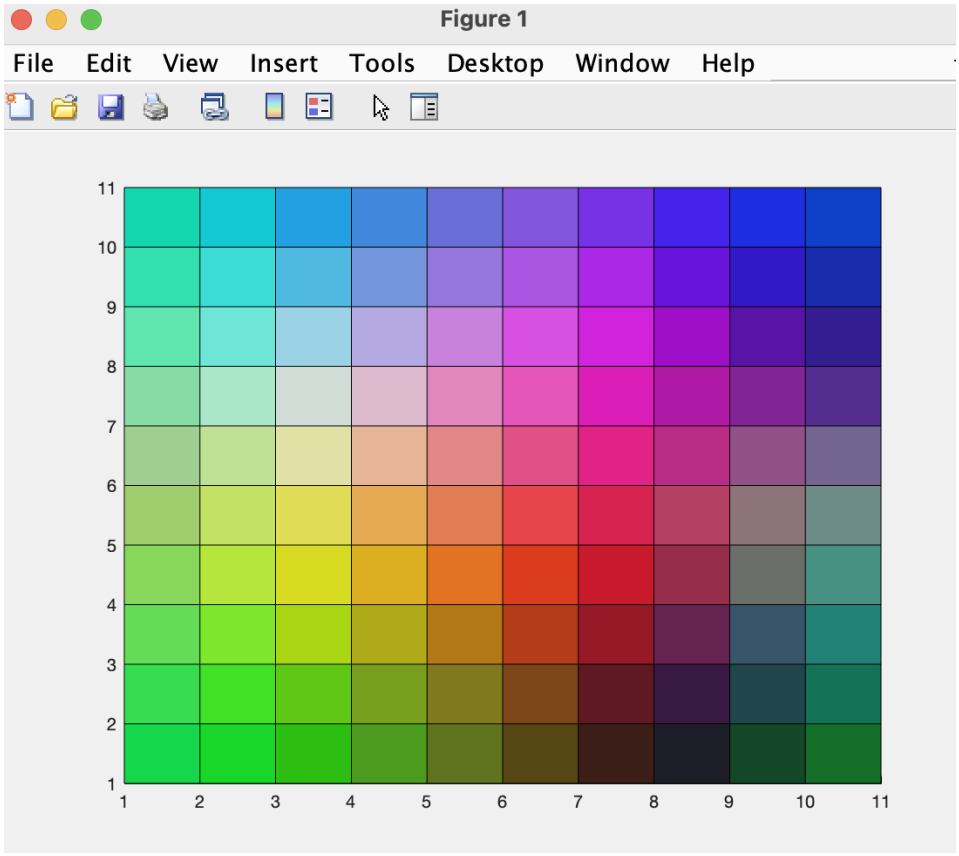
```
plotsomhits(som_P10,P30(:,101:200));
```




Vice-versa, the som_P10 was trained on a dataset with lesser variance, plotting the sample hits for P30 shows clusters as mixed because P10 has not seen the inputs in the range of P30.

**Task 2**

**Include a plot from plot_colors that shows the smoothest color map that you have been able to produce, similar to the one in Figure 3. Write what settings you used.**




```
load rgb_data
trained_som = newsom(RGB, [10 10], 'gridtop', 'linkdist', 1000, 10);
trained_som.trainParam.epochs = 1000
[trained_som, stats] = train(trained_som, RGB);
plot_colors(trained_som)
```

**Reason:**

**Too Large Neighborhood (e.g., 20 instead of 10) -> The entire map gets influenced too much, leading to less distinct regions. Colors blend too aggressively, reducing fine-grained transitions.**

**Too Many Epochs (e.g., >1000) -> The learning rate decays too much, causing stagnation. The SOM may get stuck in suboptimal positions, leading to disruptions in smooth color transitions.**

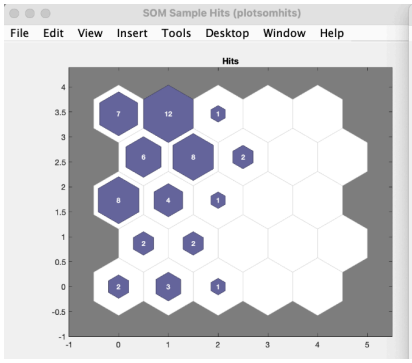
**On the other hand,**

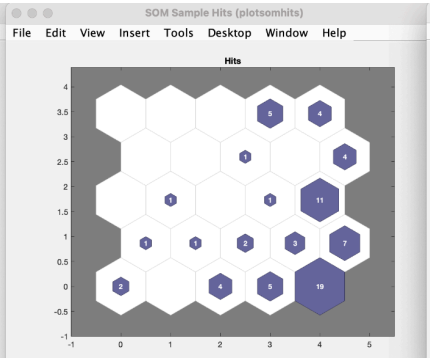
**Neighborhood too small → The SOM doesn't generalize well, causing disjointed and patchy colors instead of smooth transitions.**

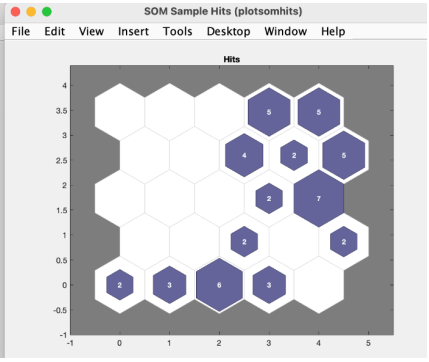
**Epochs too low → The SOM doesn't fully converge, leading to random, noisy mappings rather than an ordered color transition.**

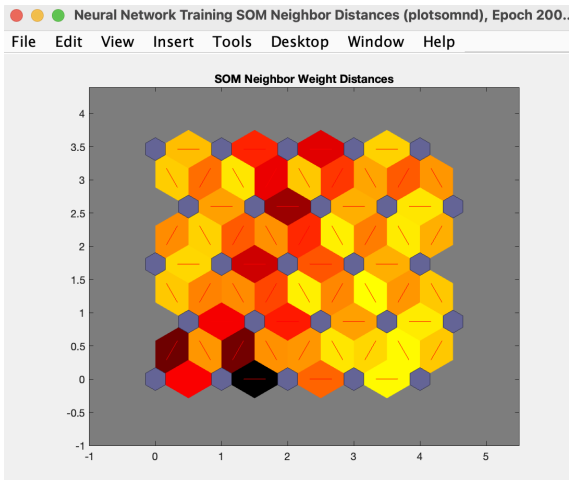
## Task 3

## Question 3: What training parameters did you use? How well does the trained sofm separate the classes in your opinion? Is some class easier to separate than the rest? If so, which one?



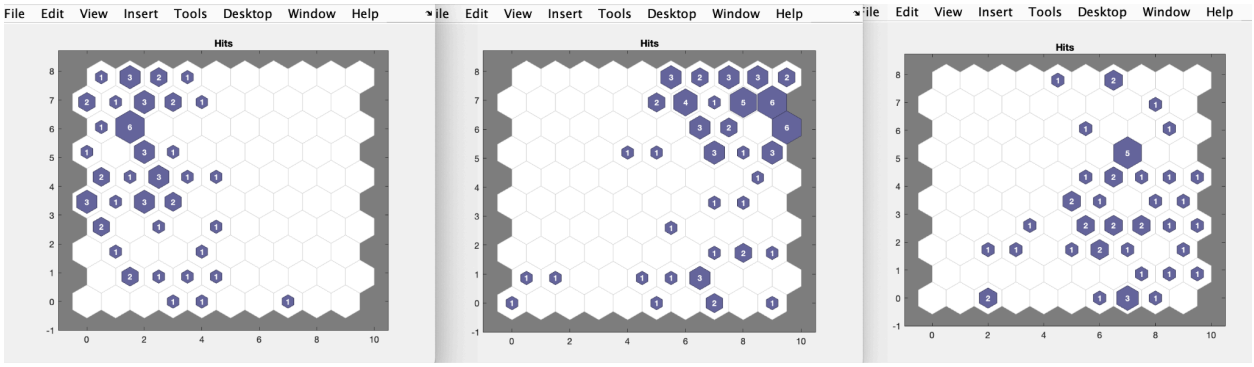

**load wine_dataset**
**som_w = newsom(wineInputs, [5 5], 'hextop', 'linkdist', 1000, 10);**
**som_w.trainParam.epochs = 2000;**

**[trained_wine, stats] = train(som_w, wineInputs);**

The SOM is able to separate the classes reasonably well; however, class 1 is clearly much easier separated than others.

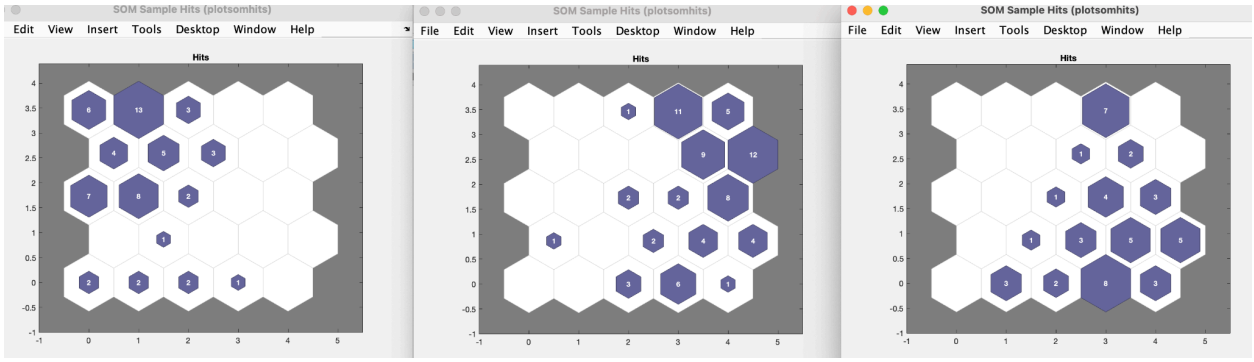
## Question 4: What training parameters did you use? What major differences do you see in the results compared to the 5 by 5-node sofm?

10 x 10 -




There are more nodes to create a better boundary but it doesn't affect the separation significantly for the 2nd and 3rd classes.

After Normalizing:

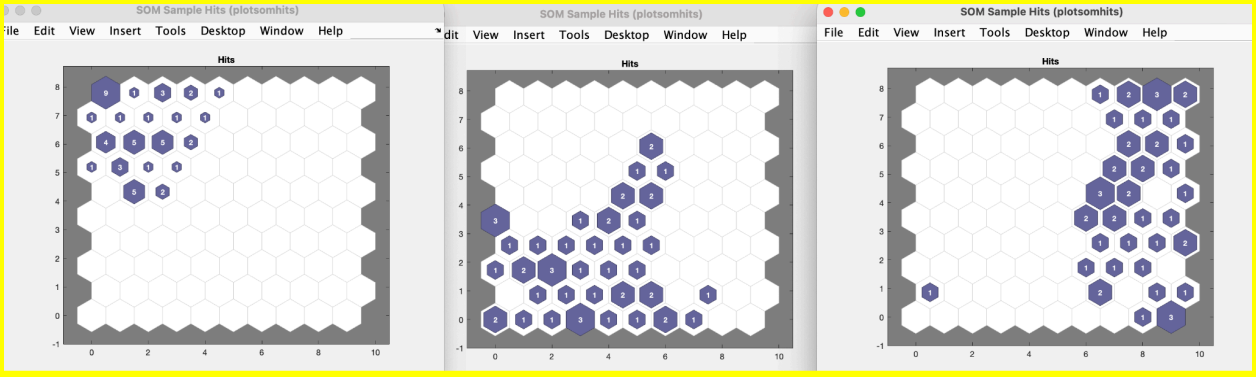



## Question 5: How well do the SOFMs separate the classes in the normalized data?

Before normalization, the features in the wineInputs have a much larger range of values; thus, the SOFM has a hard time figuring out which features are more important. Whereas after normalization everything comes to a certain range, so the model can distinguish the important features well and form better clusters.

## Task 4

## Question 6: How well does the sofm separate the classes, in your opinion?

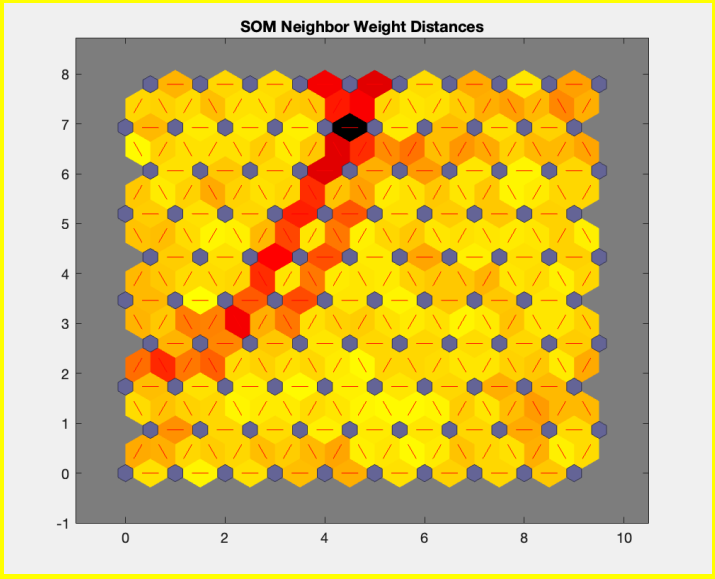



The SOFM separates the classes pretty well, because we can see 3 clear clusters being formed. The parameters used are:

load iris_dataset
som_flower = newsom(irisInputs, [10 10], 'hextop', 'linkdist', 1000, 10);
som_flower.trainParam.epochs = 2000;

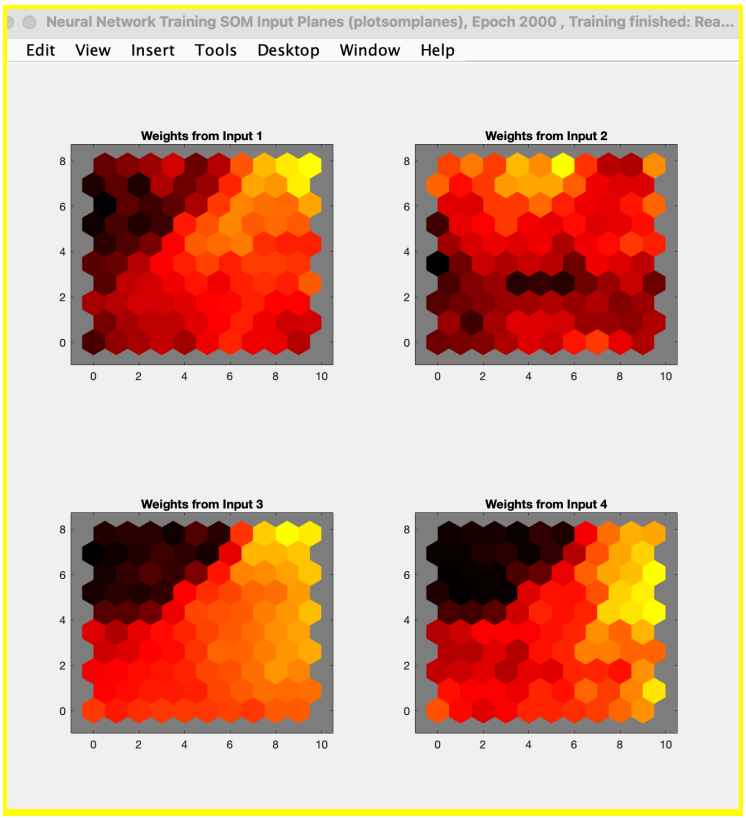
**Now open up the SOM Neighbor Distances window, and see if you can find any interesting characteristics.**

**Question 7 ⋆: If you did not already know that the data came from three different species of Iris flowers, could you have found any evidence in the neighbor distances plot to indicate that the data are not from a single species? What evidence would that be, and what does it imply about the distribution of the data? Describe your reasoning.**




The neighbor distance plot clearly shows a separating boundary for the first class, which tells us that there are at least 2 separate classes in the dataset. The second boundary is not very visible so we can only surmise that there are at least 2 classes in the dataset from this plot.
We also see that two neighboring neurons are both close in weight space but belong to different clusters, so they may both adjust strongly at the boundary, forming two distinct edges (two red lines)

**Open up the SOM Input Planes window and study the plots.**
**Question 8: Do any of the attributes seem particularly correlated? Which ones? Explain your reasoning.**



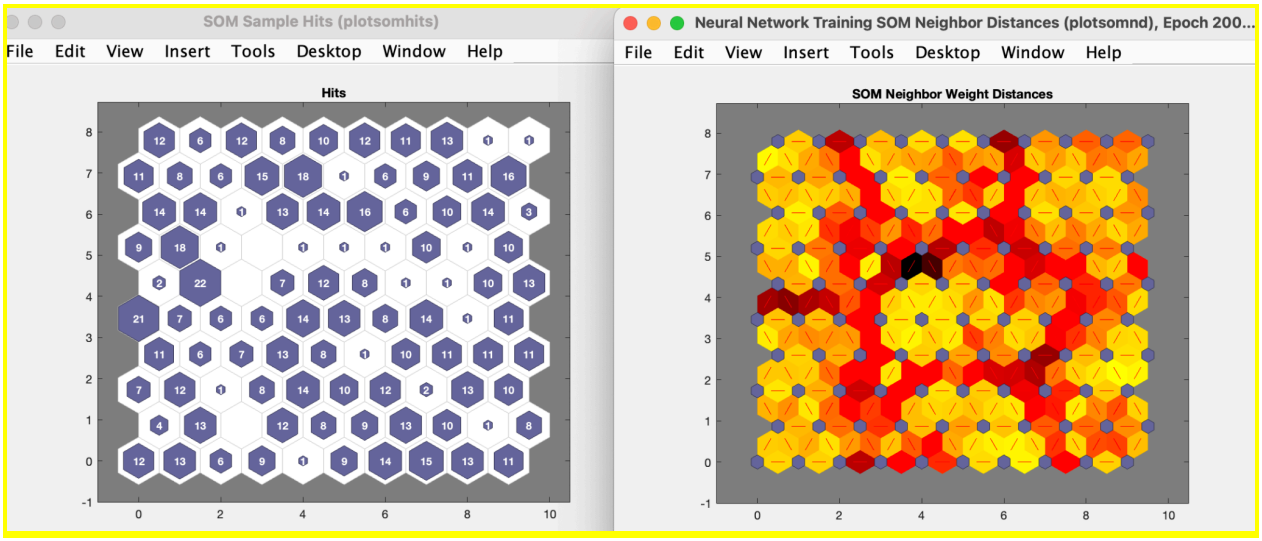

**If we combine these four, we will get three clusters like we saw above. Furthermore, we can see that the petal length and petal width show a clear separation between different flower types. Other features, such as sepal width and sepal length, have more overlapping distributions, making it harder to differentiate between classes.**

**Now study the input planes in combination with the sample hit plots for the three separate classes from before.**

**Question 9: Which species has the smallest petals? Which species has the largest petals? Explain your reasoning.**

Again, merging them would ultimately lead us to clusters we got above. Higher weight values lead to brighter colored areas (yellow in this case). This means that darker regions have the smallest petals and the largest petals are the brightest regions.

**Question 10: How many well-separated clusters are there in the data set? Explain how you found out your answer. Include figures if necessary.**




There are 7 clusters. Observing the Neighbour Distances, we can see that there are 7 regions with bright colors and the decision boundaries are of a darker color, separating the clusters.

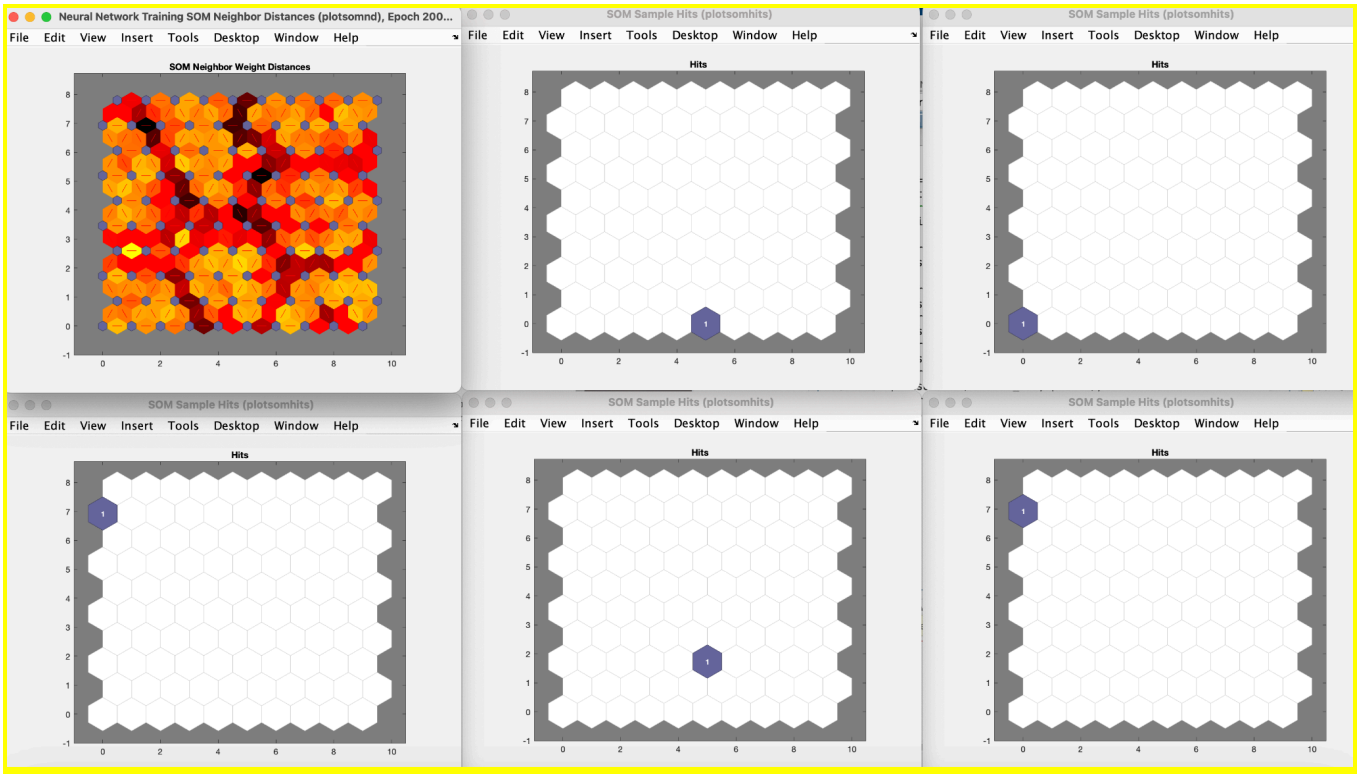
**Now consider the five single data points. Try to figure out if any of them are from the same cluster.**

**Question 11 ⋆: Which data points are from the same cluster? How did you figure it out? Include figures if necessary.**

```
load unknown_data;

som1 = newsom(unknown_data, [10 10], 'hextop', 'linkdist', 1000, 10);
som1.trainParam.epochs = 2000;
[som_unk, stats] = train(som1, unknown_data);
figure;
plotsomhits(som_unk, point1);
figure;
plotsomhits(som_unk, point2);
figure;
plotsomhits(som_unk, point3);
figure;
plotsomhits(som_unk, point4);
figure;
plotsomhits(som_unk, point5);
```

**Point 1** **Point 2**




**Point 3** **Point 4** **Point 5**

**Point 1: (5,0)**
**Point 2: (0,0)**
**Point 3: (0,7)**
**Point 4: (5,2)**
**Point 5: (0,7)**
**So, Points 1 and 4 fall in the same cluster;**
**Points 3 and 5 fall in the same cluster.**

**Question 12: Propose a real-world problem that you consider interesting and that you believe can be solved using a SOFM. Explain in a few sentences how the SOFM would be used and what kind of data it would be trained on.**

Banks need to detect fraudulent transactions that differ from normal spending behavior. A Self-Organizing Feature Map (SOFM) can help by identifying unusual patterns.

How SOFM Would Be Used
The SOFM is trained on transaction data (amount, location, time, frequency, device).
Legitimate transactions will cluster together, while fraudulent activities will appear as outliers or in sparse regions.

Another application could be a stock market sector investment suggestion model. We take N companies with attributes describing their sector and train a SOM on this data. Next, for investing in further companies, we can plot which cluster the company falls into and invest in the company with a similar amount wrt that cluster’s average.