Assignmentmt 3: Clustering

Christopher Mertin/u1010077
cmertin@cs.utah.edu
February 28, 2017

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/A3/C1.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A3/C2.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5140/A3/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=20 points, `C2` should have n=1004 points, and `C3` should have n=1000 points. We will always measure distance with Euclidean distance.

As usual, it is highly recommended that you use LaTeX for this assignment. 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 (20 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 (20 points): Run all hierarchical clustering variants on data set `C1.txt` until there are $k = 4$ clusters, and report the results as sets. It may be useful to do this pictorially.

Which variant did the best job, and which was the easiest to compute (think if the data was much larger)? Explain your answers.

- **Complete-Link:** The worst case time complexity is $\mathcal{O}\left(n^2 \log(n)\right)$. This comes about from updating the distance metric in $\mathcal{O}(n)$ after each merge iteration. We then pick the next pair to merge by finding the smallest distance that is still eligible for merging. If this is done by traversing the n sorted lists of distances, then we have done n^2 traversal steps by the end of clustering. Adding these up gives $\mathcal{O}\left(n^2 \log(n)\right)$.

- **Single-Link:** Complexity is $\mathcal{O}\left(n^2\right)$. We first compute all distances which is $\mathcal{O}\left(n^2\right)$. While doing this, we also find the smallest distance for each data point and keep them in teh 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 $\mathcal{O}\left(n\right)$. Finally, we update the next-best-merge array in $\mathcal{O}\left(n\right)$ in each step. This can be done in $\mathcal{O}\left(n\right)$ because if the best merge partner for k before merging i and j was either i or j, then after merging i and j, the best marge partner for k is the merger of i and j.

- **Mean-Link:** Complexity is $\mathcal{O}\left(n^2 \log(n)\right)$. We first compute all n^2 similarities bfor the singleton clusters, and sort them for each cluster $\left[\text{time: } \mathcal{O}\left(n^2 \log(n)\right)\right]$. In each of the $\mathcal{O}\left(n\right)$ merge iterations, we identify the pair of clusters with the highest cohesion in $\mathcal{O}\left(n\right)$, merge the pair, and update the cluster centroids, ϕ, and cohesions of the $\mathcal{O}\left(n\right)$ possible mergers of the just created cluster with the remaining clusters. For each cluster, we also update the sorted list of merge candidates by deleting the two just merged clusters and inserting its cohesion with the just created cluster. Each iteration thus takes $\mathcal{O}\left(n \log(n)\right)$. Overall, time complexity is then $\mathcal{O}\left(n^2 \log(n)\right)$.

Based on the above complexities, the best algorithm for *large* data sets is *Single-Link* as it's complexity is only $\mathcal{O}\left(n^2\right)$ compared to the other two which are bounded by $\mathcal{O}\left(n^2 \log(n)\right)$. The resulting dendrograms of the 3 cluster variants can be seen below. The numbers $\{0, 1, 2, 3\}$ denote the cluster that the number belongs to.

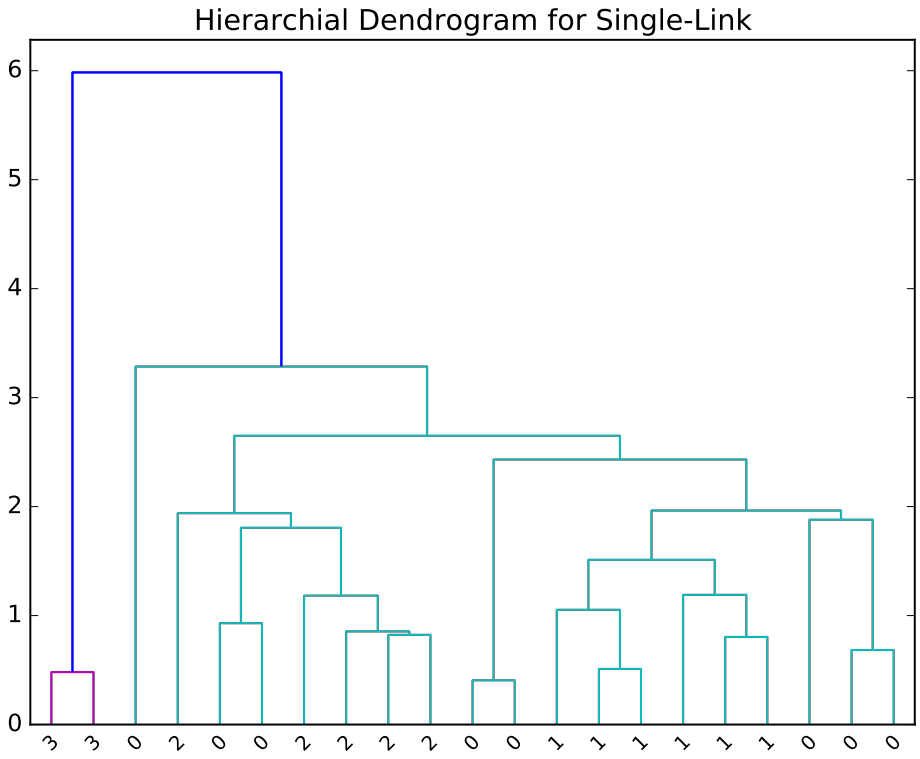


Figure 1: Dendrogram for single link

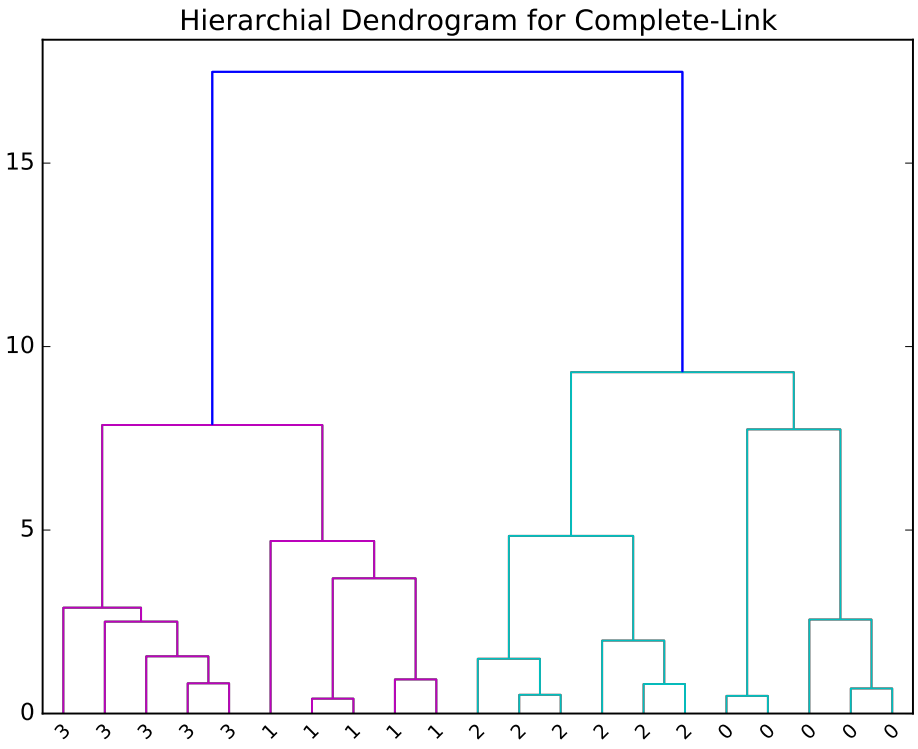


Figure 2: Dendrogram for complete link

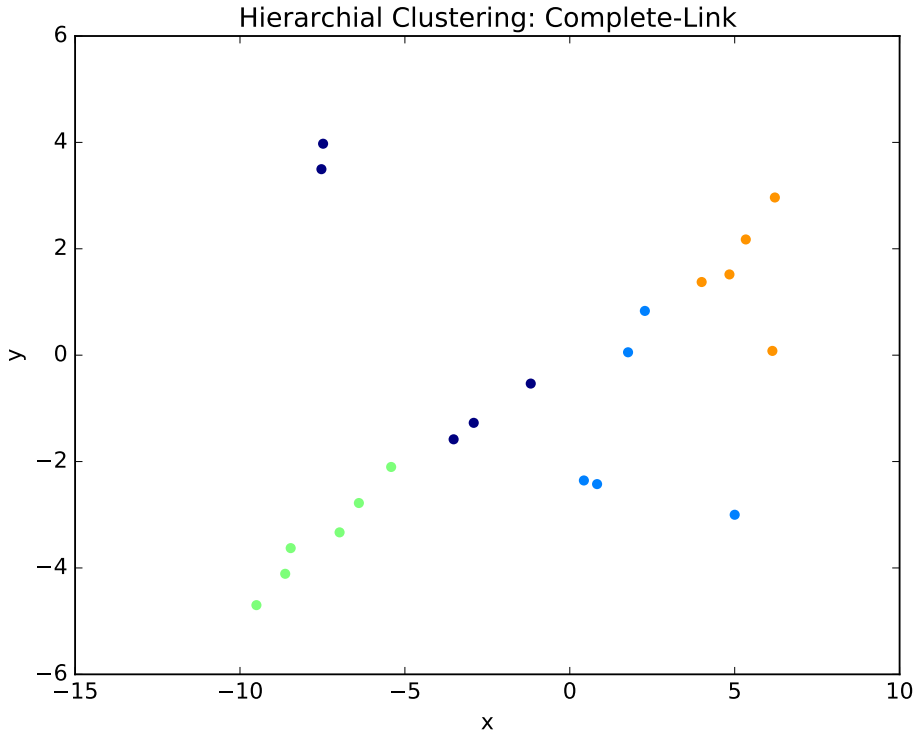


Figure 3: Clustering of points with Complete-Link

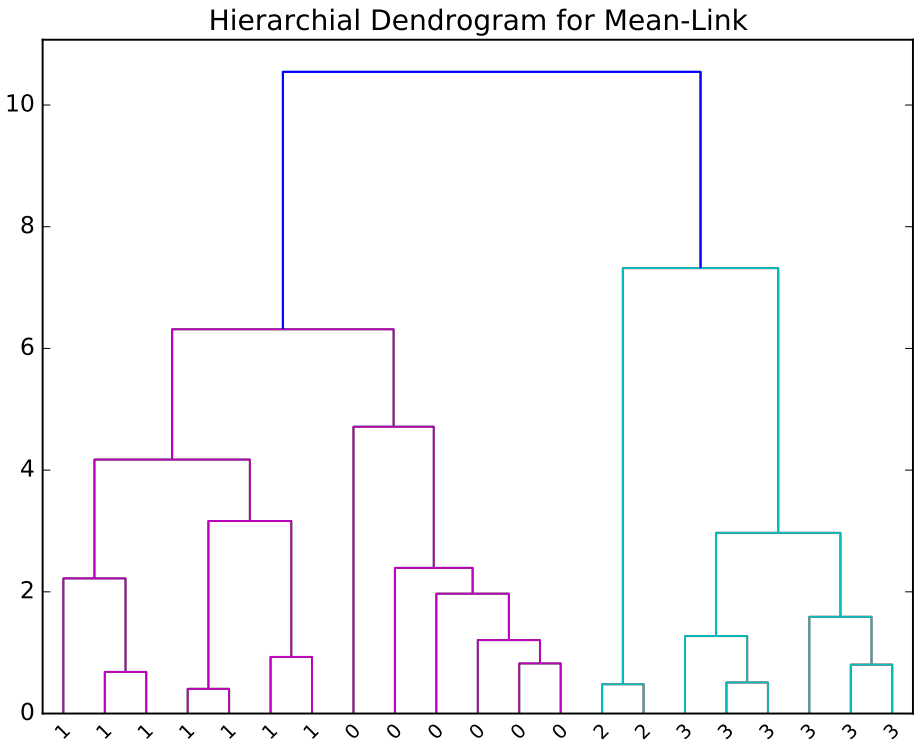


Figure 4: Dendrogram for Mean-Link

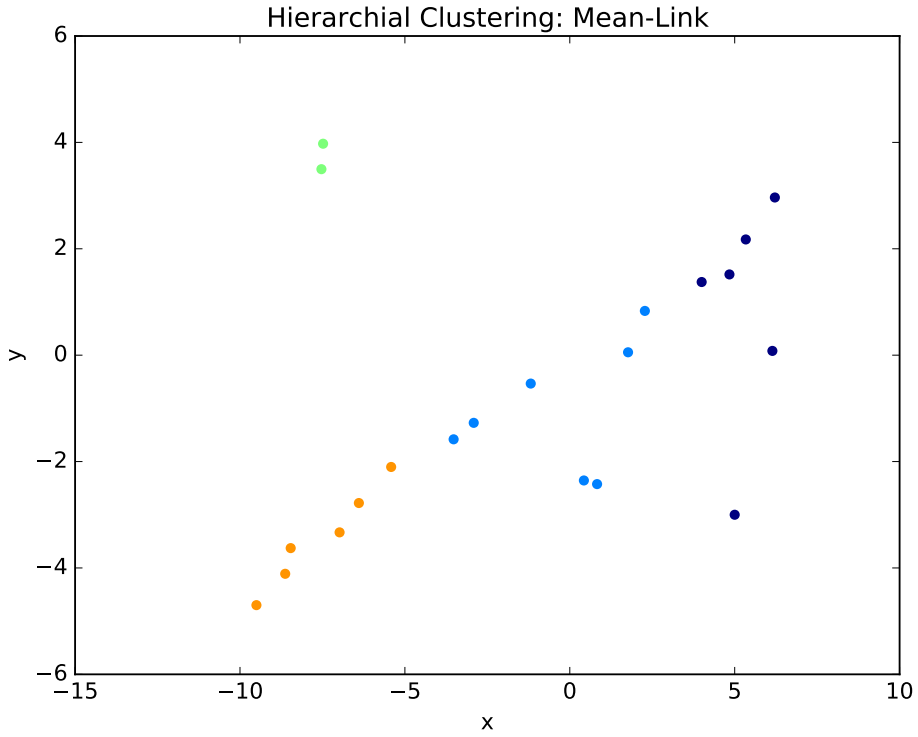


Figure 5: Clustering of points with Mean-Link

2 Assignment-Based Clustering (40 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 $\phi_C(x) = \arg\min_{c \in C} \mathsf{D}(x, c)$. All points that map to the same cluster center are in the same cluster.

Two good heuristics for these types of cluster are the Gonzalez (Algorithm 9.4.1) and k-Means++ (Algorithm 10.1.2) algorithms.

A: (20 points) Run Gonzalez and k-Means++ on data set `C2.txt` for $k = 3$. To avoid too much variation in the results, choose c_1 as the point with index `1`.

Report the centers and the subsets (as pictures) for Gonzalez. Report:

- the 3-center cost $\max_{x \in X} \mathsf{D}(x, \phi_C(x))$ and
- the 3-means cost $\sqrt{\frac{1}{|X|} \sum_{x \in X} (\mathsf{D}(x, \phi_C(x)))^2}$
  (Note this has been normalized so easy to compare to 3-center cost)

For k-Means++, the algorithm is randomized, so you will need to report the variation in this algorithm. Run it several trials (at least 20) and plot the *cumulative density function* of the 3-means cost. Also report what fraction of the time the subsets are the same as the result from Gonzalez.

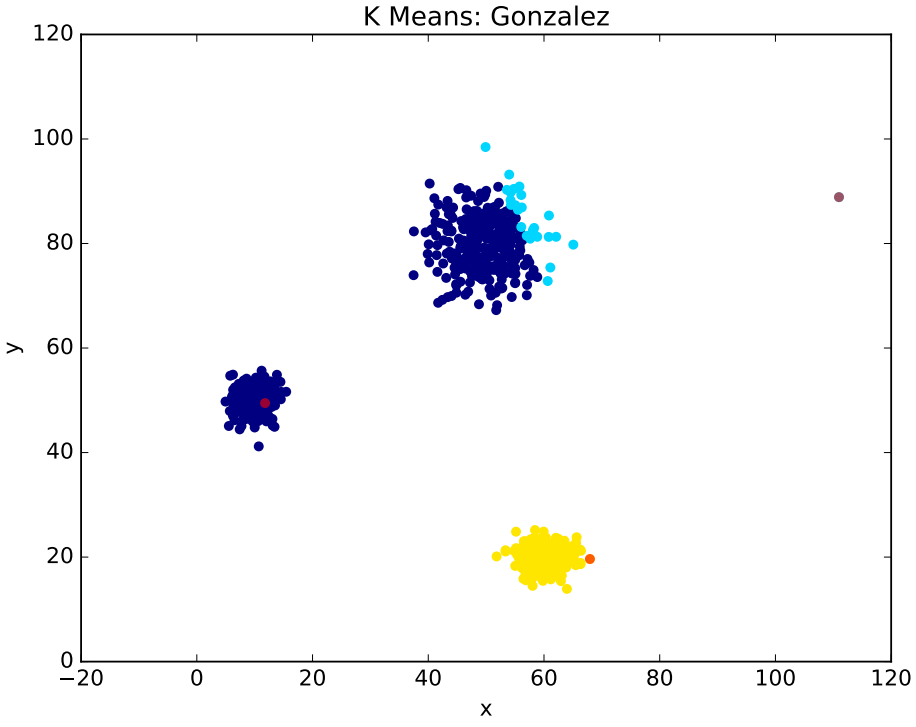


Figure 6: Clustering results with `Gonzalez`

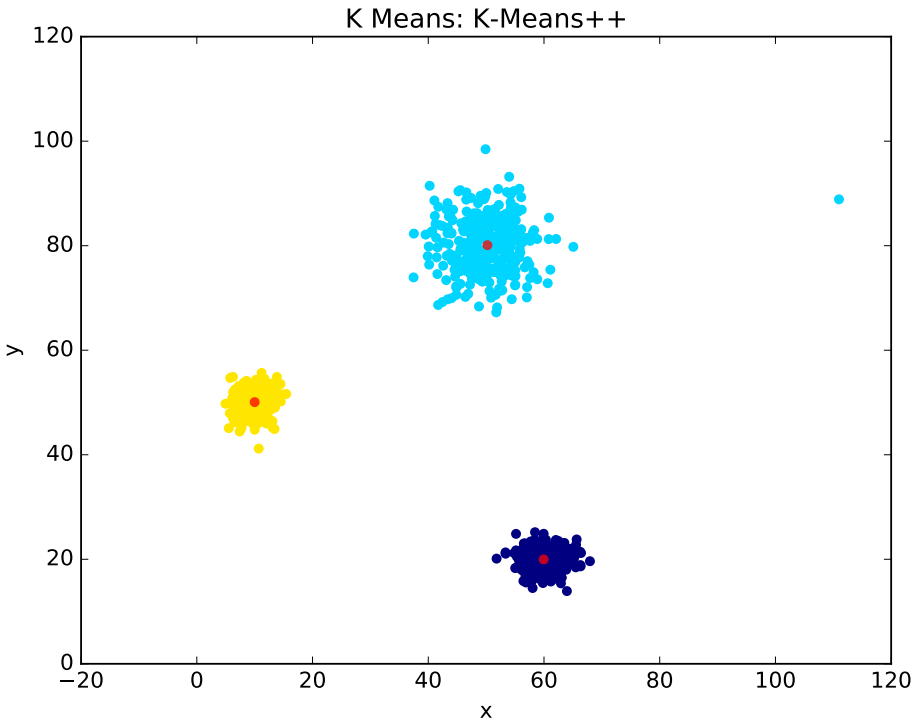


Figure 7: Clustering results with `K-Means++`

Table 1: Gonzalez Centroids

c_1	c_2	c_3
$(11.797, 49.467)$	$(110.981, 88.871)$	$(67.941, 19.630)$

Table 2: 3-Center Cost (Gonzalez)

c_1	c_2	c_3
106.725	114.176	81.528

Table 3: 3-Mean Cost (Gonzalez)

c_1	c_2	c_3
1889.118	7749.613	2737.042

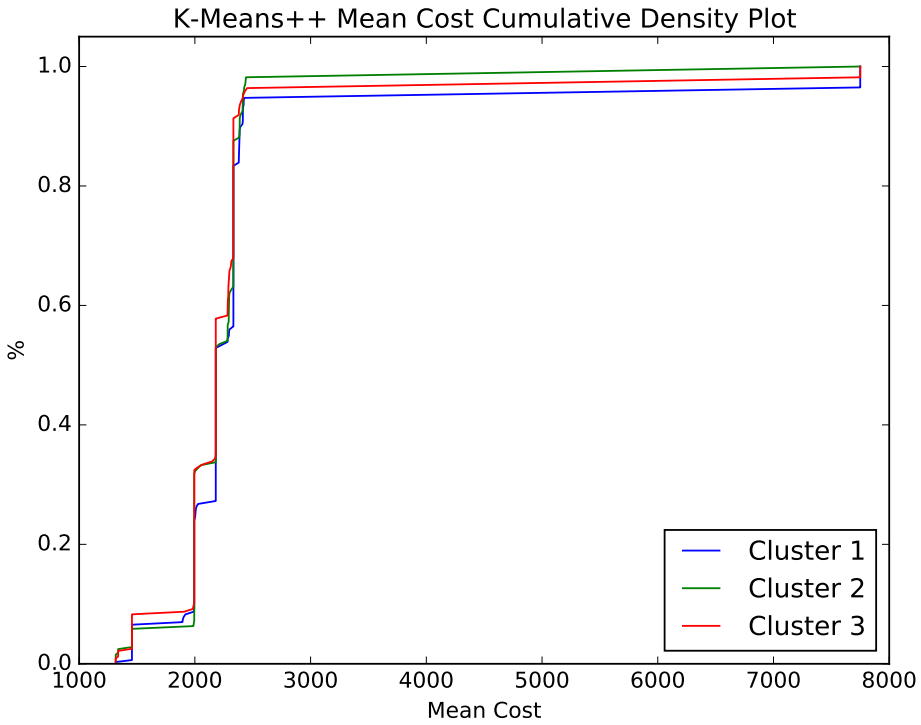


Figure 8: 3-Mean Cost of `K-Means++`

B: (20 points) Recall that Lloyd's algorithm for k-means clustering starts with a set of k centers C and runs as described in Algorithm 10.1.1.

- Run Lloyds Algorithm with C initially with points indexed $\{1, 2, 3\}$. Report the final subset and the 3-means cost.

Table 4: 3-Means Cost

c_1	c_2	c_3
1456.310	2283.676	2426.06

Table 5: Centroid Locations

	x_1	x_2
c_1	29.027	64.334
c_2	58.485	19.850
c_3	63.158	20.778

- Run Lloyds Algorithm with C initially as the output of Gonzalez above. Report the final subset and the 3-means cost.

Table 6: 3-Means Cost

c_1	c_2	c_3
1994.695	2180.608	2332.941

Table 7: Centroid Locations

	x_1	x_2
c_1	9.997	50.069
c_2	50.217	80.101
c_3	59.947	19.973

- 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++).

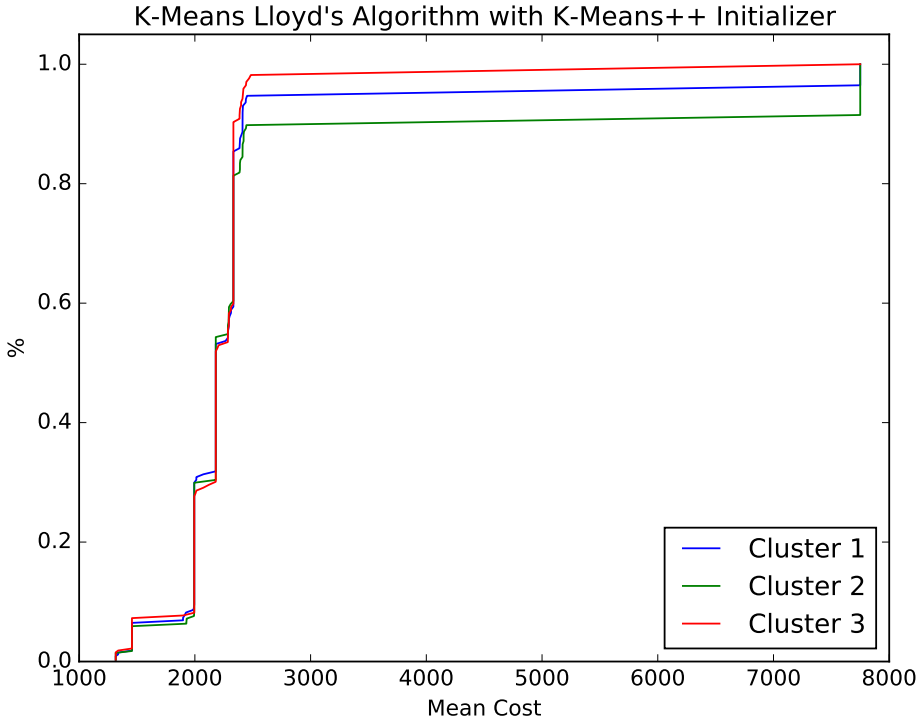


Figure 9: 3-Means cost for Lloyd's

3 High-dimensional Distances (15 points)

In class, we compared the volume of an ℓ_2 ball centered at the origin, with the smallest box containing that ball. Here I will ask you to calculate how much we will have to expand the radius of the ball to have the same volume as the box.

Recall, the volume of a d-dimensional ℓ_2 ball with radius r is

$$\text{vol}(B(d,r)) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d.$$

Here $\Gamma(z) = \int_{x=0}^{\infty} x^{z-1} e^{-x} dx$ is known as the Gamma function. It is known that $\Gamma(2) = 1$, $\Gamma(2.5) \approx 1.33$, $\Gamma(3) = 2$ and in general for integers z that $\Gamma(z) = (z-1)!$ *(that is a factorial, not just excitement).*

For any ball $B(d,r)$ (for instance one or radius $r = 1$), let $\text{vol}(\text{box}(d,r)) = (2r)^d$ be the volume of the smallest enclosing box. Determine the value of $r' = cr$ (an expanded radius greater than r) that we would have to expand the ball to so that it is the same as $\text{vol}(\text{box}(d,r))$. Solve for the expansion factor c:

1. for $d = 2$

   `1.128`

2. for $d = 3$

   `1.241`

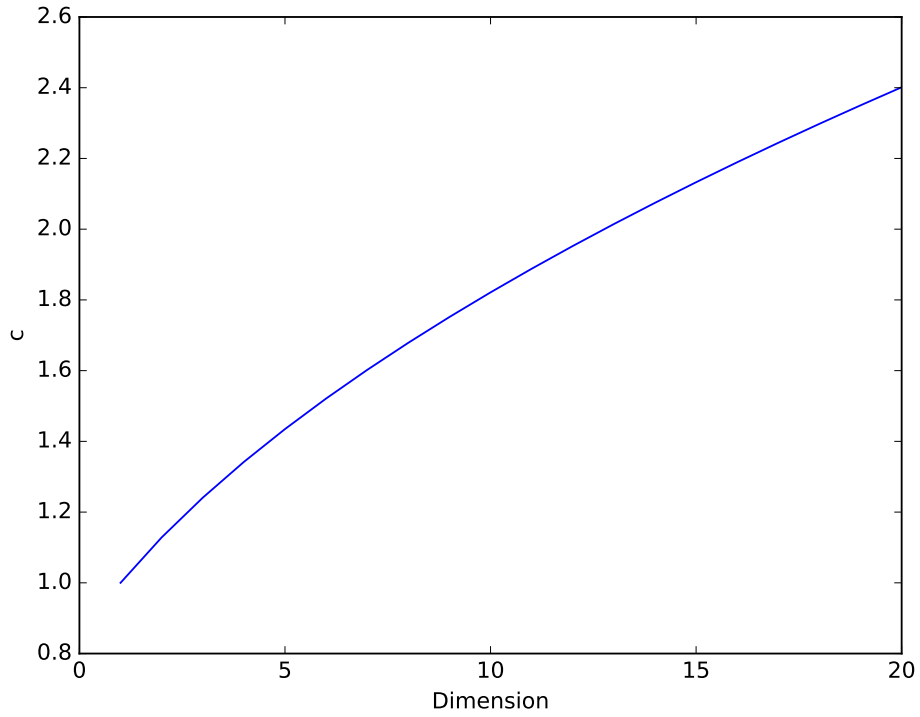
3. for $d = 4$, and

   `1.342`

4. as a function of d (for large d), restricting to even values of d.

$$c = \left(\frac{\left[2^{-d} r^d \frac{\pi^{d/2}}{\Gamma(d/2+1)} \right]}{r} \right)^{-1}$$

5. Plot the expansion factor c up to $d = 20$.



4 k-Median Clustering (25 points)

The k-median clustering problem on a data set P is to find a set of k-centers $C = \{c_1, c_2, \ldots, c_k\}$ to minimize $\mathsf{Cost}_1(P, C) = \frac{1}{|P|} \sum_{p \in P} \mathsf{D}(p, \phi_C(p))$. We did not explicitly talk much about this formulation in class, but the techniques to solve it are all typically extensions of approaches we did talk about. This problem will be more open-ended, and will ask you to try various approaches to solve this problem. We will use data set `C3.txt`.

A: (20 points) Find a set of 4 centers $C = \{c_1, c_2, c_3, c_4\}$ for the 4-medians problem on dataset `C3.txt`. Report the set of centers, as well as $\mathsf{Cost}_1(P, C)$. The centers should be in the write-up you turn in, but also in a file formatted the same was as the input so we can verify the cost you found. That is each line has 1 center with 6 tab separated numbers. The first being the index (e.g., 1, 2, 3 or 4), and the next 5 being the 5-dimensional coordinates of that center.

Your score will be based on how small a $\mathsf{Cost}_1(P, C)$ you can find. You can get 15 points for reasonable solution. The smallest found score in the class will get all 20 points. Other scores will obtain points in between.

Very briefly describe how you found the centers.

To implement this algorithm, the above cost function was implemented to be minimized with Lloyd's algorithm. Lloyd's algorithm requires some initial cluster points, so 4 points were chosen at random from the given set. As there is no way to guarenttee that the points chosen at random were the *best*, this was run for 1000 epochs, with the centroids giving the best cost being saved.

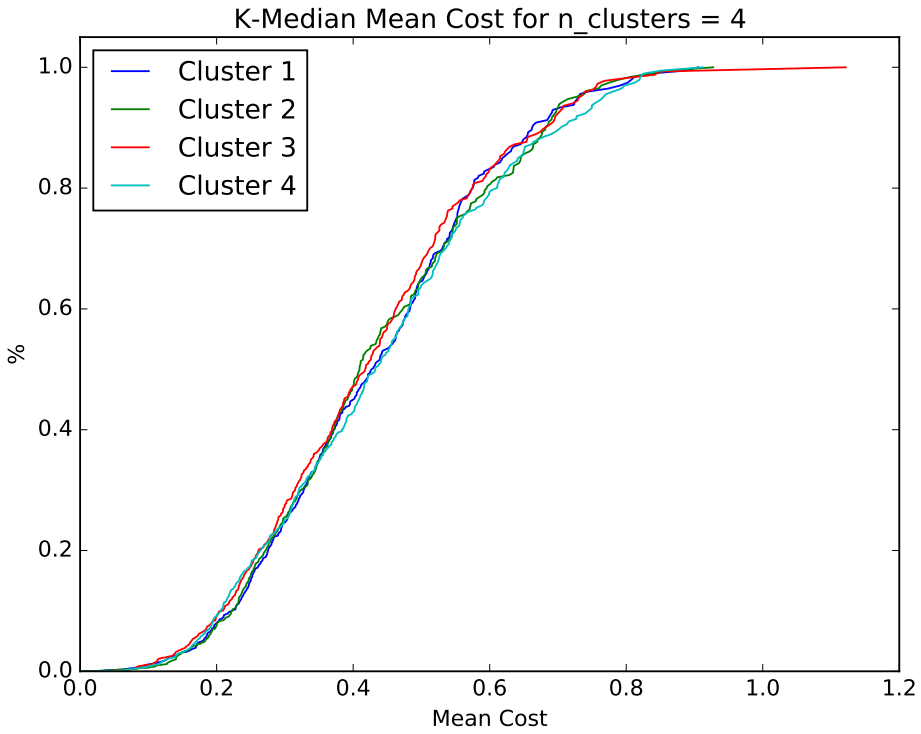
Below, the CDF of the mean cost for each cluster can be found as well for the trials.

Table 8: Centroids for $|C| = 4$

	x_1	x_2	x_3	x_4	x_5
c_1	0.188	0.845	0.141	0.132	0.037
c_2	1.007	0.120	−0.164	0.085	0.190
c_3	−0.035	0.106	0.945	0.044	−0.134
c_4	−0.257	0.766	0.209	0.191	0.141

Table 9: Mean Cost for $|C| = 4$

c_1	c_2	c_3	c_4
0.303	0.238	0.411	0.220



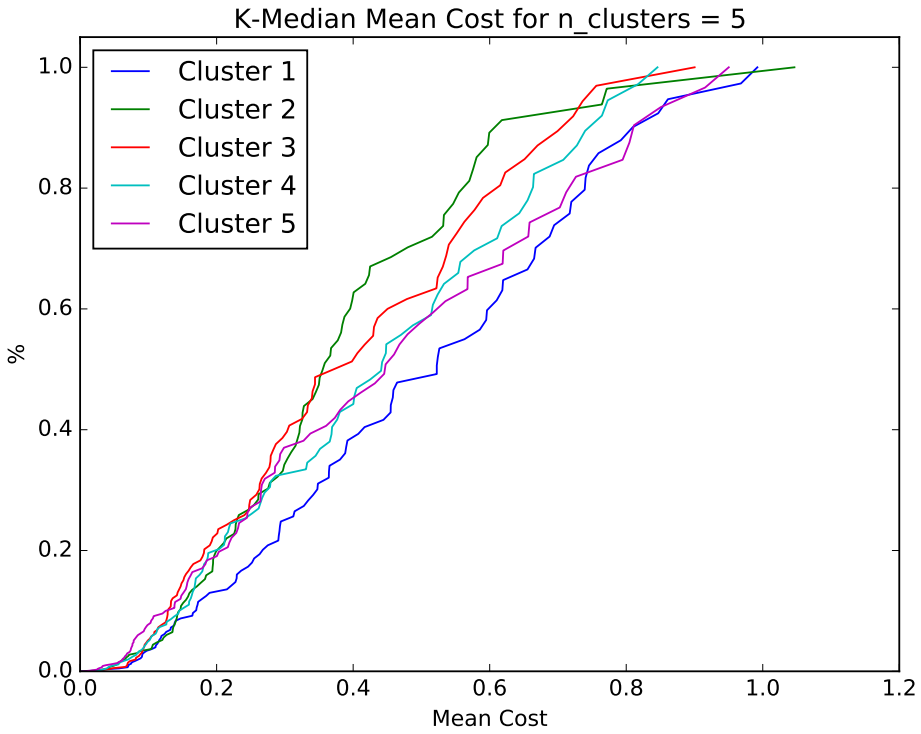
B: (5 points) Run your algorithm again for the 5-medians problem on dataset `C3.txt`. Report the set of 5 centers and the $\mathsf{Cost}_1(P, C)$. You do not need to turn in a file for these, just write it in your report.

Table 10: Centroids for $|C| = 5$

	x_1	x_2	x_3	x_4	x_5
c_1	-0.016	0.075	1.180	-0.050	-0.113
c_2	0.008	-0.096	0.633	0.135	0.034
c_3	-0.087	-0.306	0.053	0.848	0.222
c_4	-0.025	0.187	0.115	0.218	0.686
c_5	0.077	-0.147	0.743	-0.087	0.093

Table 11: Mean Cost for $|C| = 5$

c_1	c_2	c_3	c_4	c_5
0.206	0.243	0.129	0.328	0.345


K-Median Mean Cost for n_clusters = 5
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
%
Mean Cost
0.0
0.2
0.4
0.6
0.8
1.0
1.2