PHIL 7010: Formal Methods for AI, Data, and Algorithms

Week 3
K - Means Clustering

Boris Babic,
HKU 100 Associate Professor of Data Science, Law and Philosophy

Today

Learning goals

- Concept of unsupervised learning
- K-means clustering algorithm
- Challenges and considerations in K-means
- K-means clustering in R

Unsupervised Learning

- What can we do **without labels**?
- How can we even define what learning means without labels?
- Unsupervised learning is the study of learning without labels.
- It is the task of grouping, explaining, and finding structure in data.

Motivating Example

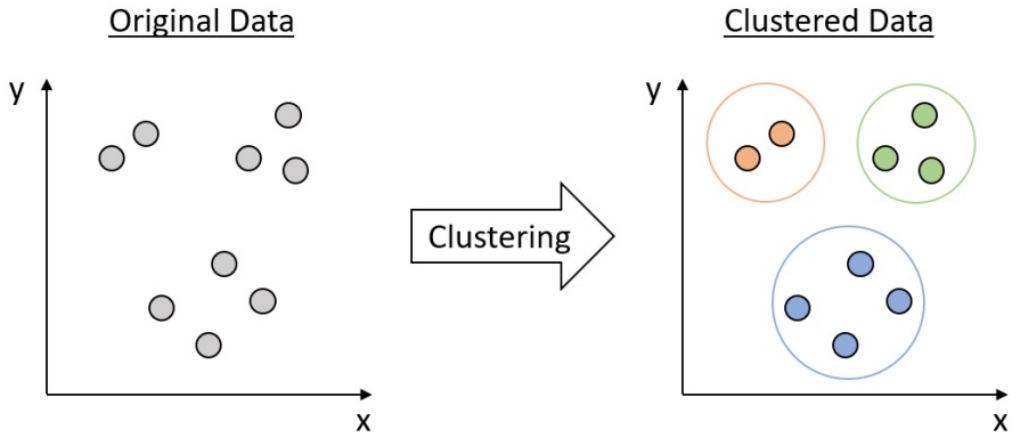
- Some examples of situations where you'd use unsupervised learning:
  - You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
  - You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from the video. You don't know the set of behaviors ahead of time.
  - You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).
- Common themes:
  - you have some data, and you want to infer some structure underlying the data.
  - you have some data, and you want to be able to make more data that looks similar.

Clustering

- Today we will look at the simplest type of unsupervised learning: clustering.
- Clustering is the task of organizing data into groups or clusters.
- We will study the simplest method for doing this: the K-means algorithm.

Clustering

- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:



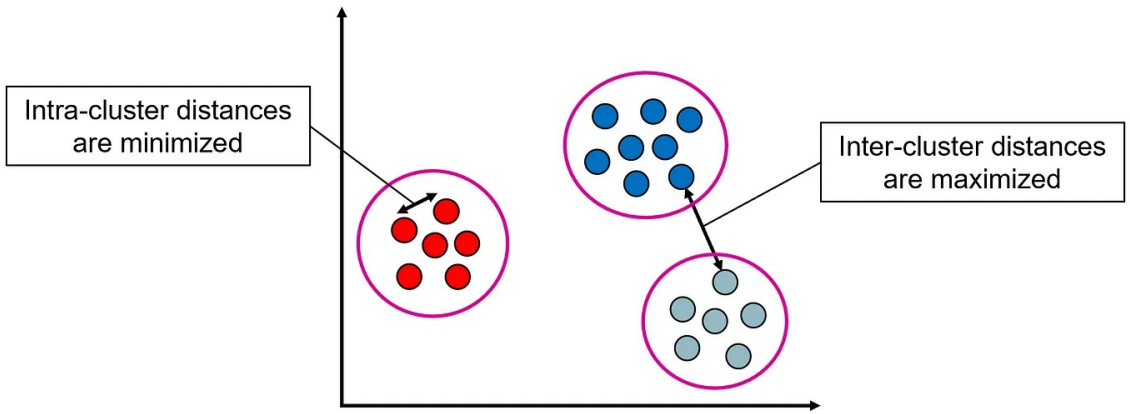
- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.
- Grouping data points into clusters, with no observed labels, is called clustering. It is an unsupervised learning technique.
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)
  - But topics are never observed (unsupervised).

Clustering

- In centroid-based clustering like the K-Means clustering, each cluster is represented by a central vector, called the cluster center or centroid, which is not necessarily a member of the dataset.
- The cluster centroid is typically defined as the mean of the points that belong to that cluster.
- Our goal in centroid-based clustering is to divide the data points into k clusters in such a way that the points are as close as possible to the centroids of the clusters they belong to.

Clustering problem

- Assume that we are given a set of n data points: $x_1, \ldots, x_n$, where each x_i is a m-dimensional vector
- Assume each data point belongs to one of K clusters
- Assume the data points from same cluster are similar, i.e. close in Euclidean distance.
- we would like to partition the data into k sets (clusters) $C_1, \ldots, C_k$, represented by centroids $c_1, \ldots, c_k$ such that
  - **Intra-cluster distances** are minimized: data points within the same cluster are as close as possible to one another.
  - **Inter-cluster distances** are maximized: data points in different clusters are as far as possible from one another.



Clustering problem

- Before jumping into how the K-means algorithms work, let's first get a sense of how we measure the distances
- Keep in mind that we want to minimize the sum of squared distances from each point to its nearest centroid
- The sum of squared distances is called WCSS (Within-Cluster Sum of Squares) or inertia, and is defined as follows:

$$WCSS = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mathbf{c}_i||^2$$

- WCSS quantifies the compactness (or cohesiveness) of the clusters. The smaller WCSS we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The K-Means Algorithm

The algorithm starts by choosing k initial centroids $c_1, \ldots, c_k$. This can be done randomly by selecting k data points from the dataset or by using various other initialization methods (described later).

Then, the algorithm alternates between the following two steps:

- **Assignment Step**: Assign each data point to the cluster with the nearest centroid, i.e., the centroid with the least squared Euclidean distance
- **Update Step**: For each cluster, compute a new centroid by calculating the mean of the data points assigned to that cluster.

These two steps are repeated until the cluster assignments no longer change or a specified number of iterations has been reached.

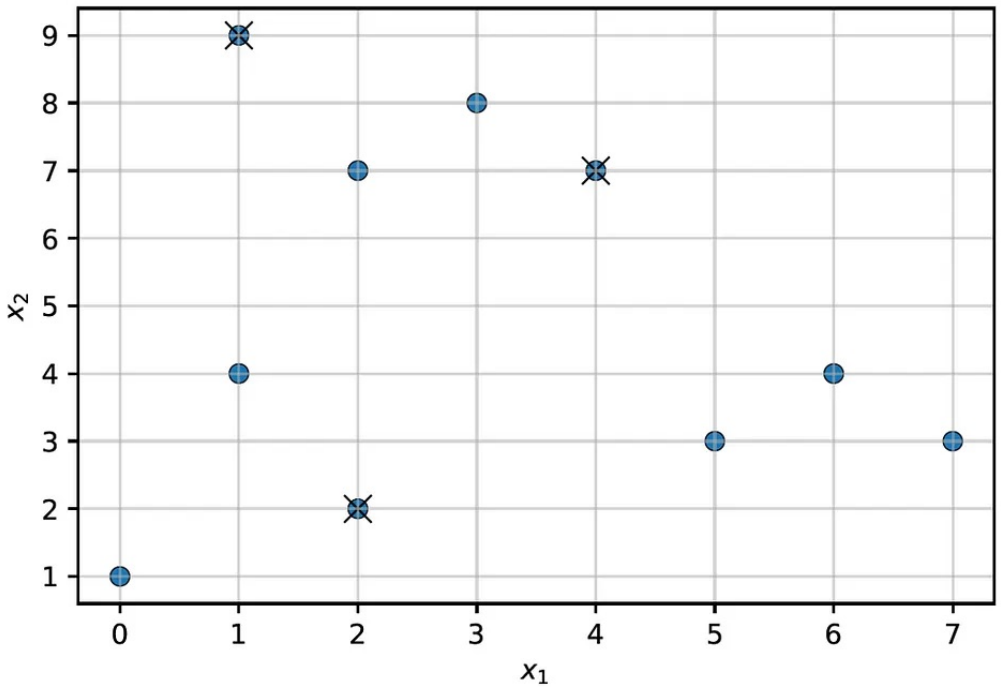
Example

Let's demonstrate the k-means algorithm on the following dataset, which consists of 10 two-dimensional points, and let's set k = 3.

i	x_1	x_2
1	0	1
2	1	4
3	1	9
4	2	2
5	2	7
6	3	8
7	4	7
8	5	3
9	6	4
10	7	3

Example

Assume that the initial centroids are: $c_1 = (1, 9)$, $c_2 = (2, 2)$, and $c_3 = (4, 7)$. The figure below shows the data points with the initial centroids marked by X:



First Iteration - Assignment

We first compute the squared Euclidean distance from each data point to the three centroids

i	x_1	x_2	distance to $c_1 = (1,9)$	distance to $c_2 = (2,2)$	distance to $c_3 = (4,7)$
1	0	1	$(0-1)^2 + (1-9)^2 = 65$	$(0-2)^2 + (1-2)^2 = 5$	$(0-4)^2 + (1-7)^2 = 52$
2	1	4	$(1-1)^2 + (4-9)^2 = 25$	$(1-2)^2 + (4-2)^2 = 5$	$(1-4)^2 + (4-7)^2 = 18$
3	1	9	$(1-1)^2 + (9-9)^2 = 0$	$(1-2)^2 + (9-2)^2 = 50$	$(1-4)^2 + (9-7)^2 = 13$
4	2	2	$(2-1)^2 + (2-9)^2 = 50$	$(2-2)^2 + (2-2)^2 = 0$	$(2-4)^2 + (2-7)^2 = 29$
5	2	7	$(2-1)^2 + (7-9)^2 = 5$	$(2-2)^2 + (7-2)^2 = 25$	$(2-4)^2 + (7-7)^2 = 4$
6	3	8	$(3-1)^2 + (8-9)^2 = 5$	$(3-2)^2 + (8-2)^2 = 37$	$(3-4)^2 + (8-7)^2 = 2$
7	4	7	$(4-1)^2 + (7-9)^2 = 13$	$(4-2)^2 + (7-2)^2 = 29$	$(4-4)^2 + (7-7)^2 = 0$
8	5	3	$(5-1)^2 + (3-9)^2 = 52$	$(5-2)^2 + (3-2)^2 = 10$	$(5-4)^2 + (3-7)^2 = 17$
9	6	4	$(6-1)^2 + (4-9)^2 = 50$	$(6-2)^2 + (4-2)^2 = 20$	$(6-4)^2 + (4-7)^2 = 13$
10	7	3	$(7-1)^2 + (3-9)^2 = 72$	$(7-2)^2 + (3-2)^2 = 26$	$(7-4)^2 + (3-7)^2 = 25$

Each data point is now assigned to the cluster with the nearest centroid (shown in yellow background). Therefore, the initial clusters are: $C_1 = \{p_3\}$, $C_2 = \{p_1, p_2, p_4, p_8\}$, $C_3 = \{p_5, p_6, p_7, p_9, p_{10}\}$.

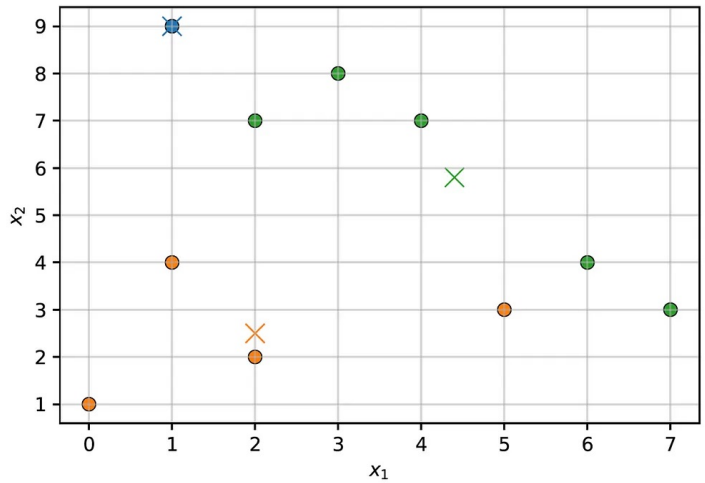
First Iteration - Update

Next, we update the centroids of the three clusters. The centroid of C_1 is simply p_3, since this cluster contains only one point. The centroids of C_2 and C_3 are computed as the mean of the points that belong to each one:

$$c_2 = \left(\frac{0+1+2+5}{4}, \frac{1+4+2+3}{4}\right) = (2, 2.5)$$

$$c_3 = \left(\frac{2+3+4+6+7}{5}, \frac{7+8+4+7+3}{5}\right) = (4.4, 5.8)$$

The new centroids are as follows:



Second Iteration - Assignment

Next, we compute the distances of the data points to the new centroids and reassign them to the clusters with the nearest centroid:

i	x_1	x_2	distance to c_1 = (1,9)	distance to c_2 = (2,2.5)	distance to c_3 = (4.4,5.8)
1	0	1	$(0-1)^2 + (1-9)^2 = 65$	$(0-2)^2 + (1-2.5)^2 = 6.25$	$(0-4)^2 + (1-7)^2 = 42.4$
2	1	4	$(1-1)^2 + (4-9)^2 = 25$	$(1-2)^2 + (4-2.5)^2 = 3.25$	$(1-4)^2 + (4-7)^2 = 14.8$
3	1	9	$(1-1)^2 + (9-9)^2 = 0$	$(1-2)^2 + (9-2.5)^2 = 43.25$	$(1-4)^2 + (9-7)^2 = 21.8$
4	2	2	$(2-1)^2 + (2-9)^2 = 50$	$(2-2)^2 + (2-2.5)^2 = 0.25$	$(2-4)^2 + (2-7)^2 = 20.2$
5	2	7	$(2-1)^2 + (7-9)^2 = 5$	$(2-2)^2 + (7-2.5)^2 = 20.25$	$(2-4)^2 + (7-7)^2 = 7.2$
6	3	8	$(3-1)^2 + (8-9)^2 = 5$	$(3-2)^2 + (8-2.5)^2 = 31.25$	$(3-4)^2 + (8-7)^2 = 6.8$
7	4	7	$(4-1)^2 + (7-9)^2 = 13$	$(4-2)^2 + (7-2.5)^2 = 24.25$	$(4-4)^2 + (7-7)^2 = 1.6$
8	5	3	$(5-1)^2 + (3-9)^2 = 52$	$(5-2)^2 + (3-2.5)^2 = 9.25$	$(5-4)^2 + (3-7)^2 = 8.2$
9	6	4	$(6-1)^2 + (4-9)^2 = 50$	$(6-2)^2 + (4-2.5)^2 = 18.25$	$(6-4)^2 + (4-7)^2 = 5.8$
10	7	3	$(7-1)^2 + (3-9)^2 = 72$	$(7-2)^2 + (3-2.5)^2 = 25.25$	$(7-4)^2 + (3-7)^2 = 14.6$

The new clusters are: $C_1 = \{p_3, p_5, p_6\}$, $C_2 = \{p_1, p_2, p_4\}$, $C_3 = \{p_7, p_8, p_9, p_{10}\}$.

Second Iteration - Update

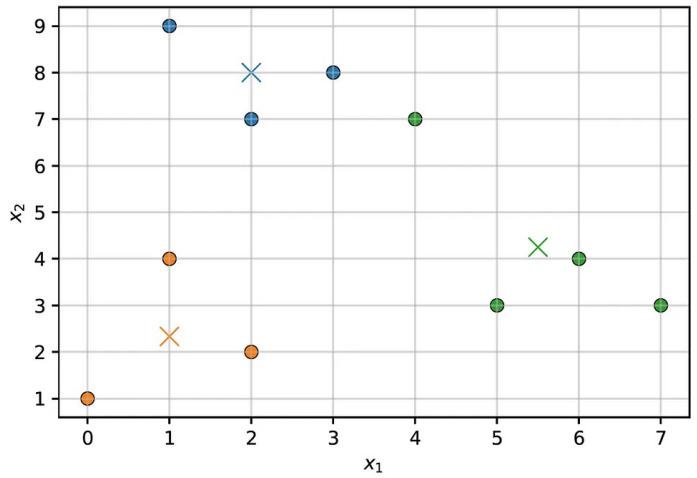
We now recalculate the centroids of these clusters:

$$c_1 = \left(\frac{1+2+3}{3}, \frac{9+7+8}{3}\right) = (2, 8)$$

$$c_2 = \left(\frac{0+1+2}{3}, \frac{1+4+2}{3}\right) = (1, 2.333)$$

$$c_3 = \left(\frac{4+5+6+7}{4}, \frac{7+3+4+3}{4}\right) = (5.5, 4.25)$$

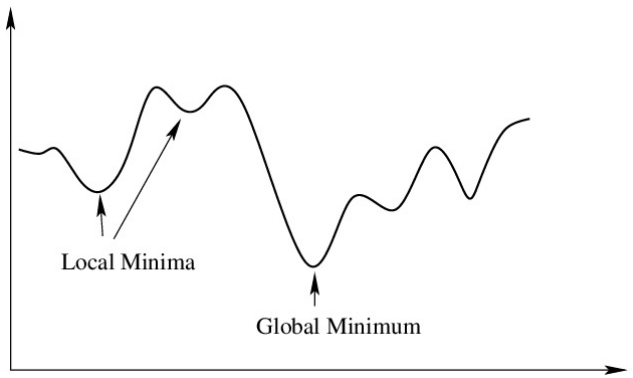
The new clusters after the second iteration:



Convergence

We will keep repeating the processes until the cluster assignments no longer change. The algorithm is guaranteed to converge to a solution. However, **the solution might be a local minimum of the WCSS, and not necessarily the global optimum**.

- **Global Minimum**: the absolute lowest possible value of the WCSS. This means that the data points are partitioned in such a way that they couldn't be arranged into any other clusters to achieve a lower WCSS.
- **Local Minimum**: a configuration where any small adjustment to the cluster assignments or centroids would increase the WCSS, but it's not necessarily the configuration with the lowest possible WCSS. In other words, while the solution is locally optimal (i.e., no nearby configuration is better), there may be other configurations (far from the current one) with a lower WCSS.



K-Means Local Minima

- The K-mans clustering algorithm is guaranteed to converge (i.e., reach a point where further iterations do not change the clusters) because each iteration reduces the WCSS or leaves it unchanged, and there's a finite limit to how much it can be reduced.
- However, the algorithm is very sensitive to our initial choice of the centroids, depending on which values we choose for our initial centroids we may obtain differing results.
- If the centroids are initialized poorly, the algorithm may get "stuck" in a local minimum where it can no longer reduce the WCSS by changing cluster assignments or centroids, even though a better (lower WCSS) global solution exists.
- Solution? Run the algorithm using different initializations of centroids several times and to pick the results of the run that yielded the lowest value of WCSS

Implement K-Means in R

To perform k-means clustering in R we can use the built-in 'kmeans()' function:

```
#perform k-means clustering with k = 4 clusters
km <- kmeans(df, centers = 4, nstart = 25)
```

where:

- **df**: Name of the dataset.
- **centers**: The number of clusters.
- **nstart**: Run k-means with n different random starting assignments (e.g. 25 in this case) and, then, R will automatically choose the best results total within-cluster sum of squares.

K-means: Pros and Cons

Pros:

- Computationally efficient
- Works well even with large datasets
- Convergence guaranteed
- Easy to interpret

Cons:

- Sensitivity to Outliers
- Dependence on Initialization

Today

Learning goals

- Concept of unsupervised learning
- K-means clustering algorithm
- Challenges and considerations in K-means
- K-means clustering in R