Algoritmo voraz para agrupamiento

Trabajo Grupal Algoritmia

Ruben Cherif, Daniel Del Barrio, Jhonny Chicaiza, Daniel Ameztoy

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Definición del Problema

Se trata de dividir un conjunto de datos en varios subconjuntos, de tal forma que los datos que sean mas cercanos deben pertenecer al mismo subconjunto, y los datos que estén mas alejados deben pertenecer a subconjuntos diferentes

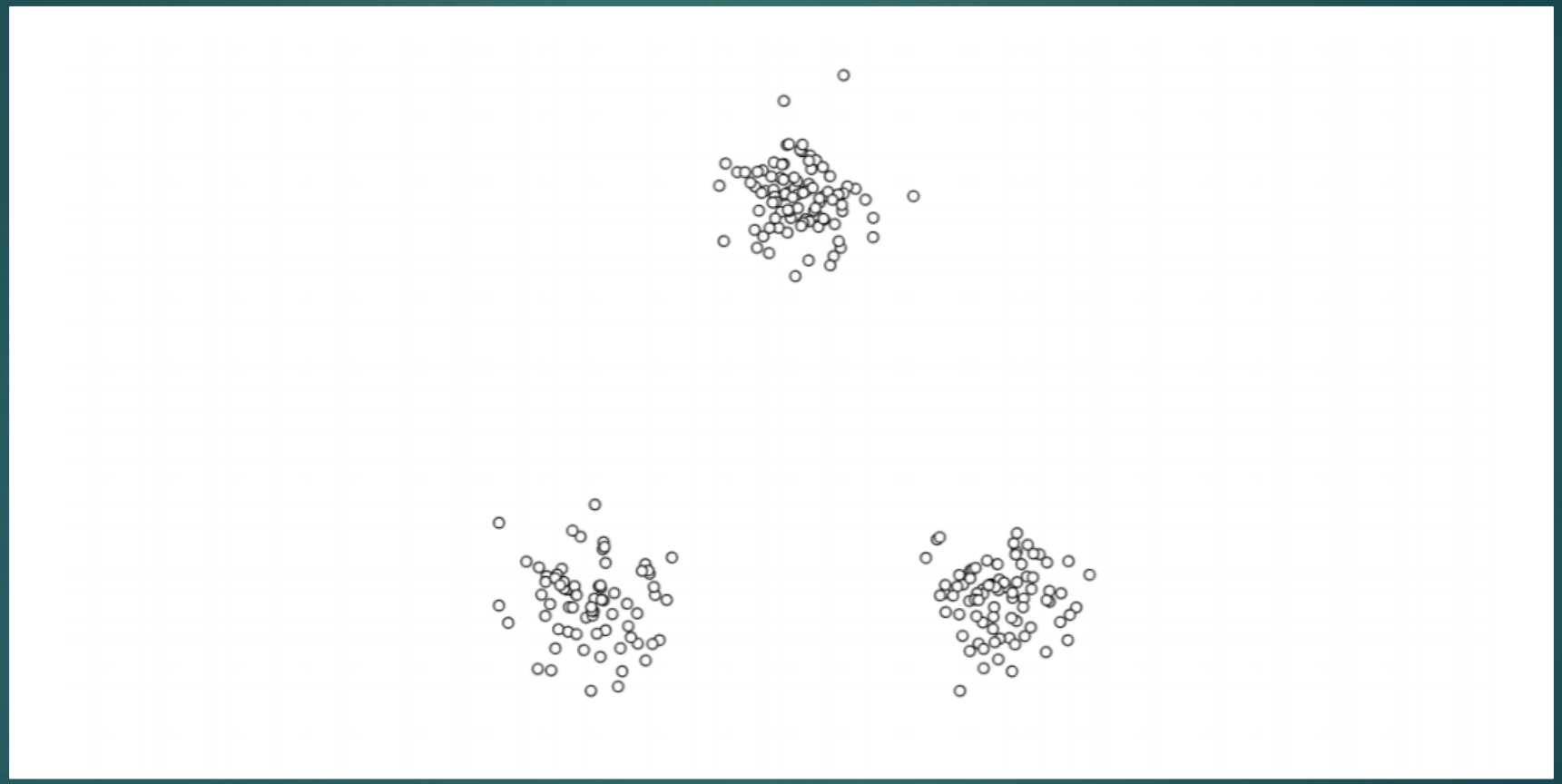
Especificación del Problema

- Sea ***M*** un espacio métrico que satisface las siguientes propiedades:
  - Reflexividad: $d(x, y) = 0 \Leftrightarrow x = y$
  - Simetría: $d(x, y) = d(y, x)$
  - Desigualdad triangular: $d(x, z) \leq d(x, y) + d(y, z)$
  - Positividad: $d(x, y) \geq 0$
- Sea ***N*** el número de elementos
- Sea ***K*** el número de grupos (***K*** ≤ ***N***)

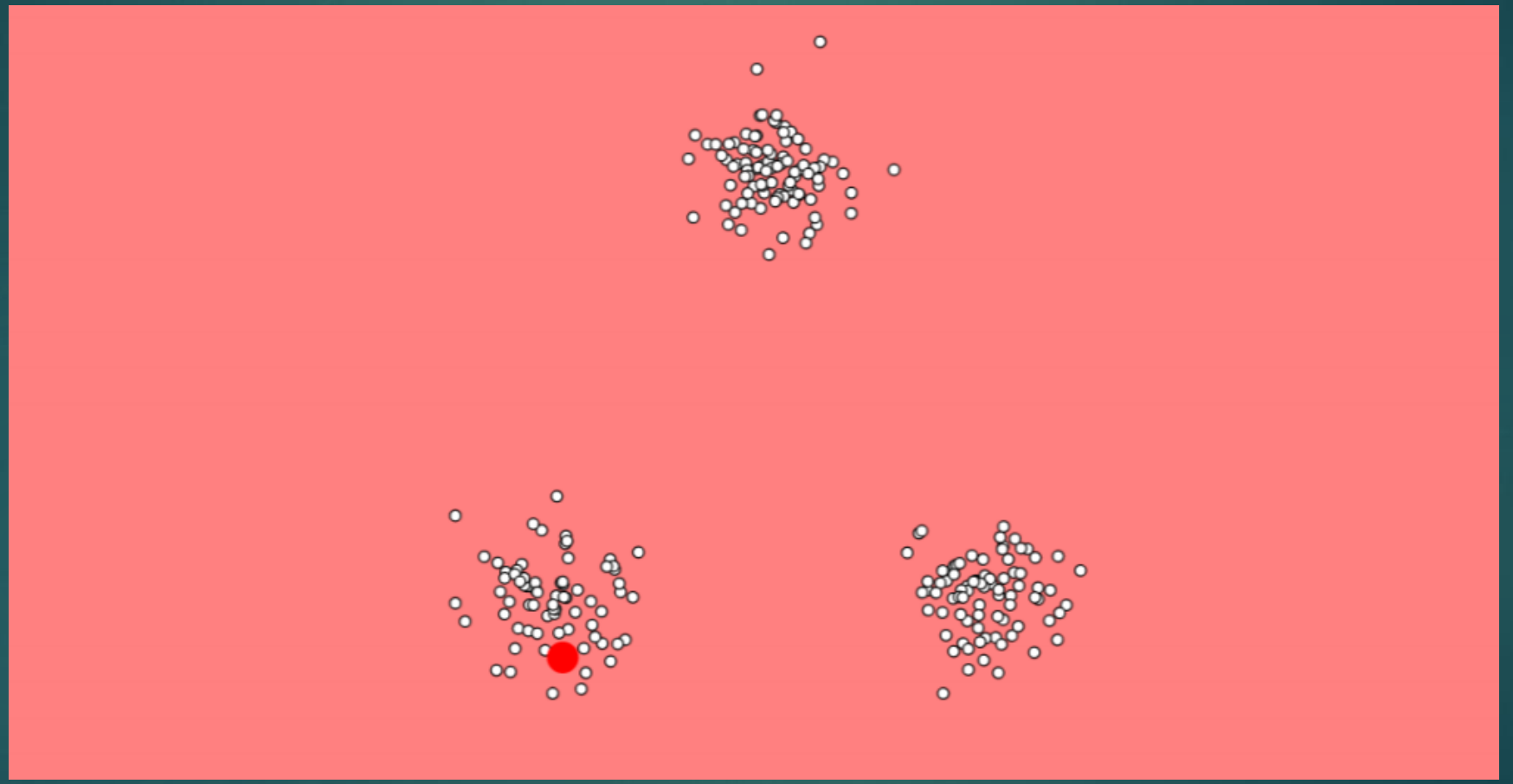
Notación Matemática Algoritmo

1. Sea C el conjunto de centros: $|C| = \emptyset$
2. Sea P el conjunto de puntos: $|P| = N \quad P = \{p_n, p_{n+1}, \dots, p_N\} \ \forall\ 0 < n \leq N$
3. $\mathrm{C} = \{\mathrm{c}_1 = p_i\} \quad |C| = 1 \quad$ i = aleatorio
4. $\mathrm{C} = \{c_1, c_2 = \max(d[c_1, p_i])\} \ \ |C| = 2 \ \forall \ 0 < i \leq |P|$
5. $\mathrm{c_k} = \max\left(\min_{\mathrm{i}}(\mathrm{d}[\mathrm{c_j}, \mathrm{p_i}])\right) \ \forall \ [\,(0 < \mathrm{i} \leq |\mathrm{P}|\,) \wedge \ (0 < \mathrm{j} < |\mathrm{C}|) \wedge (|\mathrm{C}| + 1 \leq \mathrm{k} \leq \mathrm{K})\,] \ \ |\mathrm{C}| = \mathrm{K}$
6. $C = \{c_k, c_{k+1}, \dots, c_{K-1}, c_K\} \ \forall \ (0 < k < K)$
7. $p_i \in c_x = min(d[p_i, c_j]) \ \forall \ [\,(0 < i \leq |P| = N) \wedge (0 < j < |C| = K)\,]$

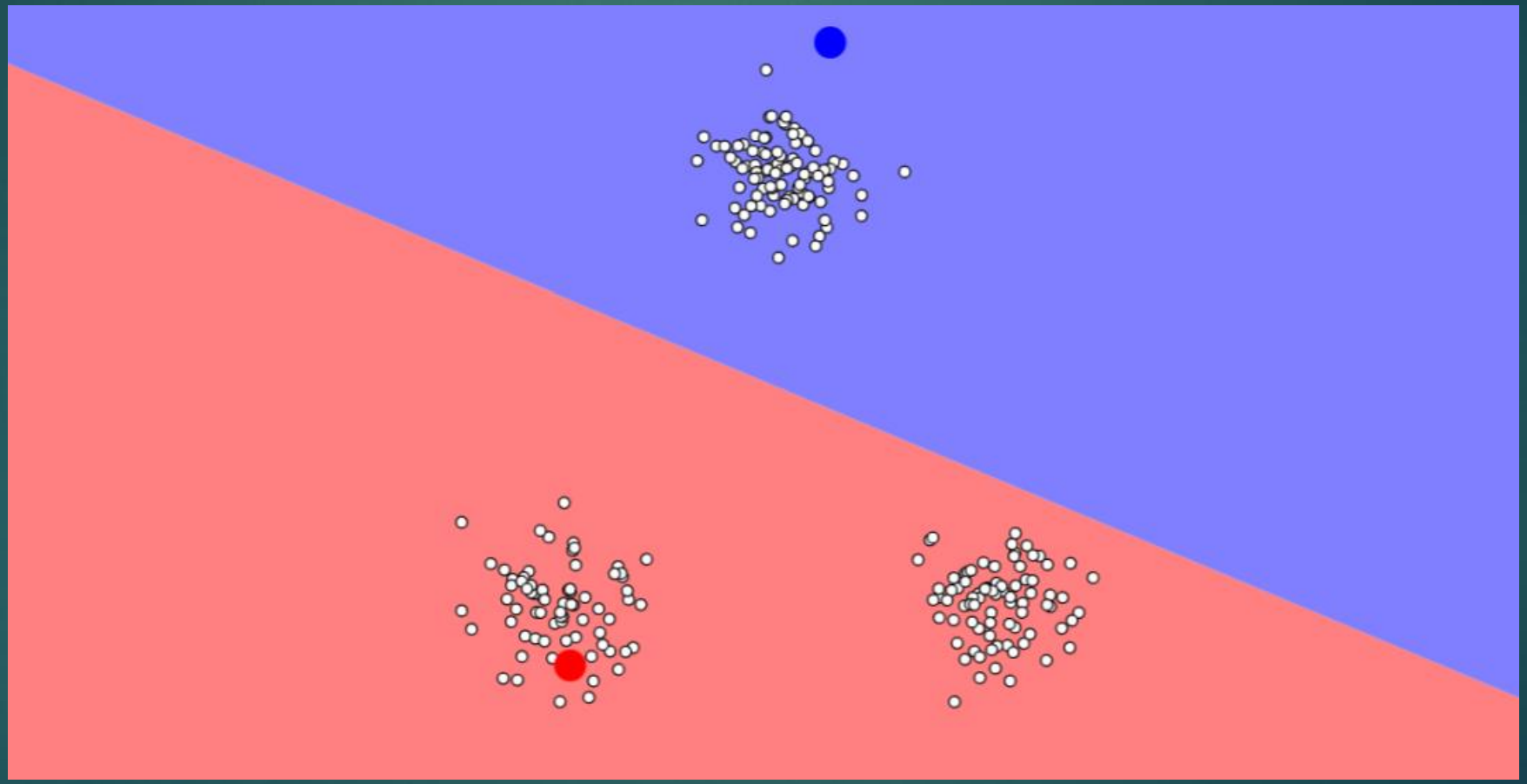
Algoritmo



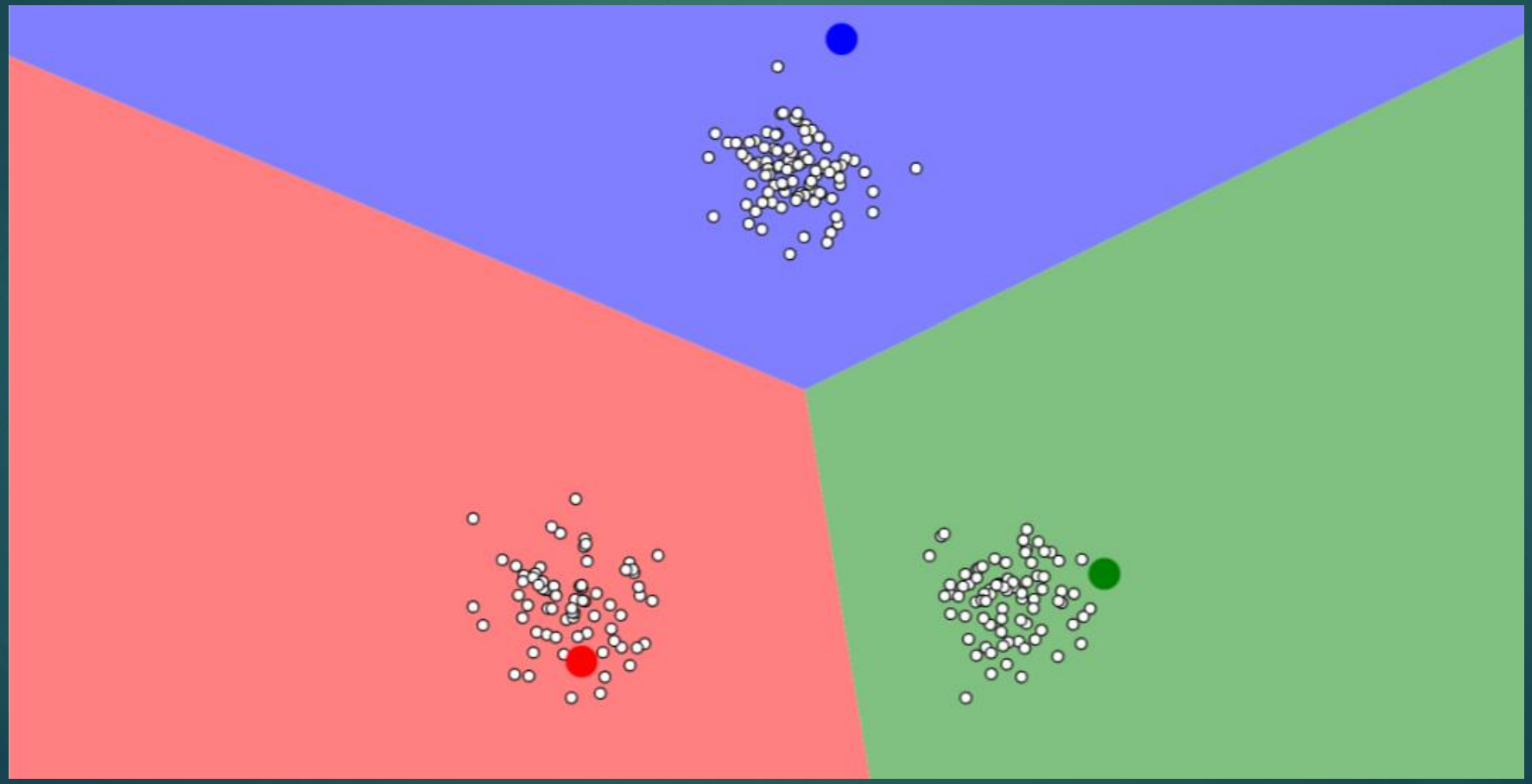
1. Elegimos un punto de forma arbitraria, y lo añadimos al conjunto de centros



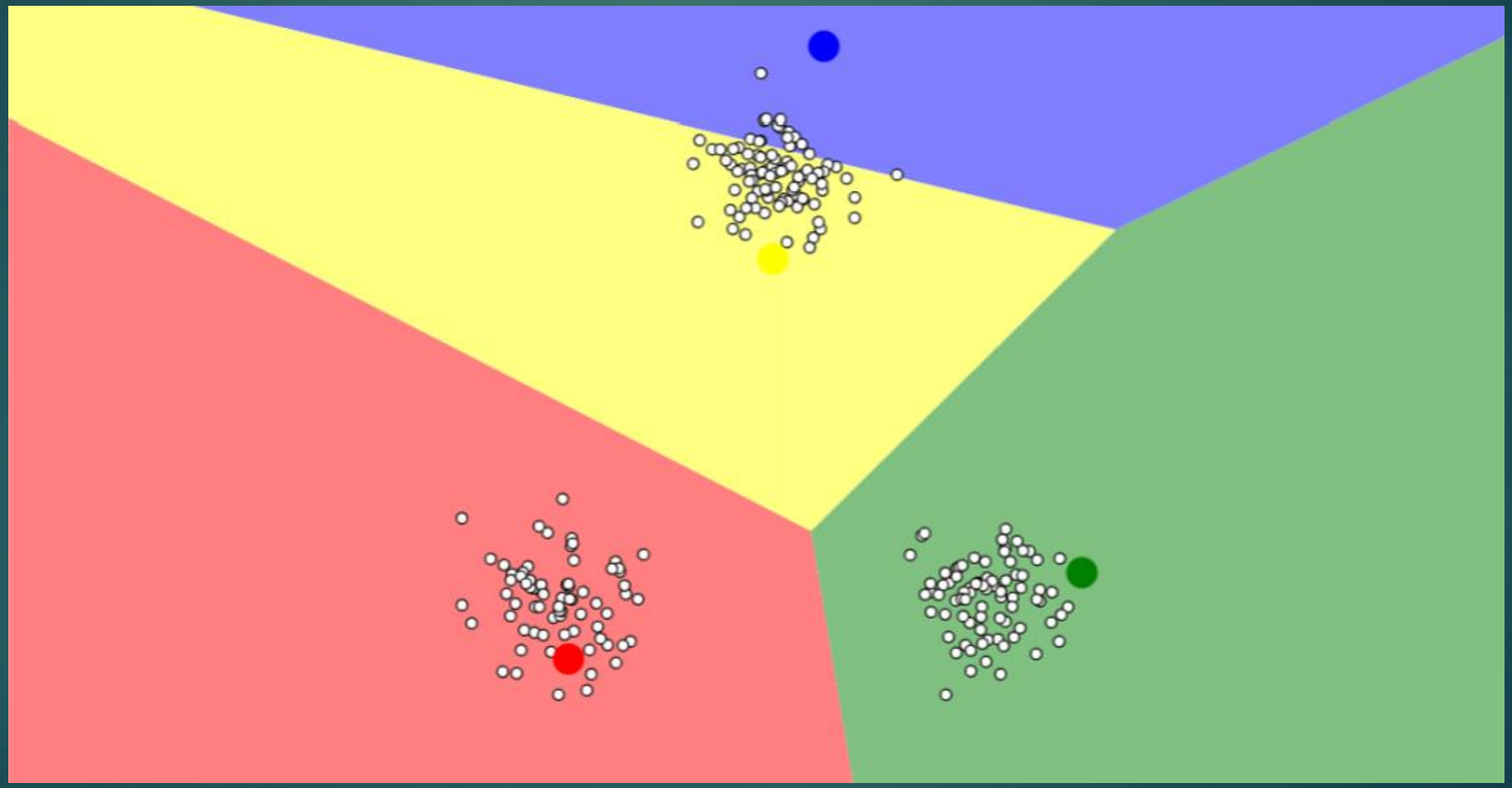
2. Calculamos el punto que se encuentre a mayor distancia del primer centro y lo añadimos al conjunto de centros



3. Calculamos las distancias entre un punto y los centros, no quedamos con la menor distancia.
Hacemos esto N veces, y de todas las mínimas distancias, nos quedamos con la máxima de ellas.
Añadimos el punto que contiene la máxima distancia y lo añadimos al conjunto de centros.
Repetimos el paso 3 hasta que el cardinal del conjunto de centros sea K.

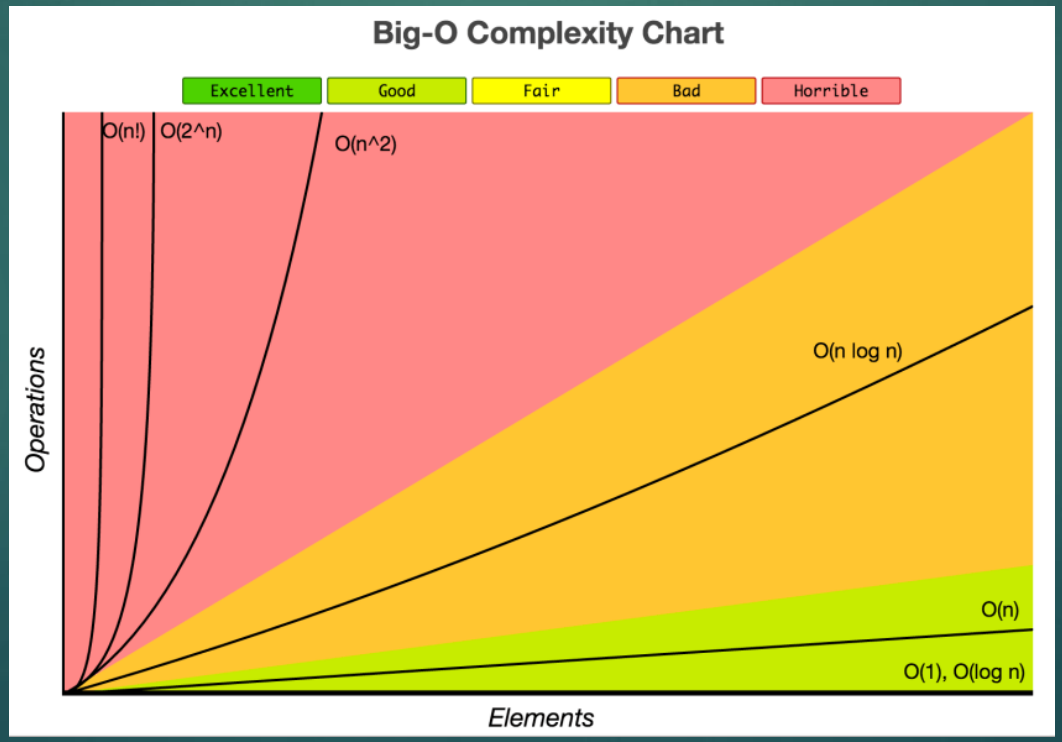


4. Una vez tengamos todos los centros, recorremos la lista calculando las distancias de cada punto hacia los centros. El punto pertenece al grupo cuya distancia sea menor



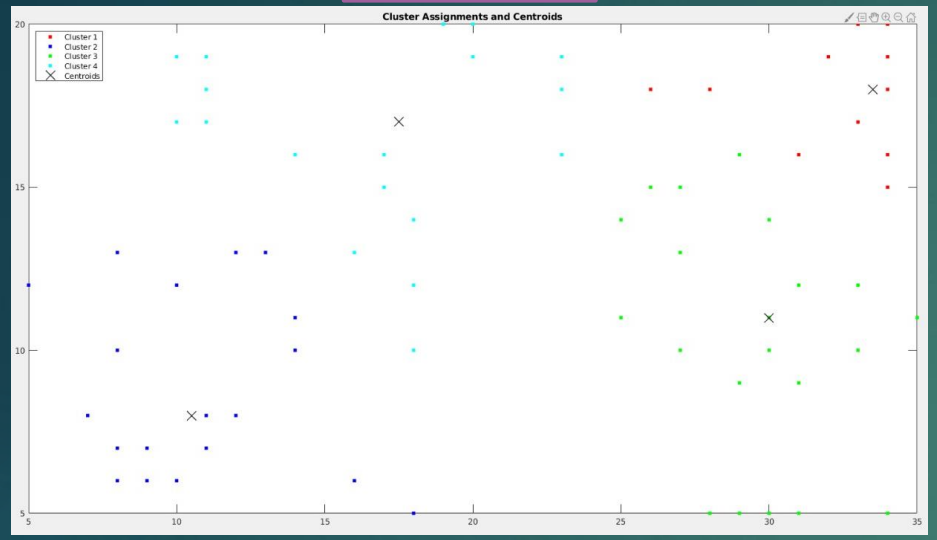
Complejidad

- Caso medio: ***O(KN)*** $\forall$ ***N*** < ***K*** | ***N***
- Peor caso: ***O(KN)*** $\forall$ ***K*** = ***N*** - 1
- Mejor Caso: ***O(N)*** $\forall$ | ***K*** = ***N*** $\vee$ ***K*** = 1 |

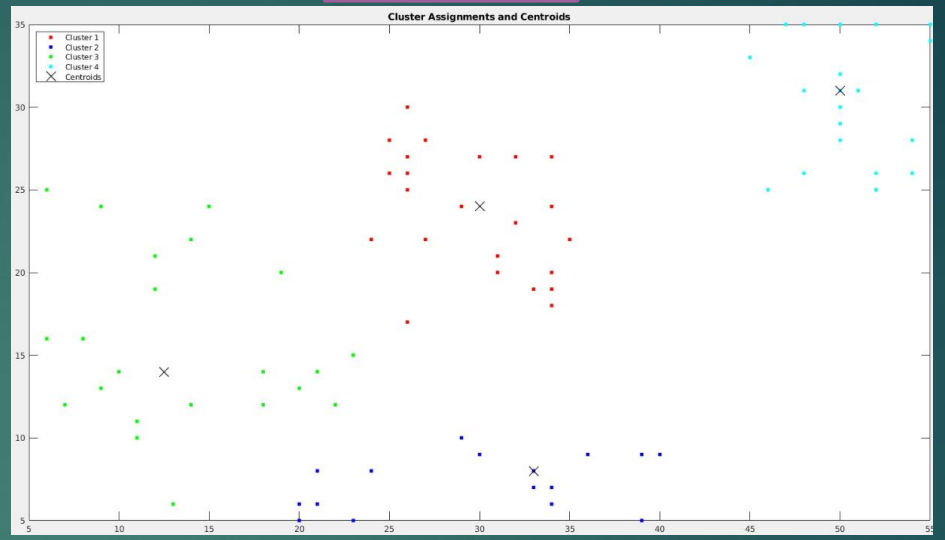


K-Means MATLAB(50000 iteraciones)

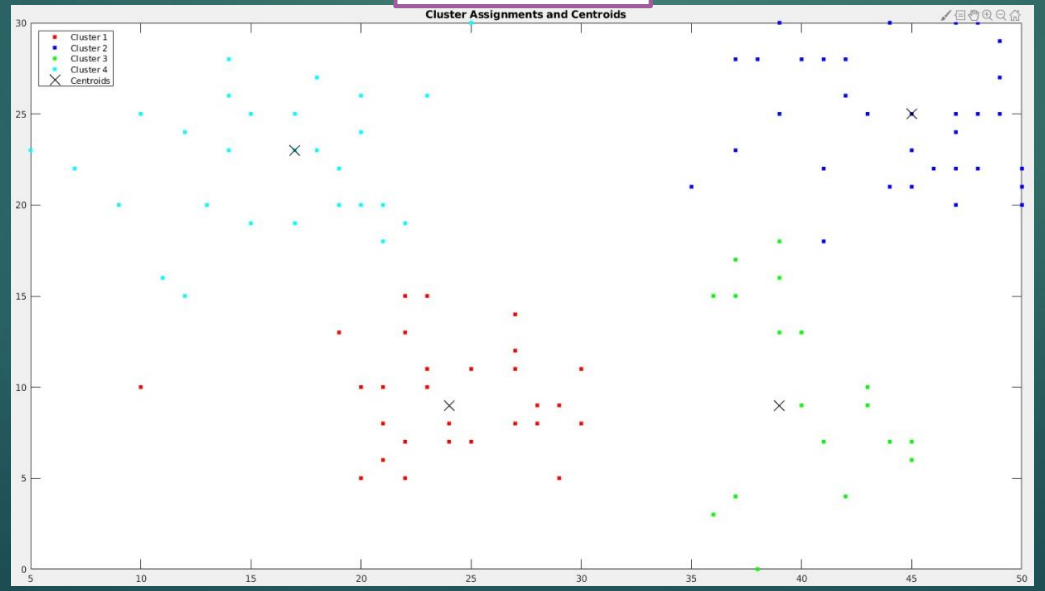
Competición 1



Competición 2



Competición 3



Estrategias Rechazada 1

1. Creamos un grafo completo ponderado uniendo cada punto con todos los demás.
2. A cada arista le asignamos como peso la distancia entre un punto y otro.
3. Buscamos el árbol recubridor de coste mínimo tantas veces como grupos deba contener(K) mediante la técnica que se quiera

Estrategia Rechazada

- Vamos a realizar una estimación de un radio: para estimarlo recorreremos la lista y calcularemos el máximo y mínimo y nos quedamos con la media
- Ahora elegimos un punto de forma aleatoria que pasara a ser nuestro primer centro
- Todos los puntos que se encuentren a mayor distancia que el radio de el centro, los eliminamos de la lista
- Volvemos a elegir otro centro, y hacemos lo mismo hasta tener K centros
- Una vez tengamos todos los centros:
  - Si no quedan puntos en la lista hemos acabado
  - Si aun quedan puntos, aumentamos el tamaño del radio en la distancia media de los puntos restantes(recorremos lista de puntos restantes), hasta finalmente no queden puntos en la lista