CS 181 Spring 2020 Section 7
Clustering
Solution

1 Motivation

We now move onto **unsupervised learning**, where the objective is to learn the structure of unlabeled data. In other words, we are looking for groups, or **clusters** among the data. Clustering algorithms are useful not only for finding groups in data, but also to extract features of the data that summarize the most important information about the data in a compressed way.

2 Setup

For most clustering algorithms, we need some kind of a metric to specify the notion of "distance" between the data points. If, for example, the points $\mathbf{x}$ and $\mathbf{x'}$ live in some Euclidean space $\mathbb{R}^m$, then the natural choice of such metric is the l_2 distance:

$$||\mathbf{x} - \mathbf{x'}|| = \sqrt{\sum_{i=1}^{n}(x_i - x_i')^2}$$

Now that the metric is well-defined, the next thing we need to do is to decide how many groups we want. Sometimes you know the ideal number of groups in advance (*e.g.* clustering the 26 letters in the alphabet). Other times, you need to decide if you'd like a more compressed representation with more information loss by having the number of groups small, or a less compressed representation with less information loss by having the number of groups large.

Suppose our data set is $\{\mathbf{x}_i\}_{i=1}^n$, then our objective is to find the ideal assignment of the data set to the clusters, by assigning to each of the n data points, a binary **responsibility vector** $\mathbf{r}_i$, which is all zeros except one component, which corresponds to the assigned cluster.

3 K-Means Algorithm

The idea is to represent each cluster by the point in data space that is the average of the data assigned to it. For some choice of K and random initialization of clusters, the

K-Means Algorithm (also called Lloyd's algorithm) is:

Repeat until convergence (none of the responsibility vectors change):

1. For each data point, update its responsibility vector by assigning it to the cluster with the closest mean.
2. For each cluster, $\{\boldsymbol{\mu}_k\}_{k=1}^K$, update its mean to be the mean of the data points currently assigned to that cluster.

3.1 Derivation

We begin by defining a loss function that the K-Means Algorithm minimizes via coordinate descent:

$$\mathcal{L}(\{\mathbf{r}_i\}_{i=1}^n, \{\boldsymbol{\mu}_k\}_{k=1}^K) = \sum_{i=1}^n \sum_{k=1}^K r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$$

First, we want to choose r_i that minimizes the loss, holding all else constant. This is when we assign data points to the clusters with means closest to them:

$$r_{ik} = \begin{cases} 1 & \text{if } k = argmin_{k'} ||\mathbf{x}_i - \boldsymbol{\mu}_{k'}|| \\ 0 & \text{otherwise} \end{cases}$$

This is the first step of each iteration of the K-means algorithm!

Second, we want to choose μ_k that minimizes the loss, holding all else constant. For a given k, the squared loss is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}_k) &= \sum_{i=1}^n r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2 \\ &= \sum_{i=1}^n r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)^T (\mathbf{x}_i - \boldsymbol{\mu}_k) \end{aligned}$$

Taking the derivative and setting it to zero,

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\mu}_k)}{\partial \boldsymbol{\mu}_k} &= -2 \sum_{i=1}^n r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k) = 0 \\ \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^n r_{ik} \mathbf{x}_i}{\sum_{i=1}^n r_{ik}} \end{aligned}$$

This is the second step of each iteration of the K-means algorithm!

3.2 Number of Clusters

There is not an especially well justified method to choose the number of clusters when using K-means. One approach is to plot K vs the objective criterion, and look for a "knee" or "kink" where progress slows down.

An advanced method is to use the "gap statistic". But this is out of scope for the course.

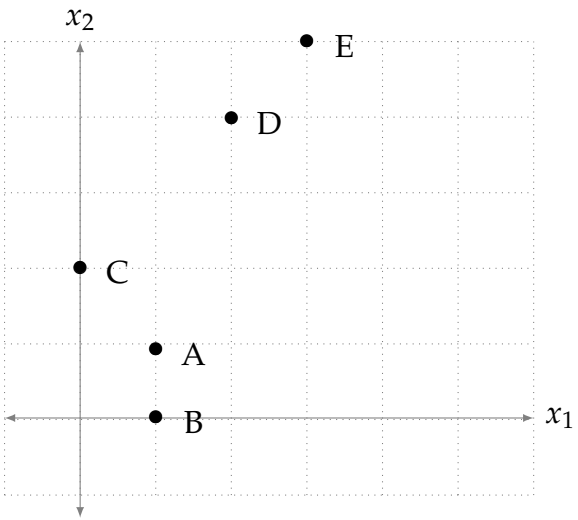
3.3 Notes

Lloyd's algorithm finds a locally optimal solution. Finding the globally optimal is NP-hard. A common strategy is to use random restarts. More recently, an algorithm called **K-Means++** has enjoyed popular usage as an alternative to random initialization. This is out of scope for the course, but the basic idea is to randomly select some of the data to be the first cluster centers. This is done by iteratively adding cluster centers, sampling them in proportion to the squared distance of each example from its nearest cluster center. Thus K-Means++ tends to favor points that are distant from the existing centers and produce a more diverse set of centers.

It is generally a good idea to **standardize** the data to account for unsatisfying result due to dimension mismatch. Lastly, when for the metric we are using for the given data set, a "mean" does not make sense, we might instead use a **K-Medoids Algorithm**. This algorithm requires the cluster centers to be a data point in the data set.

3.4 Exercise: K-Means (Di Cook)

Use K-means to cluster these examples in $\mathbb{R}^2$, looking for $K = 2$ clusters. Suppose that points A and C are randomly selected as the initial means.



Point	x_1	x_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

If we start with points A and C as our cluster, then let us set our cluster 1 mean $\boldsymbol{\mu}_1 = (1, 1)$ and cluster 2 mean to be $\boldsymbol{\mu}_2 = (0, 2)$. For each point, we will now calculate its distance from each cluster mean and assign it to the cluster where this distance is minimized.

For $\boldsymbol{\mu} = [1, 1]$, $\boldsymbol{\mu}_2 = [0, 2]$

Point	$distc_1$	$distc_2$	*cluster*
A	0	$\sqrt{2}$	1
B	1	$\sqrt{5}$	1
C	$\sqrt{2}$	0	2
D	$\sqrt{10}$	$\sqrt{8}$	2
E	$\sqrt{20}$	$\sqrt{18}$	2

Now we want to update our cluster means so that it takes the average of the coordinates

for all points assigned to this cluster.

$$\mu_1 = \frac{\sum_{i=1}^{n} r_{i1}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i1}} = [1, .5]$$
$$\mu_2 = \frac{\sum_{i=1}^{n} r_{i2}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i2}} = [\frac{5}{3}, \frac{11}{3}]$$

For $\mu_1 = [1, .5]$, $\mu_2 = [\frac{5}{3}, \frac{11}{3}]$

Point	$distc_1$	$distc_2$	$cluster$
A	.5	$\sqrt{\frac{2}{3}^2 + \frac{8}{3}^2} = 2.7$	1
B	.5	$\sqrt{\frac{2}{3}^2 + \frac{11}{3}^2} = 3.7$	1
C	$\sqrt{1 + 1.5^2} = 1.8$	$\sqrt{\frac{5}{3}^2 + \frac{5}{3}^2} = 2.3$	1
D	$\sqrt{1 + 3.5^2} = 3.64$	$\sqrt{\frac{1}{3}^2 + \frac{1}{3}^2} = .4$	2
E	$\sqrt{2^2 + 4.5^2} = 4.9$	$\sqrt{\frac{4}{3}^2 + \frac{4}{3}^2} = 1.9$	2

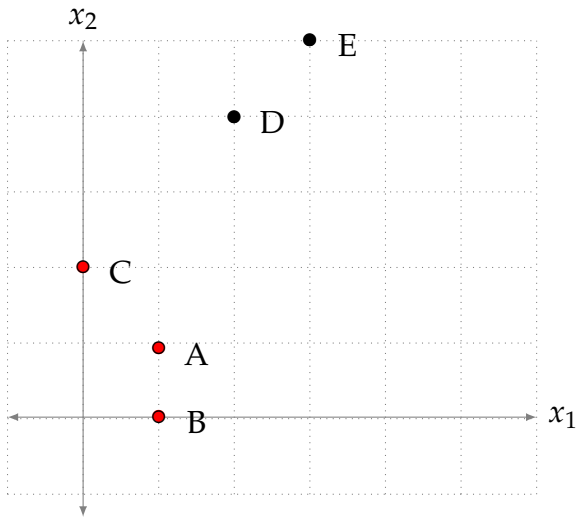
We will once again, recalculate our center means and distances from each point to the mean.

$$\mu_1 = \frac{\sum_{i=1}^{n} r_{i1}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i1}} = [\frac{2}{3}, 1]$$
$$\mu_2 = \frac{\sum_{i=1}^{n} r_{i2}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i2}} = [\frac{5}{2}, \frac{9}{2}]$$

For $\mu_1 = [\frac{2}{3}, 1]$, $\mu_2 = [\frac{5}{2}, \frac{9}{2}]$

Point	$distc_1$	$distc_2$	cluster
A	$\sqrt{\frac{1}{3}^2} = .33$	$\sqrt{\frac{3}{2}^2 + \frac{7}{2}^2} = 3.8$	1
B	$\sqrt{\frac{2}{3}^2 + 1} = 1.24$	$\sqrt{\frac{3}{2}^2 + \frac{9}{2}^2} = 4.7$	1
C	$\sqrt{\frac{2}{3}^2 + 1} = 1.24$	$\sqrt{\frac{5}{2}^2 + \frac{5}{2}^2} = 3.5$	1
D	$\sqrt{\frac{2}{3}^2 + 3^2} = 3.07$	$\sqrt{\frac{1}{2}^2 + \frac{1}{2}^2} = .7$	2
E	$\sqrt{\frac{7}{3}^2 + 4^2} = 4.63$	$\sqrt{\frac{1}{2}^2 + \frac{1}{2}^2} = .7$	2

We see that the r_{ik} assignments have not changed, and therefore the means of the clusters do not change and the algorithm has converged. The final cluster assignments are shown below.



3.5 Exercise: Convergence of K-Means (Bishop 9.1)

Consider Lloyd's algorithm for finding a K-Means clustering data, i.e., minimizing

$$\mathcal{L}(\{\mathbf{r}_i\}_{i=1}^n, \{\boldsymbol{\mu}_k\}_{k=1}^K) = \sum_{i=1}^{n}\sum_{k=1}^{K} r_{ik}||\mathbf{x}_i - \boldsymbol{\mu}_k||.$$

Show that as a consequence of there being a finite number of possible assignments for the set of responsibilities r_{ik}, and that for each such assignment there is a unique optimum for the means $\{\boldsymbol{\mu}_k\}_{k=1}^K$, the K-Means algorithm must converge after a finite number of iterations.

Since both the responsibility updates and the mean updates minimize the K-Means objective function, Lloyd's algorithm will never move to a new configuration of responsibility assignments unless the new configuration has a lower objective value. Since there are a finite number of possible assignments, each with a corresponding unique minimum with regard to the $\{\boldsymbol{\mu}_k\}_{k=1}^K$, the K-Means algorithm will converge after a finite number of steps, when no changes to the responsibilities will decrease the objective. When the responsibilities don't change in an iteration, the $\{\boldsymbol{\mu}_k\}_{k=1}^K$ also don't change.

4 Hierarchical Agglomerative Clustering

Hierarchical clustering constructs a tree over the data, where the leaves are individual data items, while the root is a single cluster that contains all of the data. When drawing

the dendrogram, for the clustering to be valid, the distances between the two groups being merged should be monotonically increasing. The algorithm is as follows:

1. Start with n clusters, one for each data point.
2. Measure the distance between clusters. This will require an inter-cluster distance measurement that we will define shortly.
3. Merge the two 'closest' clusters together, reducing the number of clusters by 1. Record the distance between these two merged clusters.
4. Repeat step 2 until we're left with only a single cluster.

The main decision in using HAC is what the distance criterion should be between groups.

4.1 The Min-Linkage Criterion

For two groups indexed by i and i', the idea is to merge groups based on the shortest distance over all possible pairs:

$$DIST_{\min}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \min_{i,i'} ||\mathbf{x}_i - \mathbf{x}_{i'}||.$$

4.2 The Max-Linkage Criterion

For two groups indexed by i and i', the idea is to merge groups based on the largest distance over all possible pairs:

$$DIST_{\max}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \max_{i,i'} ||\mathbf{x}_i - \mathbf{x}_{i'}||$$

4.3 The Average-Linkage Criterion

For two groups indexed by i and i', the idea is to average over all possible pairs between the groups:

$$DIST_{\text{avg}}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \frac{1}{nn'} \sum_{i=1}^{n} \sum_{i'=1}^{n'} ||\mathbf{x}_i - \mathbf{x}_{i'}||$$

4.4 The Centroid-Linkage Criterion

For two groups indexed by i and i', the idea is to look at the difference between the groups' centroids:

$$DIST_{\text{cent}}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = || \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \right) - \left(\frac{1}{n'} \sum_{i'=1}^{n'} \mathbf{x}_{i'} \right) ||$$

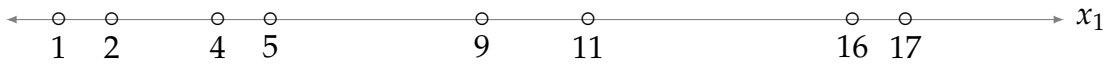
4.5 Exercise: K-means and HAC

What are three important differences between K-means and HAC?

K-Means has exactly K clusters whereas HAC can be used in a way where the number of clusters are determined after the fact, via inspecting the dendrogram. K-Means is randomized: the final clusters depend on the initial random centers whereas HAC is deterministic. HAC forms a hierarchy, which can provide additional understanding relative to a flat clustering.

4.6 Exercise: Min-Linkage and Max-Linkage Criterion

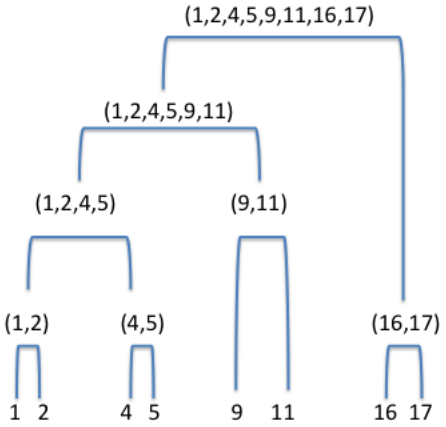
Assume the following examples lie in $\mathbb{R}$. Each example is initially in its own cluster.



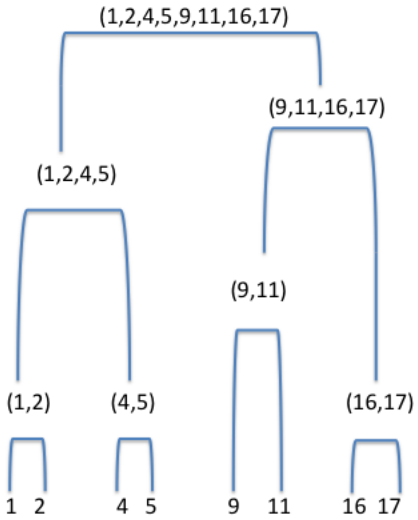
{1}{2}{4}{5}{9}{11}{16}{17}

1. Using the Min-Linkage Criterion for the HAC Algorithm, what is the clustering sequence? Draw the dendrogram.
2. Using the Max-Linkage Criterion for the HAC Algorithm, what is the clustering sequence? Draw the dendrogram.

a)
Step 1: {1} {2} {4} {5} {9} {11} {16} {17}
Step 2: {1, 2} {4, 5} {9} {11} {16, 17}
Step 3: {1, 2, 4, 5} {9, 11} {16, 17}
Step 4: {1, 2, 4, 5, 9, 11} {16, 17}
Step 5: {1, 2, 4, 5, 9, 11, 16, 17}



b)
Step 1: {1} {2} {4} {5} {9} {11} {16} {17}
Step 2: {1, 2} {4, 5} {9} {11} {16, 17}
Step 3: {1, 2} {4, 5} {9, 11} {16, 17}
Step 4: {1, 2, 4, 5} {9, 11} {16, 17}
Step 5: {1, 2, 4, 5} {9, 11, 16, 17}
Step 6: {1, 2, 4, 5, 9, 11, 16, 17}



4.7 Exercise: Clustering Complexity

What is the "big-O" complexity of HAC? What is the "big-O" complexity of K-means? Compare these.

Let's look at what each algorithm does. For K-means, we have two steps, the assignment step and the centroid update. For the assignment step, we have $O(nKm)$, where n is the size of data and m is your dimensions because you have to compare each example with each of the centroids. Then, we have $O(nm)$ for the centroid update, since we will be calculating different averages for each of the centroids. Overall, with T iterations we are left with $O(nKmT)$.

For HAC, we start with each point in its own cluster, and combine the two closest clusters until we are left with one big cluster. We get a runtime of $O(n^2m)$ because we need to calculate the pairwise distances for all n examples. In each of the T_h clustering steps, we then have $O(n^2)$ steps to use these pairwise distances and whatever linkage we're using to compute the distances between existing clusters. Generally, T_h is smaller than m and the overall complexity is $O(n^2m)$.

Comparing K-means and HAC, for large data sets we'd expect KT in K-means to be smaller than n, and thus K-means to be faster than HAC.

4.8 Exercise: Scaling to Large Dimensions

Explain the 'curse of dimensionality' and how it is related to HAC.

The curse of dimensionality refers to the problem where distances become meaningless in very large dimensional spaces. The problem is that the distance between two examples with some informative features but lots and lots of random features will be approximately the same (try this out in simulation if you don't see why!).

This means that non-parametric ('instance based') methods such as HAC that use pairwise distances between examples (vs between examples and prototypes in K-means) become less useful in higher dimensions.