Minería de Datos

K-means y funciones de proximidad

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

K-means y funciones de proximidad

métrica

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

K-means y funciones de proximidad

métrica

$$\text{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

Espacio métrico

1. **Positivity**
   (a) $d(\mathbf{x}, \mathbf{x}) \geq 0$ for all $\mathbf{x}$ and $\mathbf{y}$,
   (b) $d(\mathbf{x}, \mathbf{y}) = 0$ only if $\mathbf{x} = \mathbf{y}$.

2. **Symmetry**
   $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$.

K-means y funciones de proximidad

métrica

$$\text{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

Espacio métrico

1. **Positivity**
   (a) $d(\mathbf{x}, \mathbf{x}) \geq 0$ for all $\mathbf{x}$ and $\mathbf{y}$,
   (b) $d(\mathbf{x}, \mathbf{y}) = 0$ only if $\mathbf{x} = \mathbf{y}$.

2. **Symmetry**
   $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$.

3. **Triangle Inequality**
   $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ for all points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$.

K-means y otras funciones de proximidad

¿Métrica?

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid
cosine	mean	Maximize sum of the cosine similarity of an object to its cluster centroid

K-means y otras funciones de proximidad

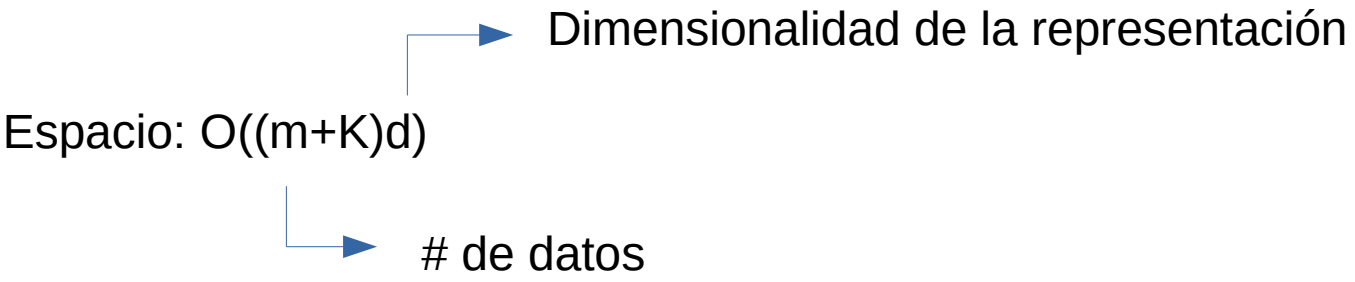
¿Métrica?

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid
cosine	mean	Maximize sum of the cosine similarity of an object to its cluster centroid

Revisar!

$$\text{Total Cohesion} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} cosine(\mathbf{x}, \mathbf{c}_i)$$

Complejidad de k-means



Complejidad de k-means

Dimensionalidad de la representación

Espacio: O((m+K)d)

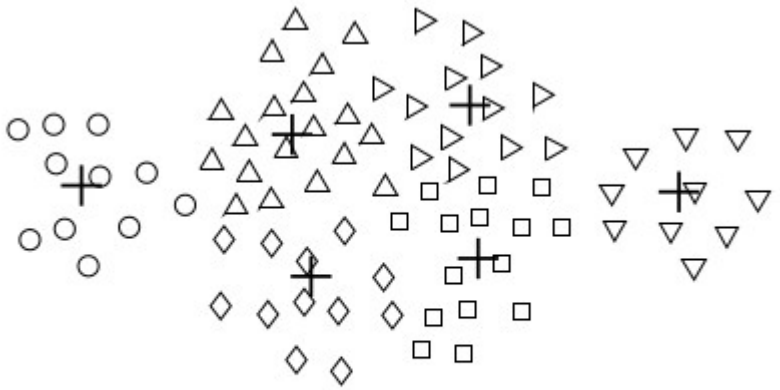
de datos

iteraciones

Tiempo: O(I*K*m*d)

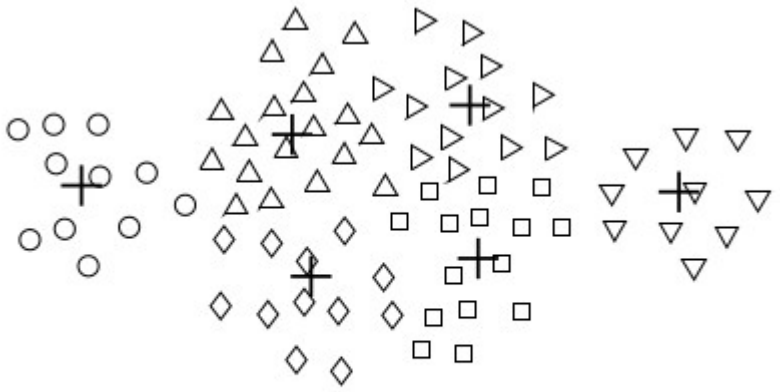
K-MEANS$(\{\vec{x}_1, \ldots, \vec{x}_m\}, K)$
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS$(\{\vec{x}_1, \ldots, \vec{x}_m\}, K)$
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{c}_k \leftarrow \vec{s}_k$
4 **while** criterio convergencia no cumplido
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $C_k \leftarrow \{\}$
7 **for** $i \leftarrow 1$ **to** m
8 **do** $k \leftarrow Min_k \mid\mid \vec{c}_k - \vec{x}_i \mid\mid$ *(encontrar el centroide mas cercano)*
9 $C_k \leftarrow C_k \cup \{\vec{x}_i\}$ *(agregar al cluster)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{c}_k \leftarrow \frac{1}{m_k} \sum_{\vec{x} \in C_k} \vec{x}$ *(recomputacion de centroides)*
12 **return** $\{C_1, \ldots, C_K\}$

Desempeño de k-means



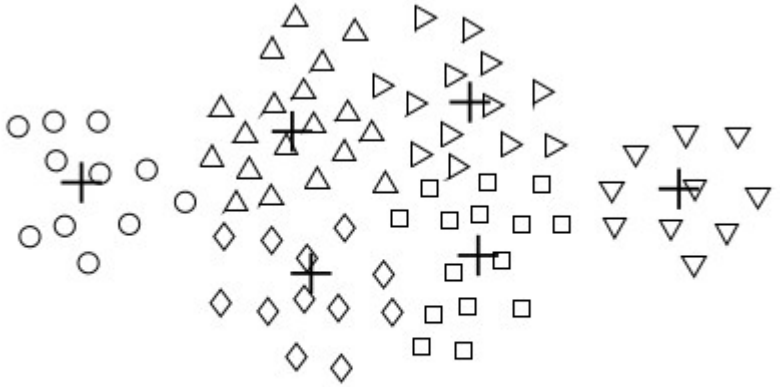
(a) Unequal sizes.

Desempeño de k-means

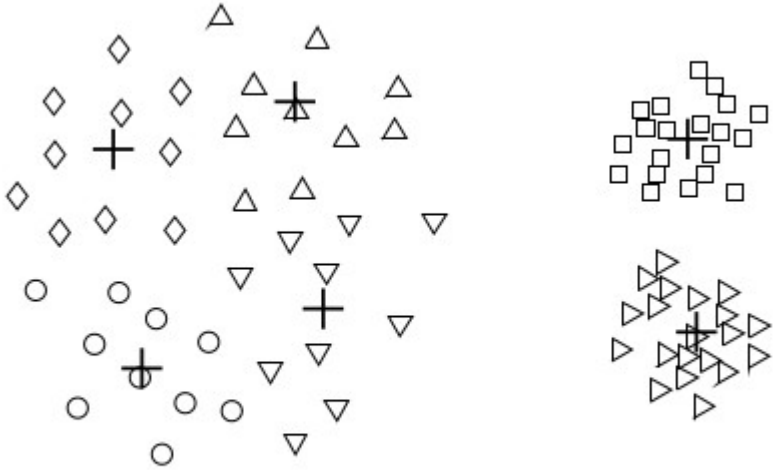


(a) Unequal sizes.

Desempeño de k-means

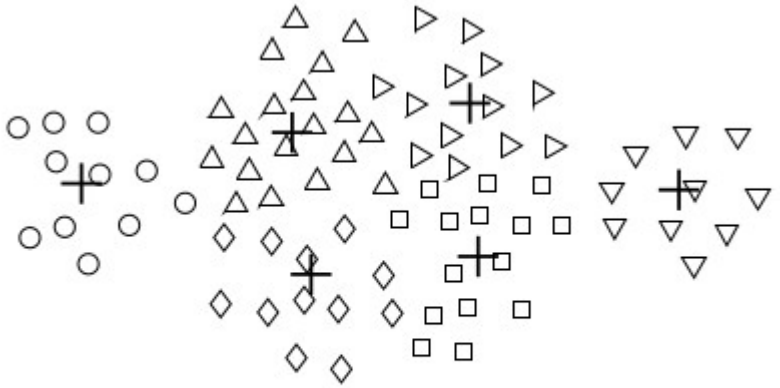


(a) Unequal sizes.

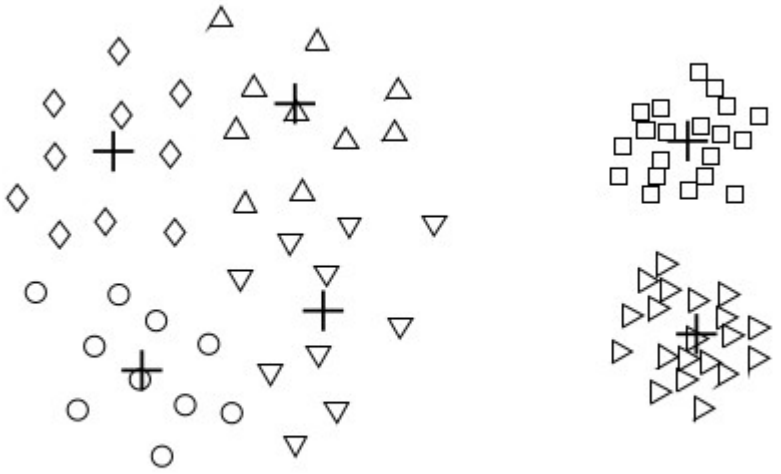


(b) Unequal densities.

Desempeño de k-means

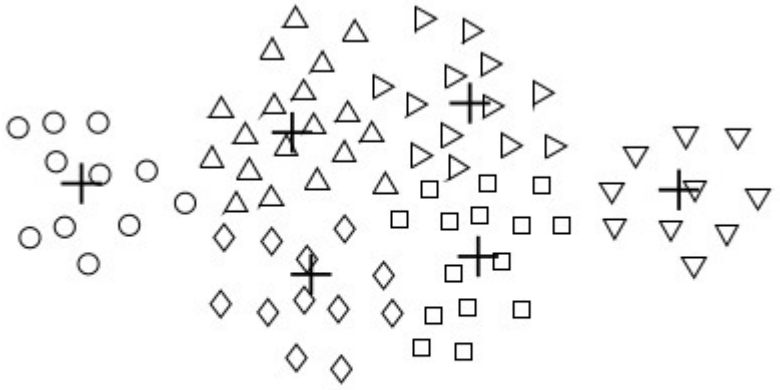


(a) Unequal sizes.

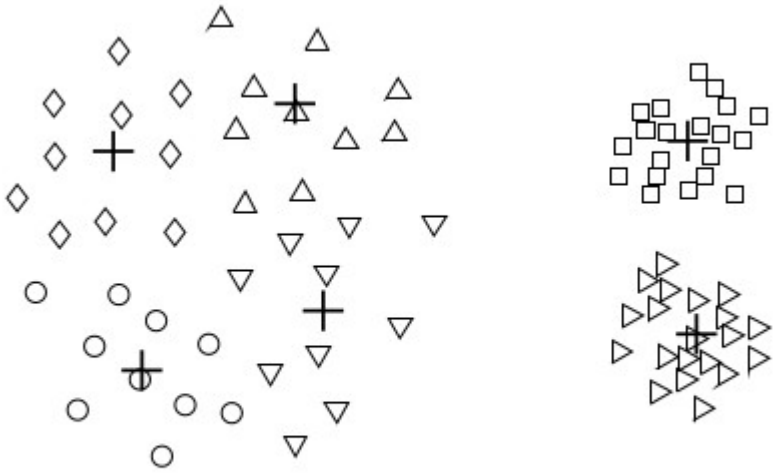


(b) Unequal densities.

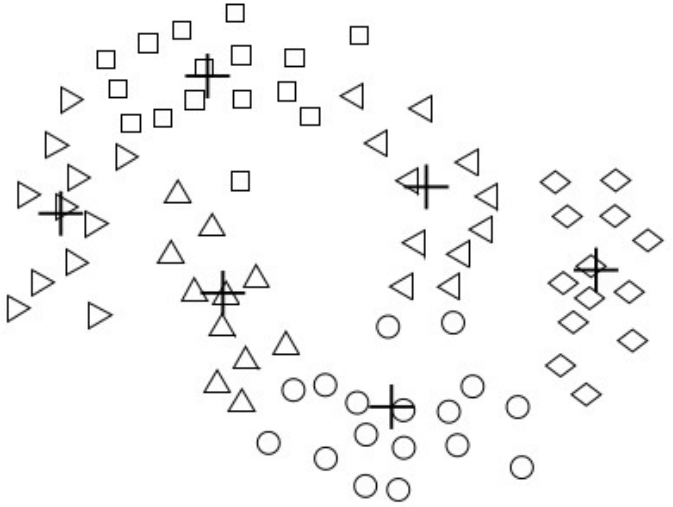
Desempeño de k-means



(a) Unequal sizes.

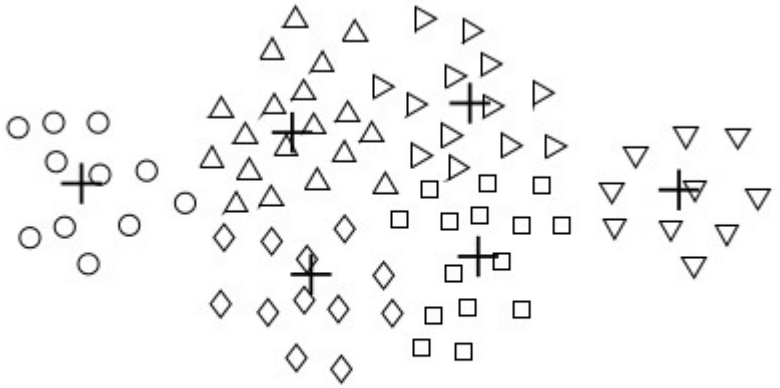


(b) Unequal densities.

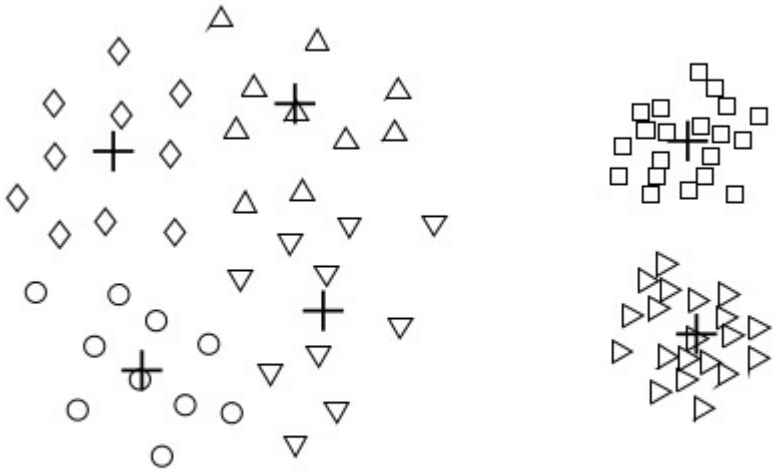


(c) Non-spherical shapes.

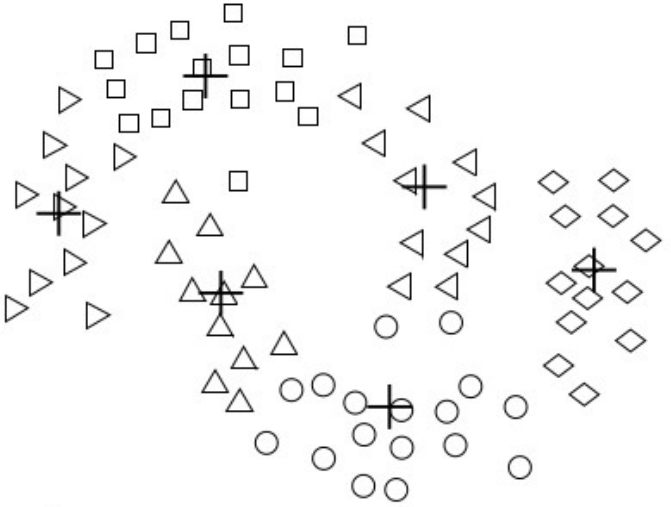
Desempeño de k-means



(a) Unequal sizes.

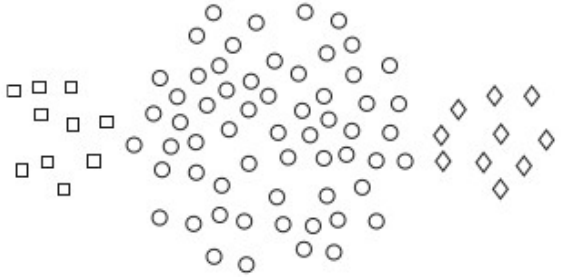


(b) Unequal densities.

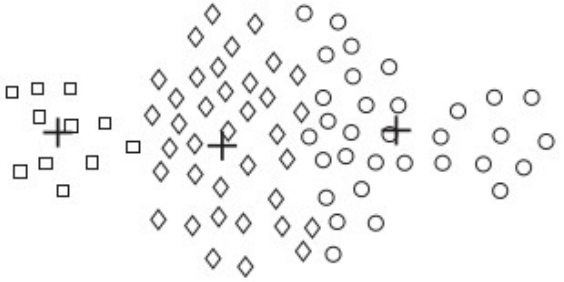


(c) Non-spherical shapes.

Desempeño de k-means

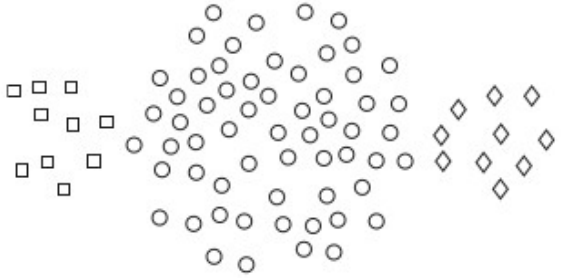


(a) Original points.

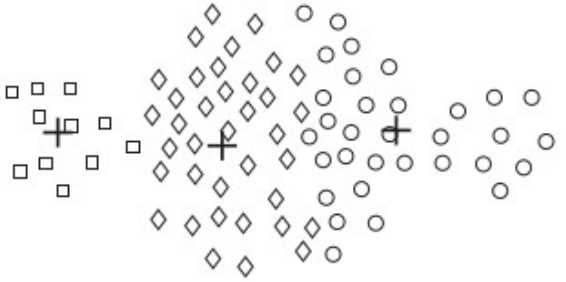


(b) Three K-means clusters.

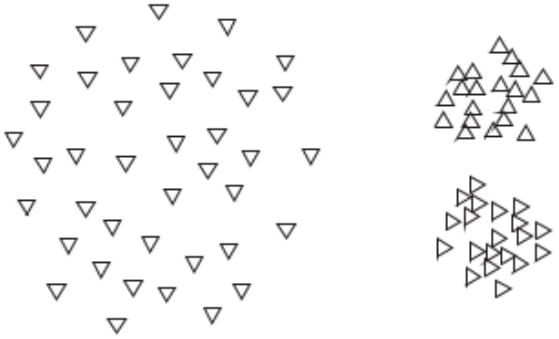
Desempeño de k-means



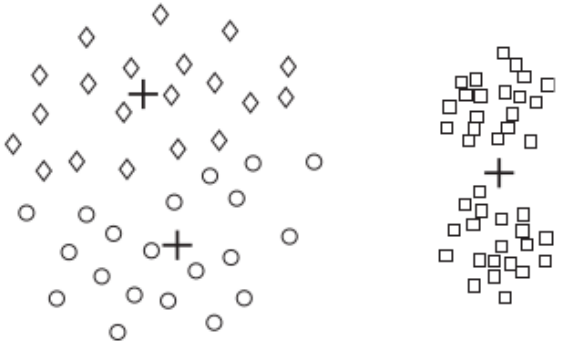
(a) Original points.



(b) Three K-means clusters.

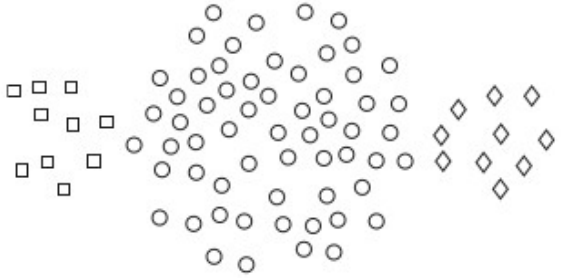


(a) Original points.

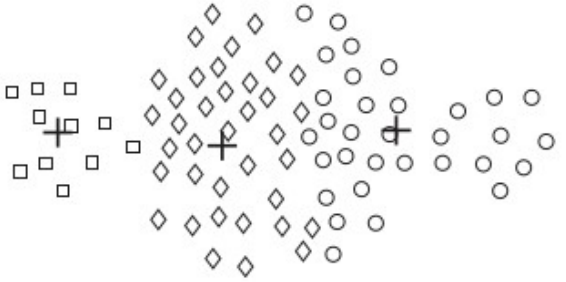


(b) Three K-means clusters.

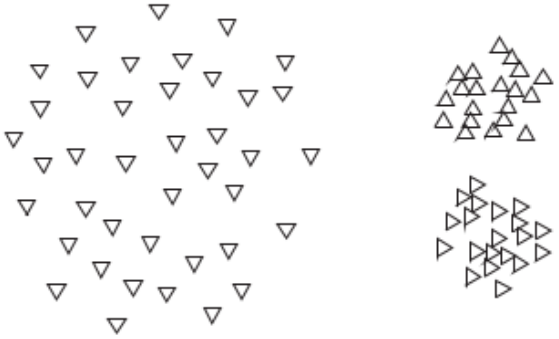
Desempeño de k-means



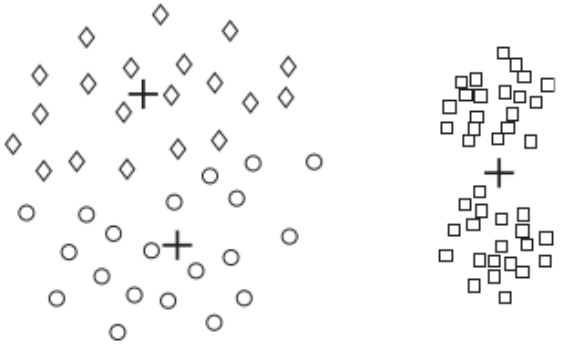
(a) Original points.



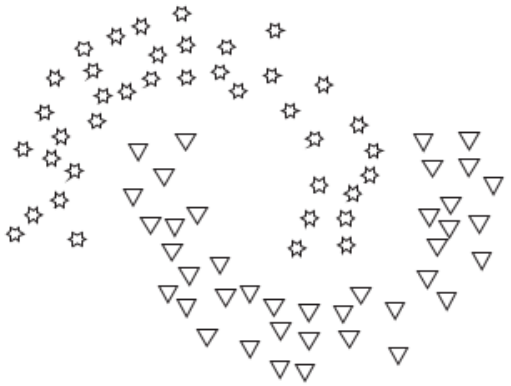
(b) Three K-means clusters.



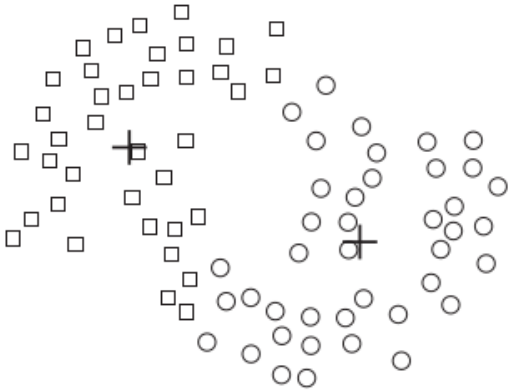
(a) Original points.



(b) Three K-means clusters.

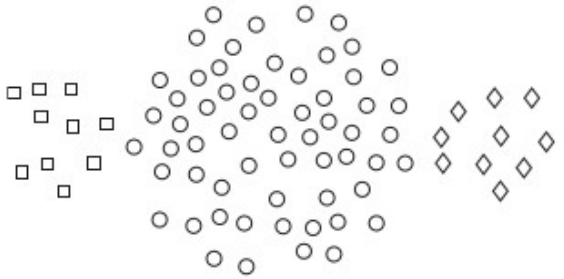


(a) Original points.

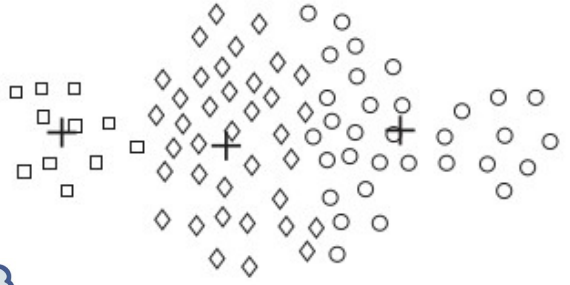


(b) Two K-means clusters.

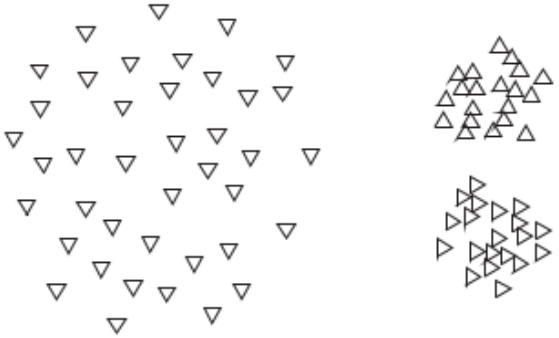
Desempeño de k-means



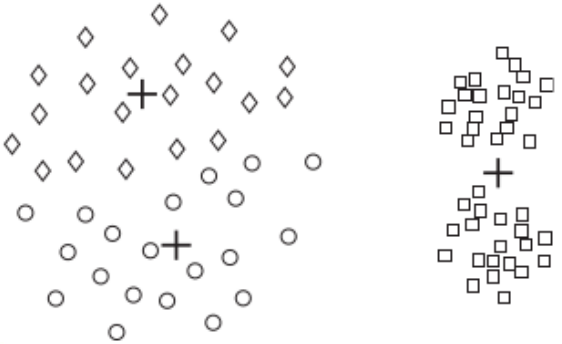
(a) Original points.



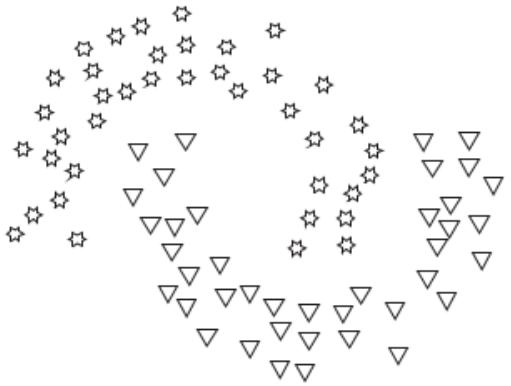
(b) Three K-means clusters.



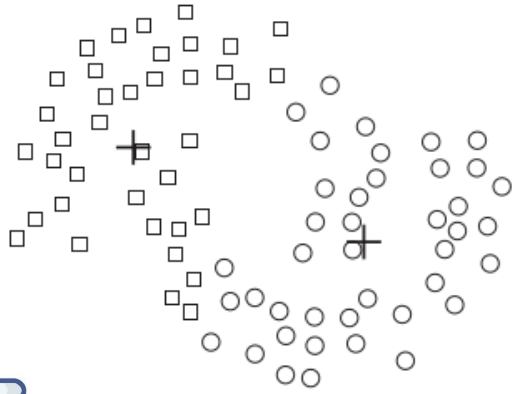
(a) Original points.



(b) Three K-means clusters.



(a) Original points.



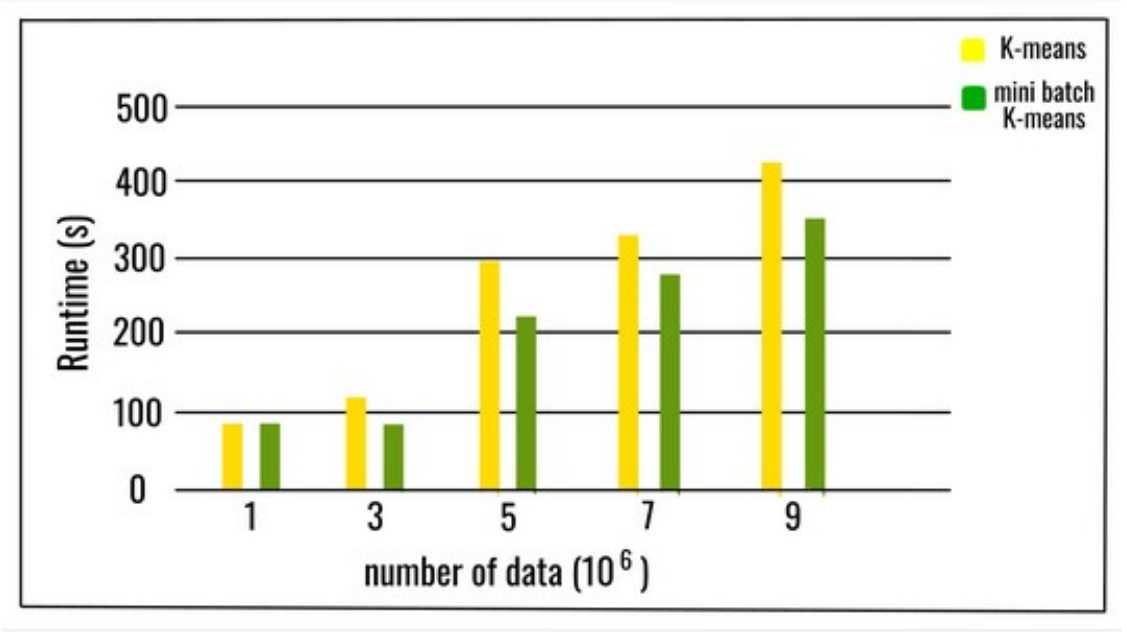
(b) Two K-means clusters.

Minibatch k-means

```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
        M ← b examples picked randomly from X
        for x ∈ M do
                d[x] ← f (C, x)              // Cache the center nearest to x
        end for
        for x ∈ M do
                c ← d[x]                     // Get cached center for this x
                v[c] ← v[c] + 1              // Update per-center counts
                η ← 1 / v[c]                 // Get per-center learning rate
                c ← (1 – η)c + ηx            // Take gradient step
        end for
end for
```

Minibatch k-means

```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
    M ← b examples picked randomly from X
    for x ∈ M do
        d[x] ← f (C, x)          // Cache the center nearest to x
    end for
    for x ∈ M do
        c ← d[x]
        v[c] ← v[c] + 1
        η ← 1 / v[c]
        c ← (1 – η)c + ηx
    end for
end for
```



Implementación en sklearn

sklearn.cluster.k_means:

```
sklearn.cluster.k_means(X, n_clusters, init='k-means++', n_init=10,
                        max_iter=300, verbose=False, tol=0.0001, n_jobs=1)
```

Parámetros:

X: datos de entrada (arreglo de vectores)
n_clusters: k
max_iter: número máximo de iteraciones (def. 300)
n_init: número de reinicios (def. 10). Se muestra el mejor resultado (min SSE).
init: {k-means++, random}, método de inicialización
tol: incremento relativo para declarar convergencia
n_jobs: número de hilos de ejecución:

```
   1: sin paralelismo
  -1: todas los procesadores son usados.
 <-1: se usan CPUs + n_jobs + 1 procesadores
      (Ej. n_jobs=-2, entonces se usan todos menos uno).
```

Implementación en sklearn

sklearn.cluster.MiniBatchKMeans:

```
sklearn.cluster.MiniBatchKMeans(n_clusters=8, init='k-means++', max_iter=100,
                batch_size=100, verbose=0, tol=0.0, max_no_improvement=10,
                n_init=3, reassignment_ratio=0.01)
```

Parámetros (diferencias con k-means):

max_no_improvement:	número consecutivo de mini batches en los que no debe haber mejora para declarar convergencia (def 10).
batch_size:	tamaño de los batches (def 100).
reassignment_ratio:	controla el número de mínimo de hits que un centroide debe registrar para no ser reasignado (def 0.01).

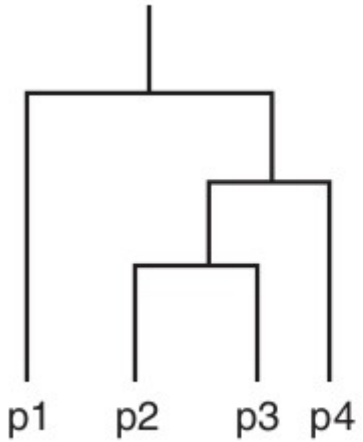
- Clustering jerárquico -

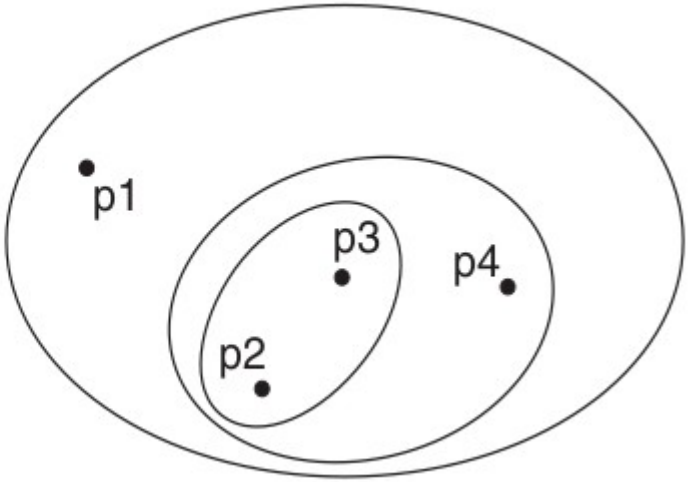
Clustering jerárquico

Algoritmo	Parámetro	Escalabilidad	Caso de uso	Geometría
K-Means	Número de clusters	Escalable Mejora con modificación MiniBatch	Propósito general flat clustering K no muy grande	Distancia entre objetos
Affinity Propagation	Coeficiente de damping	No escalable	Non-flat clustering K grande	Grafo de distancias
Mean-shift	Ancho de banda	No escalable	Non-flat clustering K grande	Distancia entre objetos
Spectral clustering	Número de clusters	Escalabilidad media	Non-flat clustering K no muy grande	Grafo de distancias
Ward	Número de clusters	Escalable	K grande	Distancias entre objetos
Clustering aglomerativo	Número de clusters	Escalable	Distancias no Euclideanas K grande	Distancias entre objetos
DBSCAN	Tamaño del vecindario	Escalable	Clusters de tamaños distintos	Grafo de vecinos más cercanos
Mezcla de Gaussianas	Muchos	No escalable	Flat clustering Estimación de densidad	Distancias Mahalanobis a centroides

Clustering Jerárquico

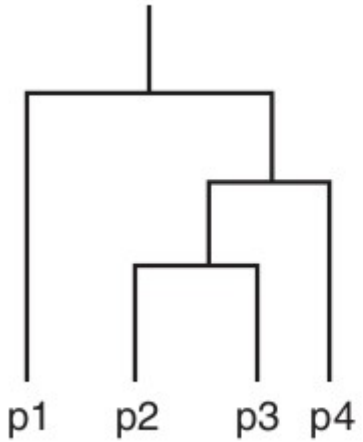
Idea:

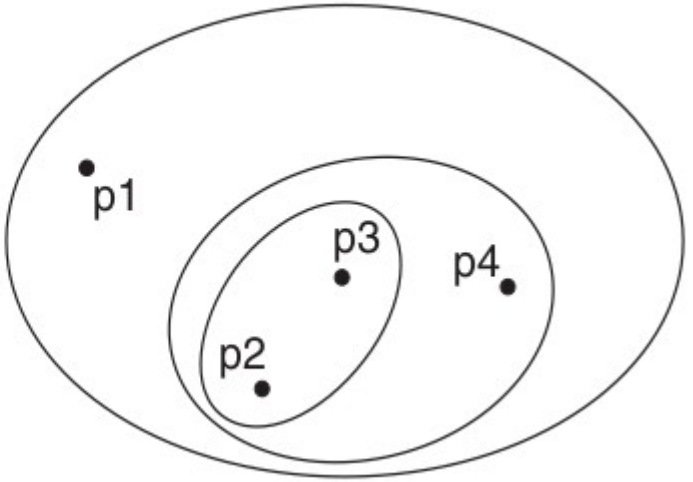




Clustering Jerárquico

Idea:

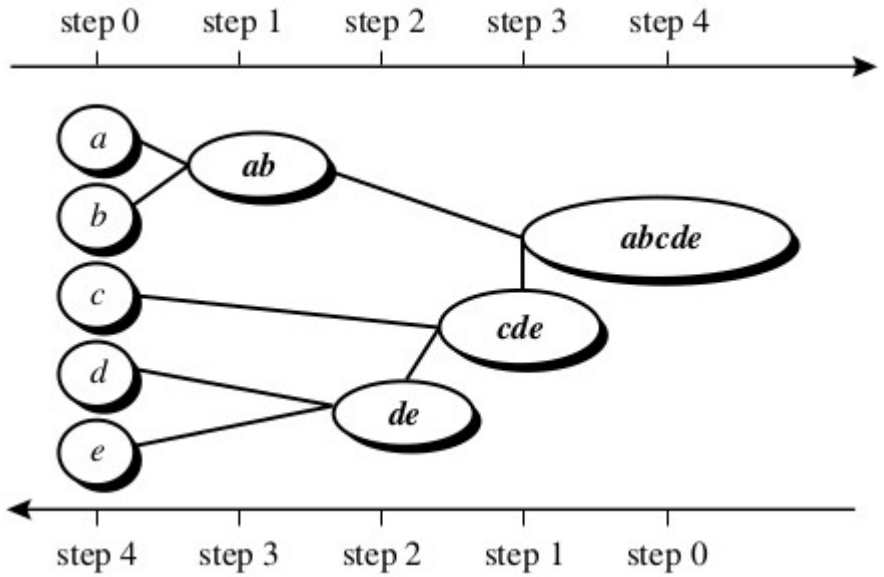




Algorithm Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

Clustering Jerárquico



Clustering Jerárquico