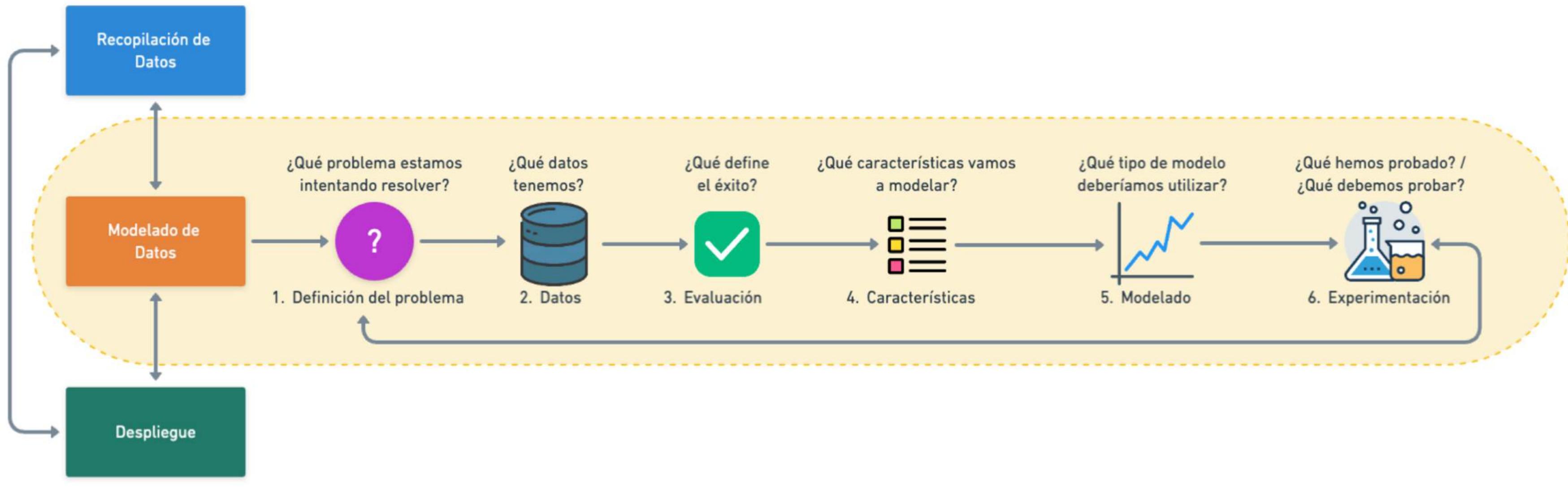


MACHINE LEARNING

MTRO. ALFONSO GREGORIO RIVERO DUARTE

FRAMEWORK



CLUSTERING

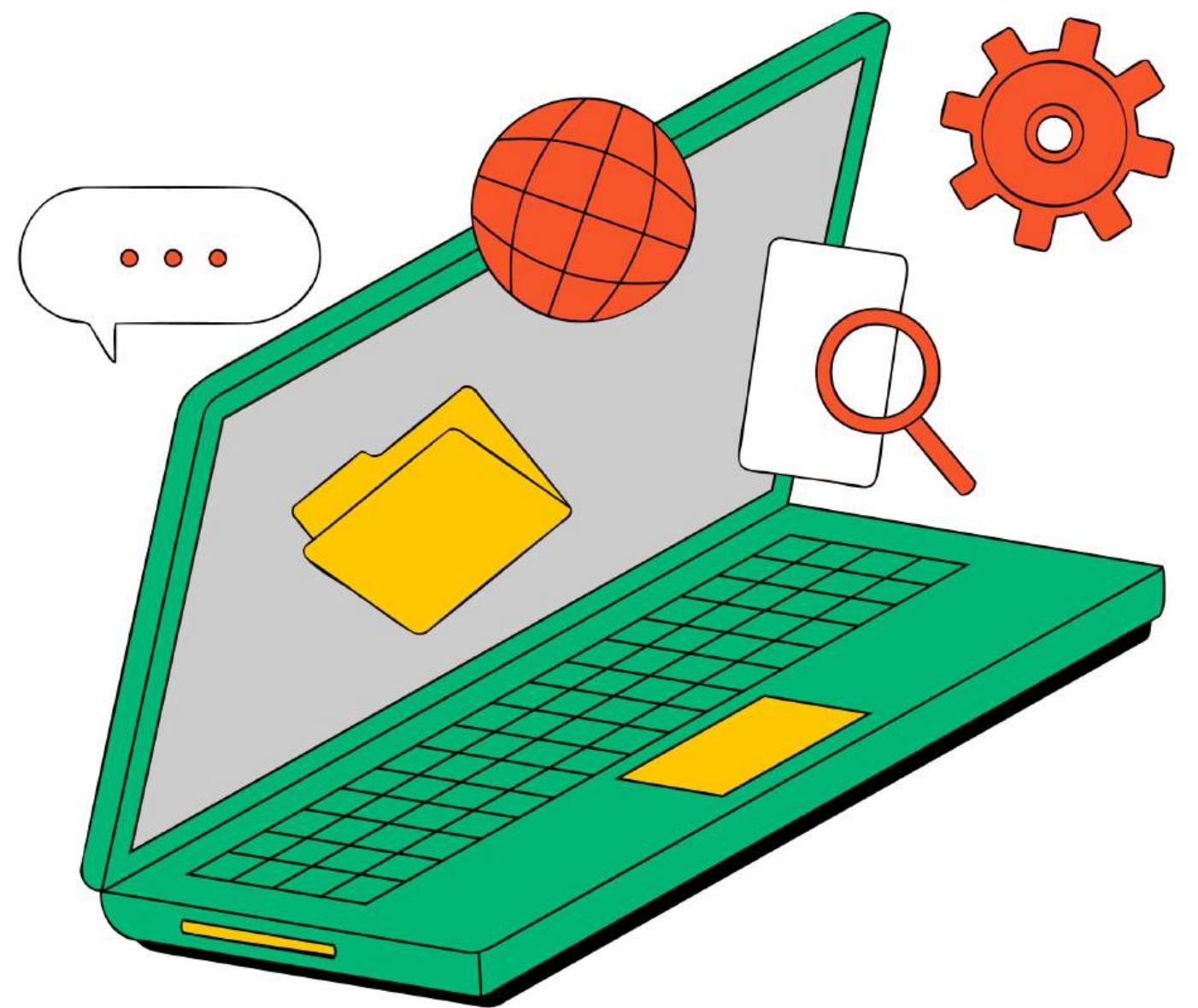
Clustering es un proceso similar al de clasificación, pero con un fundamento diferente. En el Clustering no sabes qué categorías estás buscando, si no que intentas crear una segmentación de tus propios datos en grupos más o menos homogéneos. Cuando utilizamos algoritmos de clustering en el data set, surgen de entre los datos cosas inesperadas como estructuras, clusters o agrupaciones que un humano no se habría imaginado pero la máquina crea por nosotros.



TÉCNICAS O ALGORITMOS

MODELOS DE CLUSTERING

1. Clustering con K-Means
2. Clustering Jerárquico



K-MEANS

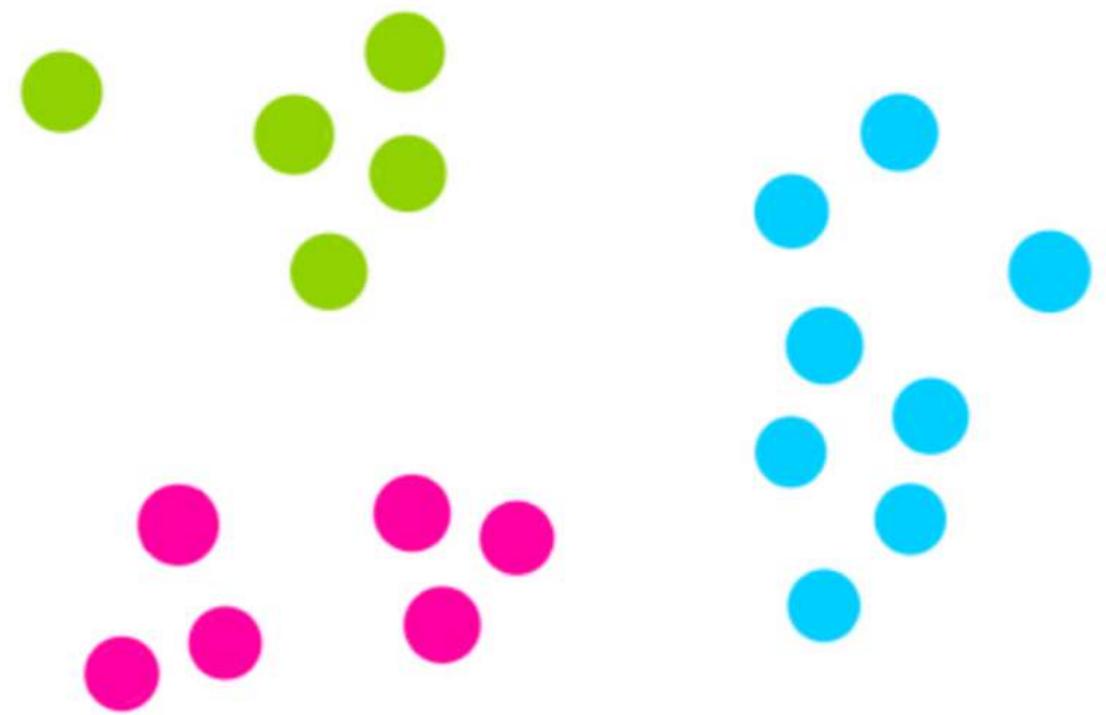
MODELOS DE CLUSTERING

- 1.Idea de K-Means
- 2.Descripción del problema
- 3.Ejemplo práctico
- 4.Trampa de inicialización aleatoria
- 5.Selección del Número de Clusters
- 6.Algoritmo de K-Means
- 7.Evaluación



IDEA DE K-MEANS

K-MEANS



El algoritmo k-means es un método de agrupamiento que divide un conjunto de datos en k grupos o clusters. Los datos se agrupan de tal manera que los puntos en el mismo clúster sean más similares entre sí que los puntos en otros clusters.

Del universo de [algoritmos de aprendizaje no supervisado](#), K-means es probablemente el más reconocido. La razón por la que existe este método es porque hoy en día la cantidad total de datos creados, capturados, copiados y consumidos globalmente es de aproximadamente 100 Zettabytes y seguirá creciendo.

K-MEANS

La idea es que vamos a tener un conjunto de puntos y queremos dividirlos, queremos segmentarlos, queremos clasificarlos, con la diferencia que en la sección anterior sabíamos que categorías queríamos asignar a cada conjunto de puntos: comprador o no comprador.

Aquí la idea es que vamos a tener un conjunto de puntos y nos interesa agrupar los que se parecen entre sí en lo que vamos a llamar una agrupación o un cluster y separarlo de los puntos que pertenecen a otro de esos clusters.

Vamos a verlo primero con un ejemplo de cuál es la idea o para qué sirve K – Means.

K-MEANS

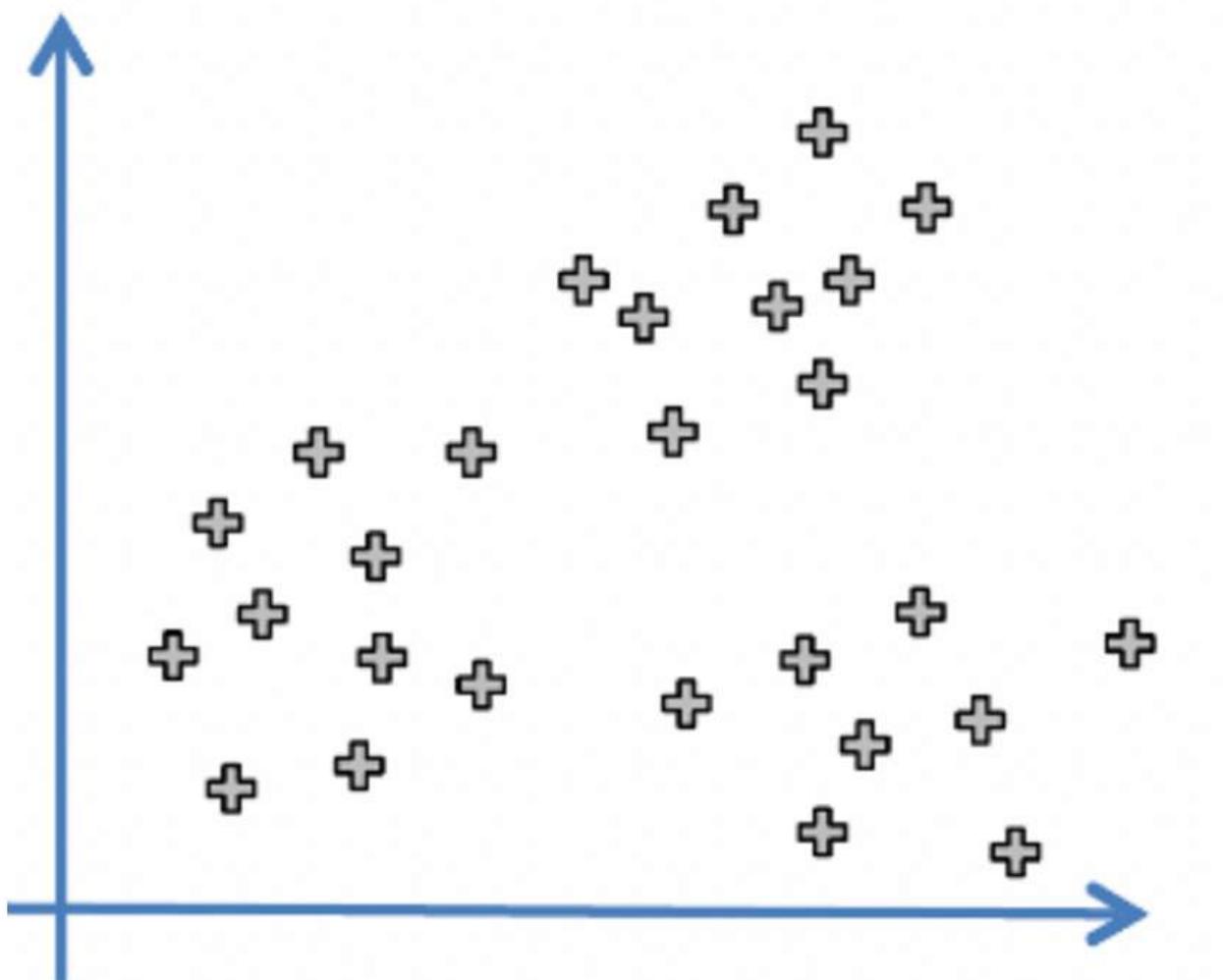
Tenemos la siguiente imagen antes de aplicar el algoritmo de K-Means una nube de puntos. En este caso hemos utilizado dos variables independientes el eje de las X y el eje de las Y, y nos ha dado por pintar una nube de puntos. Pueden ser usuarios con dos variables cualesquiera que los describan o productos u objetos o programas de televisión. Son observaciones. La idea es agruparlos.

Los agrupamos todos juntos porque están bastante juntitos, en cuyo caso sólo habría un cluster. Decidimos separarlo en dos clusters, por ejemplo, los que están más a la izquierda y los que están más a la derecha separados respectivamente.

O por ejemplo, tres clusters. El ojo humano yo creo que detectaría tres clusters fácilmente.

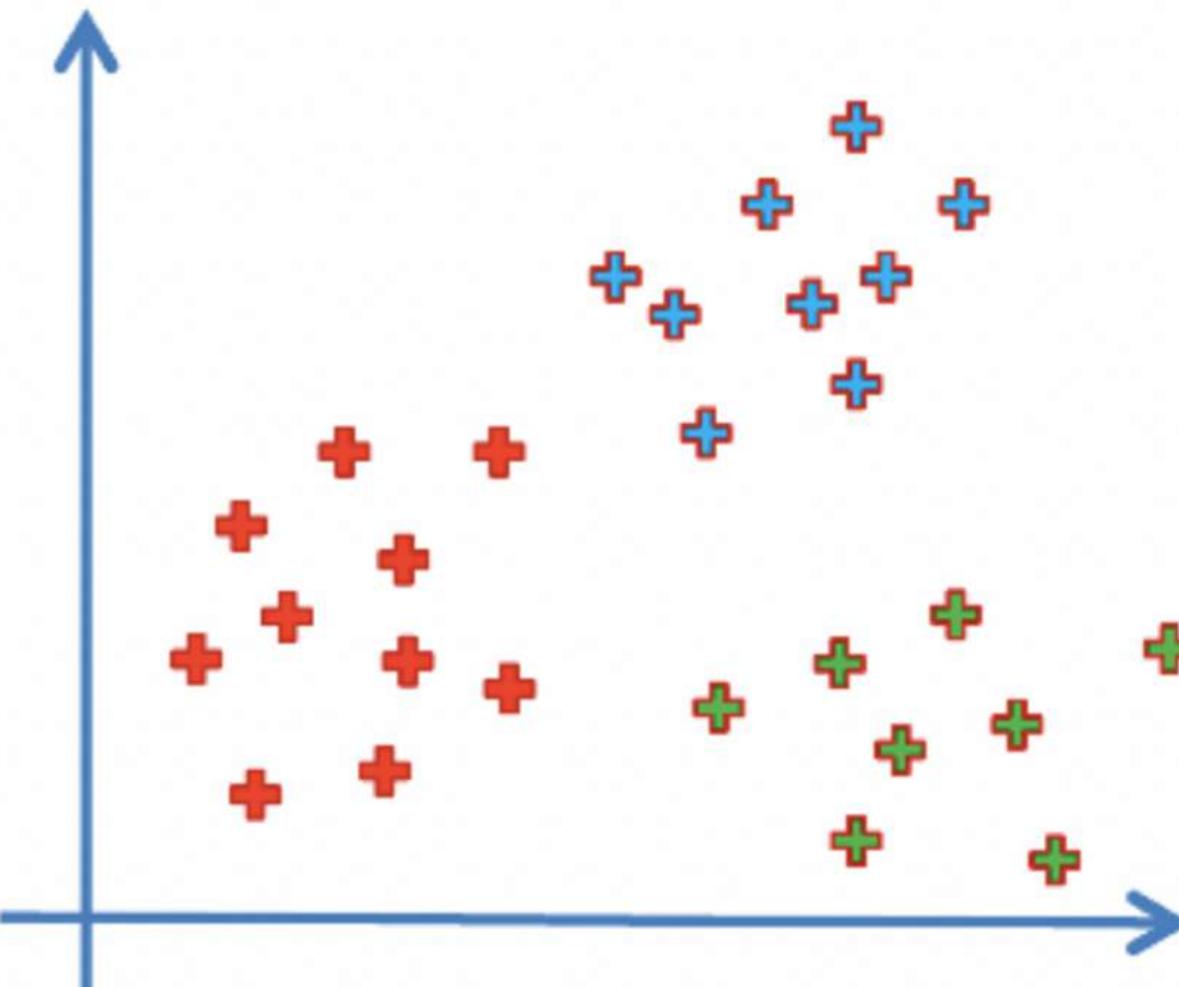
K-MEANS

Antes de K-Means



K-Means

Después de K-Means

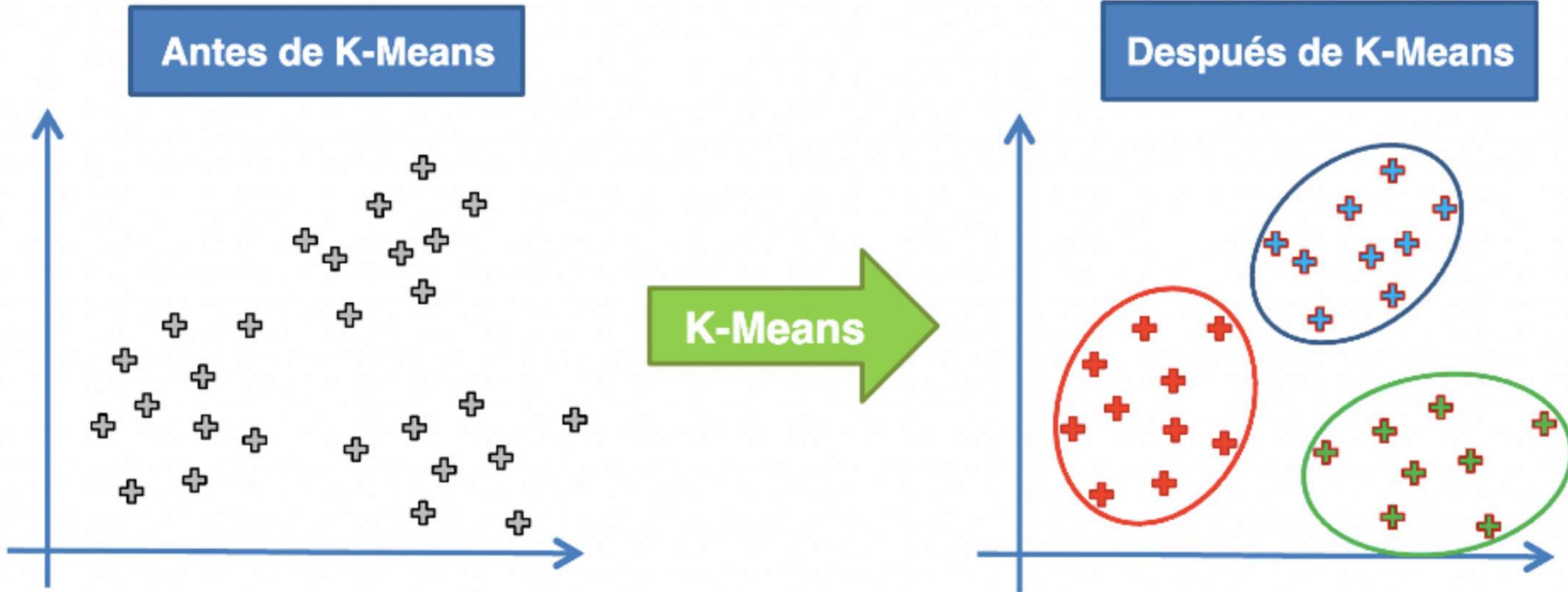


K-MEANS

Entonces la idea es que K-Means se va a encargar precisamente de hacer eso de que vamos a aplicar el algoritmo y después tendríamos una agrupación más visual, sin estar etiquetado!

Eso es la idea general de qué es lo que va a hacer el algoritmo, va a segmentar, va a dividir la información, juntando observaciones similares en un mismo cluster y separando observaciones que están lejos de las otras en clusters diferentes.

K-MEANS



K-MEANS

En primer lugar, lo que tenemos que hacer es elegir el número K de clusters. La técnica se llama K-Means, y la K es precisamente porque en función del número de clusters que elijamos nos saldrá una segmentación u otra. Esto significa que vamos a depender de una selección de un parámetro del algoritmo.

A continuación se seleccionan puntos cualesquiera del dominio donde se esté trabajando y esos van a ser los baricentros iniciales. No tienen por qué ser puntos del dataset, simplemente serán los hipotéticos centros geométricos de cada clase.

El baricentro o el centro de gravedad en el mundo de la física es simplemente el centro geométrico, el punto que quedaría en medio de cada uno de los hipotéticos clusters. También los seleccionamos al azar. Por tanto, en función de donde seleccionen esos puntos, podría ser que el algoritmo nos dé un resultado u otro.

K-MEANS

A continuación, asignaremos cada uno de los puntos del dataset al baricentro más cercano. Por tanto, cada punto se asociará al baricentro que hemos elegido que le cae más cerca. Esto será la primera conformación de clusters. Esto formará K clusters de modo que cada uno de los puntos del dataset quedará asignado a un cluster aquel cuyo baricentro caiga más cerca.

Ahora vamos a calcular y asignar el nuevo baricentro de cada cluster. Todos los puntos del primer cluster se pueden utilizar para calcular un nuevo baricentro. Todos los puntos del segundo cluster básicamente sirven para calcular su nuevo baricentro, de modo que esos centros originales quedan re calculados y todos los puntos evidentemente quedan asociados a ese nuevo baricentro.

K-MEANS

Y a continuación, reasignaríamos cada uno de los puntos de datos a su nuevo baricentro más cercano. Esto es repetir el paso tres, con la diferencia de que aquí hay una regla adicional que es que si ha habido nuevas asignaciones, si algún punto ha cambiado de baricentro, pasamos al paso cuatro y si no, hemos terminado.

La idea general es que siempre habrá k -baricentros que se irán moviendo y siempre debo asignar los puntos a ese baricentro más cercano. Hasta que hay un momento en que el baricentro no se mueve o no hay reasignaciones de los puntos con respecto a los clusters.

K-MEANS

PASO 1: Elegir el número K de clusters



PASO 2: Seleccionar al azar K puntos, los baricentros (no necesariamente de nuestro dataset)



PASO 3: Asignar cada punto al baricentro más cercano

→ Esto formará los K clusters



PASO 4: Calcular y asignar el nuevo baricentro de cada cluster



PASO 5: Reasignar cada punto de los datos a su baricentro más cercano.

Si ha habido nuevas asignaciones, ir al PASO 4, si no ir FIN.



El Modelo está Listo

K-MEANS

Tal vez no es un algoritmo 100% claro desde el punto de vista teórico, pero esta transparencia anterior es bastante ilustrativa y se marcaría como punto de referencia para que en el futuro puedan volver e investigar cómo funciona exactamente el algoritmo.

Recordar que Python lo hará automáticamente casi TODO por nosotros.



EJEMPLO PRÁCTICO

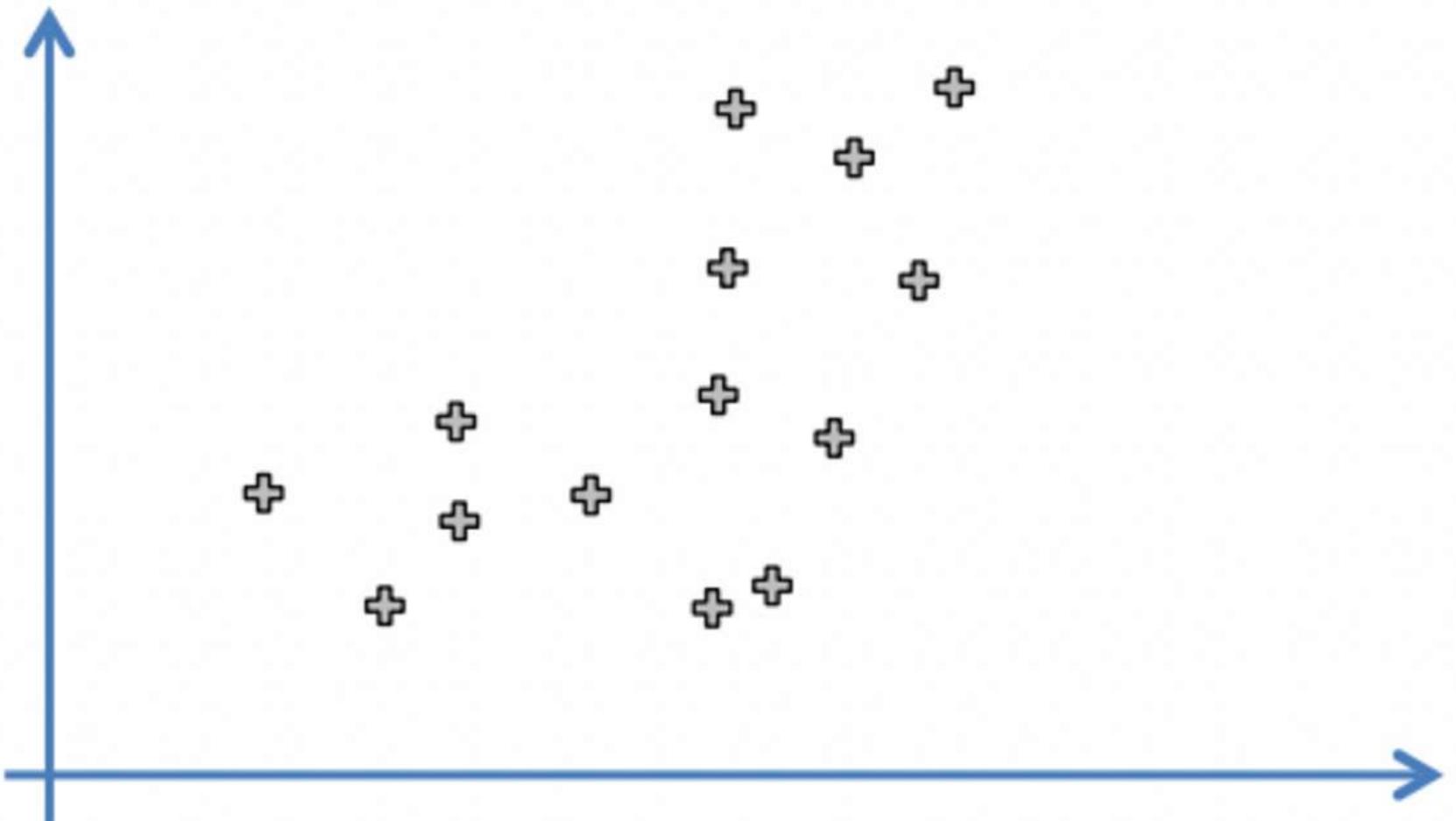
K-MEANS

Supongamos que vamos a elegir dos clusters, vamos a separar estos puntos en dos clusters, en dos agrupaciones diferentes.

¿Puedes identificar visualmente y de forma rápida los grupos finales que van a terminar seleccionándose, que van a acabar surgiendo de este conjunto de puntos? No es muy intuitivo. La pregunta es bastante difícil.

Imaginar como sería la situación de compleja si tuviéramos 3 variables, 4 variables, 20 variables.

PASO 1: Elegir el número K de clusters: K = 2



K-MEANS

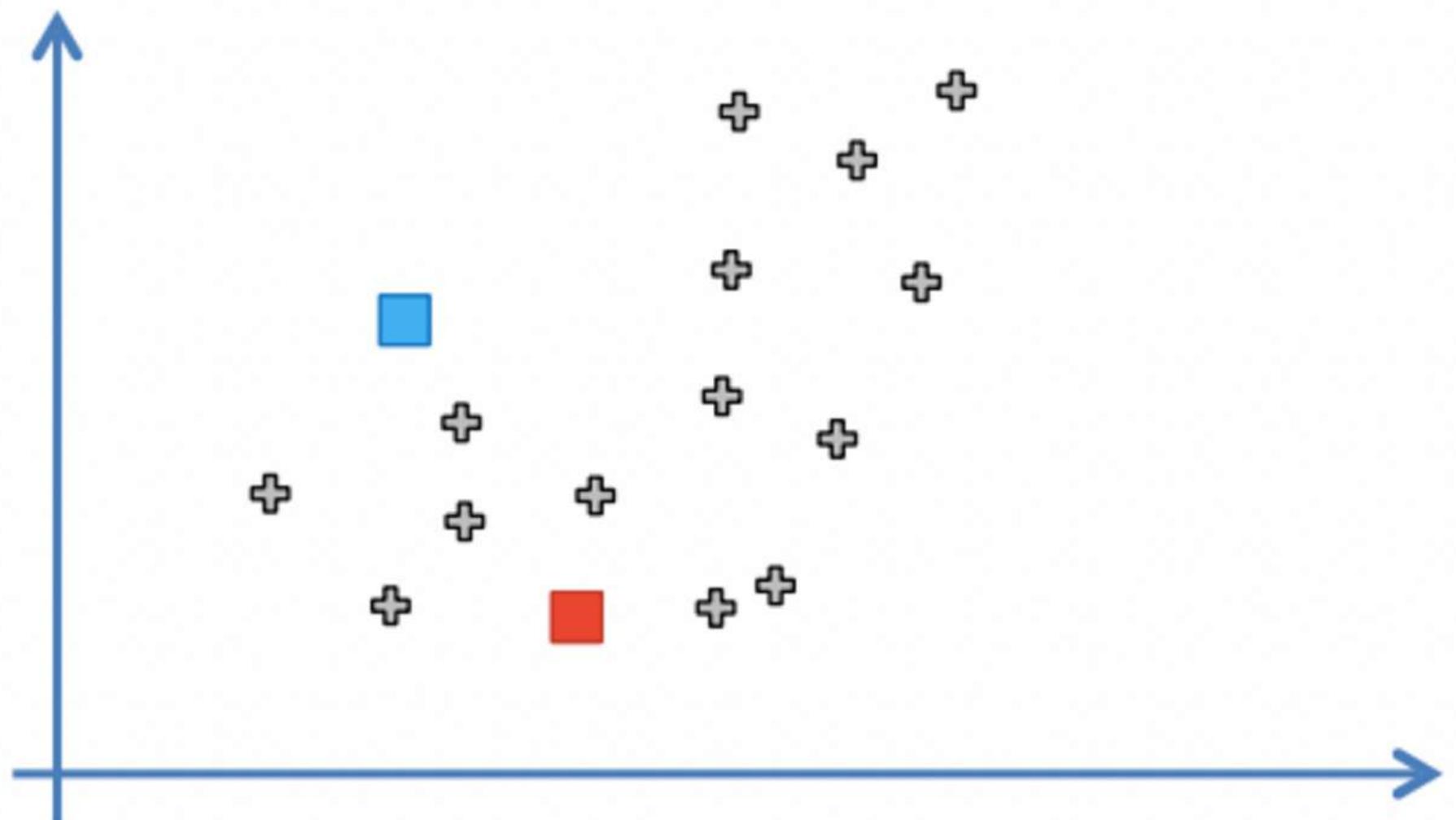
PASO 2: Seleccionar al azar K puntos, los baricentros (no necesariamente de nuestro dataset)

Lo que tengo que hacer es primero manualmente, seleccionar los dos centros hipotéticos de del cluster.

Vamos a tomar unos centros hipotéticos a partir de los que se crearán los clusters.

Entonces, paso uno Elegimos el número de clusters. (2)

Seleccionamos dos puntos cualesquiera para que sean los baricentros (rojo y azul)



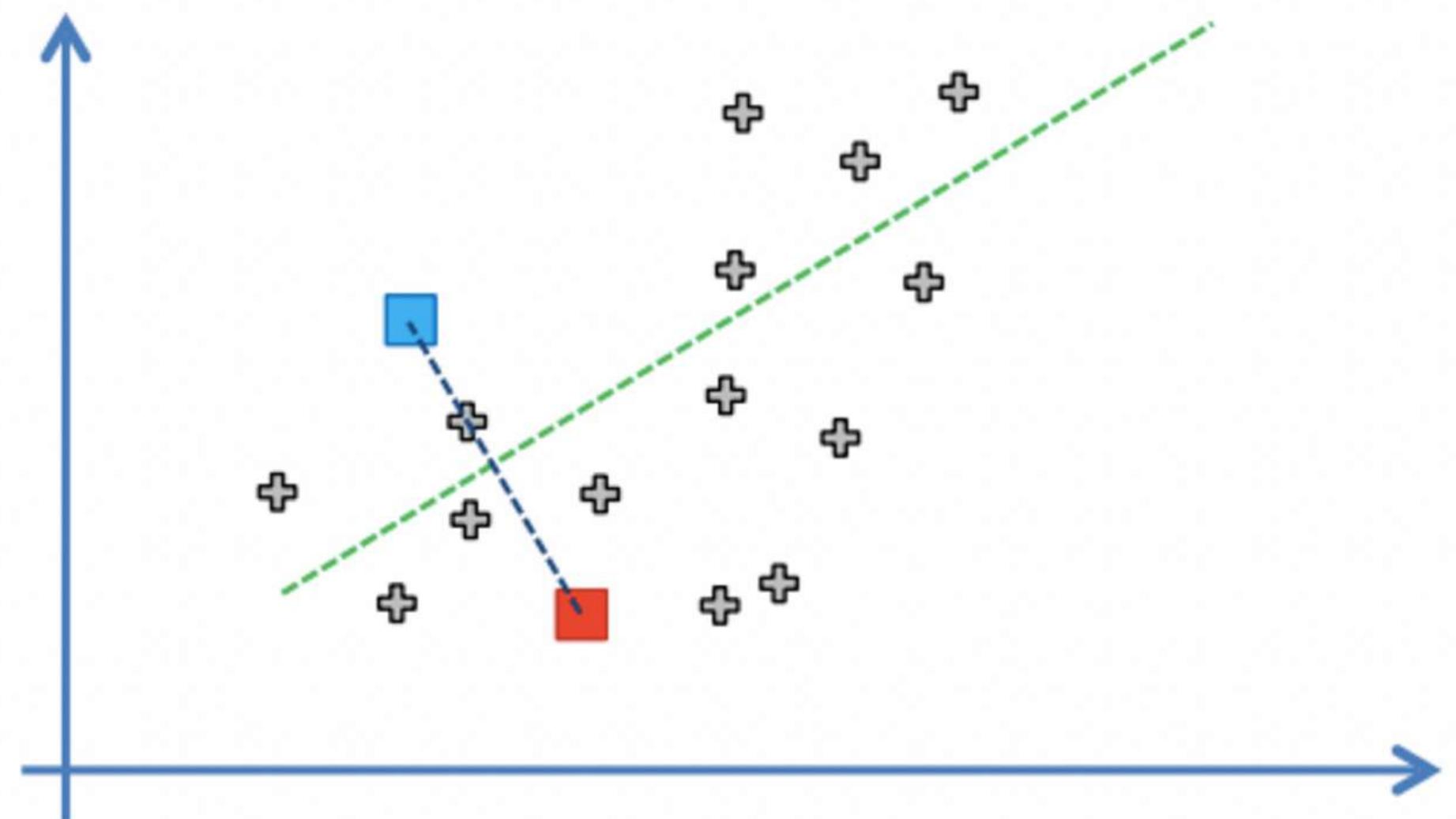
K-MEANS

Entonces, lo único que necesitamos ahora es identificar para cada uno de los puntos cuál de los dos centroides, cuál de los dos baricentros es el más cercano.

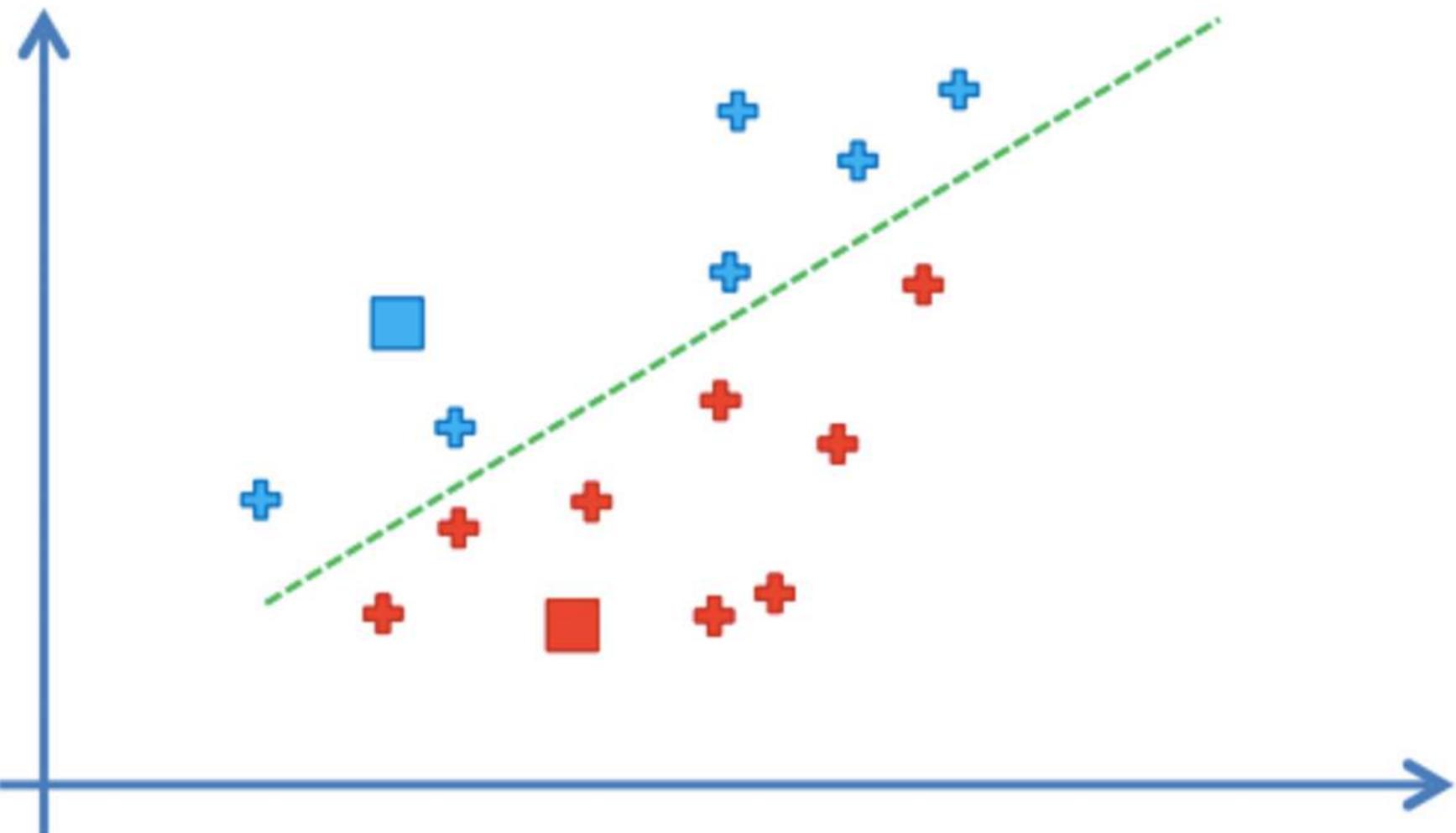
Según la distancia que establezcamos, hay que decidir cuál de los dos queda más cerca y los pintamos del color respectivo. (Geometría)

Si dibujamos la línea que une cada uno de los centroides de los baricentros con los puntos de color gris, la línea más corta corresponde al baricentro que le queda más cerca.

PASO 3: Asignar cada punto al baricentro más cercano



K-MEANS



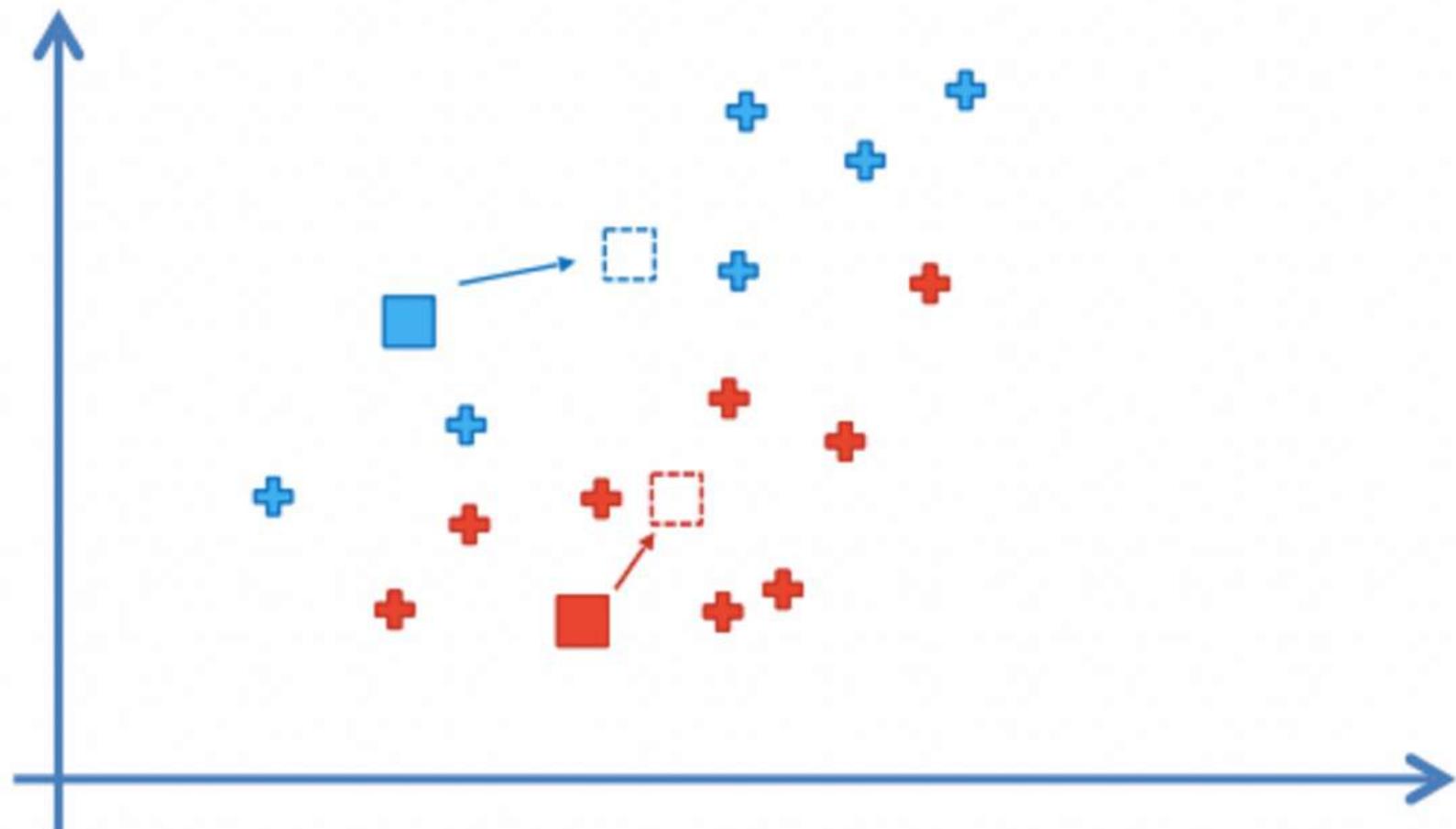
Entonces lo que se puede hacer es establecer así una perpendicular utilizando un truquito de geometría, pintar la mediatriz, la recta que divide este segmento en dos partes totalmente iguales y de forma perpendicular.

Y por los conocimientos de geometría que se tienen, lo que queda por encima de la línea de color verde está más cerca del punto azul y si no es más cerca del punto rojo.

Sobre todo por que estamos utilizando la distancia euclídea.

K-MEANS

PASO 4: Calcular y asignar el nuevo baricentro de cada cluster



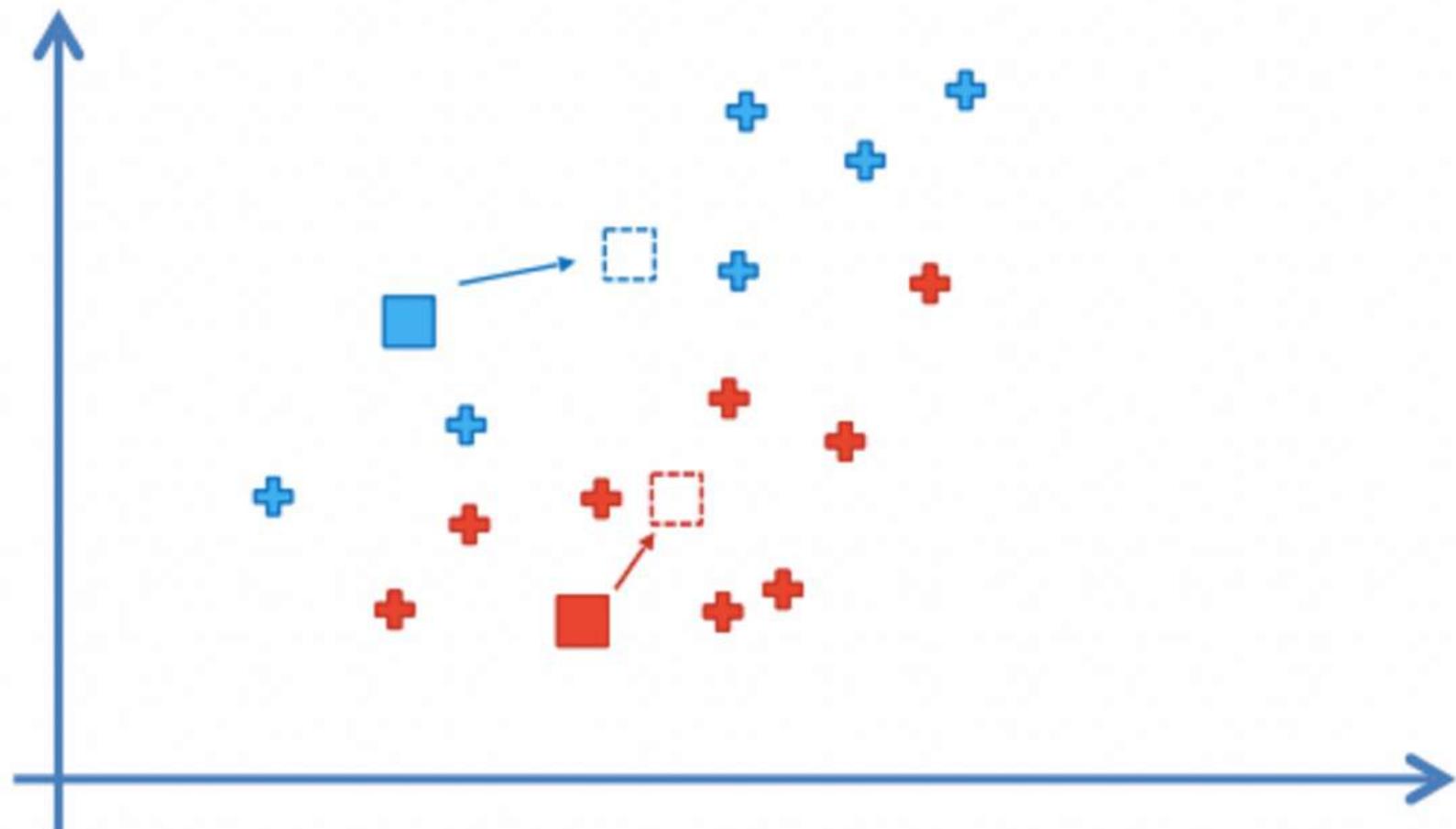
Después de haber pintado los puntos, simplemente se trata de volver a calcular el nuevo centro de gravedad.

Tomo todos y cada uno de los puntos de color azul y hago la media geométrica de ellos, que básicamente corresponde con el baricentro.

Ojo, aquí hemos utilizado una distancia euclidiana. Esto significa que los clusters nos quedarán redonditos, con formas redonditas como huevos o elipses, cosa que no sería así si utilizáramos otro tipo de distancia. Por ejemplo, la distancia Manhattan provocaría que los clusters fueran más cuadrados.

K-MEANS

PASO 4: Calcular y asignar el nuevo baricentro de cada cluster



Después de haber pintado los puntos, simplemente se trata de volver a calcular el nuevo centro de gravedad.

Tomo todos y cada uno de los puntos de color azul y hago la media geométrica de ellos, que básicamente corresponde con el baricentro.

Ojo, aquí hemos utilizado una distancia euclidiana. Esto significa que los clusters nos quedarán redonditos, con formas redonditas como huevos o elipses, cosa que no sería así si utilizáramos otro tipo de distancia. Por ejemplo, la distancia Manhattan provocaría que los clusters fueran más cuadrados.

K-MEANS

Entonces, al mismo tiempo, en matemáticas y ciencia de datos existen muchos tipos de distancia. La euclídea es la más común.

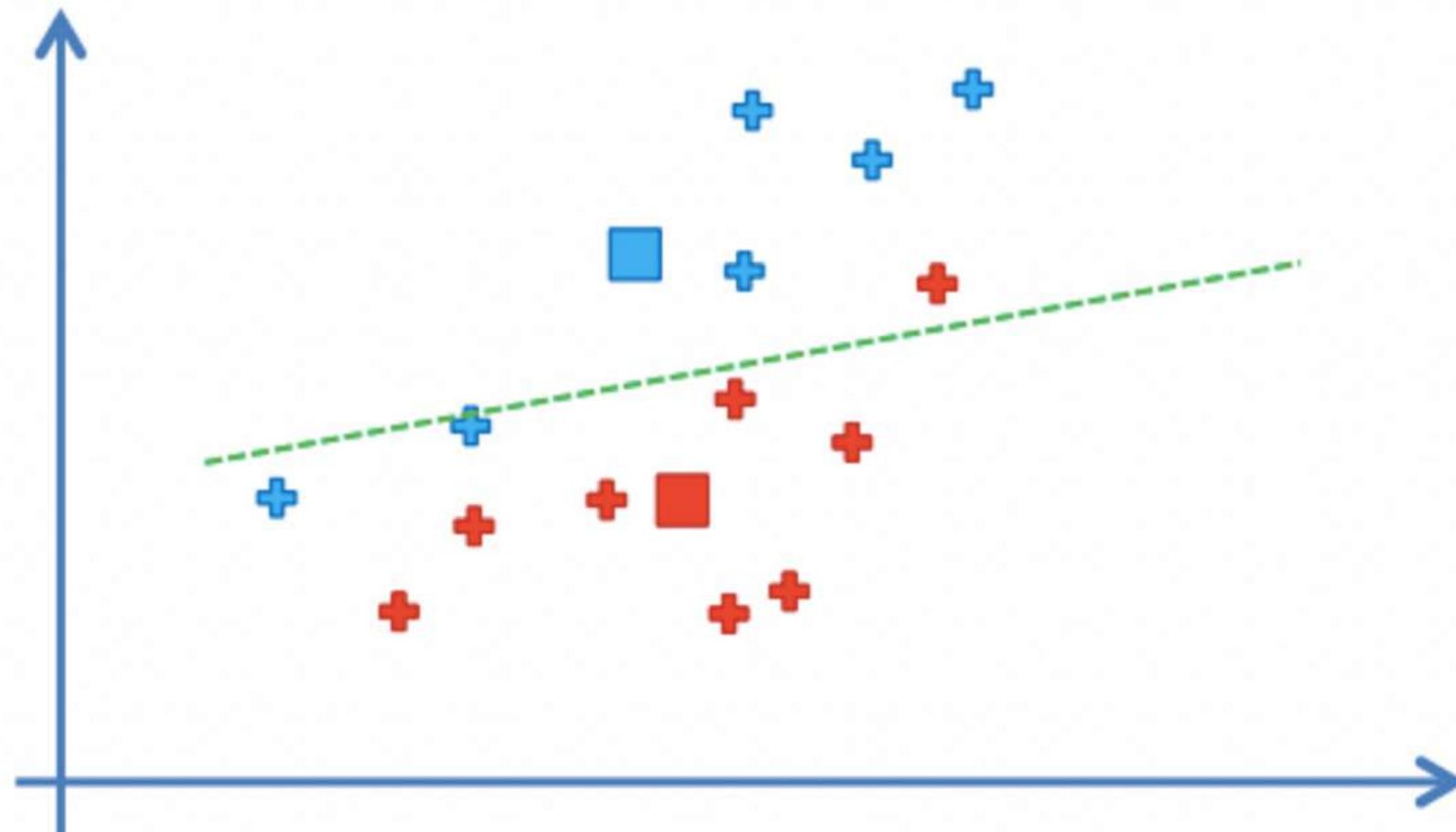
Sin embargo, si la pregunta es ¿qué distancia debo seleccionar? La respuesta va a depender del tipo de problema que se requiera resolver.

Regresando al tema del recálculo, se vuelve a calcular el centro geométrico, el centro de gravedad, y esto significa que los baricentros serán sustituidos, serán reemplazados.

Simplemente la media geométrica, el nuevo baricentro de todos los puntos azules y de los puntos de color rojo, se mueve por el espacio y se coloca en otro lado.

K-MEANS

PASO 5: Reasignar cada punto de los datos a su baricentro más cercano.
Si ha habido nuevas asignaciones, ir al PASO 4, si no ir FIN.

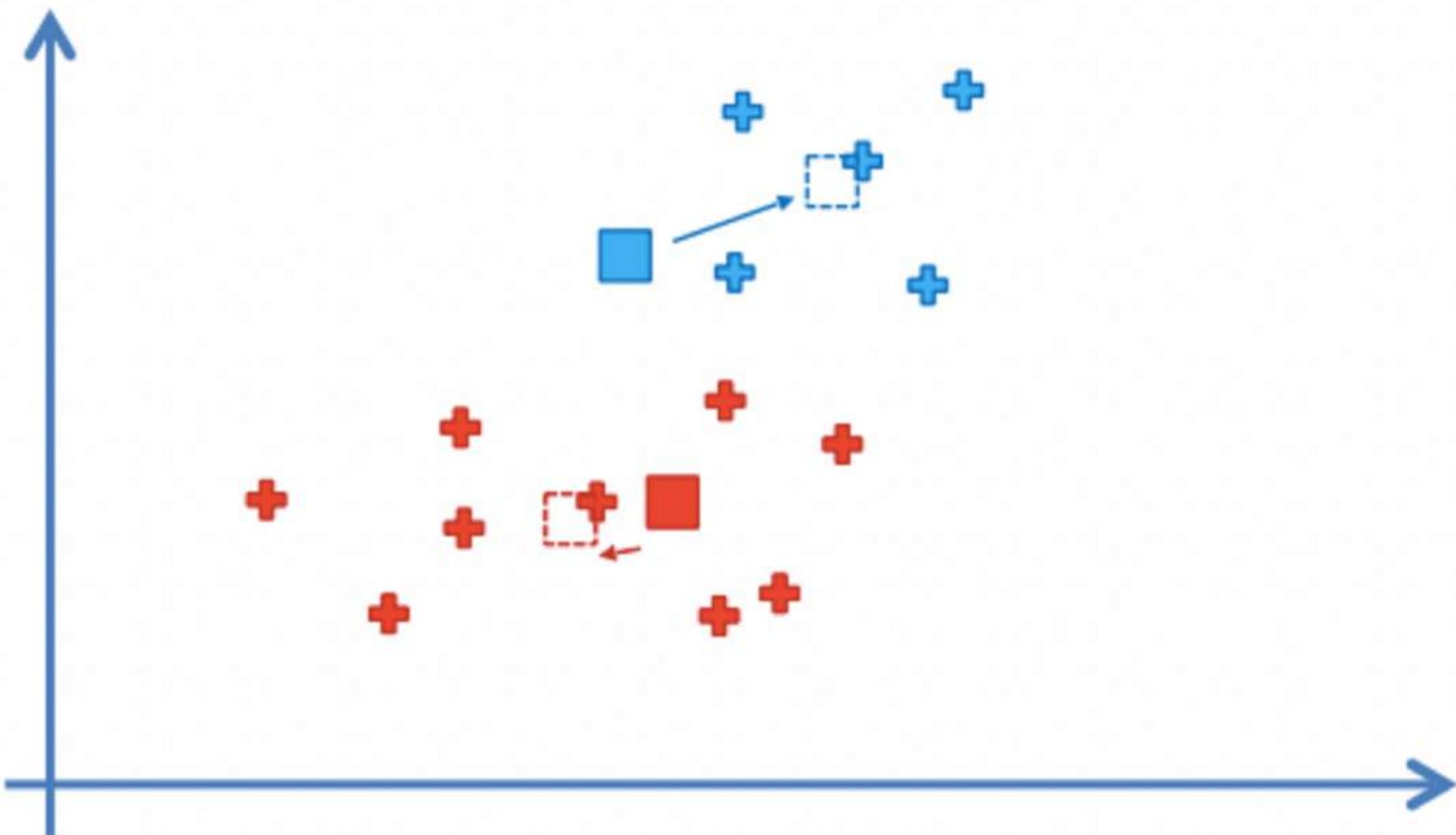


Y a continuación nos falta volver a aplicar el paso número tres, que básicamente es el paso cinco, que es reasignar cada punto de datos al nuevo baricentro que ahora le queda más cerca.

Y si hay nuevas asignaciones, si algún punto cambia de bandos, pues hay que volver a repetir el paso cuatro y volver a calcular el baricentro.

K-MEANS

PASO 4: Calcular y asignar el nuevo baricentro de cada cluster



Vemos que hay dos puntos azules en zona roja y un punto rojo en zona azul. Llevamos a cabo una recolocación.

Cada uno de los puntos es pintado de nuevo de la zona que le corresponde y volvemos a calcular el nuevo centro geométrico, el nuevo baricentro de los puntos que han vuelto a ser coloreados. Entonces ese nuevo baricentro se vuelve a mover.

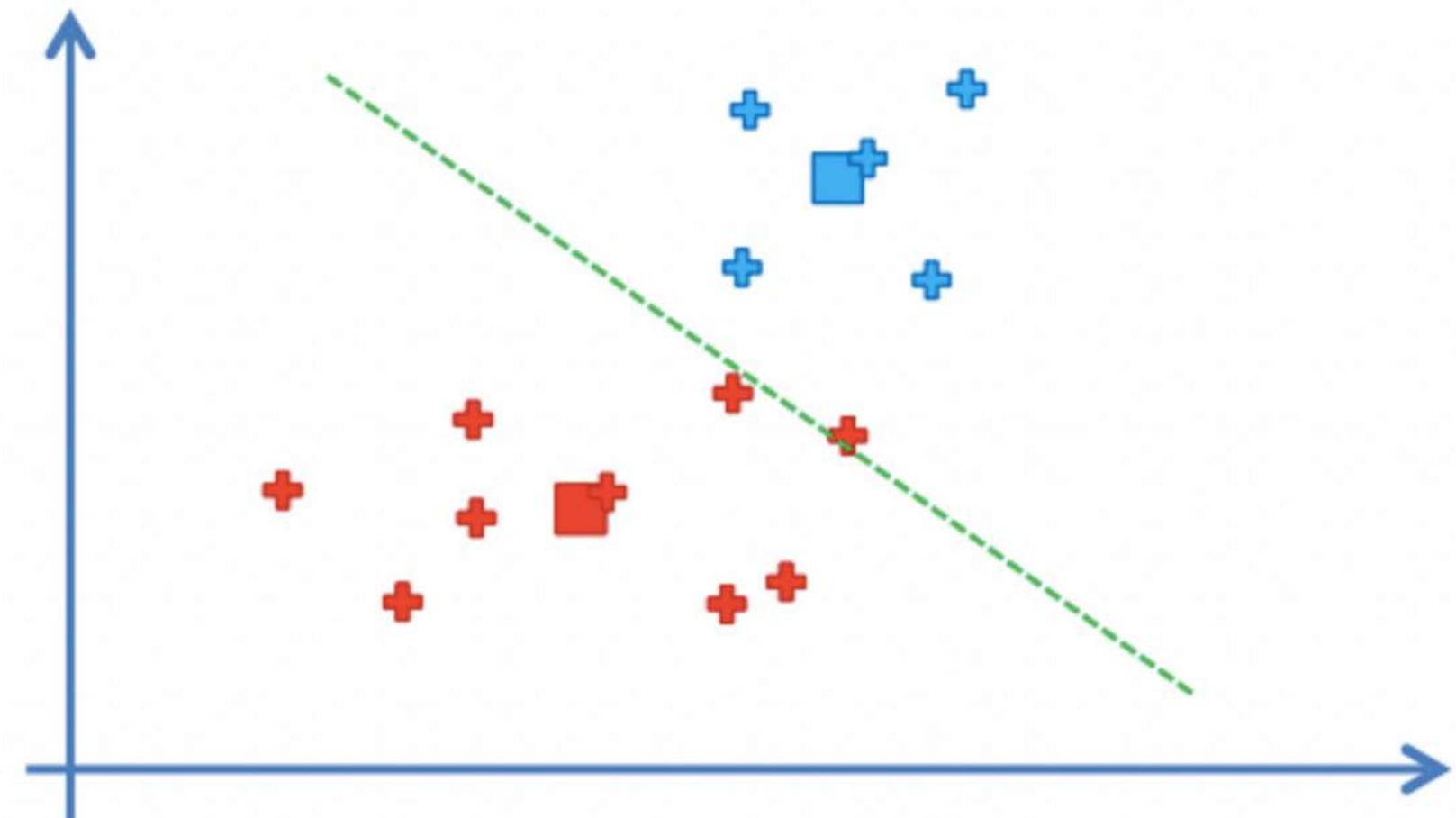
K-MEANS

Y volvemos a repetir. Calcularemos de nuevo como ha habido una reasignación de puntos el el nuevo centro geométrico.

Volveríamos a hacer el proceso de asignar cada punto a su baricentro más cercano.

Hay que continuar repitiendo el proceso. ¿Cuándo paramos?

PASO 5: Reasignar cada punto de los datos a su baricentro más cercano.
Si ha habido nuevas asignaciones, ir al PASO 4, si no ir FIN.



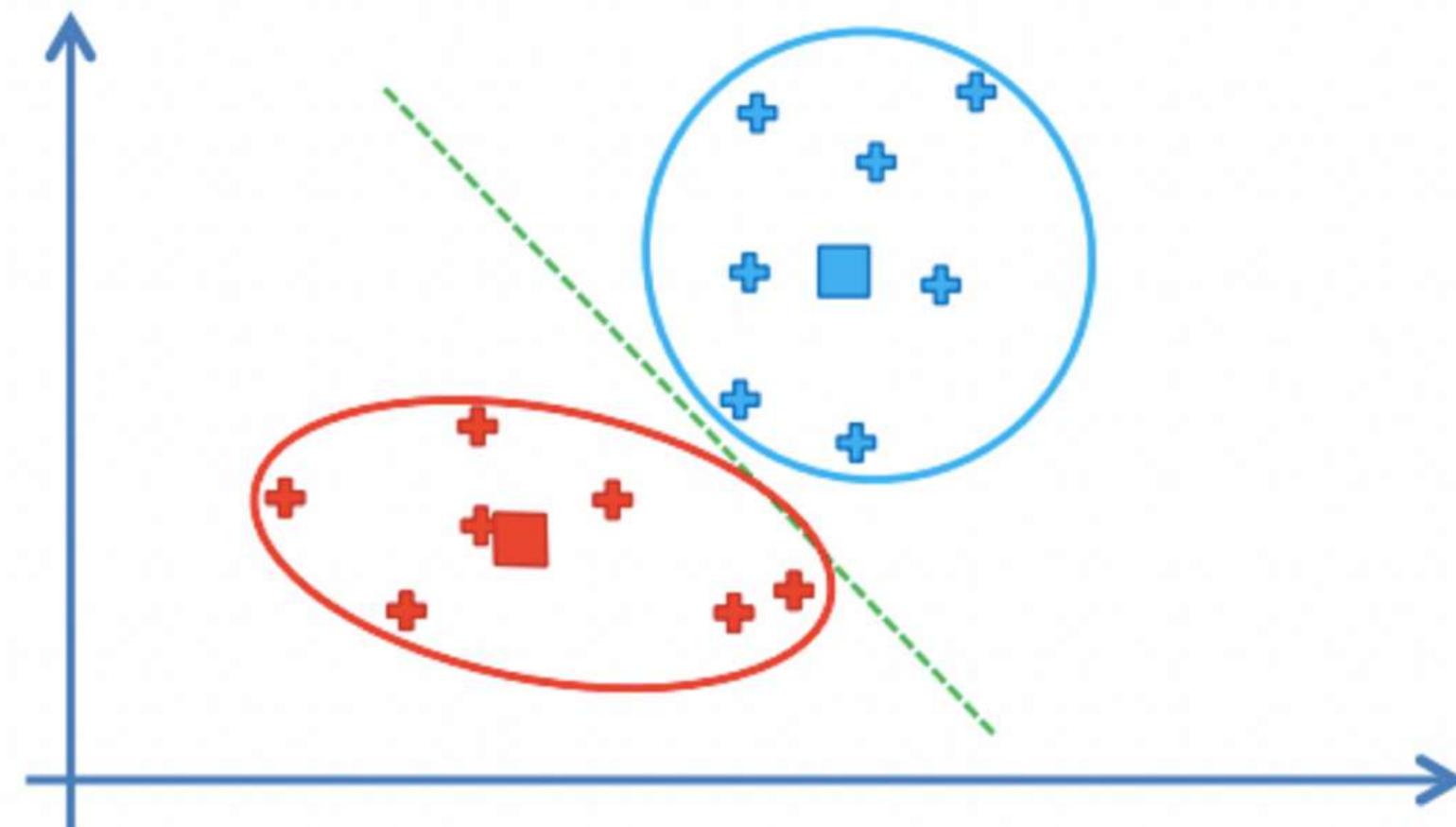
K-MEANS

Hasta encontrarnos en un punto en que no se ha realizado ninguna reasignación.

Todos los de color azul están del lado azul y todos los rojos del lado rojo.

Esto significa que el algoritmo ha convergido.

FIN: El modelo está listo





TRAMPA DE INICIALIZACIÓN ALEATORIA

K-MEANS

Recuerda que uno de los puntos clave era elegir al inicio una serie de baricentros al azar que serían los primeros que arrancarían el algoritmo de K-Means.

Veamos que según cómo se inicializa en esos centros, el resultado puede ser bastante diferente al esperado. Por tanto, hay que tener cuidado con esa elección. Volvamos a una nube de puntos como la que teníamos originalmente.

Digamos que vamos a elegir tres clusters para segmentar el resultado.

K-MEANS



Si elegimos $K = 3$ clusters...

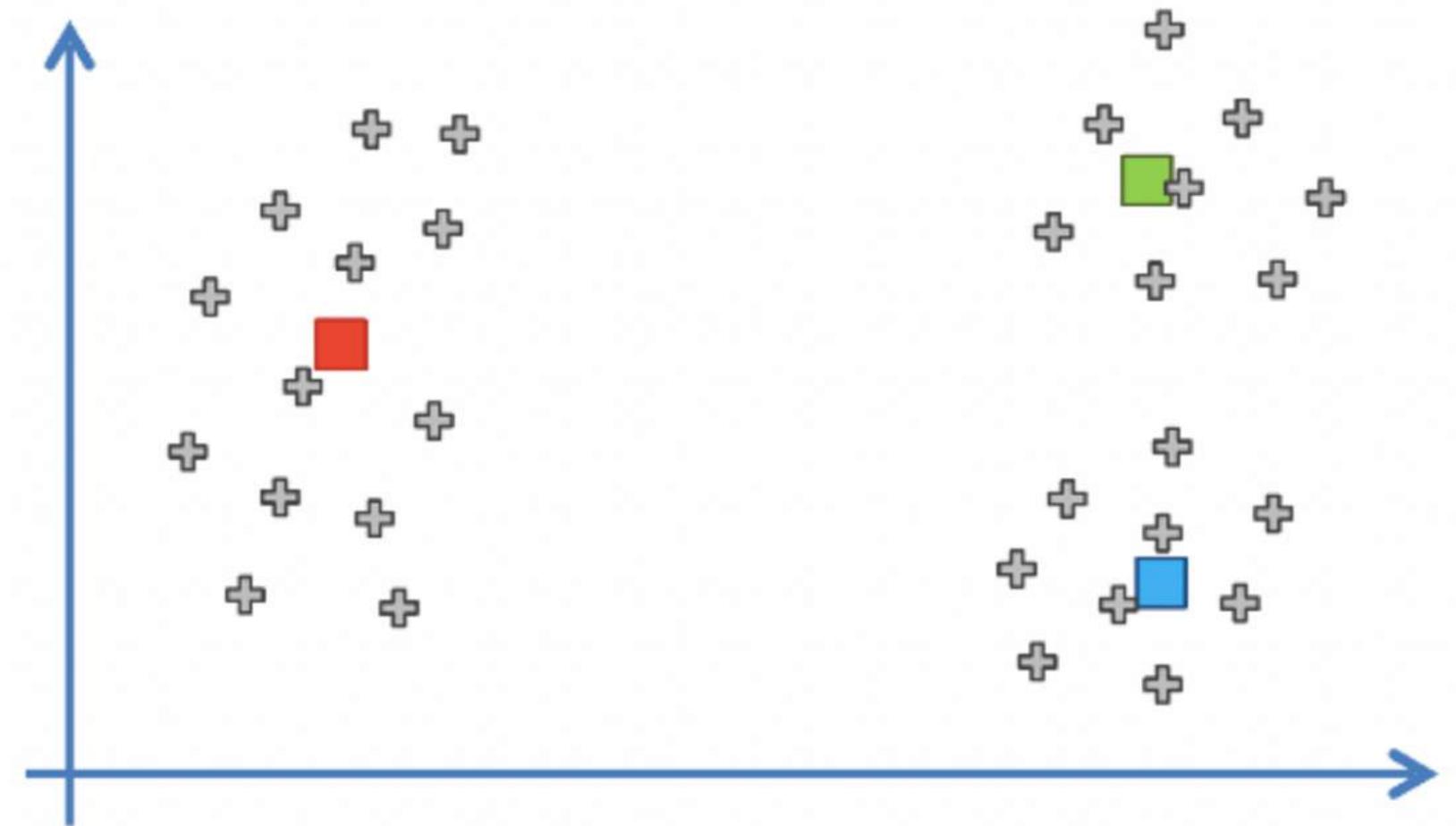
Uno evidentemente no puede decidir cómo van a quedar los clusters finales sin antes saber dónde se colocarán los puntos originales.

K-MEANS

Haremos una inicialización, llamémosla correcta, de los tres centroides.

Partimos de una ubicación a la que el ojo humano casi, casi detecta a simple vista. Y vamos a suponer que el algoritmo se ejecuta correctamente. Esto significa ya no pasar por todas las fases o etapas del algoritmo.

E incluso si nos movemos alrededor del centroide, los puntos quedan bastante bien separados entre sí, de modo que el algoritmo ha dado un resultado bastante bueno.



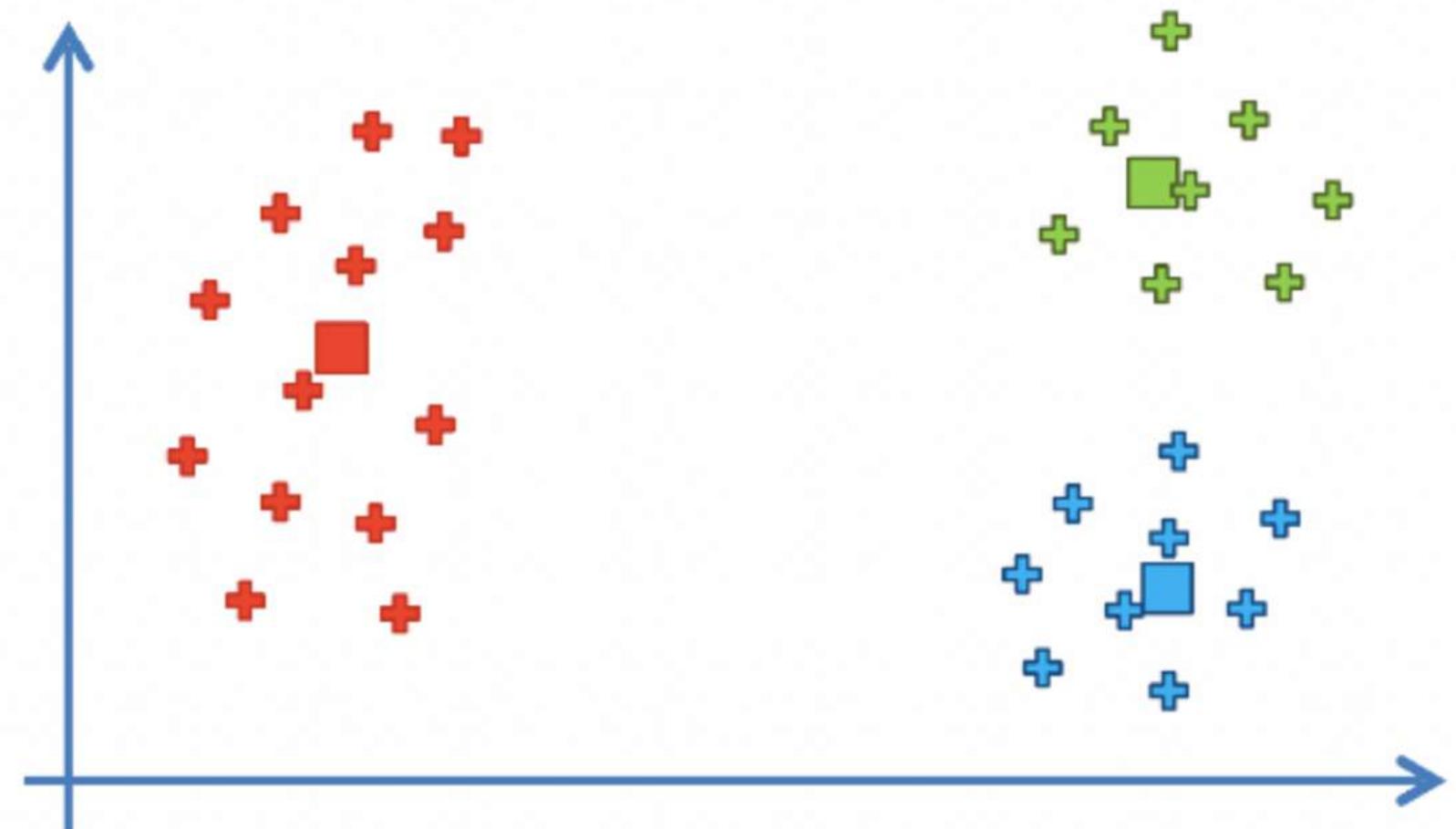
...esta inicialización correcta nos lleva a...

K-MEANS

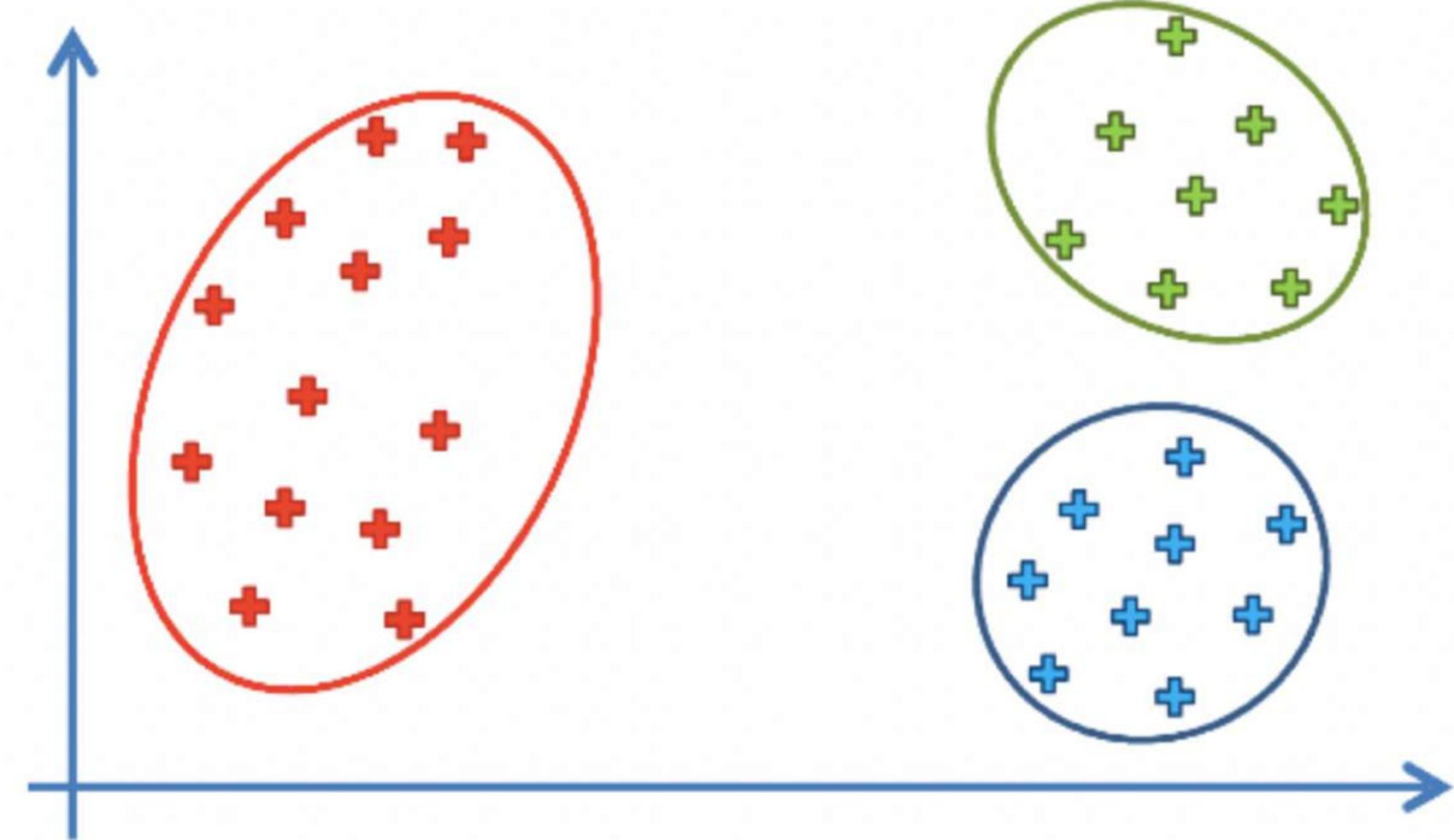
Haremos una inicialización, llamémosla correcta, de los tres centroides.

Partimos de una ubicación a la que el ojo humano casi, casi detecta a simple vista. Y vamos a suponer que el algoritmo se ejecuta correctamente. Esto significa ya no pasar por todas las fases o etapas del algoritmo.

E incluso si nos movemos alrededor del centroide, los puntos quedan bastante bien separados entre sí, de modo que el algoritmo ha dado un resultado bastante bueno.



K-MEANS



Esto sería lo fácil!

¿Qué pasaría si elegimos una mala inicialización aleatoria de los baricentros en lugar de coger un baricentro cerca?

K-MEANS

PASO 1: Elegir el número K de clusters: $K = 3$



K-MEANS

PASO 2: Seleccionar al azar K puntos, los baricentros (no necesariamente de nuestro dataset)

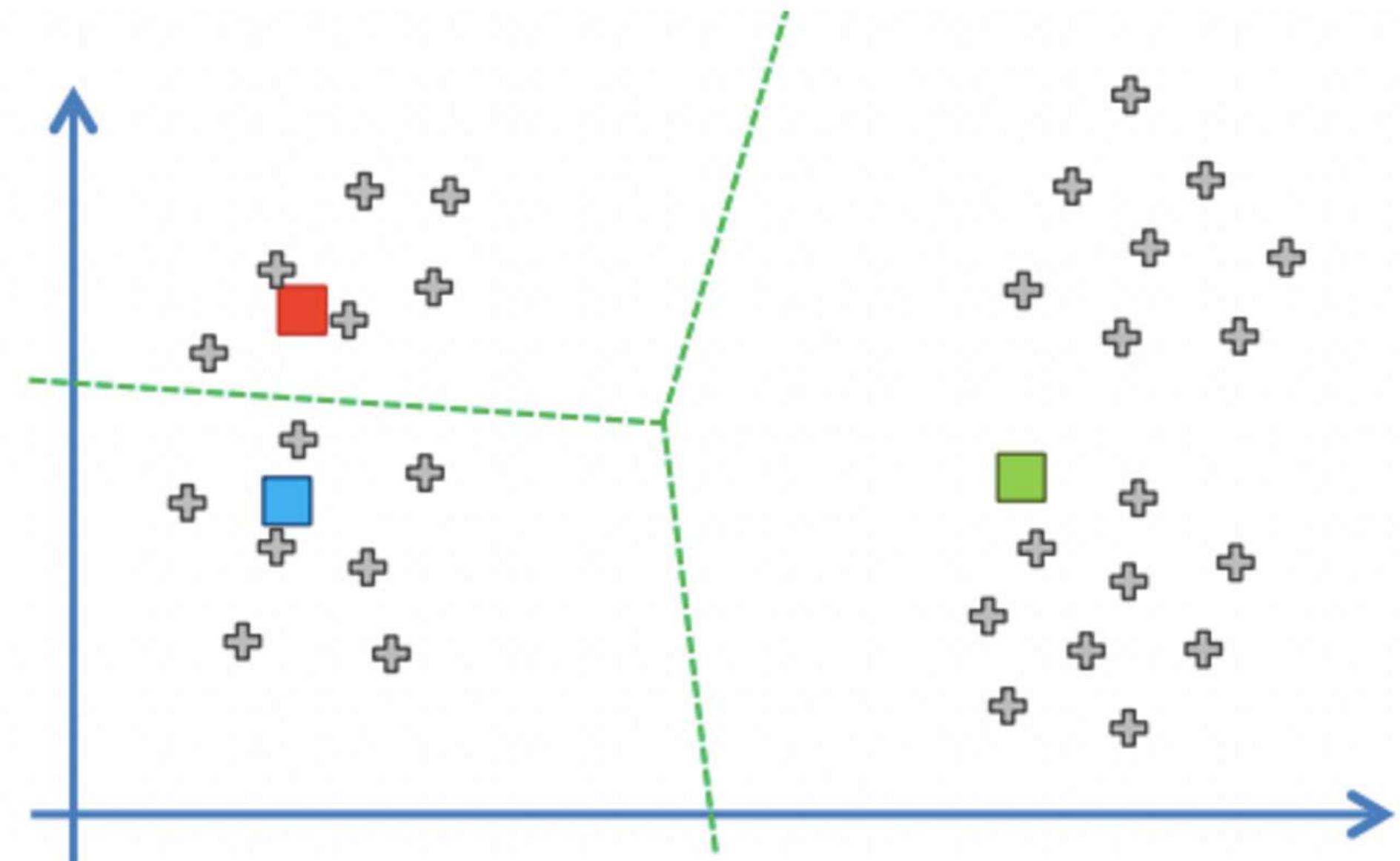


Simplemente el paso dos es seleccionar al azar puntos que serán los centros originales antes de aplicar el algoritmo

K-MEANS

Si tenemos eso en cuenta y lo único que pensamos es tener tres clusters entonces hacemos una asignación totalmente al azar de los tres baricentros originales.

Podríamos tener lo siguiente:



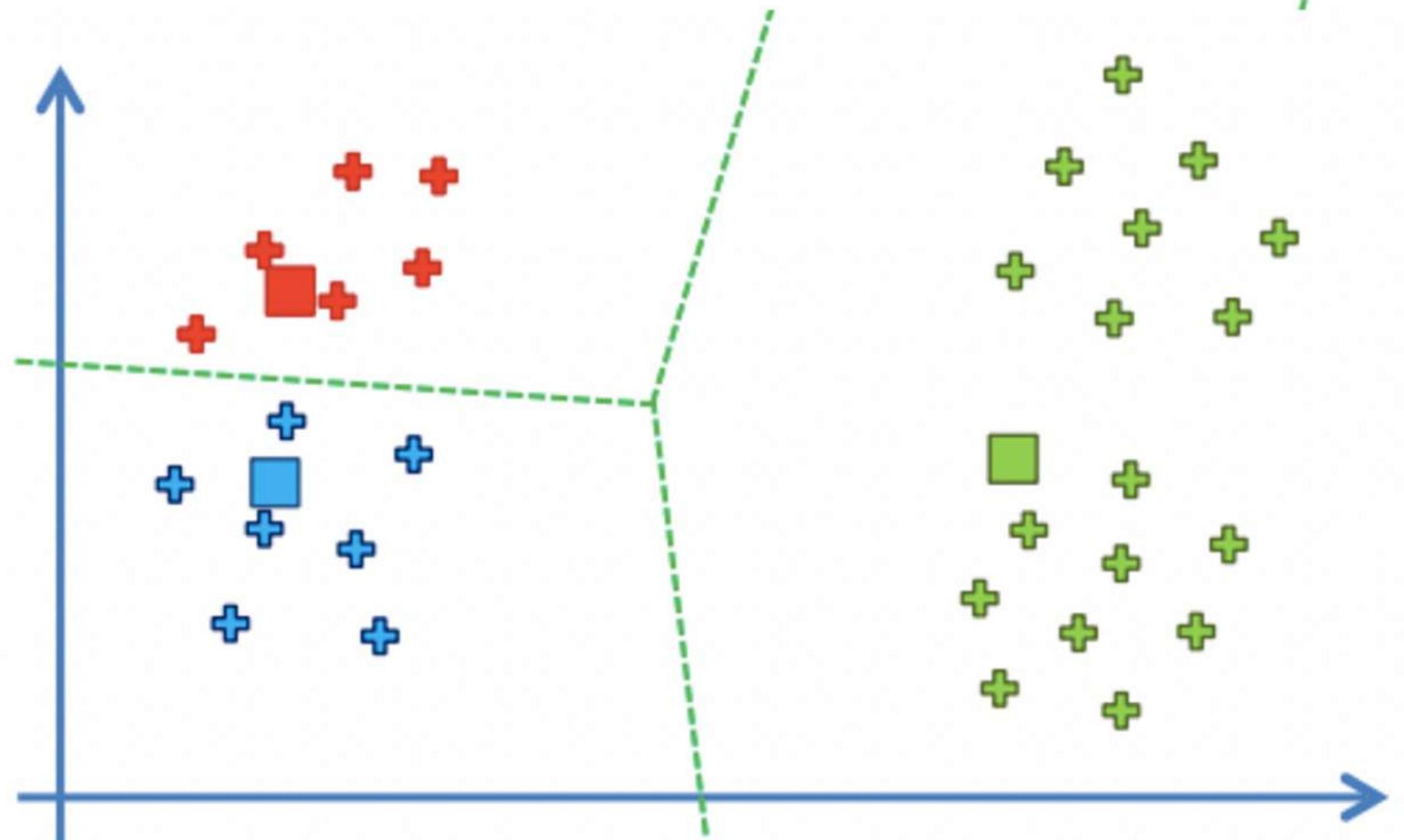
Dibujamos las líneas que separan los puntos a la misma distancia (euclidiana). Nos quedaría una división similar a esta

K-MEANS

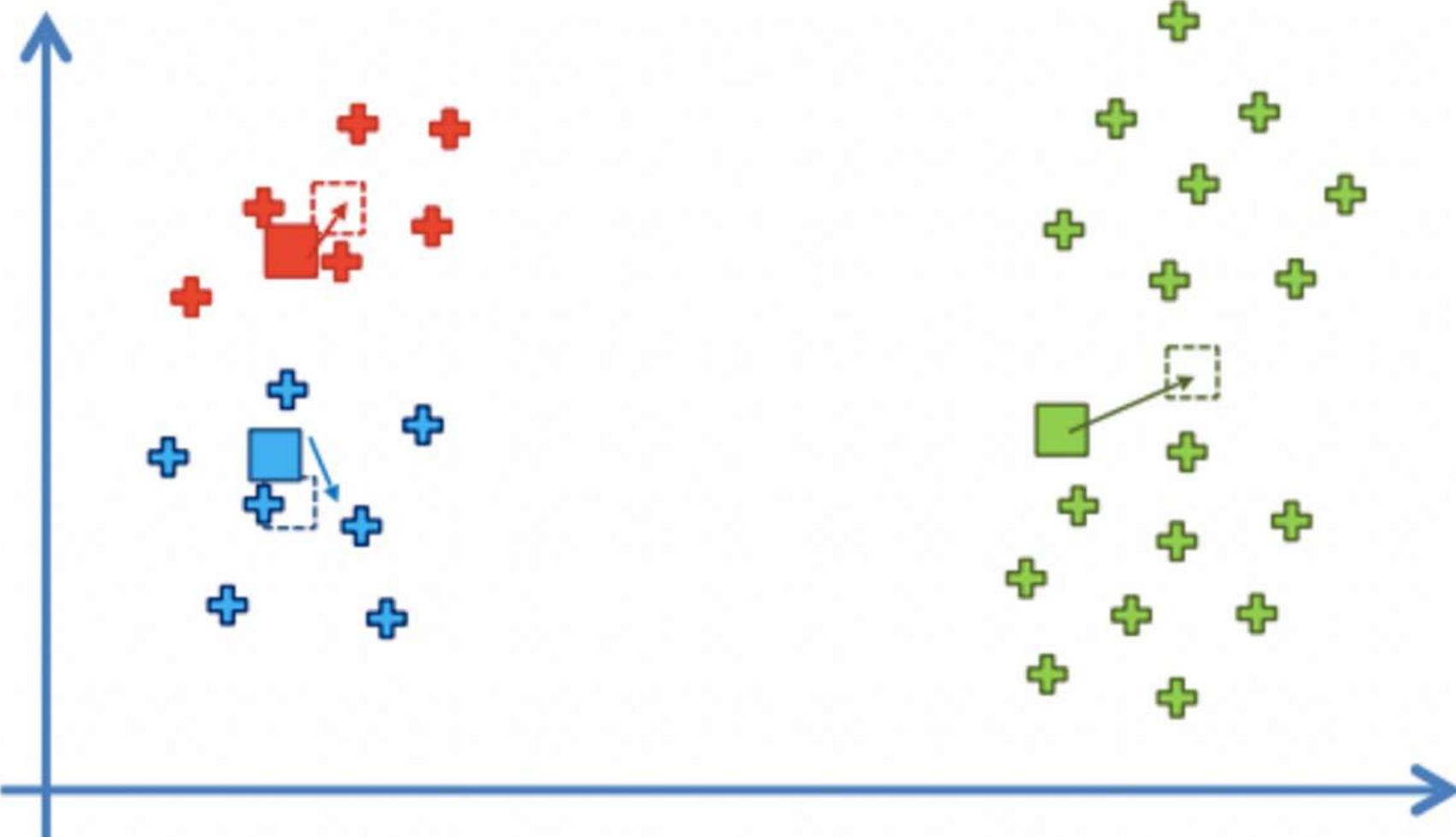
Repetiríamos lo mismo la mediatrix que separa el rojo del verde, la que separa el verde del azul y la que separa el azul del rojo.

Tendríamos esas tres divisiones, esos tres grupos, con esos tres centroide originales, y el principio sería el mismo.

PASO 3: Asignar cada punto al baricentro más cercano



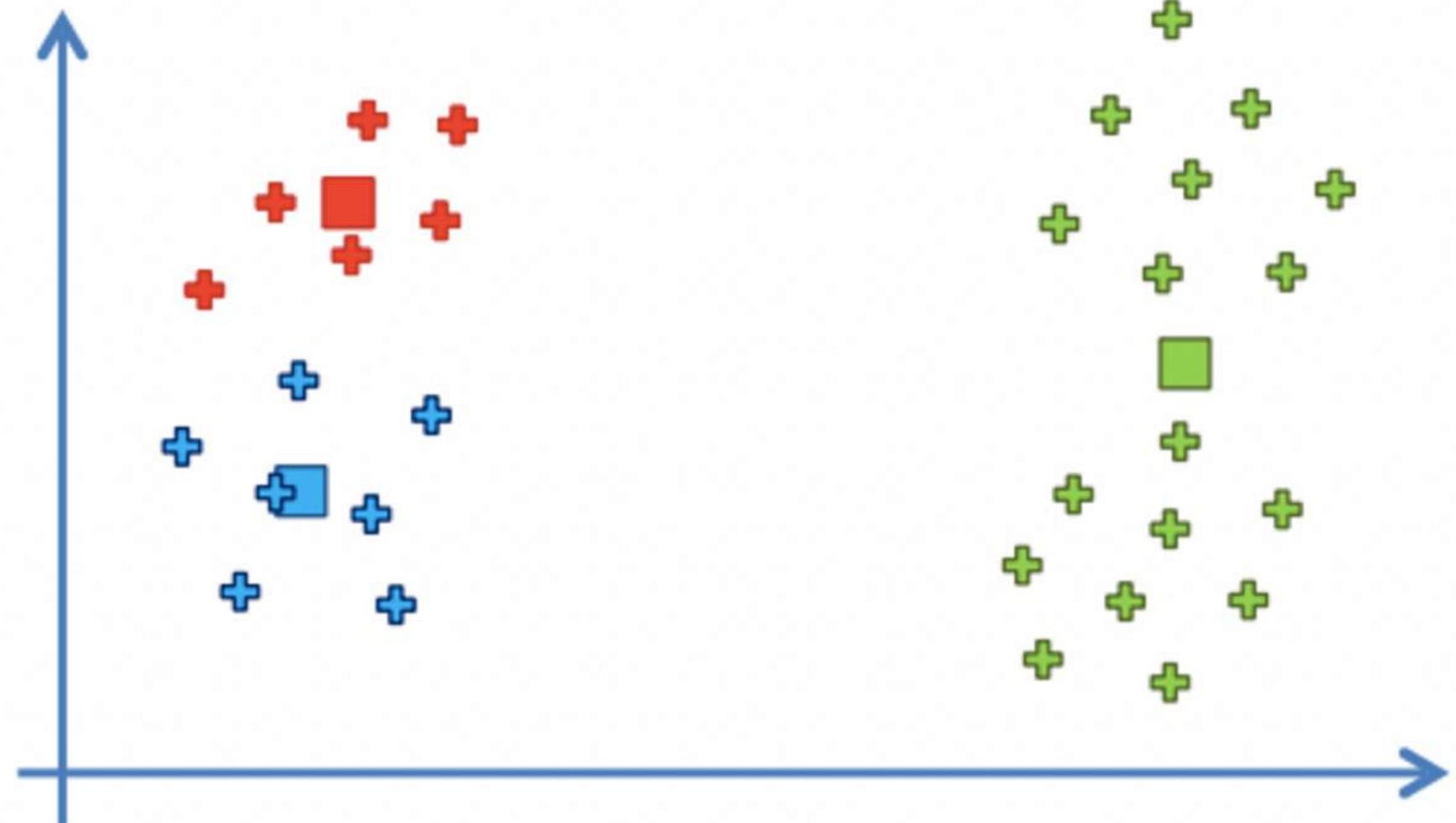
K-MEANS



Cada uno de los puntos que cae en cada una de esas regiones separadas por las líneas de puntos de color verde, es coloreada del color de su centroide más cercano y se calcula el nuevo baricentro del mismo.

K-MEANS

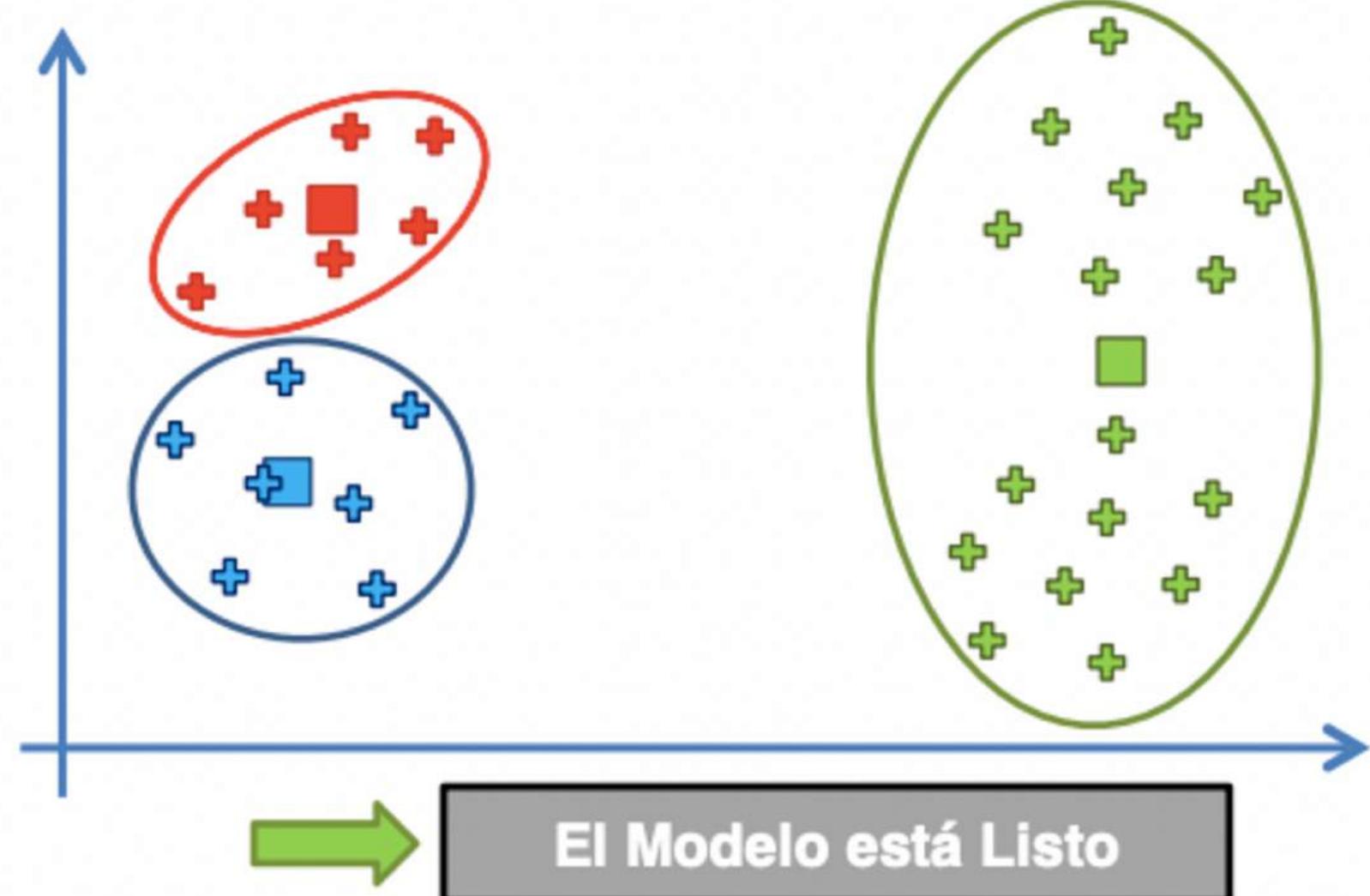
PASO 4: Calcular y asignar el nuevo baricentro de cada cluster



Esto se iría repitiendo constantemente, es decir, volveríamos a pintar las rectas que separan de forma equidistante cada uno de los centros geométricos.

K-MEANS

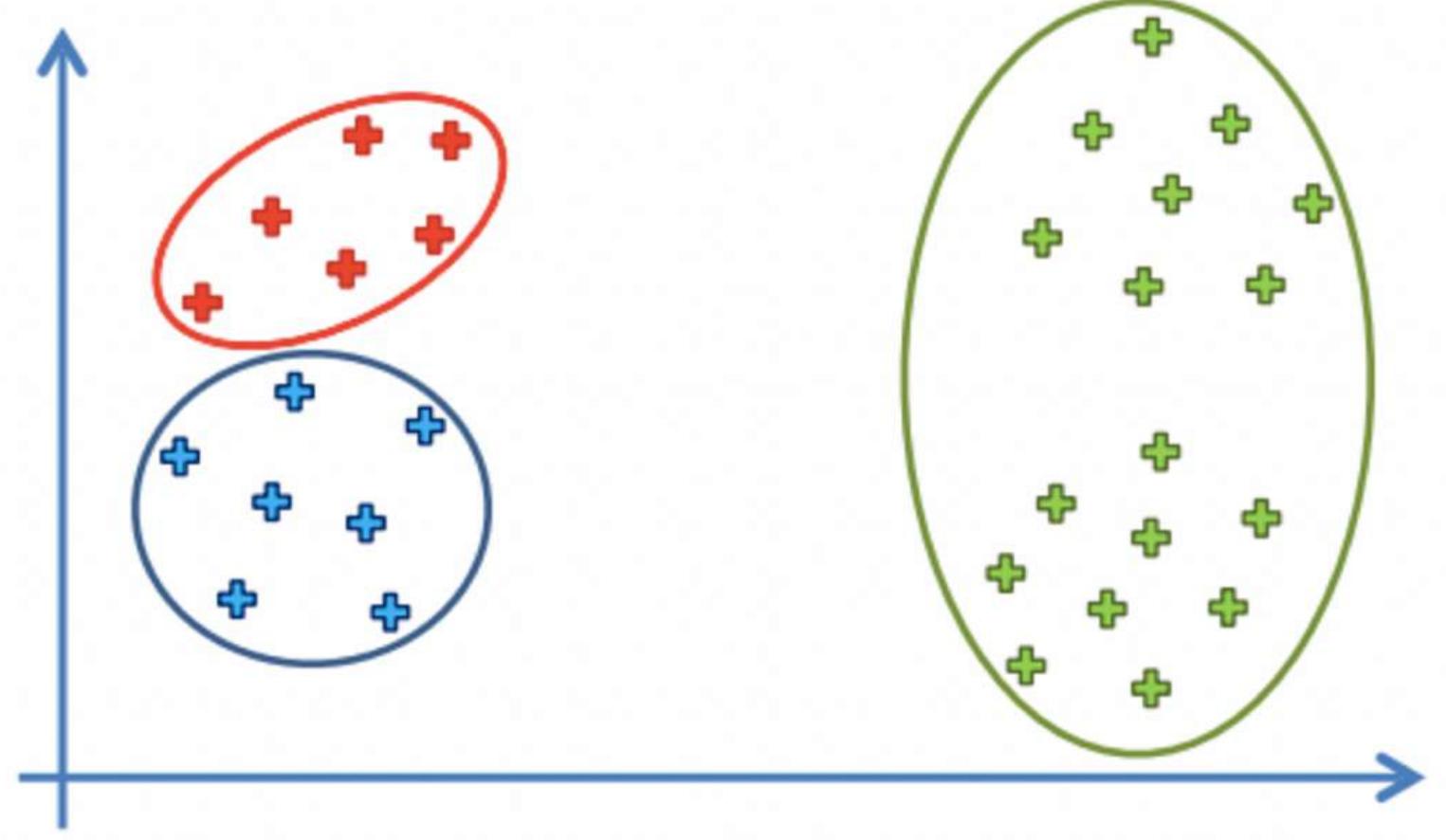
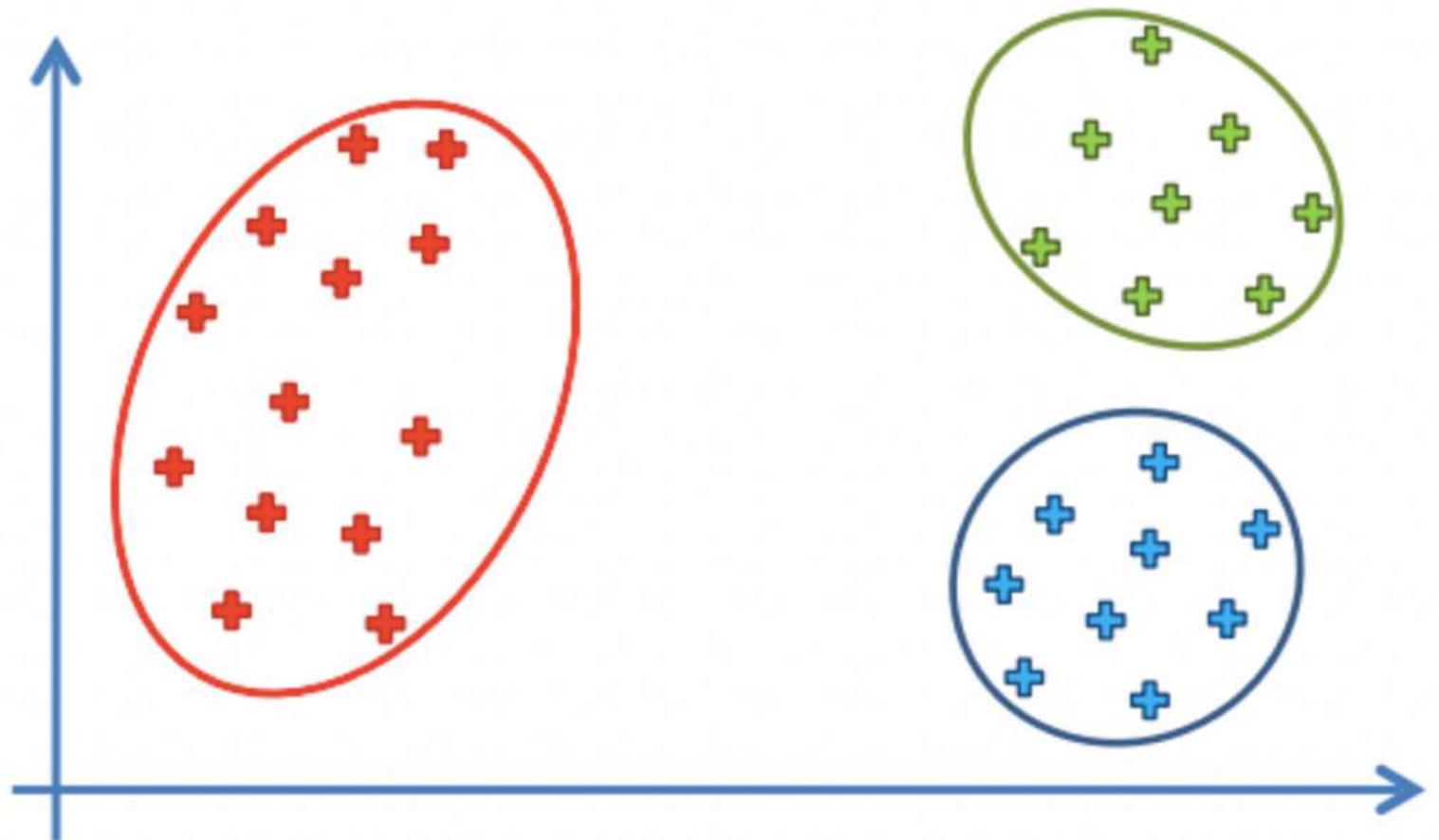
PASO 5: Reasignar cada punto de los datos a su baricentro más cercano.
Si ha habido nuevas asignaciones, ir al PASO 4, si no ir FIN.



Este sería nuestro modelo,
este sería el resultado.

Y como resultados tendríamos
estos tres clusters que se ven
aquí, y son bastante diferentes
de los que teníamos en un
principio.

K-MEANS



K-MEANS

Entonces estos tres son diferentes y tenemos una situación un poco peligrosa.

Un fenómeno en el que según como inicializamos los baricentros al origen del algoritmo, podemos sesgar cuál será el resultado final. Y esto no es bueno, porque una elección totalmente aleatoria del centroide puede llevar a una clasificación incorrecta.

¿Cómo corregimos esto?

Bueno, la respuesta no es tan sencilla. No tiene una solución realmente fácil, sino que existe una modificación del algoritmo de k-means que permite seleccionar bien correctamente los centroide iniciales del algoritmo.

Esa mejora del algoritmo se llama K-Means++

K-MEANS

Solution



K-Means++

K-MEANS

No lo veremos en este curso, es algo bastante avanzado y bastante más complicado que el algoritmo de K-Means. Tiene una estructura bastante engorrosa y difícil de entender. Es un enfoque bastante, bastante complicado.

Pero la buena noticia es que todo eso ocurre en segundo plano, de modo que el algoritmo de K-Means++ está directamente implementado en Python.

Si se desea implementar, no hace falta entender el fundamento que hay detrás de ello. Por tanto, es buena idea estar al tanto de que este problema existe y de que nos puede sesgar de forma bastante importante el resultado final que nos resulta según la elección aleatoria de los baricentros originales.

Del mismo modo, saber que existen respuestas o que la gente se ha dedicado a buscar soluciones al problema y que se puede solucionar con este caso, con el k-means++



SELECCIÓN DEL NÚMERO DE CLUSTERS

K-MEANS

K-Means implica que hay que elegir el número K de clusters que se desean crear.

¿Cómo se puede seleccionar el número correcto de clusters para resolver el problema?

Esto es un punto que vale la pena tratar, porque al final, si se decide segmentar en dos grupos, en tres, en cuatro, en ocho, va a cambiar considerablemente el resultado del algoritmo.

Bueno, partimos del mismo problema de siempre.

K-MEANS

El algoritmo de K-Means, la K evidentemente indica cuántos grupos van a salir de la segmentación.

PASO 1: Elegir el número K de clusters: $K = 3$



K-MEANS

Y por ejemplo, ¿y por qué hemos elegido tres clusters? ¿No hubiera sido mejor elegir dos clusters o diez clusters o 25 clusters? Necesitamos alguna métrica concreta que nos dé una forma de entender cómo evaluar si un cierto número de clusters es mejor o no que en comparación con otro número.

Entonces, ¿qué tipo de medida podemos imponer a nuestro algoritmo de agrupación para que nos diga algo sobre el resultado final que también hemos segmentado?

Bueno, pues aquí tenéis la métrica se llama la suma de los cuadrados del centro del cluster (within centers square sum).

$$WCSS = \sum_{P_i \in \text{Cluster 1}} d(P_i, C_1)^2 + \sum_{P_i \in \text{Cluster 2}} d(P_i, C_2)^2 + \sum_{P_i \in \text{Cluster 3}} d(P_i, C_3)^2$$

K-MEANS

Es decir, estaríamos ponderando, estaríamos calculando la suma de los cuadrados, de las distancias de cada punto con respecto al cluster que pertenece. Esa es la forma fácil de leerlo.

Esta fórmula puede parecer muy abrumadora o muy compleja, pero en realidad es súper, súper simple. Y la interpretación geométrica es extraordinaria porque lo que nos va a medir es qué tan bien clasificado está un punto con respecto al cluster al que pertenece cuando se tienen una serie de clusters.

Y como resulta que tenemos tres elementos, tres sumatorio y cada uno de los cuales se encarga de calcular los cuadrados de una distancia de los puntos de un cluster con respecto al centro del mismo.

K-MEANS

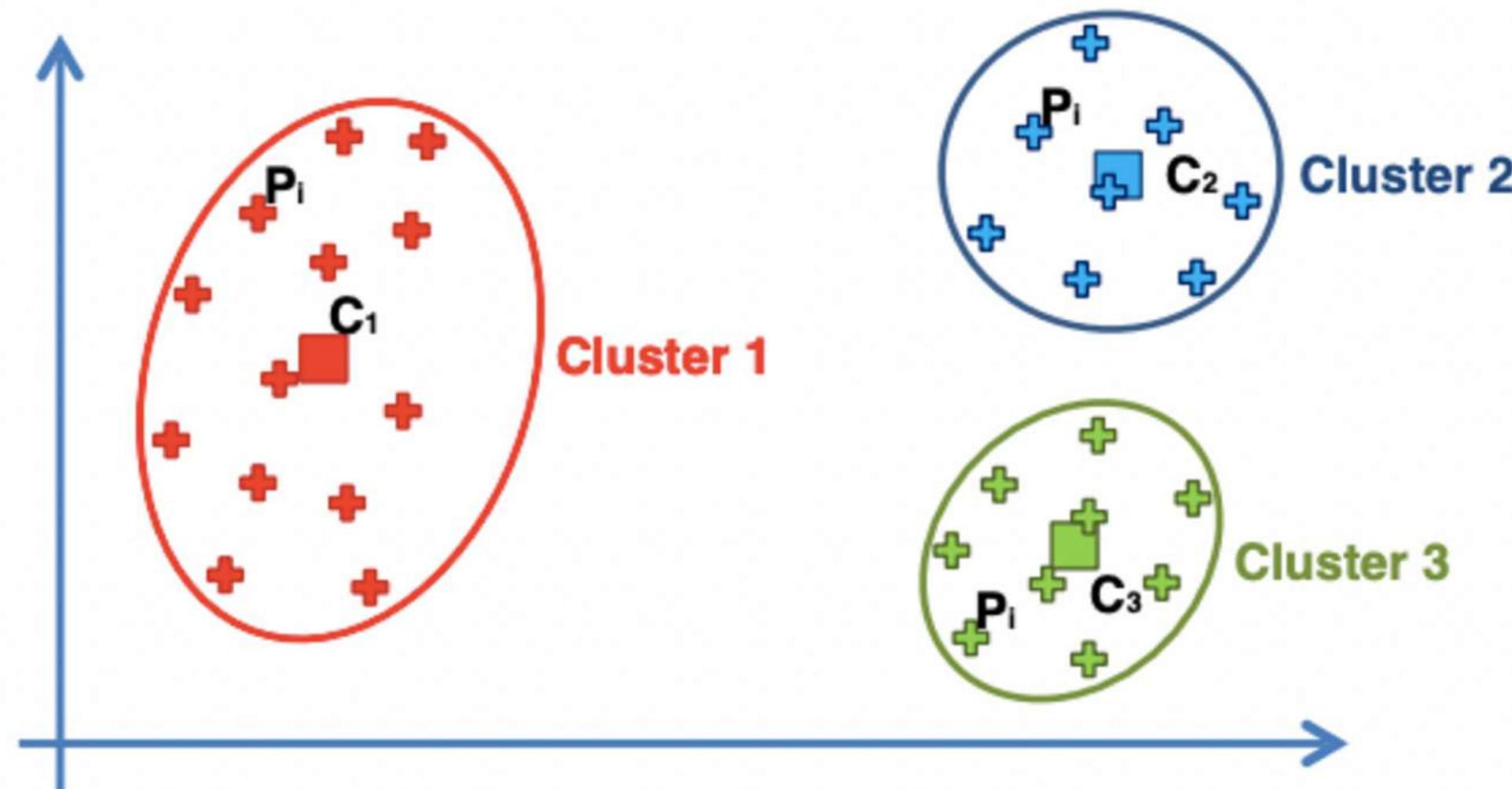
Entonces, el primer sumatorio calcula la distancia de los puntos del primer cluster con respecto a su centro geométrico.

El segundo sumatorio hace lo propio con el cluster número dos. Cuál es la distancia entre cada punto del cluster respecto al centro geométrico del mismo.

Y hacemos lo propio con el tercero.

Entonces esa es la forma matemática que tenemos de definir esta métrica.

K-MEANS



$$WCSS = \sum_{P_i \in \text{Cluster 1}} d(P_i, C_1)^2 + \sum_{P_i \in \text{Cluster 2}} d(P_i, C_2)^2 + \sum_{P_i \in \text{Cluster 3}} d(P_i, C_3)^2$$

K-MEANS

Entonces, rebobinemos un poquito en nuestro problema y veamos cómo esa métrica va a ir cambiando a medida que aumenta el número de clusters del algoritmo.

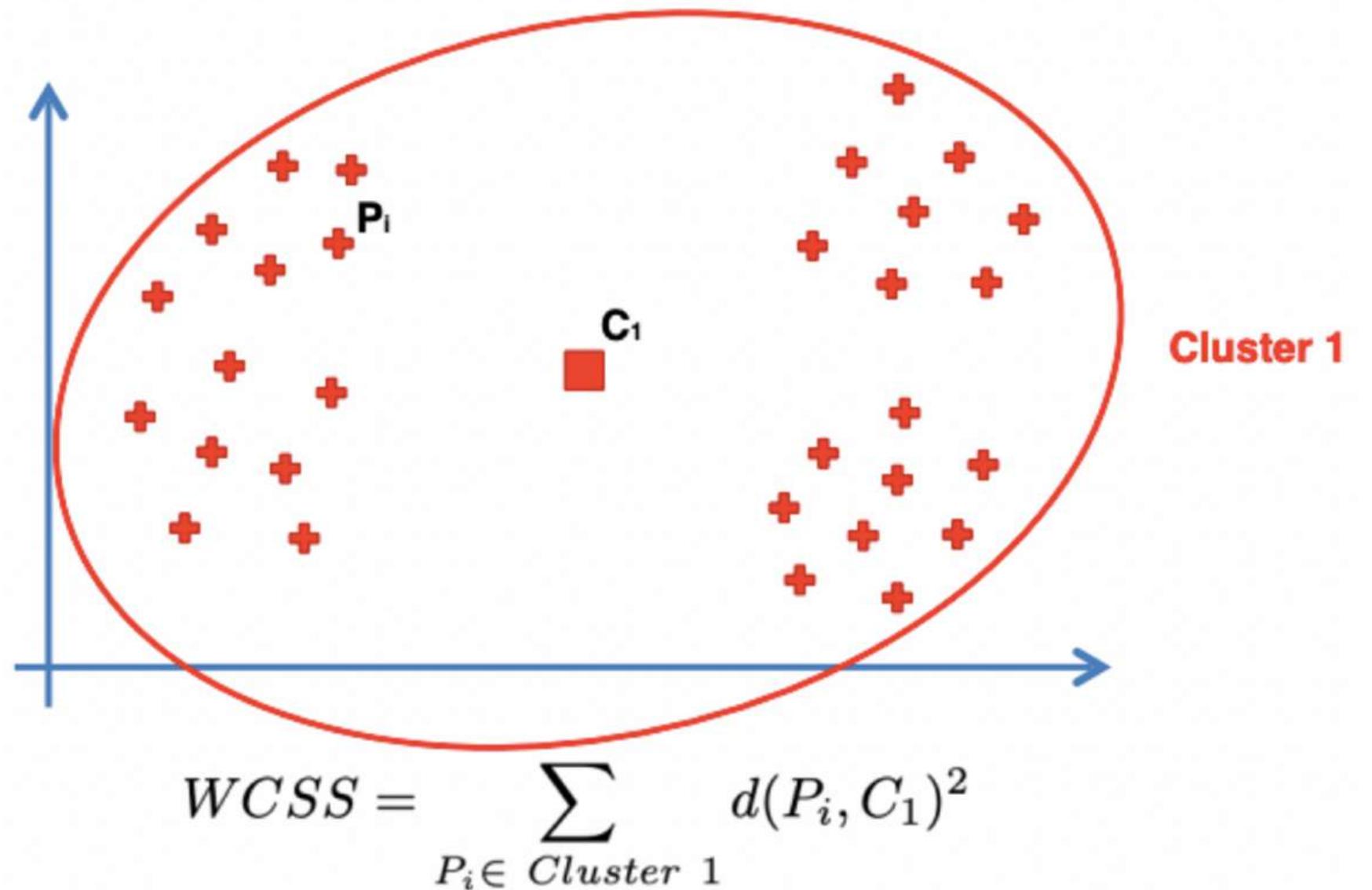
En el primer caso, imaginemos que existe un cluster enorme con una sola división. El centro geométrico del baricentro quedaría justo en el centro de ese punto.

Entonces, aquí tendríamos un valor de suma de los cuadrados, de las distancias, de los puntos al centro del cluster.

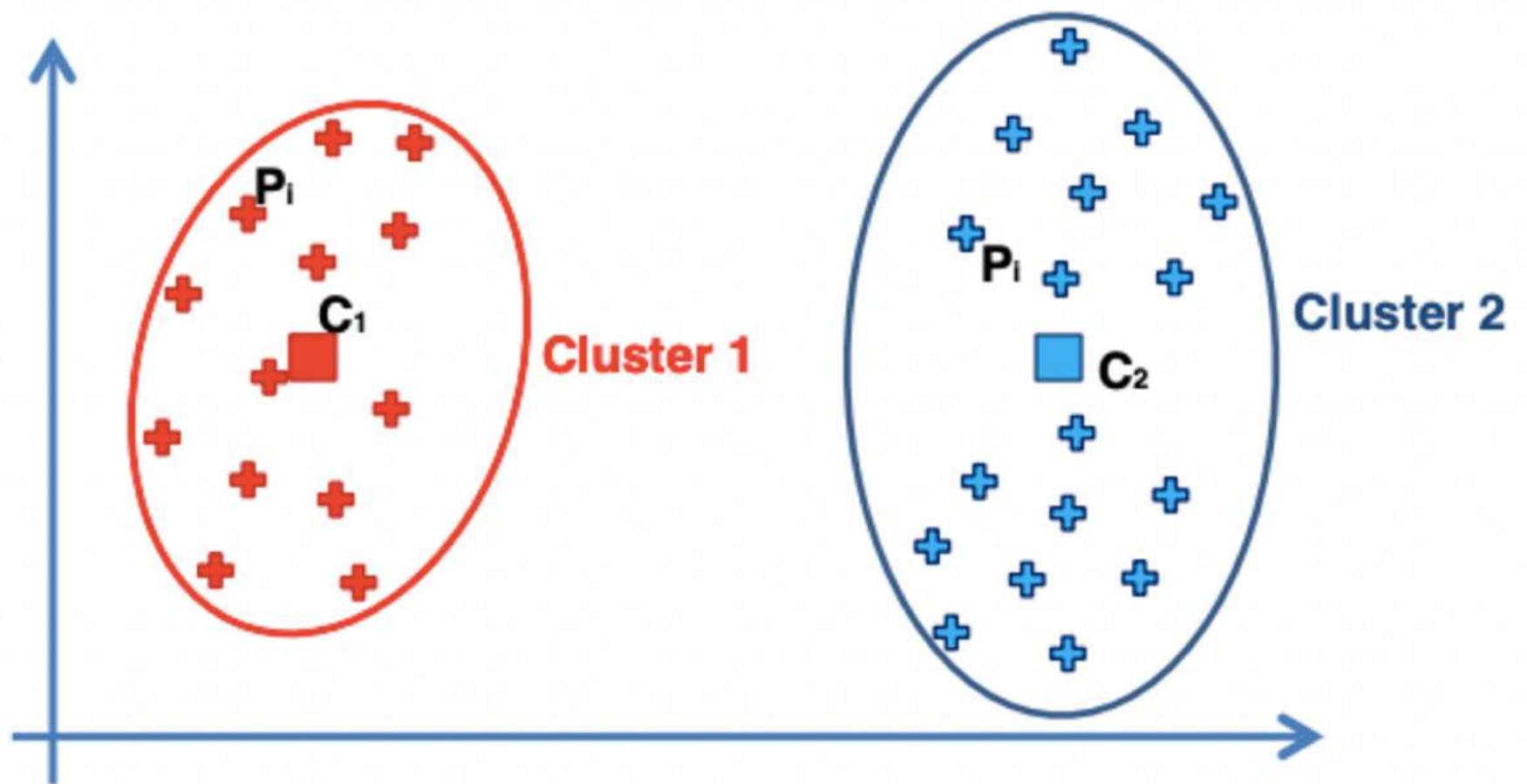
K-MEANS

Calculamos todas las distancias de cada uno de los puntos a ese centro de color rojo y parece ser que dará un valor bastante grande porque hay muchos puntos bastante alejados de ese centro del cluster al elevar al cuadrado en un número todavía bastante más grande.

Qué tan lejos, cuánta distancia global habría que recorrer para atraer cada uno de los puntos a su centro de gravedad? De momento nos quedaríamos con la idea de que sería un valor bastante grande.



K-MEANS



$$WCSS = \sum_{P_i \in \text{Cluster 1}} d(P_i, C_1)^2 + \sum_{P_i \in \text{Cluster 2}} d(P_i, C_2)^2$$

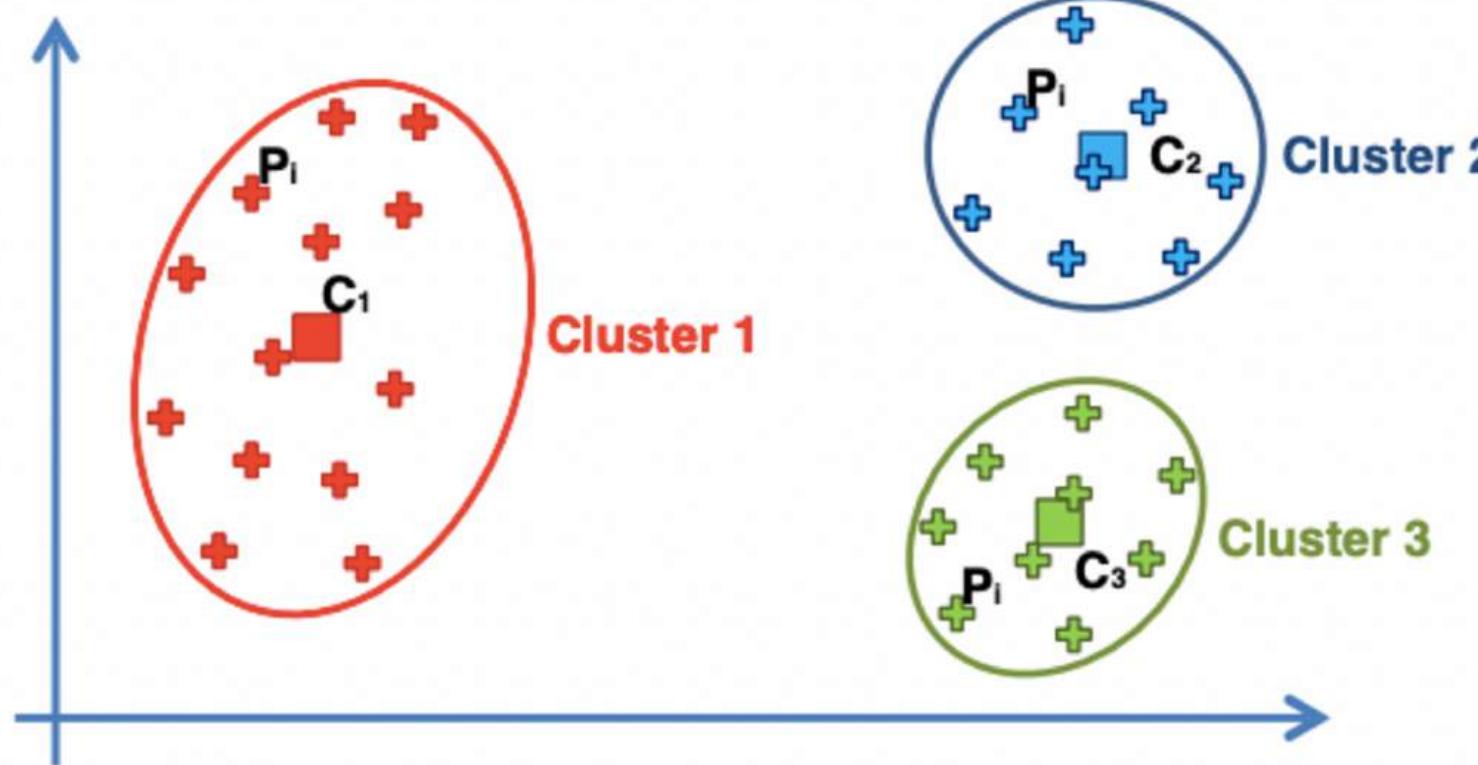
Seguiríamos con dos clusters. Uno intuitivamente entendería que el resultado sería algo similar a este.

Entonces cada uno de esos puntos ya no tiene que recorrer tanta distancia para llegar al centro del mismo.

Esto es genial porque lo que hemos hecho es disminuir la suma de los cuadrados de cada punto al centro del cluster, incrementando el valor de 2 a 3.

Entonces, la distancia total, la suma de cuadrados de distancias de cada punto al centro del cluster respectivo habría disminuido significativamente.

K-MEANS



$$WCSS = \sum_{P_i \in \text{Cluster 1}} d(P_i, C_1)^2 + \sum_{P_i \in \text{Cluster 2}} d(P_i, C_2)^2 + \sum_{P_i \in \text{Cluster 3}} d(P_i, C_3)^2$$

Entonces esto nos diría que la distancia total, la WCSS disminuye, en este caso al aumentar en uno la cantidad de clusters.

De modo que ahora todos los puntos del cluster de arriba están mucho más cerca desde su propio baricentro.

K-MEANS

Esto es genial porque en general lo que hacemos es ir juntando los puntos cuanto más posible a los baricentros, de modo que esa distancia va haciéndose cada vez más pequeña.

A medida que vamos aumentando el número de clusters, la suma de distancias al cuadrado va disminuyendo y la idea es ¿cuánto más seremos capaces de disminuirlo?

Bueno, supongamos que aumentamos la cantidad de clusters a 4, 5, 6 o 7, y así sucesivamente. Podríamos tener tantos clusters como puntos tuviera el dataset original u observaciones en nuestro conjunto de datos. En ese caso, la suma de las distancias de cada punto a su cluster llegaría a ser cero.

Cada uno de los puntos llegaría a ser su propio cluster y después de haber completado todos esos números de clusters, ya no se puede hacer disminuir más.

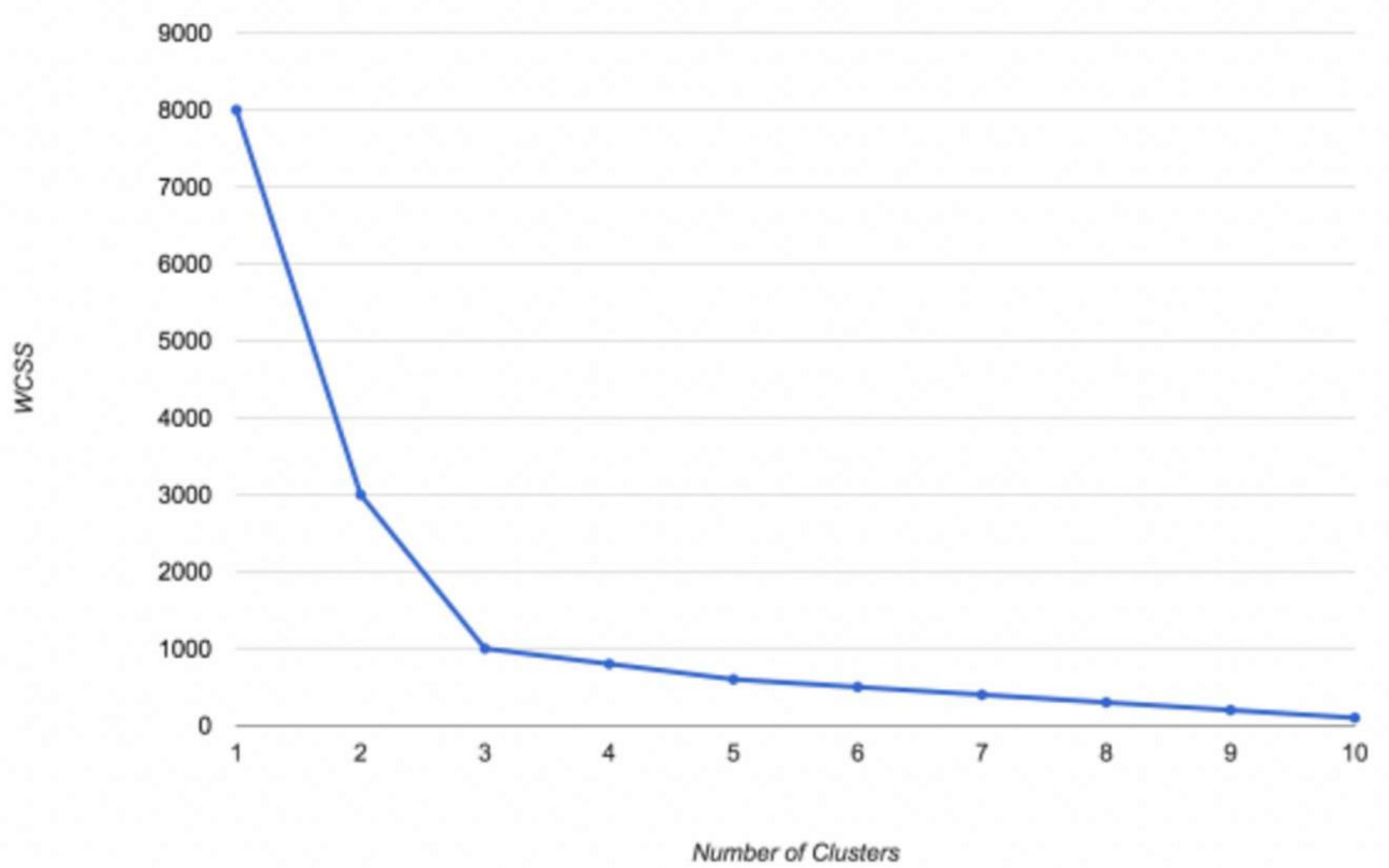
K-MEANS

De acuerdo a la evaluación de la WCSS daría evidentemente cero. Si cuadramos todo esto, vemos que la suma de los cuadrados tienden a disminuir sustancialmente cuando vamos añadiendo más y más centros de clusters.

Ese decremento en la métrica no será un decremento constante, sino que al principio sí que habrá decrecido rápidamente, habrá una mejora rápida cuando aumentas el número de clusters.

Pero habrá un punto que si ahora añadiera un 4o, 5o, o 6o cluster, los puntos están ya tan juntos que el resultado no mejoraría tan considerablemente. Se trata de buscar el mejor equilibrio, la mejor bondad de ajuste, el valor óptimo que podemos añadir de clusters de modo que la suma de los cuadrados de los puntos con respecto a los centros disminuye, y esa disminución es palpable de que realmente, de un punto al siguiente hay una caída importante.

K-MEANS



Aquí lo que hemos hecho es según el número de clusters, hemos calculado la métrica de la suma de los cuadrados, de las distancias, del punto al centro del cluster respectivo, y hemos repetido el algoritmo para un cluster para dos, para tres, para cuatro.

Podéis ver como al comienzo esa métrica empieza con un valor bastante grande, cerca de 8000 puntos.

A medida que aumenta el número de clusters, el número K, esa métrica va disminuyendo hasta que se acabaría acercando a cero.

K-MEANS

El primer salto pasa de un error, de 8000 puntos a 3000. Eso es un cambio masivo de 5000 puntos hacia abajo. Por tanto, bajamos cinco unidades en decremento.

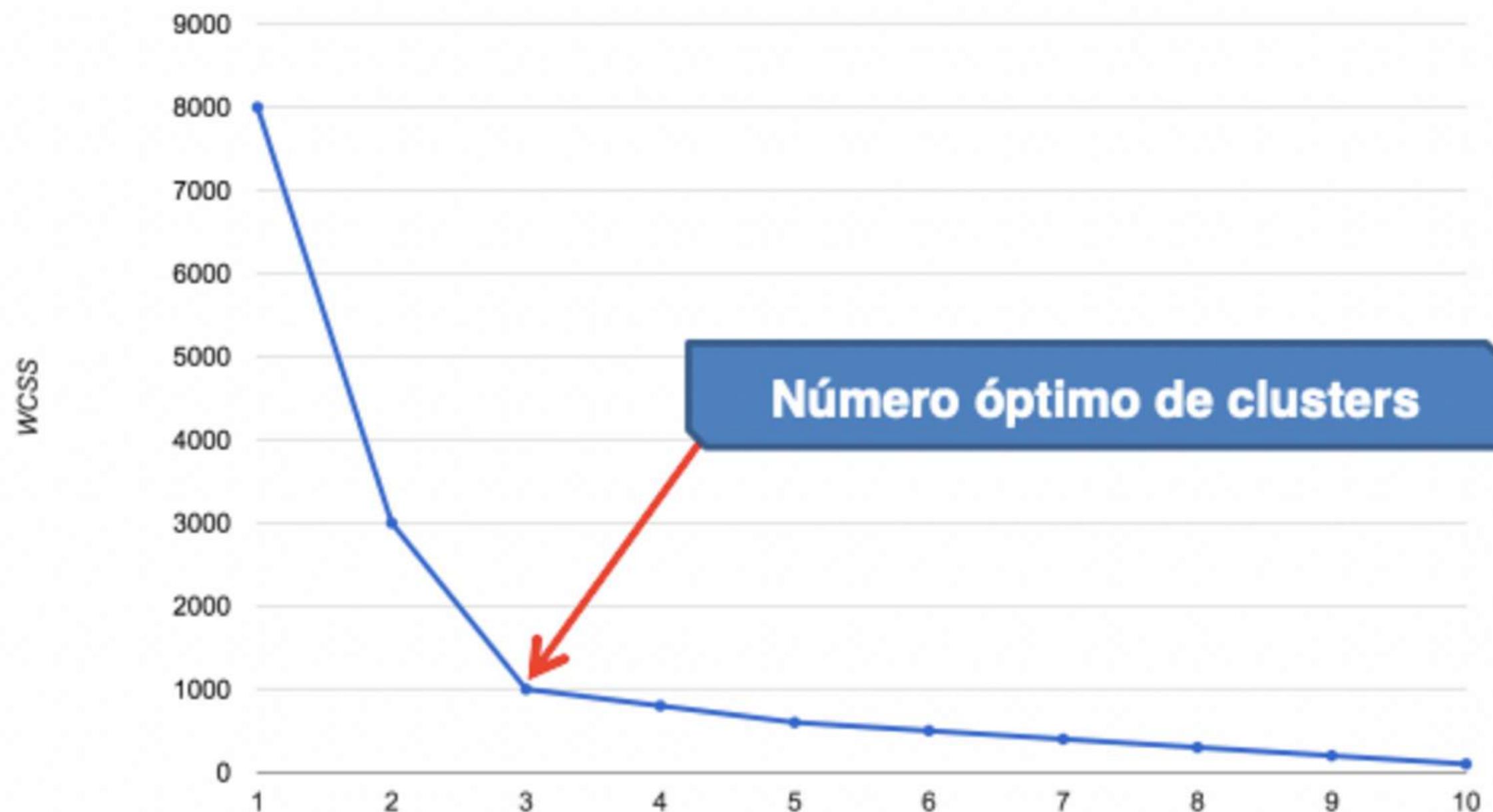
Luego de 3000 baja aproximadamente a 1000, de modo que aumentando el número de clusters de 2 a 3, hay un cambio muy significativo, bastante grande de uno al siguiente. Cuando pasamos de 3 a 4 clusters, pasará de 1000 puntos de error apenas a 800, luego a 600, 500, 400, iría bajando.

Entonces observamos que las dos primeras mejoras son bastante importantes. Hay dos cambios bastante gordos, 1 a 2 y de 2 a 3, pero a partir de 3 a 4, de 4 a 5, los cambios apenas son sustanciales, apenas se consigue una mejora en lo que a distancia se refiere.

Estos últimos no nos aporta una mejora en cuanto a disminuir la suma de los cuadrados de las distancias de los puntos respecto a los centros, y se llama la técnica del codo.

K-MEANS

La técnica del codo





DESCRIPCIÓN DEL PROBLEMA

REGRESIÓN LOGÍSTICA

Se trata de un centro comercial, un gran centro comercial de la ciudad que contiene información de los clientes. Se suscriben estos clientes a una tarjeta de membresía. Cuando se suscriben a esta tarjeta del centro comercial, se les pide información.

Entonces tenemos el género si se trata de hombres o mujeres, la edad, el sueldo anual, el ingreso anual en miles de dólares.

El centro comercial tiene un historial de las compras llevadas a cabo y aquí es donde sale la última columna, donde básicamente tienen un score, una puntuación del gasto de cada cuánto se gasta ese usuario en el centro comercial. Nos interesa que tiene valores de 1 a 100.

REGRESIÓN LOGÍSTICA

Entonces, como científico de datos, lo que queremos es segmentar a los clientes en una serie de grupos diferentes y recopilar información acerca de cuánto es cómo se deben segmentar esos clientes en base a sus ingresos anuales.

Dado que el propio centro comercial no tiene idea de cómo segmentar a esos clientes ni de cuántos segmentos va a tener.

Ahí es donde entrará en juego un problema de agrupación, un problema de clustering, porque no sabemos a priori las categorías que van a surgir del estudio de nuestro problema.



ALGORITMO

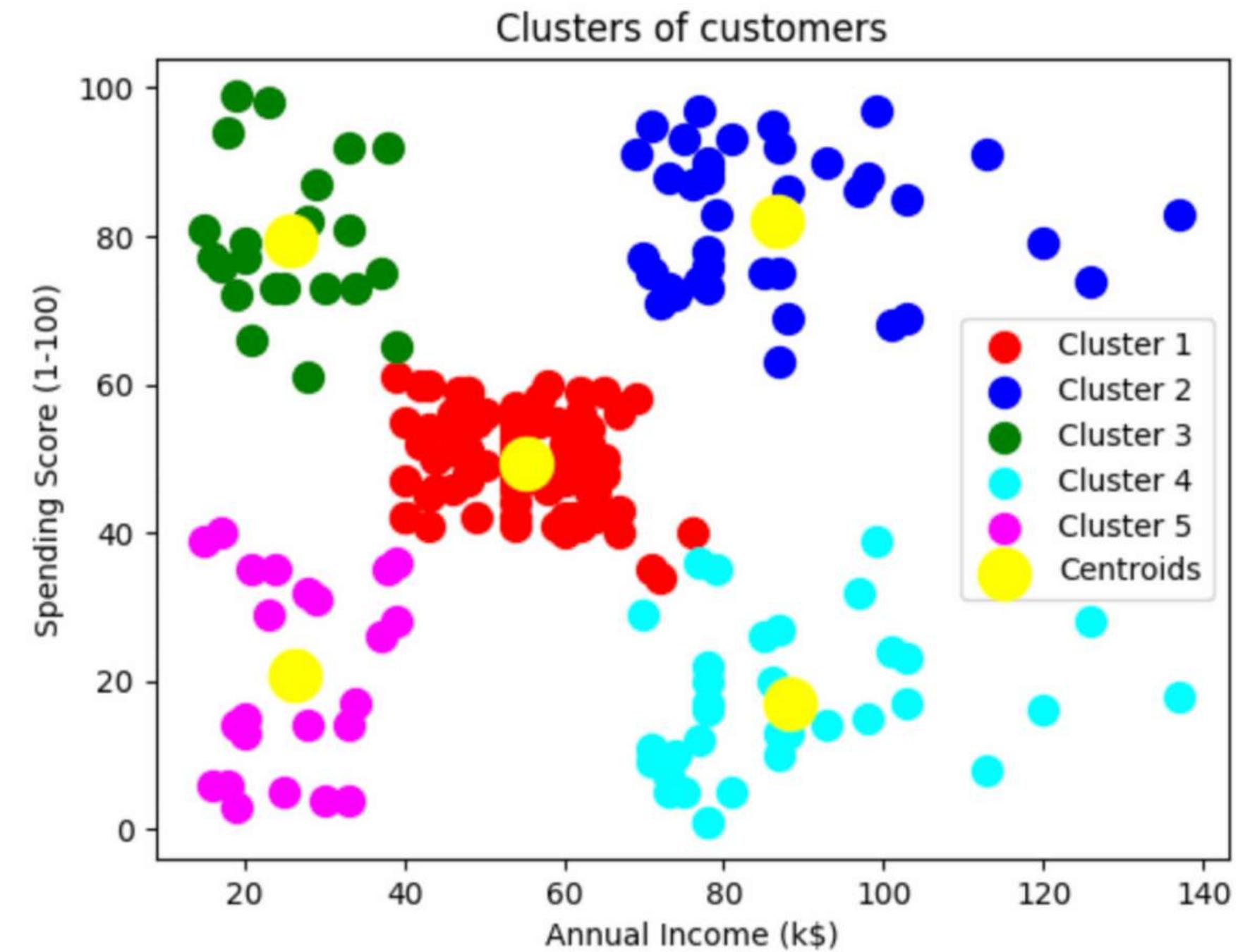
RETO

Reto:

Comencemos con:

- Active su ambiente local creado con miniconda en su equipo o su código de COLAB
- Abra el archivo [08_K_Means_Clustering.ipynb](#)
- Ejecute el código y analícelo junto al instructor

Resultado Esperado:





EVALUACIÓN

PREGUNTAS Y RESPUESTAS

Mtro. Alfonso Gregorio Rivero Duarte

Senior Data Manager - CBRE

(+52) 5528997069

devil861109@gmail.com

<https://www.linkedin.com/in/alfonso-gregorio-rivero-duarte-139a9225/>

