CSE 151A - Homework 08

Due: Wednesday, May 27, 2020

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope on Wednesday at 11:59 p.m.

Essential Problem 1.

Consider the following set of data points in $\mathbb{R}^2$.

x_1	x_2
0.2	0.5
0.4	0.1
0.3	0.1
1.0	1.2
1.3	1.1
1.4	1.0

Run k-means with $k = 2$ and initial cluster centers $\vec{\mu}^{(1)} = (1.1, 1.1)^T$ and $\vec{\mu}^{(2)} = (1.3, 0.8)^T$ until convergence. At every step, give the current cluster center positions, and show your work.

Solution: A solution notebook can be found at https://go.ucsd.edu/2ManNfm.

Whether by hand or by code, we find that this converges after only two updates! The resulting centroids for each update are found below.

```
Iteration 1
        mu1 = [0.833 0.933]         mu2 = [0.7   0.467]
Iteration 2
        mu1 = [1.233 1.1  ]         mu2 = [0.3 0.3]
Iteration 3
        mu1 = [1.233 1.1  ]         mu2 = [0.3 0.3]
Converged!
```

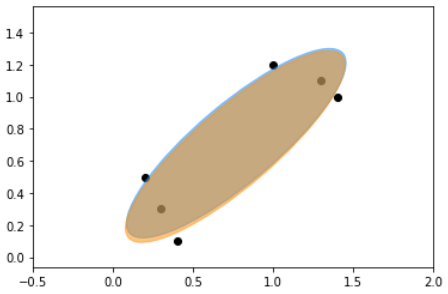
Essential Problem 2.

Using the same data as in the problem above, run the EM algorithm to fit a mixture of two Gaussians (both with full covariance matrices). Intialize the means of the Gaussians to be $\vec{\mu}^{(1)} = (1.1, 1.1)^T$ and $\vec{\mu}^{(2)} = (1.3, 0.8)^T$, set the initial covariance matrices to be the identity matrix, and set the initial mixing coefficients to both be 0.5. Run 3 iterations of the algorithm, reporting the cluster means, covariances, and mixing coefficients at each step, as well as the responsibilities of each point.

Solution: A solution notebook can be viewed at https://go.ucsd.edu/2ZMUwPz. After three iterations of the E-M algorithm, the following values are found.

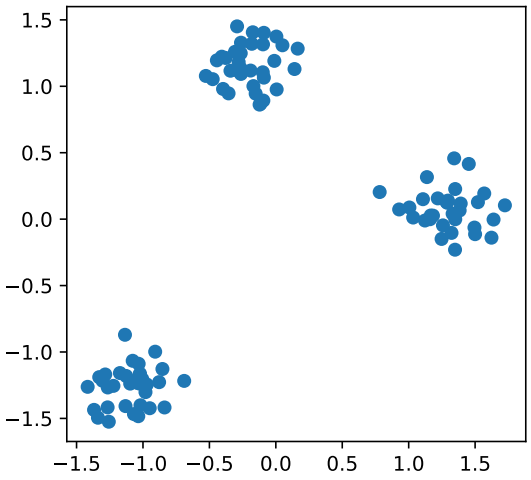
	0	1	2	3
mu0	[1.1 1.1]	[0.769 0.709]	[0.769 0.71]	[0.77 0.71]
mu1	[1.3 0.8]	[0.764 0.691]	[0.764 0.69]	[0.764 0.69]
cov0	[[1. 0.] [0. 1.]]	[[0.235 0.172] [0.172 0.175]]	[[0.235 0.172] [0.172 0.175]]	[[0.236 0.172] [0.172 0.174]]
cov1	[[1. 0.] [0. 1.]]	[[0.236 0.175] [0.175 0.178]]	[[0.236 0.175] [0.175 0.178]]	[[0.235 0.175] [0.175 0.179]]
pi0	0.5	0.503	0.503	0.503
pi1	0.5	0.497	0.497	0.497
w0	None	[0.516 0.476 0.496 0.529 0.506 0.494]	[0.52 0.473 0.495 0.527 0.506 0.496]	[0.522 0.47 0.495 0.527 0.506 0.498]
w1	None	[0.484 0.524 0.504 0.471 0.494 0.506]	[0.48 0.527 0.505 0.473 0.494 0.504]	[0.478 0.53 0.505 0.473 0.494 0.502]

Note that we appear to have stumbled on an unsatisfactory local minimum!

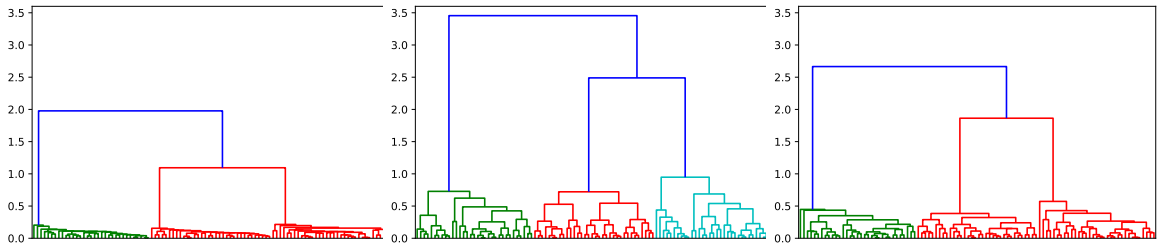


Essential Problem 3.

Consider the following set of data points:



Single-linkage, average-linkage, and complete-linkage clustering were all performed on this data. Which of the dendrograms below is the result of single-linkage, which is the result of average-linkage, and which is the result of complete-linkage? Explain your reasoning.



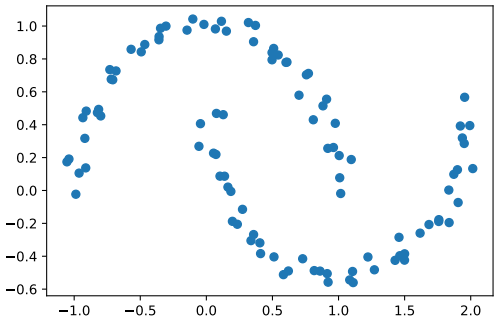
(a) (b) (c)

Solution: The easiest way is to take a look at the last merge that is made by the agglomerative algorithm and the value of the linkage function when this merge occurred. Remember that single linkage looks at the closest pair of points, complete linkage looks at the furthest pair of points, and average linkage looks at the average distance between the points in the two clusters.

Since among all of the dendrograms, dendrogram (a) features the "shortest" last merge, it must correspond to single-linkage. Likewise, (b) is for complete-linkage, and (c) must be for average-linkage.

Essential Problem 4.

Consider the data below:



Out of all of the clustering methods discussed this week, which method would you use to cluster this data if you believe that there are two clusters? Explain your reasons.

Solution: We want to choose the clustering method that will result in the two visually-separable clusters above.

Suppose we wanted to use k-means or mixtures of Gaussians for this problem. In that case, the two clusters would have respective means at roughly $(0.0, 0.4)$ and $(1.0, 0.0)$. Because these means actually both lie within their *opposite* cluster, we realize that any method which relies on cluster means will not work as intended!

Of the linkage methods we've been exposed to, it is likely that some form of **single-linkage** will produce the best results, as points within the clusters are dense and there is ample space with no noise between the two clusters. Complete-linkage and average-linkage may suffer due to the overlapping nature of the two clusters.

Plus Problem 1. (8 plus points)

In lecture, it was claimed that Lloyd's algorithm will converge to a local minimum of the k-means objective function. In this problem, we'll prove that claim.

a) Let $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)}$ be a set of n vectors in d-dimensional space. Define:

$$L(\vec{v}) = \sum_{i=1}^{n} \|\vec{x}^{(i)} - \vec{v}\|^2.$$

You can think of L as a convex function measuring how well the vector $\vec{v}$ represents the set of data points, with smaller numbers being better. Prove that L is minimized by the mean, $\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}^{(i)}$.

Solution: Recall that $\|\vec{x}\|^2 = \sum x_j^2$. This allows us to rewrite L in terms of each dimension of $\vec{v}$.

$$L(\vec{v}) = \sum_{i=1}^{n}\sum_{j=1}^{d}(x_j^{(i)} - v_j)^2$$

Why are we doing this? So that we can take the gradient of $L(\vec{v})$ to minimize it! Recall we can compute the gradient as $\nabla L(\vec{v}) = \begin{bmatrix} \frac{\partial L}{\partial v_j} \\ \vdots \end{bmatrix}$, allowing us to compute the derivative of a single dimension

$$\begin{aligned} \frac{\partial L}{\partial v_j} &= \sum_{i=1}^{n} 2(-1)(x_j^{(i)} - v_j) \equiv 0 \\ \Rightarrow \quad \sum_{i=1}^{n} v_j &= \sum_{i=1}^{n} x_j^{(i)} \\ \Rightarrow \quad v_j &= \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)} \end{aligned}$$

Therefore $\nabla L(\vec{v}) = \vec{0}$ at

$$\vec{v} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}^{(i)}$$

We are given that L is convex (and can prove it similar to one of our previous homeworks!), thus we have achieved the minimum.

b) Recall that the k-means objective function is defined to be:

$$\text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}) = \frac{1}{n}\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$$

Prove that the value of the k-means objective decreases (or stays the same) between iterations of Lloyd's algorithm.

Hint: Consider an arbitrary iteration of Lloyd's algorithm. Let $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}$ be the centers before the iteration, and let $\vec{\nu}^{(1)}, \ldots, \vec{\nu}^{(k)}$ be the centers after the iteration. Show that the cost of the new centers is smaller than the cost of the old centers. Remember that the nearest cluster center to a data point can change in an iteration, so make sure to take this into account.

Solution: Let $C_1, \ldots, C_k$ be the clusters of points assigned to labels $1, \ldots, k$ prior to iteration, and $D_1, \ldots, D_k$ be the clusters of points assigned to labels $1, \ldots, k$ after the iteration.

Define the following notation as equivalent to our cost before the iteration.

$$\text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)} \mid C_1, \ldots, C_k) = \sum_{j=1}^{k} \sum_{\vec{x}^{(i)} \in C_j} \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$$

During the first step of Lloyd's algorithm, points are assigned to their nearest cluster center. The only reason some point $\vec{x}^{(i)}$ with label j will change its label to another ℓ during this step is if $\|\vec{x}^{(i)} - \mu_\ell\|^2$ is less than the distance from the point to $\|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$.

The cost of our old cluster centers with the **new** cluster assignments must therefore less than or equal to the cost of our old cluster centers with the **old** cluster assignements.

$$\text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)} \mid D_1, \ldots, D_k) \leq \text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)} \mid C_1, \ldots, C_k)$$

During the second step of Lloyd's algorithm, cluster centers are updated to the mean of the points with the same label. We proved in part (a) that the loss function $L(\vec{v}) = \sum \|\vec{x}^{(i)} - \vec{v}\|^2$ is minimized by the mean. So, we know that by setting the cluster centers to be the respective cluster means then the cost of each individual cluster D_j is minimized, giving us

$$\sum_{\vec{x}^{(i)} \in D_j} \|\vec{x}^{(i)} - \vec{\nu}^{(j)}\|^2 \leq \sum_{\vec{x}^{(i)} \in D_j} \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$$

The cost of our **new** cluster centers with the new cluster assignments must therefore be less than or equal to the cost of our **old** cluster centers with the new cluster assignments.

$$\text{Cost}(\vec{\nu}^{(1)}, \ldots, \vec{\nu}^{(k)} \mid D_1, \ldots, D_k) \leq \text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)} \mid D_1, \ldots, D_k)$$

Thus, we have shown that the cost after iteration will be equal to or less than the cost before iteration.

$$\text{Cost}(\vec{\nu}^{(1)}, \ldots, \vec{\nu}^{(k)}) \leq \text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)})$$

After you have proven (a) and (b), it only remains to be shown that the algorithm terminates. It can be shown that k-means takes only a finite number of iterations, and must converge eventually. Here's the idea: apart from before the initial iteration, a cluster center must be the mean of a subset of data. There are exponentially-many subsets of the data set, but an exponential number is finite. Therefore there are finitely-many cluster centers. Once the algorithm has reached a certain configuration of cluster centers, this configuration will never be seen again (as it can be shown that the objective function strictly decreases unless the centers stay the same). Therefore the algorithm terminates eventually (perhaps after an exponential number of iterations).

Plus Problem 2. (6 plus points)

Consider the following set of points in 1-dimensional space:

$$\{0, 1, 3, 6, 10, 15\}$$

Simulate the robust single linkage algorithm discussed in class with $k = 2$ and $\alpha = 0.5$ and show the clusters that the algorithm forms at every step from time $r = 1$ to $r = 9$. If you use code, please provide it in your submission.

Solution: A solution notebook can be found at https://go.ucsd.edu/3dbIIKQ.

At each timestep $r = 1 \ldots 9$ with $k = 2$ and $\alpha = 0.5$, the following Vertices are admitted, the following

Edges are found, and the following Clusters are formed.

```
Timestep r=1
        V = set()                       E = set()
        Clusters = []

Timestep r=2
        V = {1}                         E = set()
        Clusters = [{1}]

Timestep r=3
        V = {0, 1, 3}                   E = {(0, 1)}
        Clusters = [{0, 1}, {3}]

Timestep r=4
        V = {0, 1, 3, 6}                E = {(0, 1), (1, 3)}
        Clusters = [{0, 1, 3}, {6}]

Timestep r=5
        V = {0, 1, 3, 6, 10}            E = {(0, 1), (1, 3)}
        Clusters = [{0, 1, 3}, {6}, {10}]

Timestep r=6
        V = {0, 1, 3, 6, 10}            E = {(0, 1), (0, 3), (1, 3), (3, 6)}
        Clusters = [{0, 1, 3, 6}, {10}]

Timestep r=7
        V = {0, 1, 3, 6, 10}            E = {(0, 1), (0, 3), (1, 3), (3, 6)}
        Clusters = [{0, 1, 3, 6}, {10}]

Timestep r=8
        V = {0, 1, 3, 6, 10}            E = {(0, 1), (1, 3), (6, 10), (3, 6), (0, 3)}
        Clusters = [{0, 1, 3, 6, 10}]

Timestep r=9
        V = {0, 1, 3, 6, 10, 15}        E = {(0, 1), (1, 3), (6, 10), (3, 6), (0, 3)}
        Clusters = [{0, 1, 3, 6, 10}, {15}]
```