K-means como algoritmo de clustering

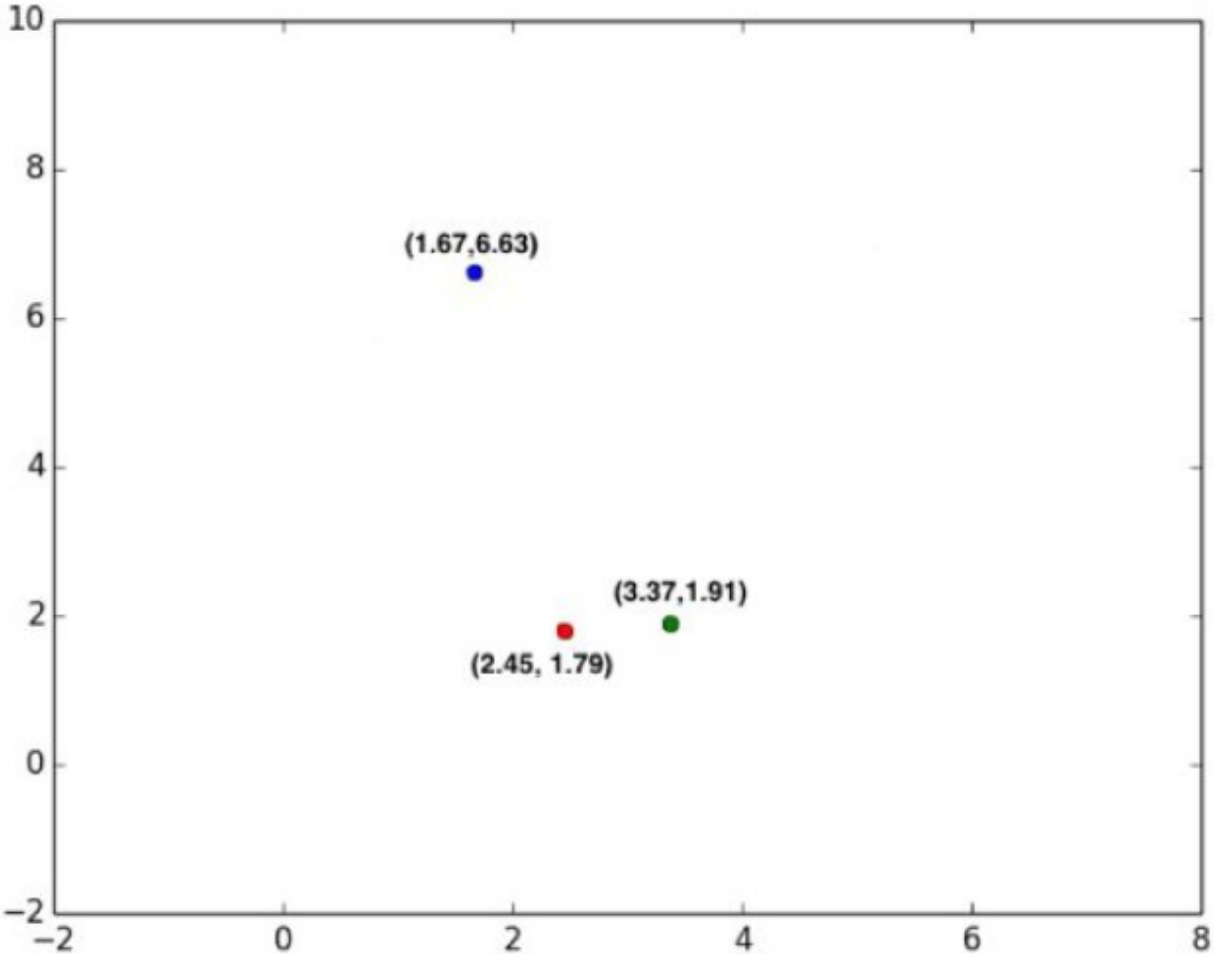
El K-means es un método de Clustering que separa 'K' grupos de objetos (Clusters) de similar varianza, minimizando un concepto conocido como inercia, que es la suma de las distancias al cuadrado de cada objeto del Cluster a un punto 'μ' conocido como Centroide (punto medio de todos los objetos del Cluster).

El funcionamiento de este algoritmo comienza eligiendo un centroide para cada uno de los 'K' Clusters. El método de elección de estos centroides puede ser cualquiera; siendo los dos más comunes, el inicializarlo de forma aleatoria o el de elegir 'K' objetos del data set, bien sea de forma aleatoria o haciendo un pre-procesamiento de los datos. Lo recomendable sería la segunda opción e inicializar estos Clusters con objetos del data set.

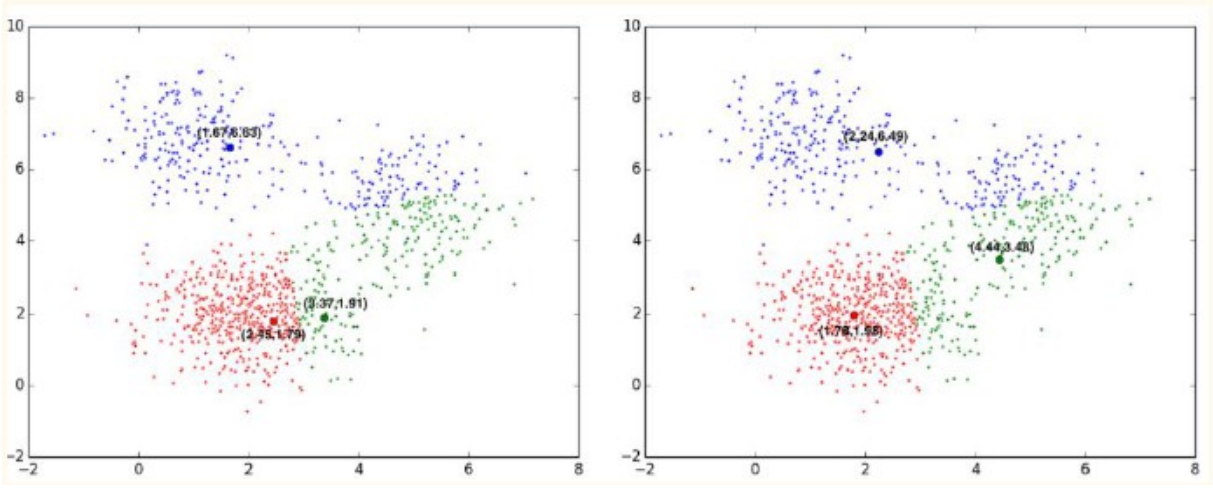
Una vez inicializados los centroides, el algoritmo continúa alternando los dos siguientes pasos (asignación y actualización) de forma iterativa hasta que los centroides converjan.

Veamos a continuación un ejemplo de ejecución del K-means para 3 Clusters, dado un data set en el que cada objeto está representado por un punto en un espacio de dos dimensiones:

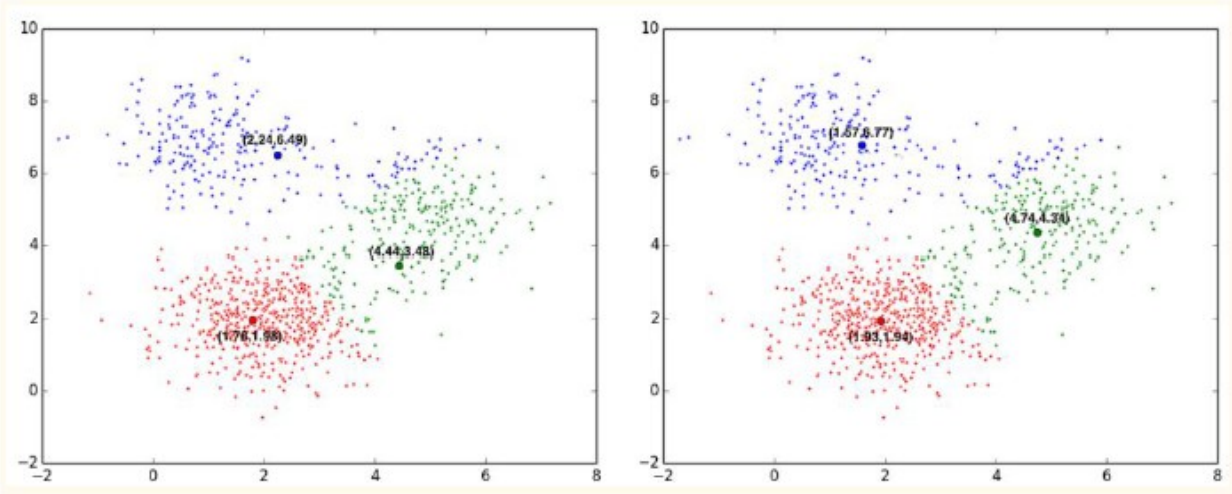
1. **Inicialización de los Clusters:** Para ello cogemos al azar 3 puntos del data set y los asignamos a un Cluster. Como cada Cluster solo tiene un punto, será ese punto el centroide del Cluster:



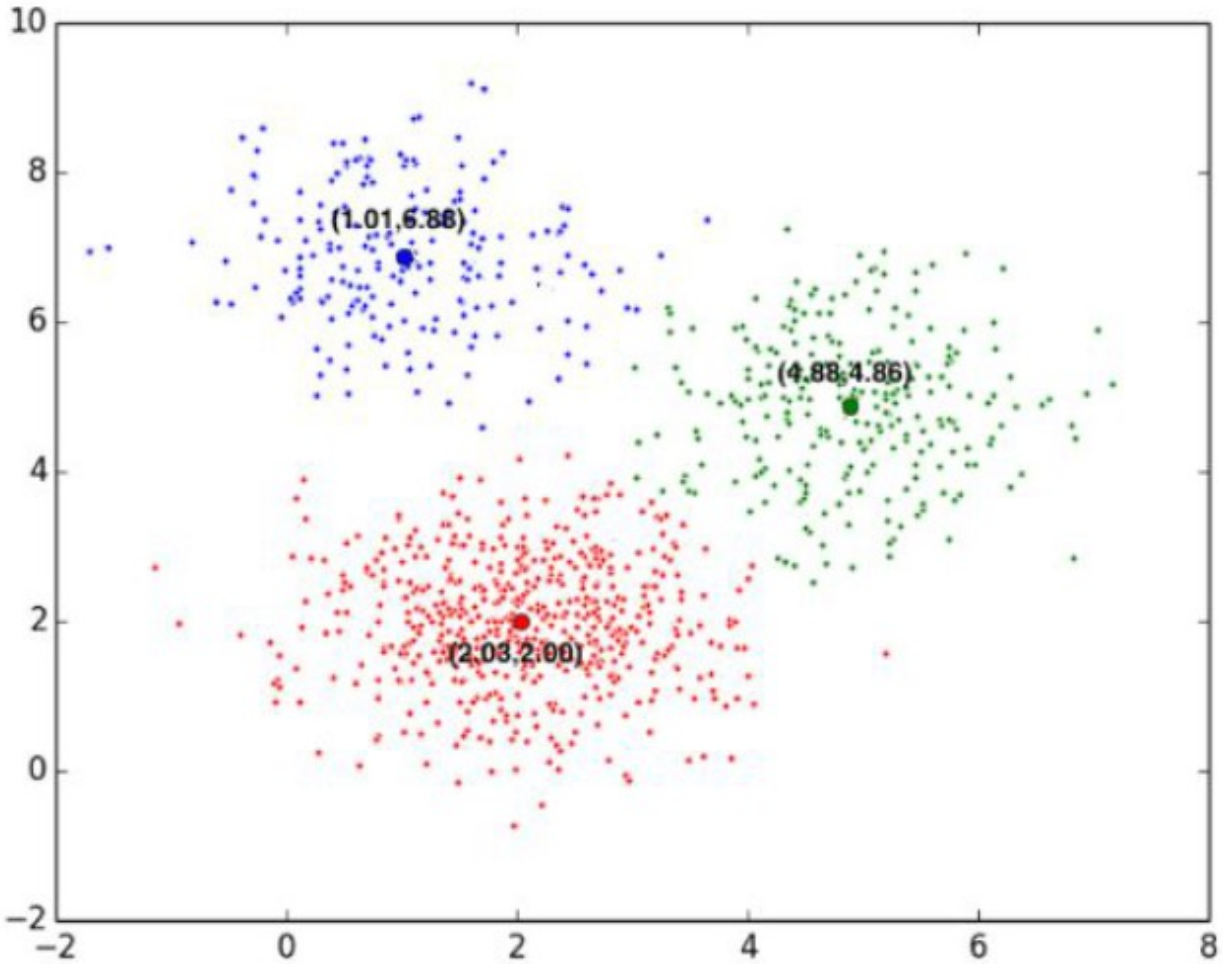
2. **Primera asignación y actualización:** Tras haber elegido al azar 3 Clusters, se asigna cada punto al Cluster más cercano. Una vez que están asignados todos los puntos, se calcula un nuevo centroide siendo este el valor medio de todos los puntos.



3. **Segunda asignación y actualización:** Con los nuevos centroides, volvemos a calcular para cada punto cuál es el centroide más cercano y asignamos ese punto al centroide. Una vez asignados los puntos a los Clusters, volvemos a calcular los centroides.



4. **Convergencia y resultado final:** Estos pasos de asignación y actualización se repiten hasta que los centroides de los Clusters converjan; es decir, hasta que el valor de los centroides de la última iteración de actualización coincida con el valor de los centroides de la iteración anterior de actualización:



Definido el funcionamiento del algoritmo paso por paso y con un ejemplo, pasamos a mostrarlo en pseudocódigo.

El algoritmo va a ser un proceso iterativo donde a partir de una primera inicialización ,cada muestra se va a asignar al representante más cercano y esta asignación se va a utilizar para recalcular los representantes de cada cluster

```
K = num_clusters
1.- Inicializar K Clusters con sus centroides μ1,…,μk de forma aleatoria
2.- while not converge:
   for i in range(dataset):
        c_k := argmin‖x_i − μ_k‖²
   for j in range(K):
        μ_j := (1/N) Σ_{i=1}^{N} x_i
```

En general los pasos del algoritmo se pueden resumir en 4 pasos:

- Paso 1 : Establece un grupo K como centroides en el espacio representado
- Paso 2 : Calcula la distancia de cada objeto con cada uno de los centroides de K y asígnalo al centroide del cual su distancia sea menor.
- Paso 3: Cuando todos los objetos sean asignados, entonces recalcula la posición de los centroides.
- Paso 4: Repite los pasos 2 y 3 hasta que los centroides no se muevan mas.

Para el paso 1 usualmente se suele tomar alguna medida estadística como la media o la mediana, algún dato que represente perfectamente a cada grupo o bien se utilizan datos aleatorios. Dependiendo del número de grupos que demos serán los cluster que obtendremos.

Para el paso 2 se puede utilizar cualquier fórmula de distancia. Cada objeto será comparado con todos los centroides y se asignará al centroide que haya tenido una distancia menor.

Para el paso 3, usualmente se utiliza alguna heurística personalizada para determinar cómo modificar los centroides, aunque en la práctica se suele calcular la media o mediana de sus objetos y movilizar los centroides a esa posición

Para el paso 4, de la misma manera se suele utilizar una heurística que nos permita determinar alguna condición de paro, estas van desde correr el proceso un número determinado de ocasiones, sumar las distancias más cortas del paso 2 y si se obtiene una suma inferior continuar con el proceso (siendo esta la más usada y en algunos casos considerada obligatoria) o bien determinar un mecanismo personalizado para el problema en cuestión.

K-means es un algoritmo que encontrará k clusters para un conjunto de datos dado.

El número de clústeres k está definido por el usuario. Cada grupo se describe por un solo punto conocido como el centroide. **Centroide** significa que está en el **centro de todos los puntos del cluster.**

El algoritmo k-means funciona de este modo. En primer lugar, los k centroides se asignan aleatoriamente a un punto. A continuación, cada punto del conjunto de datos se asigna a un clúster. La asignación se realiza encontrando el centroide más cercano y asignando el punto a ese grupo. Después de este paso, los centroides se actualizan tomando el valor medio de todos los puntos de ese grupo.

Así es como se vería el pseudo-código:

```
Crear k puntos para iniciar centroides (a menudo al azar)
Mientras que cualquier punto ha cambiado la asignación de clúster
        Para cada punto de nuestro conjunto de datos:
                Para cada centroide
                        Calcular la distancia entre el centroide y el punto
                        Asignar el punto al cluster con la distancia más baja
                Para cada grupo calcular la media de los puntos en ese grupo
                        Asignar el centroide a la media
```