Tarea 3 Reconocimiento de Patrones

Fecha de entrega: domingo 10 de marzo, 22PM

Un colab con código Python para agrupamiento y la contraparte en R está en el material de la clase de 28 de febrero.

En `https://www.r-graph-gallery.com/340-custom-your-dendrogram-with-dendextend.html` se pueden encontrar algunas herramientas para trabajar con dendogramas. En particular `tanglegram` para comparar dos dendogramas es de vez en cuando util. No he encontrado su contraparte en Python.
Por ejemplo `https://www.python-graph-gallery.com/dendrogram/` no contiene tanto.

Ejercicios:

1. Verifica la igualdad que vimos en la clase:

$$\frac{1}{2}\sum_{k=1}^{K}\sum_{i:g(i)=k}\sum_{j:g(j)=k}||x_i-x_j||^2=\sum_{k=1}^{K}N_k\sum_{i:g(i)=k}||x_i-\mu_k||^2 \text{ con } \mu_k=promedio\{x_i:g(i)=k\} \tag{1}$$

   donde N_k es el número de elementos en cluster k.

   Puedes limitarte al caso cuando $x \in \mathcal{R}$.

2. Sea la fórmula de **average linkage** que se usa para un **Algoritmo Jerarquico Aglomerativo**

$$d(\mathcal{C}_i,\mathcal{C}_j) \;=\; \frac{1}{|\mathcal{C}_i|\cdot|\mathcal{C}_j|}\sum_{\mathbf{x}\in\mathcal{C}_i}\sum_{\mathbf{y}\in\mathcal{C}_j}d(\mathbf{x},\mathbf{y}).$$

   donde $|\mathcal{C}_i|$ y $|\mathcal{C}_j|$ representan la cardinalidad de los clusters $\mathcal{C}_i$ y $\mathcal{C}_j$ respectivamente, y $d(\mathbf{x},\mathbf{y})$ una medida de distancia entre $\mathbf{x}$ y $\mathbf{y}$.

   En cada paso, los clusters mas cercanos $\mathcal{C}_i$ y $\mathcal{C}_j$ se combinan en un nuevo cluster $\mathcal{C}_i \cup \mathcal{C}_j$. Muestra que la distancia del cluster $\mathcal{C}_i \cup \mathcal{C}_j$ a otro cluster $\mathcal{C}_k$ se puede calcular mediante la fórmula recursiva:

$$d(\mathcal{C}_i \cup \mathcal{C}_j,\mathcal{C}_k) \;=\; \frac{|\mathcal{C}_i|\cdot d(\mathcal{C}_i,\mathcal{C}_k)+|\mathcal{C}_j|\cdot d(\mathcal{C}_j,\mathcal{C}_k)}{|\mathcal{C}_i|+|\mathcal{C}_j|}$$

3. Un plot de diagnóstico para evaluar un agrupamiento es un `stripes plot`.

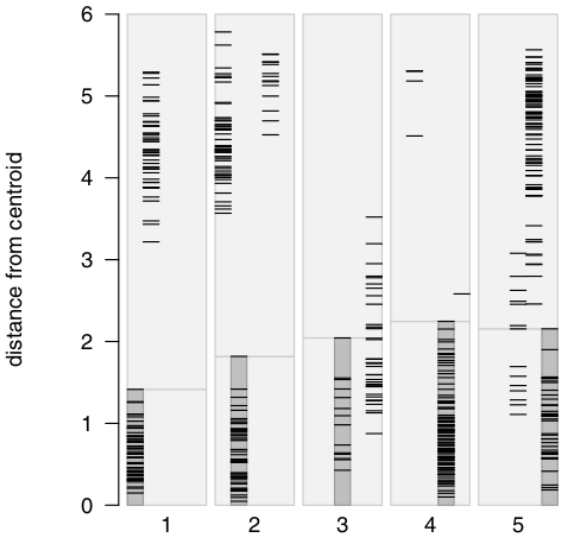

Fig. 6.19. Stripes plot of k-means solution for artificial data.

Se visualiza por cluster la distancia de las observaciones al centroide del cluster, por cluster. En R se puede usar la función `stripes` de la libraria `flexclust`. En Python no lo he encontrado.

Escribe tu propia función en Python o en R siguiendo la misma idea pero puedes proponer tu variante propia. Ilustra su funcionamiento para algunos conjuntos de prueba generadas a partir de gausianas.

4. Usa los métodos de agrupamiento vistos en clase para buscar (y evaluar y discutir) grupos en los datos del heptatlón.

   Para construir un `elbow plot`, puedes calcular para diferentes valores de k `kmeans(data, k)$tot.withinss` en R o en Python:

```
km = KMeans(n_clusters=k)
km.fit(data)
km.inertia_
```

   Aprovecha también el stripes plot del inciso anterior.

   Una posible visualización de los resultados es proyectar los datos y los representantes sobre los primeros dos componentes principal del

conjunto (completo) de los datos (aunque tiene sus restricciones por supuesto).