
CS 181 LECTURE 2/29/24

Scribe Notes

Contents

1	Introduction to Unsupervised Learning	3
2	How Do We Evaluate Unsupervised Models?	3
3	How Do We Find the Representations?	3
3.1	Real World Examples	3
4	Setup: Clustering	4
5	K-Means Algorithm	4
5.1	What types of clusters does K-means find?	4
5.2	Understanding K-means	5
5.3	K-Means++	5
5.4	K-Medoids	5
6	HAC	6
6.1	Observations on HAC (HAC vs K-means)	7
7	Concept Check	7
8	Concept Check Answers	8
9	Pictures of the Board	8

1 Introduction to Unsupervised Learning

Today's content will be on Chapter 6, and we will begin looking at unsupervised learning.

In this scenario, we only have x 's in our training data set and there are no y 's. There are no targets to aim for. Instead, we take the data and produce a summary of the data or some different view that captures the relevant important things about the data.

Why would we want to do unsupervised learning?

1) The first reason is for summarization. We have a collection of data where it is too large to analyze item by item, then having a summary can help surface important insights about the data.

2) Another reason is for compression. We cannot store the whole data so we compress to allow for storage

3) Visualizing the data and interpreting the data is another reason.

4) Sometimes we identify latent structures in the data. Unsupervised learning helps us uncover the latent structure.

5) We might try to organize the data in a more accessible way to people.

6) Sometimes we have some labeled data and a lot of unlabeled data. It can be helpful to first look at the unlabeled data first as a preprocessing step to looking at supervised learning.

Consider the cake analogy where the cake is unsupervised learning, the icing is supervised learning and the cherry on top is the reinforcement learning. The sizes of each layer represents the amount of data needed and the importance of each type of learning.

2 How Do We Evaluate Unsupervised Models?

Reconstruction!

We take a data point x , we encode the data, then we decode the data, and we minimize the loss.

$$||x - \text{decode}(\text{encode}(x))||^2 \tag{1}$$

3 How Do We Find the Representations?

Today, we focus on clustering. We try to find group structure that is in the data.

3.1 Real World Examples

1) Content personalization: For example, companies build groups of broadly identifiable groups (ie. white, young, male) and then the company may advertise different things to different cohorts of people. Government organizations may also think about what policies are fit for different groups of people.

2) Community detection: Political groups will look at people's digital interactions and track different groups. Social scientists use these methods to identifies that are happening in society.

3) News Digest: New app on your phone has different categories.

4 Setup: Clustering

- data $x_1, \dots, x_n$
- distance $d(x, x')$
- Goal: group the x 's into $C_1, \dots, C_k$

Let Z_{nk} = indicator function indicating x_n is in cluster k . Let n_k be the size of the cluster k .

5 K-Means Algorithm

K-means will define prototypes μ_k , one per cluster.

Then, the goal is to find $\mathbf{Z}$, $\mu_1, \dots, \mu_k$ that minimizes

$$\sum_{n=1}^N \sum_{k=1}^K Z_{nk} \|x_n - \mu_k\|^2$$

This summation is telling us how close the datapoints are to the clusters. In other words, this is a notion of how good the clusters are.

In this case, we are encoding the data into clusters and then we are decoding the clusters into the μ_k 's.

Now, since we have the discrete variable Z_{nk} , this objective is a difficult one to solve. K-means is an approximate way to solve this hard objective.

Lloyd's Algorithm:

1. Randomly assign points to a cluster - randomly assign Z_{nk} .
2. Update step: then, we recompute the means of the clusters.

$$\mu_k = \text{mean}(x_n \text{ s.t. } Z_{nk} = 1)$$

We can also write this as

$$\mu_k = \frac{1}{n_k} \sum_{n=1}^N Z_{nk} x_n$$

3. Assignment step: We assign each x_n to the closest mean. The x_n is assigned to the C_k that minimizes $\|x_n - \mu_k\|^2$.

We repeat steps 2 and 3.

5.1 What types of clusters does K-means find?

Each point ends up in its closest cluster.

We see that on there is a line equidistant between two clusters. This linear boundary comes from our use of the Euclidean distance. So, we could use other $d(x, x')$ metrics but things could get complicated. In the assignment step, we could use a different metric. For the update step, we are using the empirical mean, which again is using L2 distance. It is

not so clear how we could do the update step without the L2 distance. Some other metrics of mean are very difficult to compute.

How would you deal with different shapes of clusters? Choosing different metrics of distance. The current L2 distance metric is good for capturing circles but not ovals.

5.2 Understanding K-means

1) The main observation is that in each step, we are making progress towards lowering loss.

For the assignment step, we are only assigning points to cluster that it is closer to. So, the loss is only going to go down.

For the update step, we can show that the mean is the point that is closest to all the points. So, when we update the means, we are also updating the loss so that the loss gets smaller.

2) We stop the algorithm after we reach convergence. But a down side is that convergence can be very slow.

3) The step 1 of initialization has a huge impact on the outcome of the algorithm.

5.3 K-Means++

K-Means++ is a smart way to initialize - to find the initial clusters.

Algorithm 1 K-means++ Initialization

- 1: Initialize centroids C as an empty set
 - 2: Select the first centroid c_1 uniformly at random from the data points
 - 3: Add c_1 to C
 - 4: **for** $k = 2$ to K **do**
 - 5: Compute the distance $D(x)$ from each data point x to the nearest centroid in C
 - 6: Choose the next centroid c_k from the data points with probability proportional to $D(x)^2$
 - 7: Add c_k to C
 - 8: **end for**
-

The algorithm starts by randomly selecting the first centroid from the data points. For each subsequent centroid, K-means++ computes the distance of each data point to the nearest centroid already chosen. The probability of selecting each data point as the next centroid is proportional to the square of its distance to the nearest centroid. This means that data points that are further away from existing centroids are more likely to be chosen as new centroids.

By iteratively selecting centroids based on their distance from existing centroids, K-means++ tends to select centroids that are well-spaced and representative of different parts of the data distribution. This initialization strategy often leads to faster convergence and better clustering results compared to random initialization in traditional K-means.

5.4 K-Medoids

Instead of computing means in the update step, we compute the medians of the cluster.

Advantages:

K-medoids is less sensitive to outliers compared to K-means because it uses medoids, which

are actual data points, as cluster centers. Medoids are less influenced by outliers than mean-based centroids.

Since medoids are actual data points, they are easier to interpret and understand compared to centroids, which are geometric centers that may not correspond to any actual data point.

Disadvantages:

The downside is that this is hard to solve. K-medoids has a higher computational complexity compared to K-means, especially for large datasets, because it requires pairwise distance computations between all data points.

6 HAC

HAC Algorithm:

1. Each x_n is its own cluster.
2. While there is still more than one cluster, we will merge the closest clusters.
3. Dendrogram representing the clustering hierarchy

We create this hierarchy graph.

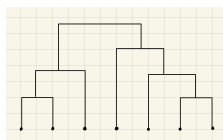


Figure 1: Example dendrogram (hierarchy graph).

What does it mean for two clusters to be close together?

We need to define a linkage function, which is a notion that defines distance between two clusters, $d(C, C')$.

- 1) Single-Linkage: We could define this as looking at distance between the two closest points in two different clusters.
- 2) Complete-Linkage: Another possible linkage function is getting the distance between the two furthest points in the two clusters.
- 3) Average Linkage: is the average distance between data points in the first cluster and data points in the second cluster.
- 4) Centroid Linkage: computes the distance between the averages.

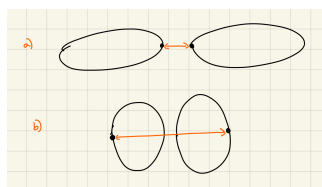


Figure 2: a) Depicts clustering using single-linkage b) Depicts clustering using complete-linkage.

Single-Linkage tends to privilege, long thin clusters. The complete-linkage typically creates squished clusters.

The other two linkages normally give us something in between.

6.1 Observations on HAC (HAC vs K-means)

- 1) No initialization issue.
- 2) Another advantage: this is a single pass algorithm.
- 3) The cost: this is a greedy algorithm! Once we merge something, we are stuck with it. There are no revisions made along the way.
- 4) Another downside is that the optimization objective is not clear.
- 5) HAC is more flexible than K-means.

7 Concept Check

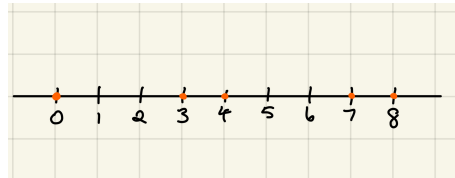


Figure 3: Concept check data.

- 1) HAC with average linkage, what does my cluster look like?
- 2) Initialize k-means with $\{0\}$, $\{3, 4, 7, 8\}$. Does the solution change?

8 Concept Check Answers

1)

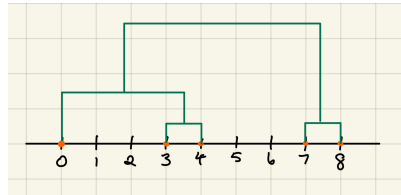
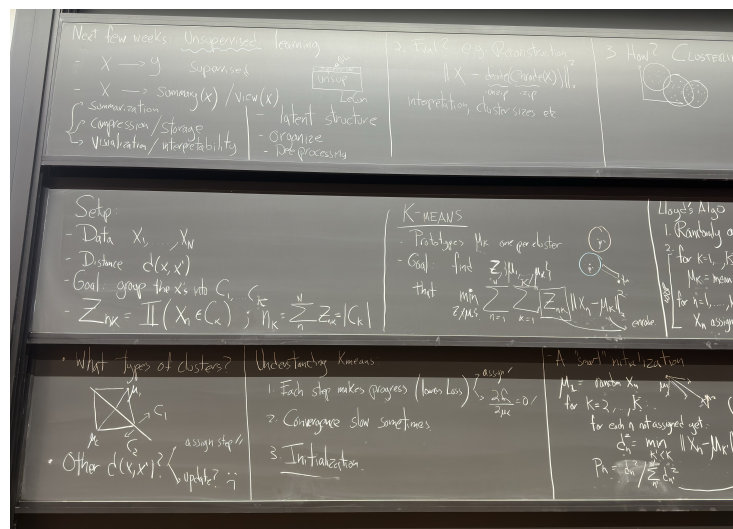
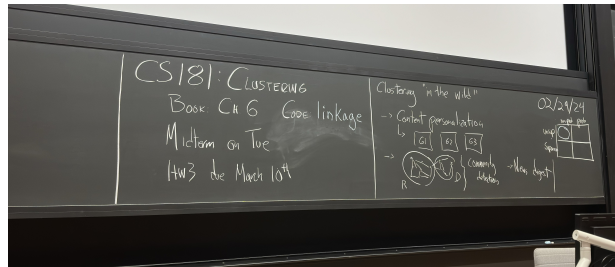


Figure 4: Average-linkage hierarchy.

2) The solution will not change! The only thing that can change is 3 moving to cluster with 0.

9 Pictures of the Board



Update step:

$$\mu_k = \text{Mean}(X_n \text{ st } Z_{nk}=1) = \frac{1}{n_k} \sum_{n=1}^N Z_{nk} X_n$$

- A variant: K-medoids ("median")

$$\mu_k = \arg\min_{X_n \text{ st } Z_{nk}=1} \sum_{n=1}^N Z_{nk} \|X_n - \mu_k\|$$

• μ_k medoid
• Z_{nk} assigns

Define "linkage" for

$$d(C_i, C_j) = \begin{cases} \min_{x \in C_i, y \in C_j} d(x, y) & \text{"Single" linkage} \\ \max_{x \in C_i, y \in C_j} d(x, y) & \text{"complete" linkage} \\ \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y) & \text{"average" linkage} \\ d(\text{Mean}(C_i), \text{Mean}(C_j)) & \end{cases}$$



HIERARCHICAL AGGLOMERATIVE CLUSTERING (HAC)

→ Choose clusters as you go

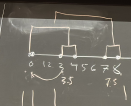
Algo:

1. Each X_n its own cluster
2. While #clusters > 1, merge closest clusters



Issues on HAC

- No initialization issue
- Single-pass / irreversible
- Objective not clear
- Flexible



1) HAC v centroid linkage, dendrogram? $\mu=0, \mu_2=22$

2) Initialize K-means $\{0\}, \{3, 4, 7, 8\}$? Will solution move? $d(0, 3)=2, d(0, 4)=1, d(0, 7)=8, d(0, 8)=1$

3) $\{0, 3, 4\}$ and $\{7, 8\}$? How does it compare to (2) in terms of d ?