Question 1

a. (L3, 2') Given the same parameters, does DBSCAN always output the same cluster assignment or not? Briefly explain.

No, because border point that are reachable from more than one cluster can be part of either cluster depending on the order the data is processed.

b. (L3, 2') With improper choice of initial cluster centers, K-Means can give us a very bad clustering output. Could you propose a method to alleviate the issue?

We can run some other algorithm such as DBSCAN to select the possible clustering center to start with or tryout for some times.

c. (L1, 2') Between k-Medoids (PAM) and k-Means, which is more efficient? Briefly explain.

The average runtime of K-Means algorithm is greater than the average runtime of K-Medoids algorithm for both normal and uniform distributions. Though K-Medoids is efficient when the data size is small.

d. (L1, 2') OPTICS is superior to DBSCAN because we do not have to set the paramenter ε. True or False? Briefly explain.

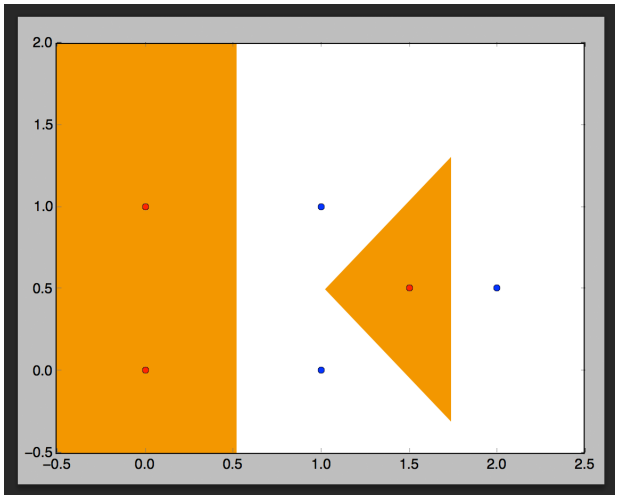
True, OPTICS use a priority queue, so that it gets the nearest neighbor. It's a better than DBSCAN when we don't know what paramenter ε to give. However, it is slower than DBSCAN.

e. (L1, 2') Is it true that Lazy learners (e.g., kNN) is more efficient than non-lazy learners (e.g., decision trees)?

False. Lazy learner take less time in training but more time in predicting. It's less efficient, if there is only a small sample to train but a huge sample to predict.

Question 2

a. (L2, 2') Plot the given labeled data in the x-y plane, and highlight the regions where data will be classified as '+1' by a 1NN classifier (i.e, kNN for k = 1).



b. (L2, 2') Suppose we use a 3NN classifier based on 100 (labeled) data points uniformly distributed in a d-dimensional unit cube, i.e. $[0,1]^d$. For d = 2 (i.e, the unit square), what is the minimum size of a d-dimensional cube centered at q for it to cover at least 3 neighbors of q in expectation? Show your steps. (Note that there is no need to consider the special case in which the query point q lies near the boundary of the unit cube.)

Since the data is uniformly distributed. In order to cover 3, the needed size would be to 1/10^2.

c. (L2, 2') Calculate the minimum cube size asked in (b) for d = 100.

Same as question (b), the needed size should equals to 1/10^100.

d. (L3, 2') According to your calculations in (b) and (c), do you see any problem in applying a kNN classifier to data in high-dimensional spaces (e.g. d = 100)? Briefly explain. (Hint: consider the underlying principle of why kNN works generally.)

As the dimensions of data increase, the amount of calculations grows exponentially. Therefore, with high dimension, the algorithm would be very costly.

e. (L3, 2') What are the pros and cons of using a large k for the kNN classifier?

With a large k, the computational cost would be high. The distance would become so small that it can't classify efficiently.

Question 3

a. (L2, 2') Given $x_1, x_2 \in \{0,1\}$, design a neuron that implements the logical operation AND. That is, the neuron takes x_1 and x_2 as input and computes $y = x_1$ AND x_2 as output ($y \in \{0, 1\}$) (Hint: use the activation function threshold(a) = 1 if $a > 0$ and 0 otherwise.)

weighted vectors: {1, 1}, bias = -1

activation function (a) = 1 if vector sum plus bias $a > 0$ and 0 otherwise

b. (L2, 2') Design a neuron that outputs $y = x_1$ OR x_2.

weighted vectors: {1, 1}, bias = 0

activation function (a) = 1 if vector sum plus bias $a > 0$ and 0 otherwise

c. (L2, 2') Design a neuron that takes a single input $x \in \{0, 1\}$ and outputs y = NOT x.

weighted vectors: {-1}, bias = 1

activation function (a) = 1 if vector sum plus bias $a > 0$ and 0 otherwise

d. (L3, 4') Design a neural network that takes $x_1, x_2 \in \{0, 1\}$ as input and computes $y = x_1$ XORx_2 as output (x_1 XOR$x_2 = 1$ if and only if $x_1 /= x_2$). (Hint: you can reuse the neurons you designed in (a)–(c))

Layer1: Input x_1, perform NOT, get x'_1. Input x_2, perform NOT, get x'_2.

Layer2: Input x_1 and x'_2, perform AND, output y_1. Input x_2 and x'_1, perform AND, output y_2.

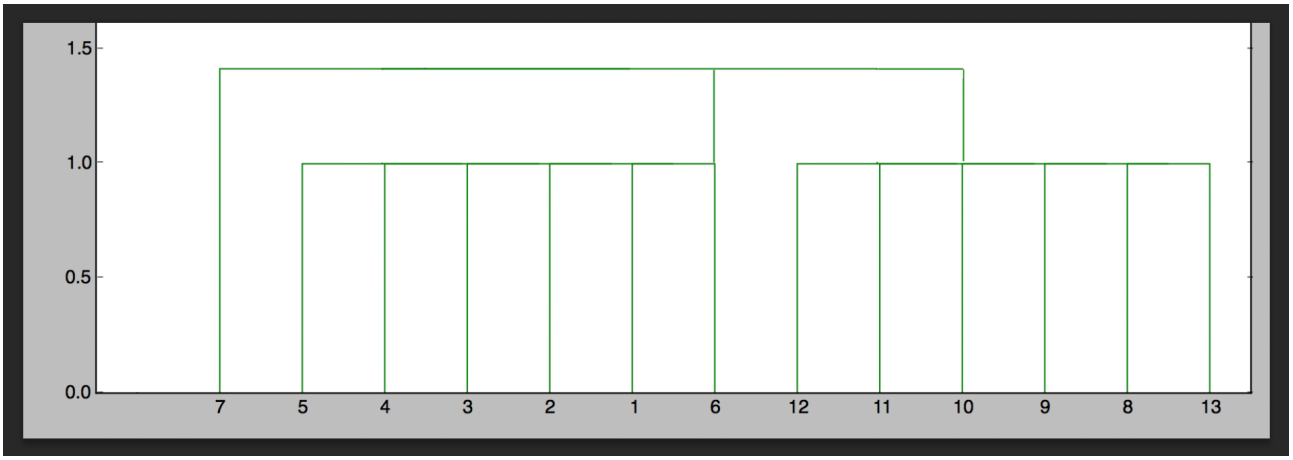
Layer3: Input y_1 and y_2, perform OR, output result

e. (L3, 2') Is it possible to implement XOR using a single layer neural network (without any hidden layer)? Explain your answer.

No, because XOR is not a linearly separable function.

Question 4

a. (L2, 4') Draw the dendrogram using AGNES. Please use single link and Euclidean distance as the dissimilarity measure.



b. (L2, 2') If we want to cluster the dataset into 3 groups based on the dendrogram, what are the members of each of the 3 groups?

Group1{p1,p2,p3,p4,p5,p6}

Group2{p7}

Group3{p8,p9,p10,p11,p12,p13}

c. (L2, 2') Based on the given ground truth, what are the B-Cubed precision and recall of the output?

Point i	1	2	3	4	5	6	7	8	9	10	11	12	13
Pi	1	1	1	1	1	1	1	1	1	1	1	1	1
Ri	6/7	6/7	6/7	6/7	6/7	6/7	1/7	1	1	1	1	1	1

Precision = 1

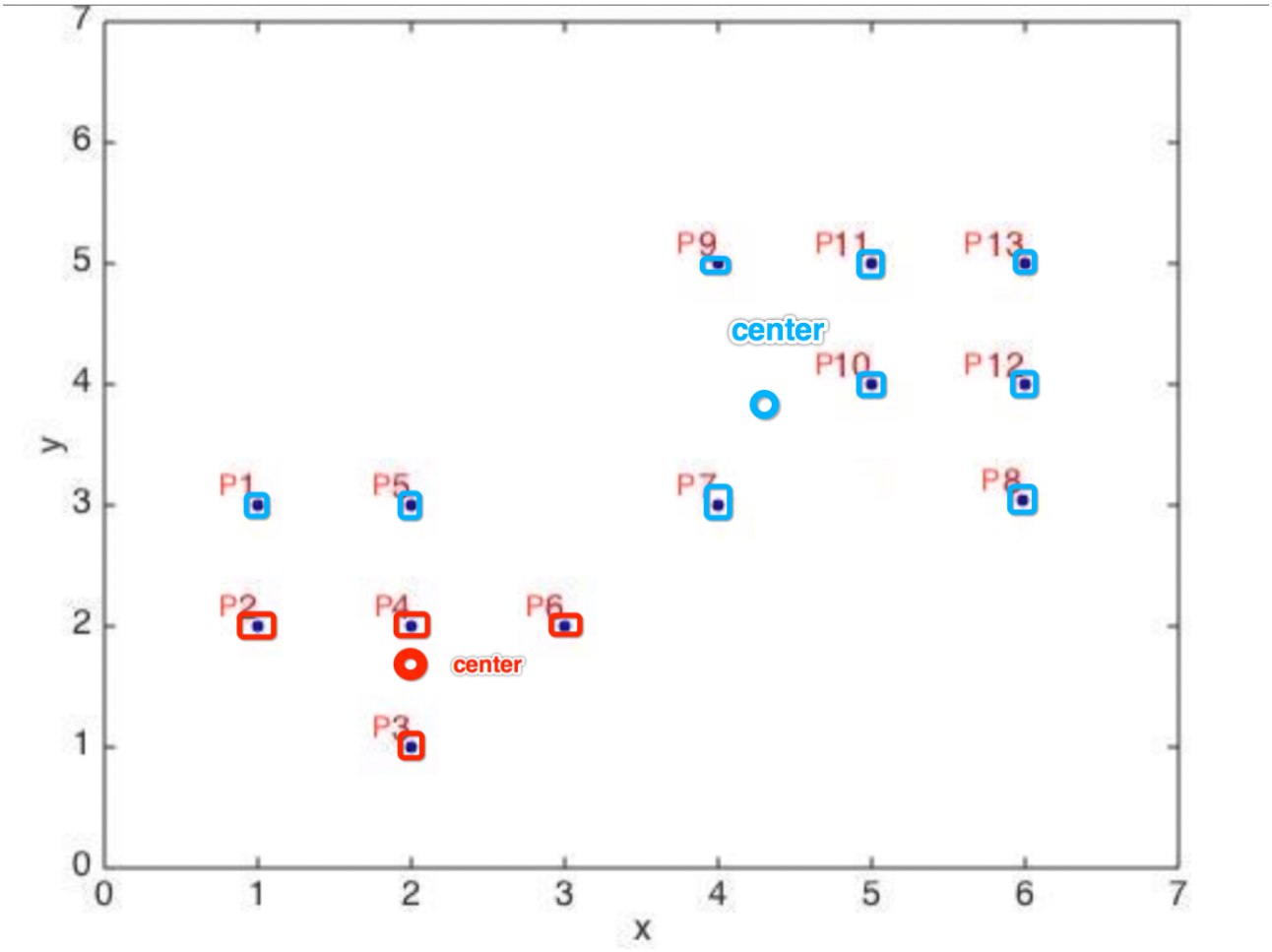
Recall = (6*6/7 + 1/7 + 6*6/6) /13 = 79/91

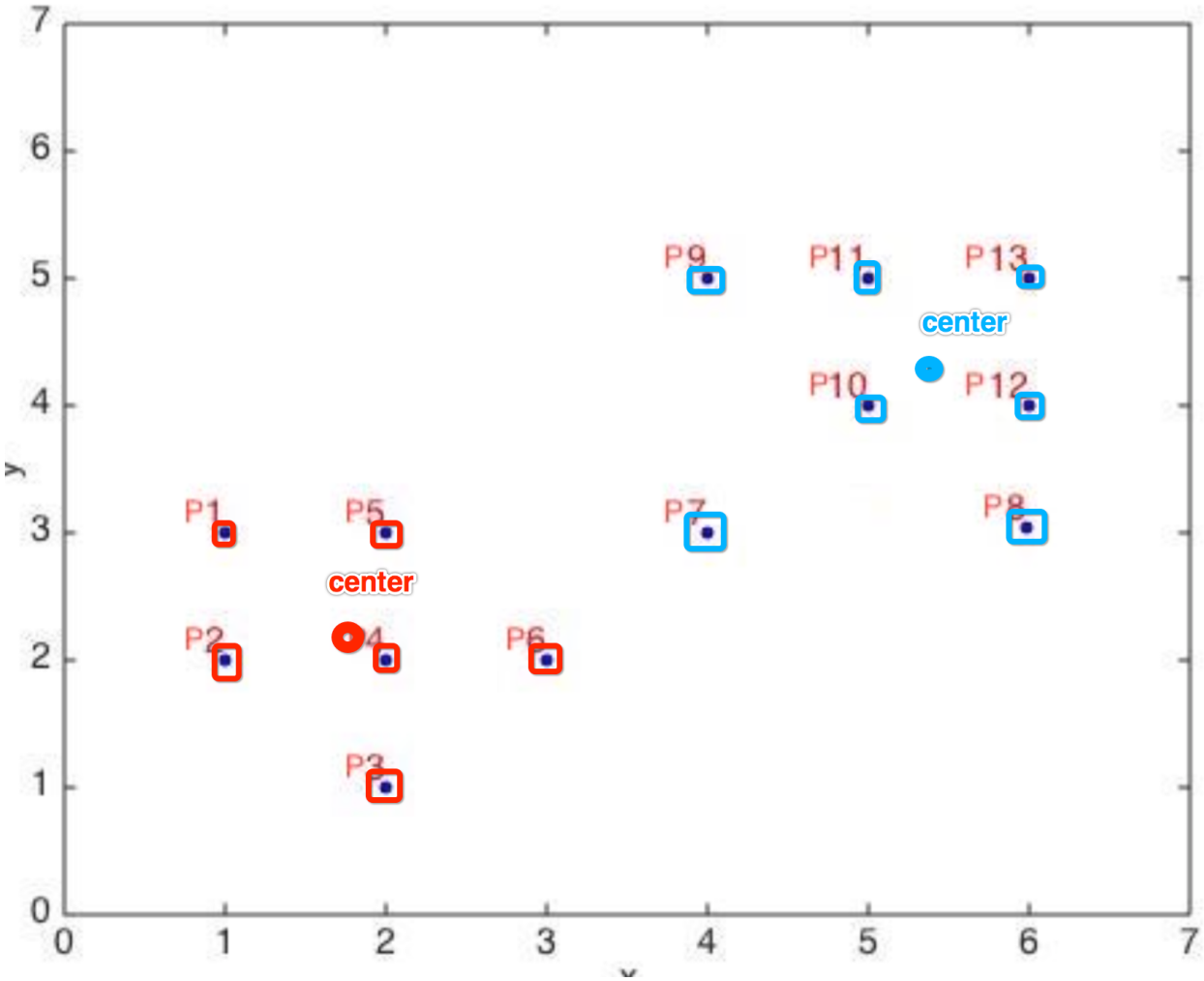
Question 5

a. (L2, 4') Perform k-means using Euclidean distance with k = 2 and the initial cluster centers P1 and P2.

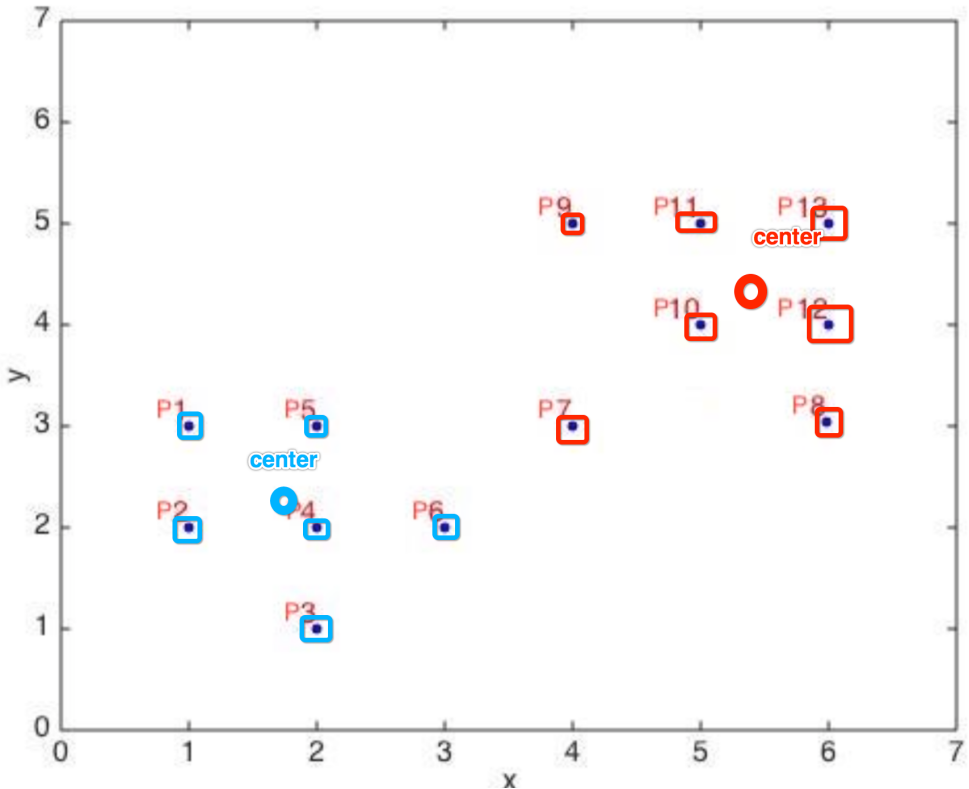
Center1 (1, 2) -> (2, 1.75) -> (1.83, 2.17)

Center2 (1, 3) -> (3.89, 4.33) -> (5.14, 4.14)





b. (L2, 2') Perform k-means using Euclidean distance with k = 2 and the initial cluster centers P2 and P12.



Center1 (3, 1) -> (1.83, 2.17)

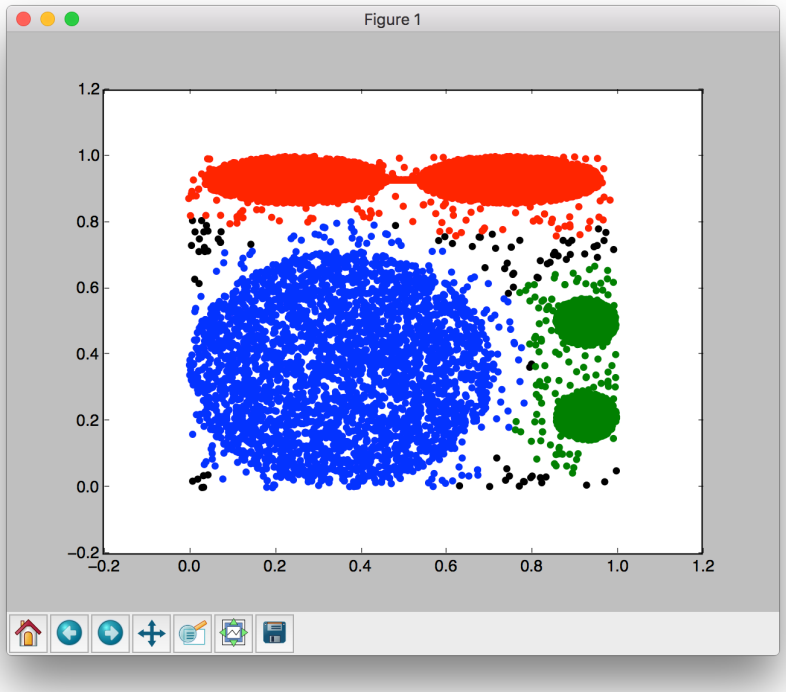
Center2 (4, 6) ->(5.14, 4.14)

c. (L3, 2') As you can see in sub-questions a and b, choices of initial cluster centers can affect the speed of the clustering process. Assume k = 2 and the data are 2- dimensional, suggest a general and reasonable way to select initial cluster centers to speed up the clustering process. Briefly justify your choice.

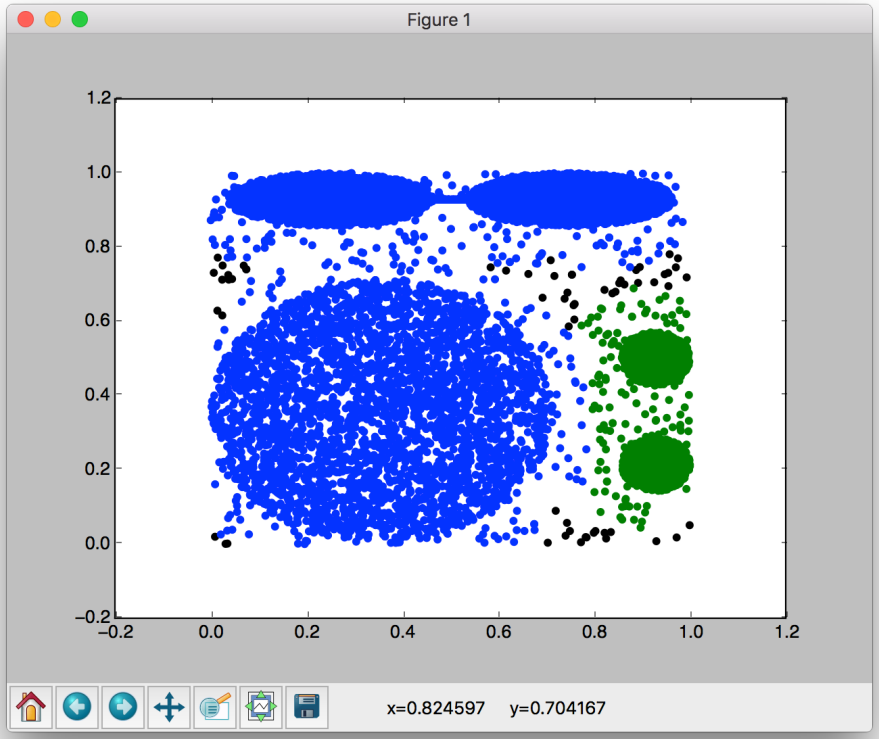
We can start with points that are far away from each other, so they would separate the different clusters quickly. If two initial points are closed to each other, there would be more intermediate steps.

Machine Problem

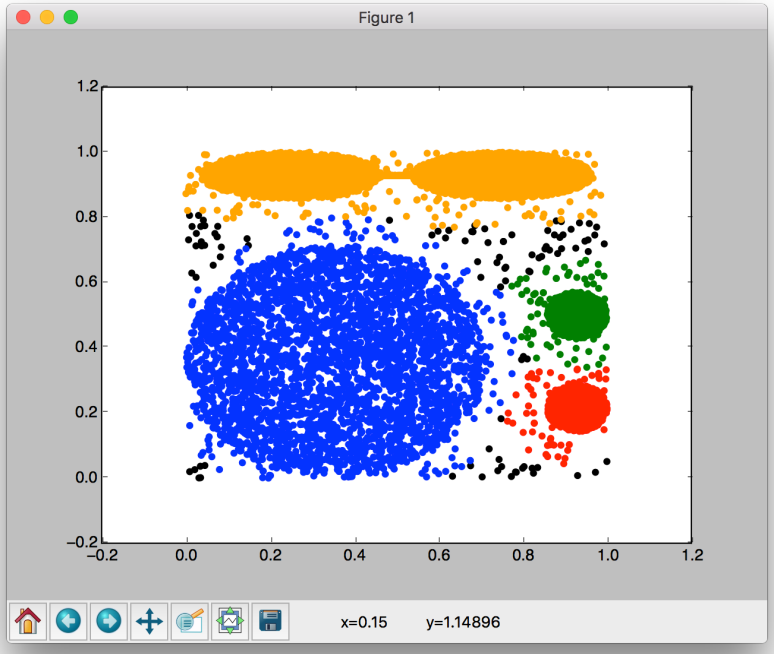
Step1 plot



Step2a plot



Step2b plot



- Question to ponder A: How long does it take for your machine to run DBSCAN on the given data? What is the time complexity of

DBSCAN in the worst case scenario? Can you describe a worst case scenario?

Runtime : 53.339835 seconds

The worst runtime for DBSCAN is O(n^2)

The worst case scenario is that we have to go through all points when finding the points in Eps-neighborhood. If we use data structure such as kdtree, the computational time would reduce.

Question to ponder B: Usually, a good clustering algorithm should put similar points to the same cluster, and dissimilar points to different clusters. Based on that intuition, what do you think of the clustering result? Given the same MinPts, should we increase or decrease to get more intuitive clustering results?

Based on the observation of the graph, I think the clustering result is good. Data are well separated into groups. With the same MinPts, I think we should decrease the distance to get more separated clusters, since there is a big cluster in blue that could be further separate.

	Eps	number of cluster	number of outlier
Step1	0.065	3	84
Step2a	0.07	2	65
Step2b	0.06	4	100

- Question to ponder C: Which one of three distance is the best? Could you suggest a general way to tune parameters MinPts and Eps for DBSCAN?

Step2b is the best, because it is similar with the ground truth. Compare to Step1, it managed to distinguish the two cluster on the down right corner. The way of tuning the parameters is highly dependent on the meaning of the constants. If each point represent a location, and we want to find some locations to build post station. We would choose

MinPts according to how many residents each post location can serve and choose Eps according to the travel radius of the mail carrier.

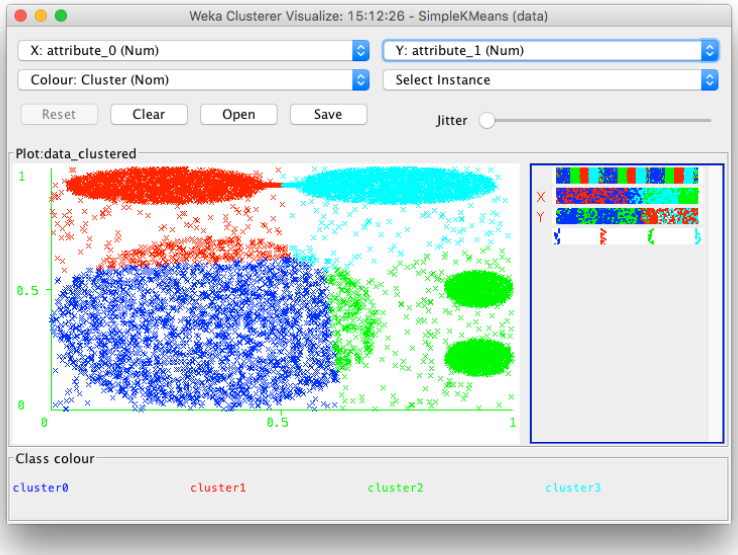
- Run B-Cubed with the outputs from Step 1 and Step 2. Create a table to report the precision and recall corresponding to each of the outputs.

	precision	recall
Step1	0.856190070008	1
Step2a	0.485835695064	1
Step2b	0.961592979835	1

- Question to ponder D: Can you suggest a way to combine B-Cubed precision and recall into a single measure so that it would be easier to compare different results?

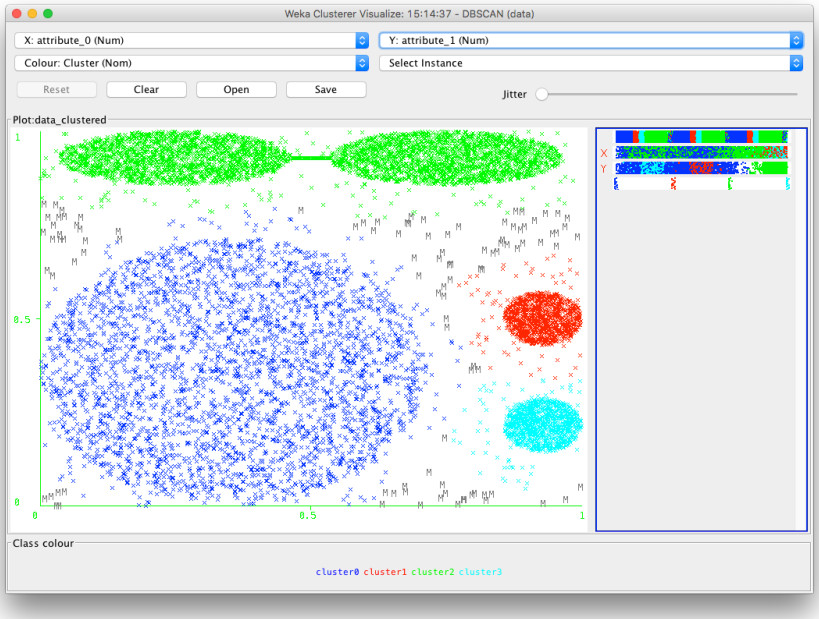
Precision represent the accuracy of the labeling. Recall represent the completeness of the labeling. To combine, we could do (precision^2 + recall^2)/2. The larger the number is, the better the result is.

- Run SimpleKMeans with different number of clusters. Report the best number of clusters, and the plot showing the corresponding clustering result. You can generate the plot by right clicking on the entry of the Result list, and choose Visualize cluster assignments.



Best number of Cluster is 4.

Run DBSCAN with the best minPts and Eps in Step 2. Show the plot for the corresponding clustering result.



Question to ponder E: Compare the results from k-Means and DBSCAN. Is k- Means or DBSCAN more suitable for the provided dataset? Why?

DBSCAN is more suitable, because we don't know how many clusters we should use with K-Mean without running DBSCAN. Since we don't know the number of classes is advance, it's better to run DBSCAN to get an idea of what clusters are there. Also, the density of the data represent the distribution better.