# CSCI 1315 - Tutorial 2
# Data Clustering

**Name:** __________________________ **B00:** ________________

**Goal:** In this tutorial, we will use the $K$-Means algorithm for splitting a set of points into disjoint sets or clusters. You will see this again in your degree, for example in CSCI 3152 - Foundations of Machine Learning[1].

## Setting

In many areas of computer science you will collect large sets of data points, and then try and group these data points into sets or **clusters**. These clusters have the property that the elements in the same cluster are more similar than elements in different clusters.

For example, say you want to train a computer to recognize different animals in images. Your goal is for your program to be able to distinguish the difference between a cat and a dog. You can train your program by giving it many test points (photos of cats and dogs) and tell it which ones are which. Through this process, you will end up creating a large set of data points, for which you will separate them into the set “cats” and the set “dogs”.

In this tutorial, we will consider one of the simpler clustering algorithms called the **$K$-Means algorithm**.

## $K$-Means

The $K$-Means clustering algorithm is considered a **hard clustering** in the sense that “hard” means that a data point can only be in a single cluster (whereas **soft clustering** means that a data point can belong to more than one cluster). The algorithm is set up as follows:

Given a set of $N$ ordered pairs (or $n$-tuples, but we’ll focus on ordered pairs, viewed as points on a plane), we start by choosing $K$ points randomly from our set and designating them as **centroids** (a centroid is the “center” of each cluster. A centroid doesn’t have to be a point in our set, but in our initial set-up we will choose our centroids to be elements of our original set). We then iterate through the list of $N$ points in our set, considering the distance between each point and each centroid, and assigning each point in our original set to the nearest centroid[2] (called **reassignment**). This forms the $K$ clusters of points for our first iteration – every point in our original set is in exactly one of the $K$ clusters. After we have designated a centroid to each of the data points, we recompute the centroids based on the “average” of the data points in the cluster (called **recomputation**), and then start the reassignment process again. This continues until the data points and their centroids converge, which means that the data points stay in the same cluster after recomputing the centroids.

[1] Some of the references to this material comes from Dr. Milios’ course notes in CSCI 3152, as well as Wikipedia.

[2] In the case of any ties, we can randomly assign the point to one of the centroids that minimizes the distance.

Formally, this process is explained in the following algorithm:

**K-Means**

Input: a set of $N$ points $\{x_1, \dots, x_n\}$, an integer $K$ for the number of clusters

Output: $K$ clusters (or sets) with each points assigned to the cluster whose centroid it is closest to.

```
$(s_1, s_2, \dots, s_K)$ ←SelectRandomSeeds($\{x_1, \dots, x_n\}, K$) \\ randomly choose $K$ seeds, denoted by $s_i$, which the clusters will grow around
for $k$ from 1 to $K$ \\ initialize each centroid $\mu_k$ as seed $s_k$
do $\mu_k \leftarrow s_k$ while stopping criterion has not been met
do    for $k$ from 1 to $K$ \\ Initialize the clusters as empty sets
      do $\omega_k \leftarrow \{\}$
      for $n \leftarrow 1$ to $N$ \\ Find $j$ that minimizes the distance between centroids $\mu_{j'}$ and point $x_n$
      do $j \leftarrow \arg\min_{j'} |\mu_{j'} - x_n|$
          $\omega_j \leftarrow \omega_j \cup \{x_n\}$ \\ Reassignment of $n$-tuples into cluster $\omega_j$ where $j$ is that minimum $j'$
      for $k$ from 1 to $K$
      do $\mu_k \leftarrow \frac{1}{\omega_k} \sum_{x \in \omega_k} x$ \\ recomputation of centroids $\mu_k$ by taking the average of the points $x$ in cluster $\omega_k$
return $\{\mu_1, \dots, \mu_K\}$ \\ This returns the clusters, but it's equivalent to just returning the centroids. Why?
```

We will be using this algorithm to find 3 clusters (so $K = 3$) of the following set of data points:

$$P = \{(-2,1),\ (-1,1),\ (-1,2),\ (0,0),\ (1,-1),\ (1,1),\ (1,2),\ (2,-2),\ (2,-1),\ (2,0)\}$$

To do this, we first start by choosing our seeds.

**Question 1**: Select the first 3 points from $P$ to act as **seeds** for which the centroids will grow around.

**Answer:**

$$s_1 =$$

$$s_2 =$$

$$s_3 =$$

Once we have our seeds (which we use as our initial centroids), we need to compute the distance between each element in $P$ to each centroid. We do this by using the **squared distance formula**. That is, for two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$, the squared distance between them (squared so that we get rid of the square root in the usual distance formula) is

$$d^2(A, B) = (x_1 - x_2)^2 + (y_1 - y_2)^2.$$

After doing this, we determine which cluster to add the point to by assigning it to the centroid that minimizes distance.

**Question 2:** Fill in the following table, where $\mu_i$ are the centroids (original seeds) that you chose in Question 1 (so $\mu_i = s_i$), $d^2(A, \mu_i)$ is the squared distance between the given point $A$ and the centroid $\mu_i$, and the Cluster is the number associated to the centroid (so 1 if the smallest number is in the $\mu_1$ column, 2 if the smallest number is in the $\mu_2$ column, or 3 if the smallest number is in the $\mu_3$ column). If you have any ties, randomly assign that point to a cluster with a centroid of minimal distance.

**Answer:**

| | $\mu_1 =$ | $\mu_2 =$ | $\mu_3 =$ | |
|---|---|---|---|---|
| Point $A$ | $d^2(A, \mu_1)$ | $d^2(A, \mu_2)$ | $d^2(A, \mu_3)$ | Cluster |
| $(-2, 1)$ | | | | |
| $(-1, 1)$ | | | | |
| $(-1, 2)$ | | | | |
| $(0, 0)$ | | | | |
| $(1, -1)$ | | | | |
| $(1, 1)$ | | | | |
| $(1, 2)$ | | | | |
| $(2, -2)$ | | | | |
| $(2, -1)$ | | | | |
| $(2, 0)$ | | | | |

We now have our first iteration of clusters!

**Question 3:** Write out the three clusters as sets. The set associated with the centroid $\mu_i$ will be named $\omega_i$.

**Answer:**

$$\omega_1 =$$

$$\omega_2 =$$

$$\omega_3 =$$

Let's now recompute the centroids for a second pass through the algorithm.

**Question 4:** Recompute the centroids by calculating the average point in each cluster. That is, evaluate the following sums where $P$ is a point in the cluster $\omega_i$, $P_x$ is the $x$-coordinate of the point $P$, $P_y$ is the $y$-coordinate of the point $P$, and $(\mu_{i_x}, \mu_{i_y})$ is the new centroid point $\mu_i$ in each cluster.

**Answer:**

$$\mu_{1_x} = \frac{1}{|\omega_1|} \sum_{P \in \omega_1} P_x =$$

$$\mu_{1_y} = \frac{1}{|\omega_1|} \sum_{P \in \omega_1} P_y =$$

$$\mu_{2_x} = \frac{1}{|\omega_2|} \sum_{P \in \omega_2} P_x =$$

$$\mu_{2_y} = \frac{1}{|\omega_2|} \sum_{P \in \omega_2} P_y =$$

$$\mu_{3_x} = \frac{1}{|\omega_3|} \sum_{P \in \omega_3} P_x =$$

$$\mu_{3_y} = \frac{1}{|\omega_3|} \sum_{P \in \omega_3} P_y =$$

Our new centroids are:

$\mu_1 = (\quad , \quad)$ $\qquad$ $\mu_2 = (\quad , \quad)$ $\qquad$ $\mu_3 = (\quad , \quad)$

Finally, we can repeat the algorithm and see how it affects the clusters.

**Question 5**: Complete the following table, where $\mu_i$ are the new centroids you calculated in Question 4.

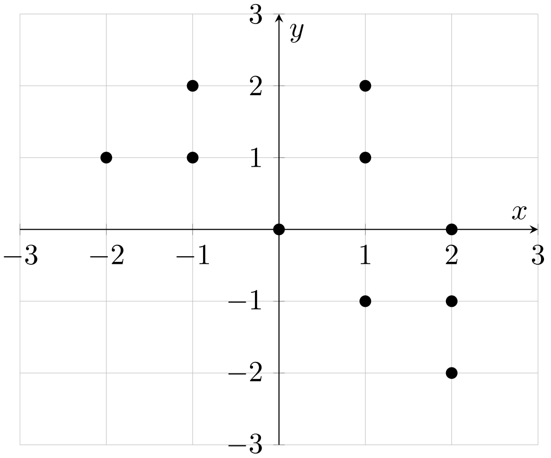
**Answer:**

| | $\mu_1 =$ | $\mu_2 =$ | $\mu_3 =$ | |
|---|---|---|---|---|
| Point $A$ | $d^2(A, \mu_1)$ | $d^2(A, \mu_2)$ | $d^2(A, \mu_3)$ | Cluster |
| $(-2, 1)$ | | | | |
| $(-1, 1)$ | | | | |
| $(-1, 2)$ | | | | |
| $(0, 0)$ | | | | |
| $(1, -1)$ | | | | |
| $(1, 1)$ | | | | |
| $(1, 2)$ | | | | |
| $(2, -2)$ | | | | |
| $(2, -1)$ | | | | |
| $(2, 0)$ | | | | |

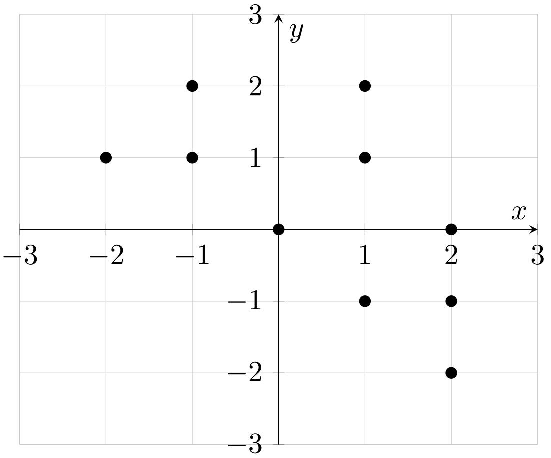
**Question 6**: Indicate your clusters for the first iteration (Question 2) and second iteration (Question 5) on the plots below. Did the clusters change? What observations do you have? What do you think might happen if you completed further iterations?

**Answer:**

First iteration



Second iteration



**Reflection**: That completes our tutorial on **Data Clustering**! What did you learn today? What are the important messages you have taken with you? What did you find easy/hard? How does this connect our content in the course to computer science?