Asmt 4: Clustering

Turn in through Canvas by 2:45pm:
Wednesday, February 19
100 points

Overview

In this assignment you will explore clustering: hierarchical and point-assignment. You will also experiment with high dimensional data.

You will use three data sets for this assignment:

- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A4/C1.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A4/C2.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A4/C3.txt`

These data sets all have the following format. Each line is a data point. The lines have either 3 or 6 tab separated items. The first one is an integer describing the index of the points. The next 2 (or 5 for `C3`) are the coordinates of the data point. `C1` and `C2` are in 2 dimensions, and `C3` is in 5 dimensions. `C1` should have n=19 points, `C2` should have n=1040 points, and `C3` should have n=1000 points. We will always measure distance with Euclidean distance.

It is recommended that you use LaTeX for this assignment (or other option that can properly digitally render math). If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: `http://www.cs.utah.edu/~jeffp/teaching/latex/`

1 Hierarchical Clustering (35 points)

There are many variants of hierarchical clustering; here we explore 3. The key difference is how you measure the distance $d(S_1, S_2)$ between two clusters S_1 and S_2.

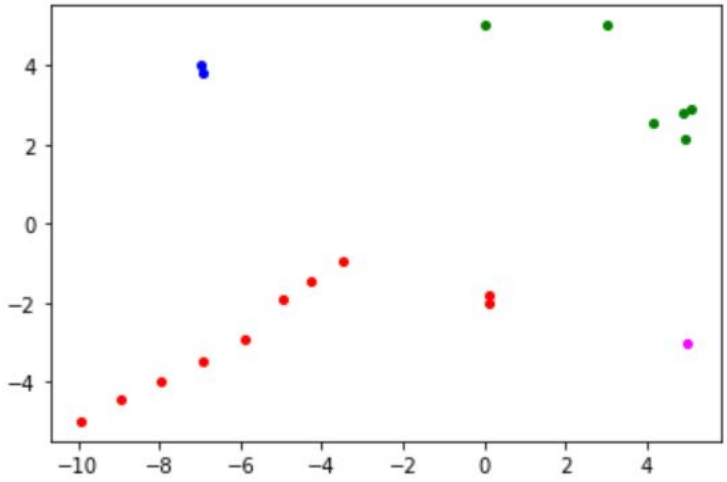
Single-Link: measures the shortest link $d(S_1, S_2) = \min_{(s_1,s_2)\in S_1\times S_2} \|s_1 - s_2\|_2$.

Complete-Link: measures the longest link $d(S_1, S_2) = \max_{(s_1,s_2)\in S_1\times S_2} \|s_1 - s_2\|_2$.

Mean-Link: measures the distances to the means. First compute $a_1 = \frac{1}{|S_1|}\sum_{s\in S_1} s$ and $a_2 = \frac{1}{|S_2|}\sum_{s\in S_2} s$ then $d(S_1, S_2) = \|a_1 - a_2\|_2$.

A (30 points): Single Link
Cluster 1: [[-6.91 3.8] [-6.99 4.]]
Cluster 2: [[-9.92 -4.98] [-8.96 -4.41] [-7.98 -3.98] [-6.91 -3.47] [-5.91 -2.91] [-4.94 -1.91] [-4.24 -1.44] [-3.45 -0.97] [0.13 -1.8] [0.1 -2.]]
Cluster 3: [[4.91 2.82] [5.07 2.88] [4.93 2.12] [4.15 2.56] [3.01 5.01] [0. 5.]]
Cluster 4: [[5. -3.]]

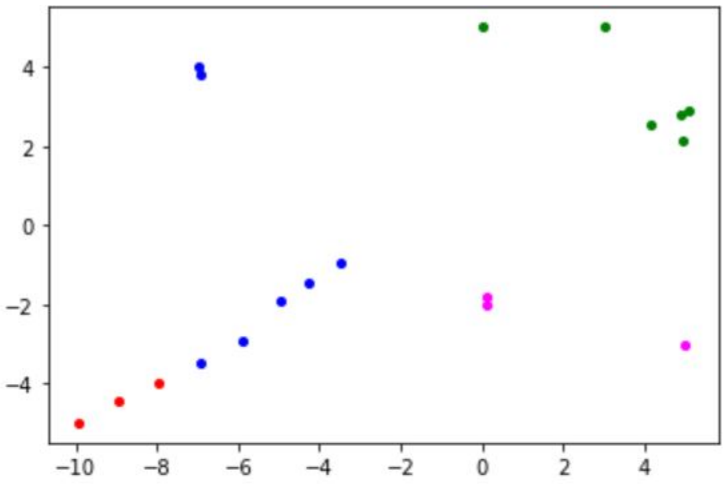
```
first cluster |  second cluster |  distance
10  |  12  |  0.16944488242854633
first cluster |  second cluster |  distance
13  |  14  |  0.19528929532997621
first cluster |  second cluster |  distance
0  |  1  |  0.21391809640396484
first cluster |  second cluster |  distance
9  |  10  |  0.7044274570741265
first cluster |  second cluster |  distance
9  |  10  |  0.8091530719434338
first cluster |  second cluster |  distance
6  |  7  |  0.8426573444474812
first cluster |  second cluster |  distance
6  |  7  |  0.9194131222054369
first cluster |  second cluster |  distance
2  |  3  |  1.0712595569036494
first cluster |  second cluster |  distance
1  |  2  |  1.1203622073662078
first cluster |  second cluster |  distance
2  |  3  |  1.1447658914304835
first cluster |  second cluster |  distance
1  |  2  |  1.1896814692761608
first cluster |  second cluster |  distance
1  |  2  |  1.387606917151757
first cluster |  second cluster |  distance
2  |  6  |  2.69726307540558
first cluster |  second cluster |  distance
2  |  5  |  3.0058339424054465
first cluster |  second cluster |  distance
1  |  3  |  3.6758553047574276
```



Complete Link:
Cluster 1: [[-6.91 3.8] [-6.99 4.] [-6.91 -3.47] [-5.91 -2.91] [-4.94 -1.91] [-4.24 -1.44] [-3.45 -0.97]]
Cluster 2: [[-9.92 -4.98] [-8.96 -4.41] [-7.98 -3.98]]
Cluster 3: [[4.91 2.82] [5.07 2.88] [4.93 2.12] [4.15 2.56] [0. 5.] [3.01 5.01]]

Cluster 4: [[0.13 -1.8] [0.1 -2.] [5. -3.]]

```
first cluster |  second cluster |  distance
10  |  12  |  0.16944488242854633
first cluster |  second cluster |  distance
13  |  14  |  0.19528929532997621
first cluster |  second cluster |  distance
0  |  1  |  0.21391809640396484
first cluster |  second cluster |  distance
9  |  10  |  0.774955686868893
first cluster |  second cluster |  distance
6  |  7  |  0.8426573444474812
first cluster |  second cluster |  distance
8  |  9  |  0.9785629383103536
first cluster |  second cluster |  distance
2  |  3  |  1.0712595569036494
first cluster |  second cluster |  distance
3  |  4  |  1.1447658914304835
first cluster |  second cluster |  distance
4  |  5  |  1.7614691339807294
first cluster |  second cluster |  distance
1  |  2  |  2.187883535423716
first cluster |  second cluster |  distance
7  |  8  |  3.0058339424054465
first cluster |  second cluster |  distance
2  |  3  |  4.261377574286882
first cluster |  second cluster |  distance
4  |  5  |  5.019634753076082
first cluster |  second cluster |  distance
3  |  5  |  5.712700554477763
first cluster |  second cluster |  distance
0  |  2  |  7.465552725809934
```



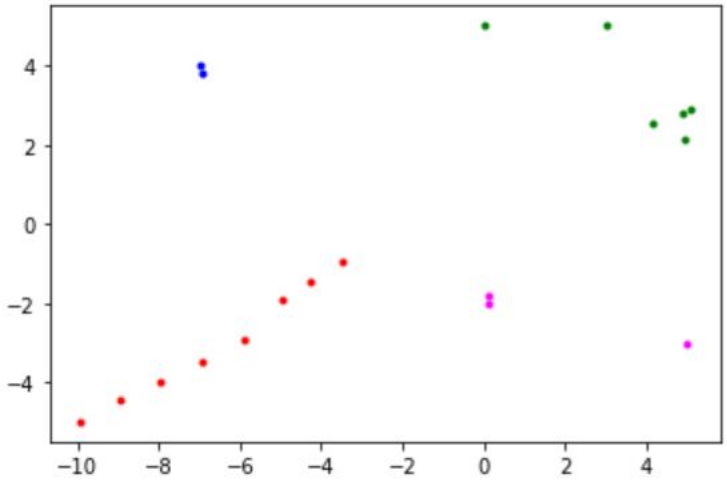
Mean-Link:

Cluster 1: [[-6.91 3.8] [-6.99 4.]]

Cluster2: [[-9.92 -4.98] [-8.96 -4.41] [-7.98 -3.98] [-6.91 -3.47] [-5.91 -2.91] [-4.94 -1.91] [-4.24 -1.44] [-3.45 -0.97]]

Cluster 3: [[4.91 2.82] [5.07 2.88] [4.93 2.12] [4.15 2.56] [3.01 5.01] [0. 5.]]
Cluster 4: [[0.13 -1.8] [0.1 -2.] [5. -3.]]

first cluster	second cluster	distance
10	12	0.16944488242854633
13	14	0.19528929532997621
0	1	0.21391809640396484
9	10	0.7356692785483661
9	10	0.8268857474710963
6	7	0.8426573444474812
2	3	1.0712595569036494
3	4	1.1447658914304835
4	5	1.3403466527683667
1	2	1.65351856976932
2	3	2.8045414127112185
3	7	2.9874825241339704
1	2	4.506896584930611
2	5	4.815686959793644
3	4	5.0082485659887



B (5 points): Mean link seems to outperform other methods, the clusters seems to make more sense as they are on average close to one another. Complete link could potentially grab outliers. Single link is probably easiest to compute but a drawback of this method is that it tends to produce long thin clusters in which nearby elements of the same cluster have small distances
Time complexity of complete-link: clustering is at most $O(n^2 logn)$. One $O(n^2 logn)$ algorithm is to compute the n^2 distance metric and then sort the distances for each data point (overall time: $O(n^2 logn)$.
Time complexity of single-link clustering is $O(n^2)$. We first compute all distances in $O(n^2)$. While doing this we also find the smallest distance for each data point and keep them in a next-best-merge array. In each of the n-1 merging steps we then find the smallest distance in the next-best-merge array. We merge the two identified clusters, and update the distance matrix in $O(n)$.
Time complexity of average-link clustering is $O(n^2 logn)$. It is similar to complete link but at each iteration we have to compute the average, so this is the most expensive and maybe have to apply dimensionality reduction.

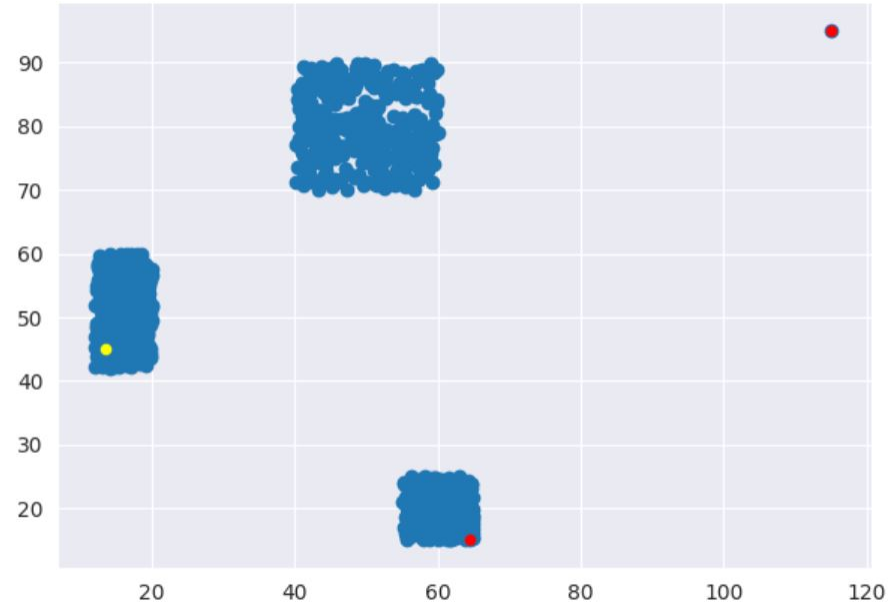
2 Assignment-Based Clustering (65 points)

Assignment-based clustering works by assigning every point $x \in X$ to the closest cluster centers C. Let $\phi_C : X \to C$ be this assignment map so that

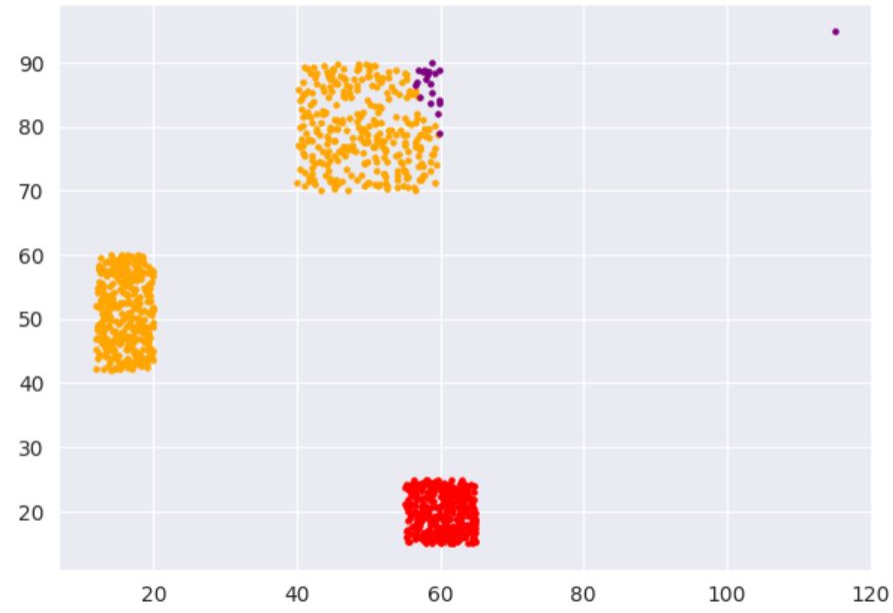
Two good heuristics for this type of clustering are the Gonzalez (Algorithm 8.2.1 in M4D book) and k-Means++ (Algorithm 8.3.2) algorithms.

A: (15 points) Gonzalez:
We choose c1 as the first point in the dataset. The Gonzalez algorithm finds the next center which is farthest from the current center. And it then finds the third center which is farthest from the first two centers.

Cluster Centeroids: [[13.51372985, 45.03355641], [115., 95.], [64.53270534, 15.09821553]]
3-center cost: 60.09493752598304
3-means cost: 29.590556306970086
Yellow point is the first initial centeroid, red points generated by algorithm

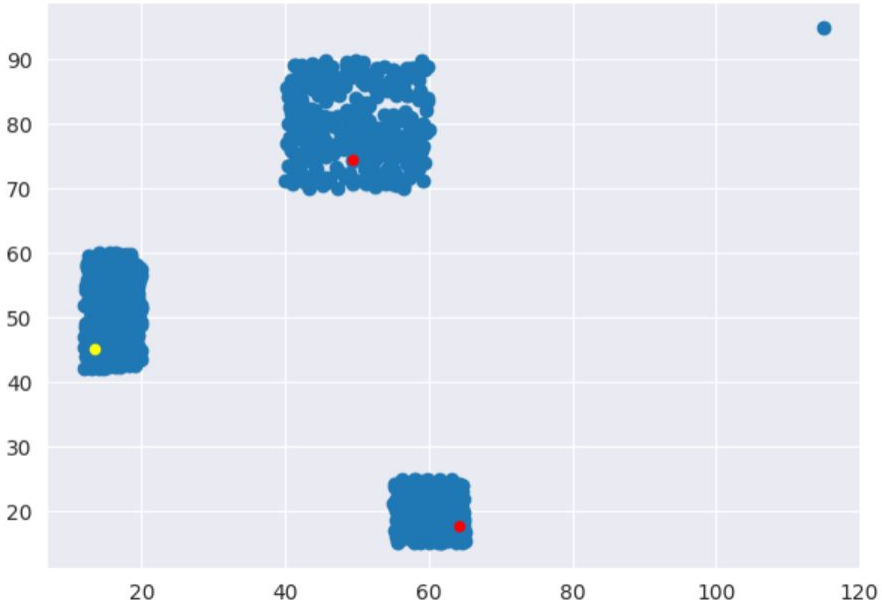


From the plot we can clearly observe that the point at the top right is an outlier and since Gonzalez algorithm is biased towards outliers it picks that point as one of the cluster centers.
Subsets by colors:

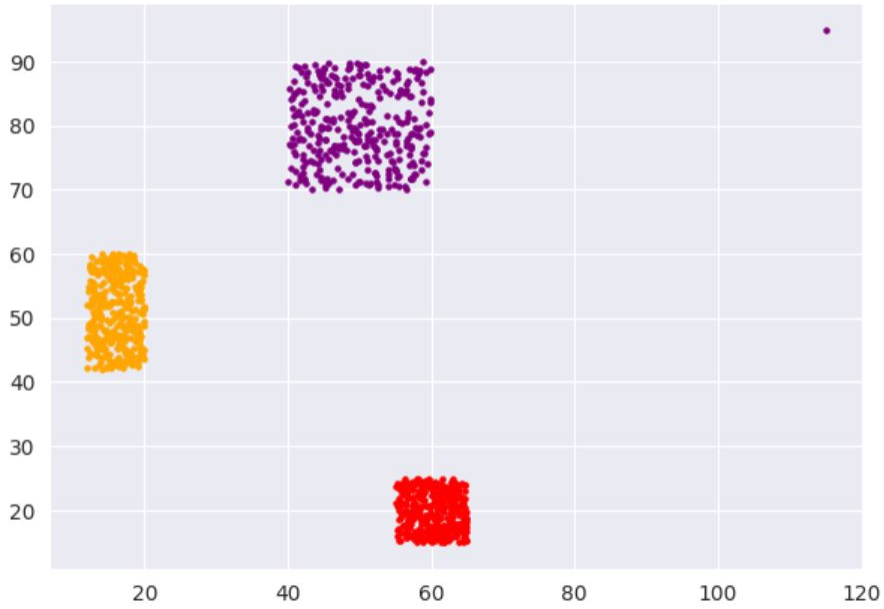


B: (20 points) K-Means++
We choose c1 as the first point in the dataset. We then determine the next c_i by picking an element with a probability proportional to the $(Distance)^2$.
We basically fit in the values we get into a distribution and then randomly pick a center from the distribution. Once we obtain the center we then update the points to the new centers if they are closer to the new center then the previous centers and then repeat the whole process until we end up with k clusters.

Cluster Centeroids: [[13.51372985 45.03355641] [64.29245108 17.69646212] [49.4824992 74.37994042]]
3-center cost: 68.68573191456251
3-means cost: 8.604109877189153
Yellow point is the first initial centeroid, red points generated by algorithm

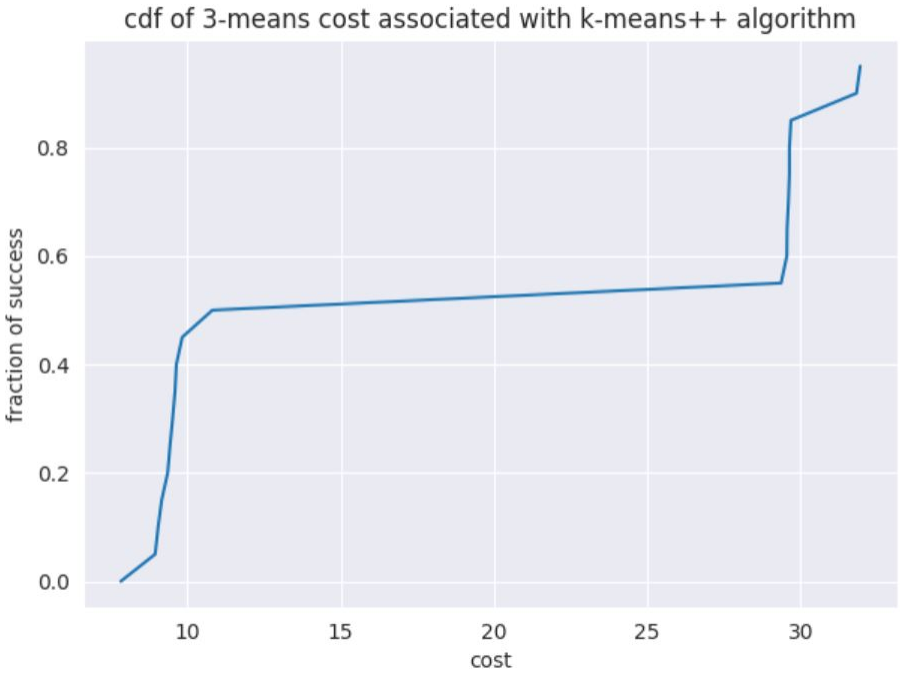


Subsets by colors:

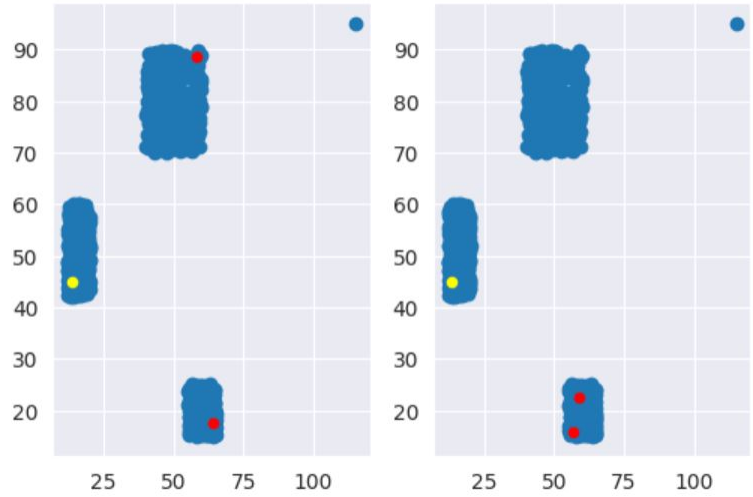


Kmeans yields better results than Gonzalez as it does not grab outliers.

Running 20 trials , we get these variations for
3-means costs:
[9.43658 9.04801 29.64258 9.83396 7.83159 31.95149 9.6438 31.83162 29.56669 9.59884 9.522 29.37196 10.81921 29.69235 9.16522 8.95182 29.61109 29.55782 9.36048 29.64256]
centroids:
[[[13.513729853983081, 45.033556411507824], [59.988874023477905, 22.274462216197847], [52.52590218071763, 70.73577721729384]], [[13.513729853983081, 45.033556411507824], [61.20574620043293, 22.735533528863808], [57.224897235758746, 76.73661216871683]], [[13.513729853983081, 45.033556411507824], [56.637296105355205, 18.29141651486513], [62.99100540430472, 17.281050233393078]], [[13.513729853983081, 45.033556411507824], [58.552091338498656, 74.50270456589206], [63.44553840817501, 17.8417449361255]], [[13.513729853983081, 45.033556411507824], [51.023921789061184, 78.58377515596645], [57.54558124438772, 21.16783392989684]], [[13.513729853983081, 45.033556411507824], [58.658466123311186, 73.1758589900369], [55.12623467284194, 73.05495448762835]], [[13.513729853983081, 45.033556411507824], [61.20574620043293, 22.735533528863808], [59.67521681187035, 78.7729651610088]], [[13.513729853983081, 45.033556411507824], [46.473799469822104, 87.73736100649961], [45.09216605802008, 71.38291604929103]], [[13.513729853983081, 45.033556411507824], [61.03378377212428, 22.097236963648506], [60.114176659008855, 15.071885176481585]], [[13.513729853983081, 45.033556411507824], [58.008674814101155, 20.63008344337366], [45.26322630719664, 70.86466404866877]], [[13.513729853983081, 45.033556411507824], [44.68411735362903, 88.22407274801745], [59.406271671531286, 17.211522895309464]], [[13.513729853983081, 45.033556411507824], [61.06197698151743, 18.83361807536688], [57.943785426320645, 24.85947690909029]], [[13.513729853983081, 45.033556411507824], [41.14955242295295, 89.26264840339012], [63.338424476271534, 16.41346373821168]], [[13.513729853983081, 45.033556411507824], [62.87643576643215, 18.366143825396424], [64.38707418972835, 18.212663432078863]], [[13.513729853983081, 45.033556411507824], [60.7532729651175, 15.915484394098044], [43.7965875056429, 85.46079879591508]], [[13.513729853983081, 45.033556411507824], [59.05425486042056, 18.16804527142228], [43.7965875056429, 85.46079879591508]], [[13.513729853983081, 45.033556411507824], [61.50881455460594, 16.586723893891833], [58.08321750233146, 20.706514335319692]], [[13.513729853983081, 45.033556411507824], [58.58144571635126, 22.67312082617855], [58.762301841513484, 15.063419670858998]], [[13.513729853983081, 45.033556411507824], [59.97617149322907, 19.138513742408268], [45.742184504858706, 71.09798629221457]], [[13.513729853983081, 45.033556411507824], [55.579930586494726, 15.1694010883857], [59.97617149322907, 19.138513742408268]]]



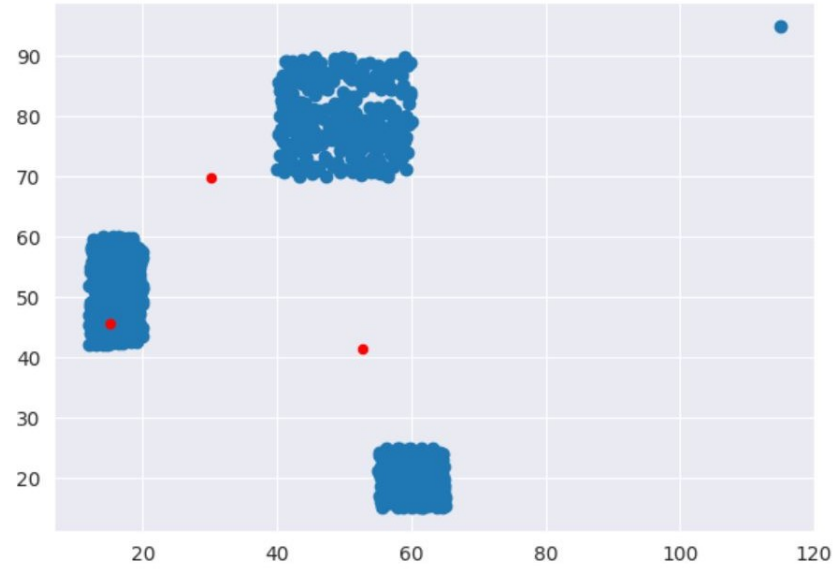
Out of 20 trials, Close 3-means compared to Gonzalez is about 5
There's approximately only two kinds of outputs the CDF will produce as shown in the picture above. The centeroids on the left plot falls in the first rapid growth stage of the CDF (cost: 1 - 11), followed by the second rapid growth stage of the CDF (cost:$>$ 29.59) on the right. The right growth stage is when KMean++ picks up outliers as represented in Gonzales, which happens but not often
Variants of the algorithm:



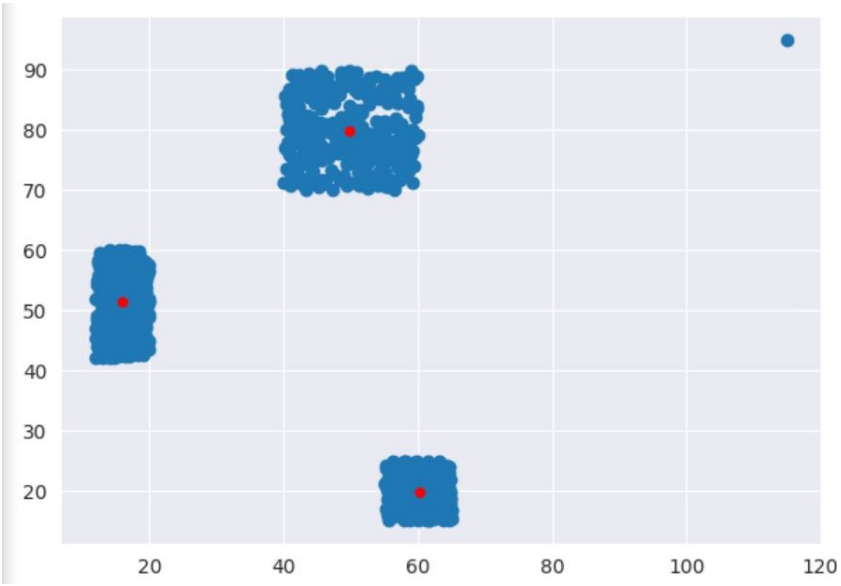
C: (30 points) In Lloyds algorithm we initally choose a set of k centers randomly and then perform the averaging operation to get a better cluster center.

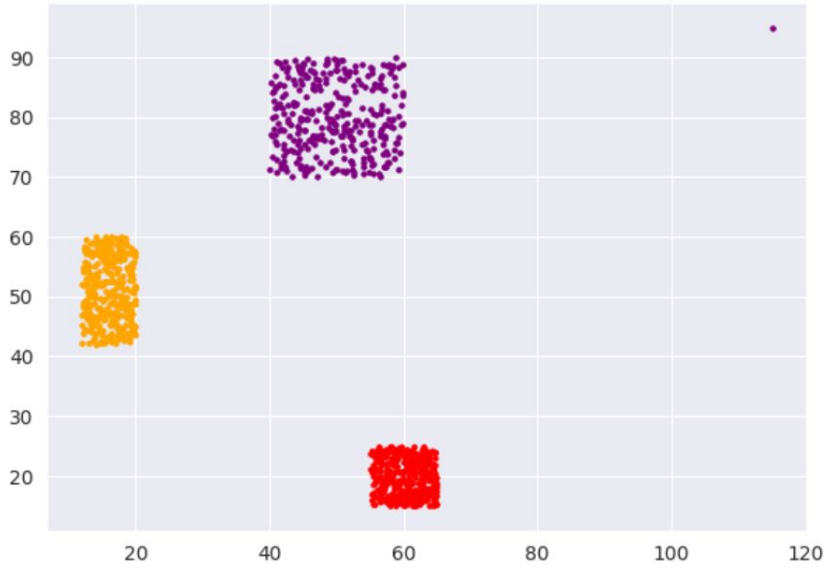
1: Run Lloyds Algorithm with C initially with points indexed $\{1, 2, 3\}$. Report the final subset and the 3-means cost.
Run Lloyds Algorithm with C = [[13.51372985398308, 45.03355641150783], [15.075594712855686, 55.97440969831794], [19.71316191003929, 50.68096395347217]]

Centroids seems to converge quickly after a few iterations.
This is after one iterations



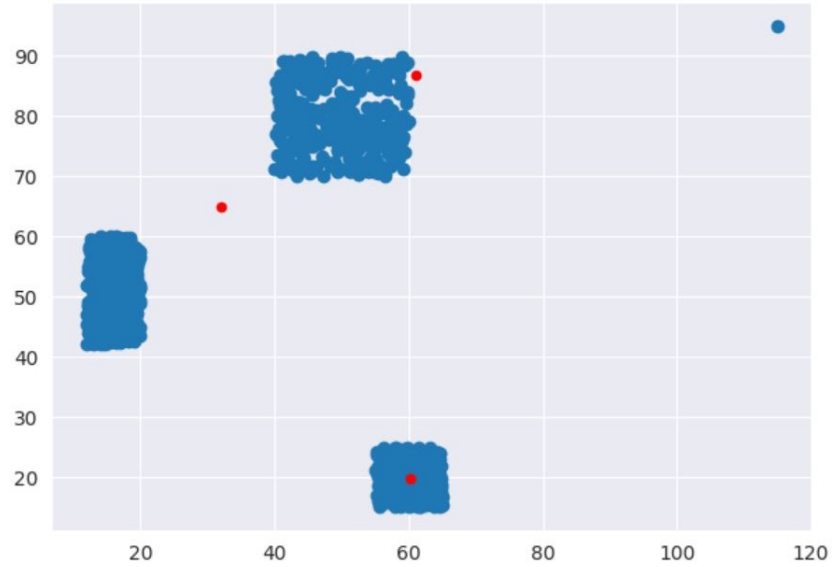
After 5 iterations, the centroids are improved significantly
New Cluster Centeroids: [[16.05121, 51.32798], [49.73053, 79.88286], [60.28161, 19.80588]]
3-means cost: 6.4764399645966



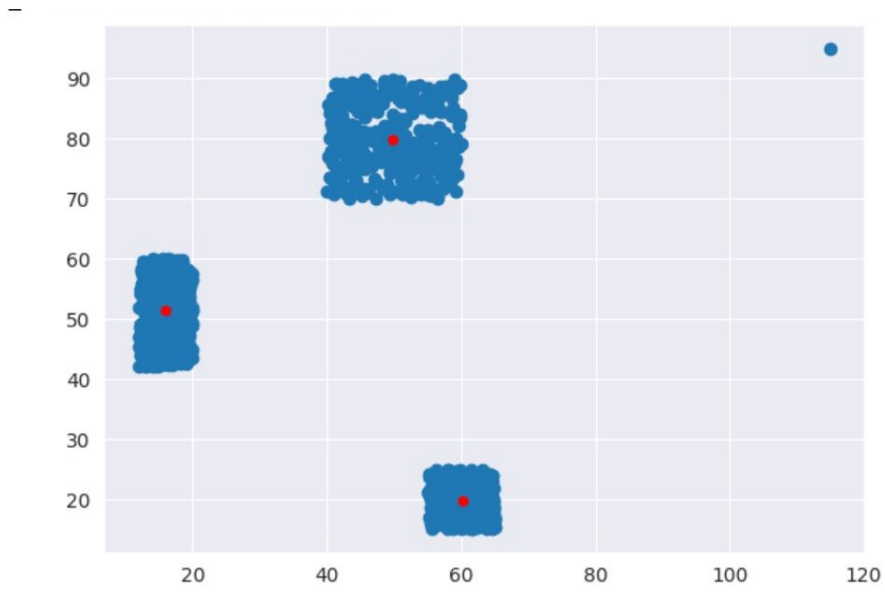


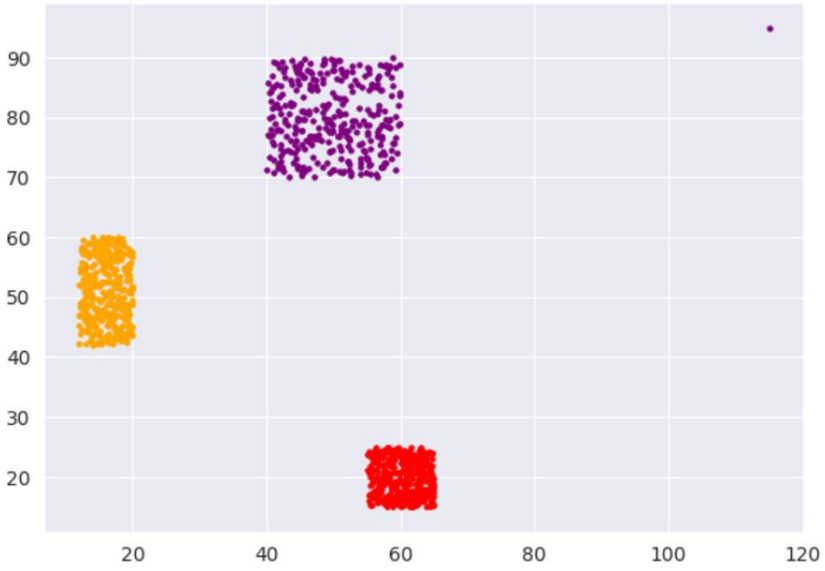
2: Run Lloyds Algorithm with C initially as the output of Gonzalez above. Report the final subset and the 3-means cost. Run Lloyds Algorithm with C initially as the output of Gonzalez above [[13.51372985, 45.03355641], [115., 95.], [64.53270534, 15.09821553]]
Centroids seems to converge quickly after a few iterations.

This is after one iterations



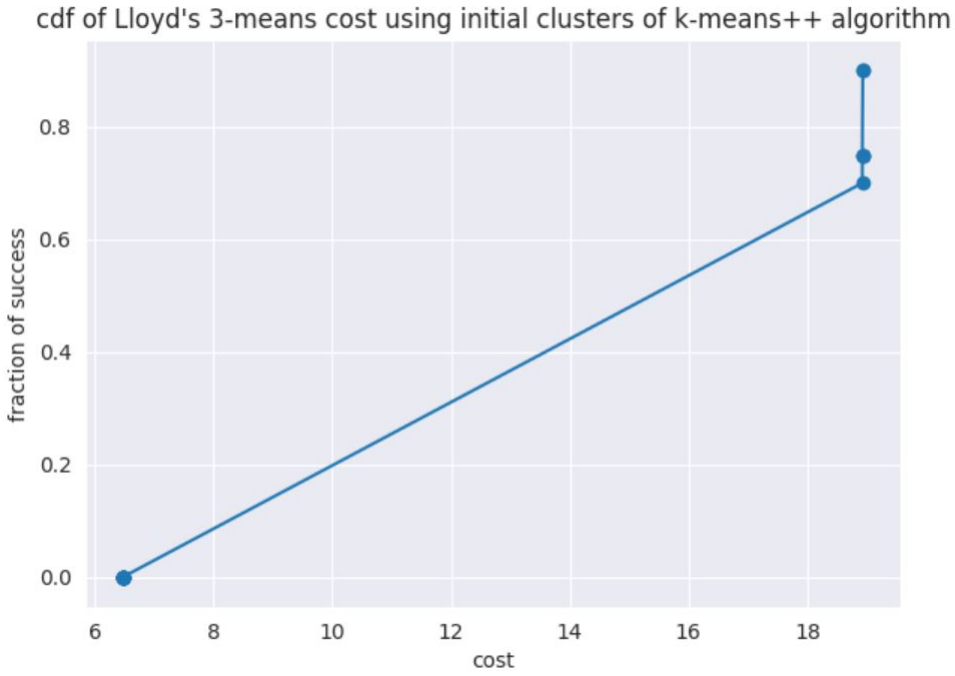
After 5 iterations, the centroids are improved significantly
After 5 iterations,
New Centeroids: [[16.05121, 51.32798], [49.73053, 79.88286], [60.28161, 19.80588]]
3-means cost: 6.4764399645966





3: Run Lloyds Algorithm with C initially as the output of each run of k-Means++ above. Plot a *cumulative density function* of the 3-means cost. Also report the fraction of the trials that the subsets are the same as the input (where the input is the result of k-Means++).

Using the 20 trials of KMeans above, we plot the CDF
About 75 percent of the trials will result in an average cost of 6. The other 25 percent cost is way higher than the optimal case.



Fraction of the trials that the subsets are the same as the input is 0. In the Kmeans algo, we initialize one of the center is the first point, Lloyd algo performs averaging operations to get better clusters so intuitively these new centers should be different than Kmeans centers.