Sistemas de Aprendizaje Automático

Conforme a contenidos del «Curso de Especialización en Inteligencia Artificial y Big Data»

Universidad de Castilla-La Mancha

Escuela Superior de Informática
Ciudad Real

Capítulo 7

Otros algoritmos de aprendizaje no supervisado

Ricardo García Ródenas
José Ángel Martín Baos

7.1. Algoritmos jerárquicos

El algoritmo de las $K-$medias es el representante por antonomasia de los llamados *métodos clusters particionales* en los que directamente se busca una partición del conjunto de datos mediante la optimización de un cierto criterio. Los algoritmos jerárquicos son métodos alternativos basados también en el concepto de distancia. El aspecto relevante de estos métodos es que son capaces de establecer una jerarquía entre grupos, lo que le permite al usuario definir un nivel de granularidad del problema y determinar la partición final. Existen dos enfoques en el desarrollo de métodos jerárquicos: i) **aglomerativos** en los que inicialmente cada objeto es su propio grupo y un par de estos grupos se van juntando en cada iteración y ii) **divisivos** en los que se parte de un solo grupo y en cada iteración se divide uno de los grupos existentes.

El coste computacional de lo algoritmos jerárquicos aglomerativos es $O(n^3)$ lo que los hace aplicables a tamaños de datos pequeños mientras que algoritmos jerárquicos divisivos en los que se hace una búsqueda exhaustiva para determinar que grupo particionar tiene una complejidad exponencial $O(2^n)$ que los hacen prohibitivos incluso para tamaños de datos muy pequeños. Por este motivo nos centramos en el primer tipo de algortimos.

Los métodos **jerárquicos aglomerativos** son procedimientos iterativos en los que inicialmente se considera que cada objeto constituye un cluster individual. En cada iteración se agrupa los dos clusters mas próximos de acuerdo a una **distancia intercluster** $d(C_k, C_s)$, definida a partir de una distancia dada entre objetos, y finaliza cuando todos los objetos son asignados a un único cluster. Notar que en la

implementación del algoritmo solo se requiere la matriz de distancia entre todos los objetos, no es necesario conocer las propias observaciones. A partir de esta matriz que almacena en cada celda i, j la distancia $d(x_i, x_j)$ se puede calcular la distancia intercluster $d(C_k, C_s)$.

El resultado del algoritmo es un árbol de grupos denominado **dendograma** (ver lado derecho de la figura 7.1) que muestra la unión de los diferentes grupos. En el eje OY se representa la distancia interclusters. Las conexiones marcan qué grupos se han juntado y su altura con qué distancia intercluster lo ha hecho. Mediante un **corte** en el dendograma a un nivel deseado de la distancia entre clusters, se obtiene el agrupamiento de los datos en grupos disjuntos. Una revisión de estos métodos se encuentra en [MC12]

Un método jerárquico queda completamente definido a partir de la distancia entre objetos y la distancia entre cluster. La tabla 7.1 recoge las distancias intercluster más usuales.

Tabla 7.1: Distancias interclusters en el análisis cluster jerárquico

Distancia interclusters	Definición matemática
• **Vecino más próximo** *(Single)*	$d_S(C_k, C_s) = \min_{\substack{i_k \in C_k \\ i_s \in C_s}} \{d(x_{i_k}, x_{i_s})\}$
• **Vecino más alejado** *(Complete)*	$d_A(C_k, C_s) = \max_{\substack{i_k \in C_k \\ i_s \in C_s}} \{d(x_{i_k}, x_{i_s})\}$
• **Distancia media** *(Average)*	$d_M(C_k, C_s) = \frac{1}{\lvert C_k \rvert \cdot \lvert C_s \rvert} \sum_{\substack{i_k \in C_k \\ i_s \in C_s}} d(x_{i_k}, x_{i_s})$
• **Distancia entre centroides** *(Centroide)*	$d_{Ce}(C_k, C_s) = d(x_k^c, x_s^c)$ donde x_k^c y x_s^c son respectivamente los centroides de C_k y C_s
• **Mínima varianza** *(Ward)*	$d_V(C_k, C_s) = \frac{1}{\lvert C_k \cup C_s \rvert} \sum_{i \in C_k \cup C_s} d(x_i, x^c)$ donde x^c es el centroide de $C_k \cup C_s$

Un algoritmo cluster jerárquico prototipo se describe en la tabla 7.2.

La figura 7.1 muestra los resultados del análisis cluster con un método jerárquico obtenido, empleando la distancia euclídea y la distancia entre cluster distancia al vecino más cercano. El dendograma muestra a que distancia se han ido juntando los distintos grupos. Esta representación nos permite aproximar el número adecuado de cluster. La existencia de un *outlier* en los datos hace que el valor verdadero $K = 2$, sea inadecuado. Si consideramos el valor verdadero $K = 2$ obtenemos que el *outlier* conforma su propio grupo y el resto de los datos el otro grupo.

Tabla 7.2: El algoritmo cluster jerárquico aglomerativo

Paso 0. (**Inicialización**). Asignar cada objeto a un cluster individual. El número inicial de cluster es n.

Paso 1. (**Agrupación**). Encontrar los dos clusters más cercanos de acuerdo a $d(C_k, C_s)$ y únelos en un nuevo cluster. Decrecer el número total de clusters en uno.

Paso 2. (**Criterio de paro**). Repetir el paso 1, hasta que todos los objetos sean agrupados en un número predefinido de clusters.

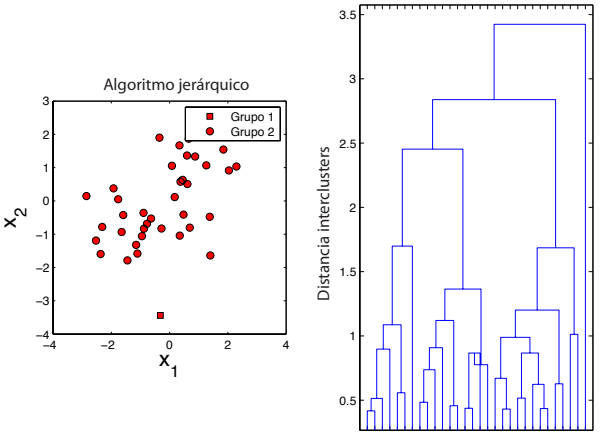

Figura 7.1: Resultados de aplicar un análisis cluster jerárquico al ejemplo 6.3.2.

El listado 7.1 muestra un ejemplo en Python que implementa un algortimo de clustering jerárquico. En este ejemplo, la variable X contiene el dataset empleado. Notar que en este ejemplo se utiliza como distancia intercluster el vecino más alejado (complete). Finalmente, se establece un umbral (threshold) de 2,5 como distancia a partir de la cual se considerará que los grupos a distancia mayor son grupos independientes (y por lo tanto no se agruparan).

Listado 7.1: Ejemplo de algoritmo de clustering jerárquico en Python

```
# 1. Obtener la matriz de distancia entre todos los elementos
import sklearn.neighbors
dist = sklearn.neighbors.DistanceMetric.get_metric('euclidean')
D = dist.pairwise(X)

# 2. Constuimos el dendograma (establecemos threshold=2.5)
from scipy import cluster
```

```
threshold = 2.5
clusters = cluster.hierarchy.linkage(D, method='complete')
cluster.hierarchy.dendrogram(clusters, color_threshold=threshold)
plt.show()

# 3. Obtenemos el grupo al que pertenece cada observación
labels = cluster.hierarchy.fcluster(clusters, threshold , criterion = 'distance')
# ¿Cuántos grupos hay? Contamos el número de "labels"distintas en el vector
print("Número de clusters {}".format(len(set(labels))))
```

Ejercicio. Considérese que la distancia entre cinco objetos viene definida por la siguiente tabla

	O1	O2	O3	O4	O5
O1	0.0	0.9	1.9	2.0	1.0
O2	0.9	0.0	1.0	2.0	2.0
O3	1.9	1.0	0.0	1.0	2.0
O4	2.0	2.0	1.1	0.0	0.8
O5	1.0	2.0	2.0	0.8	0.0

Aplica un algoritmo jerárquico con la distancia intercluster el *vecino más alejado* (complete).

Iteración 1.

	O1	O2	O3	O4 – O5
O1	0.0	0.9	1.9	2.0
O2	0.9	0.0	1.0	2.0
O3	1.9	1.0	0.0	2.0
O4-O5	2.0	2.0	2.0	–

Iteración 2.

	O1 – O2	O3	O4 – O5
O1-O2	–	1.9	2.0
O3	1.9	0.0	2.0
O4-O5	2.0	2.0	–

Iteración 3.

	O1 – O2 – O3	O4 – O5
O1–O2–O3	–	2.0
O4–O5	2.0	–

Iteración 4.

	O1 – O2 – O3 – O4 – O5
O1–O2–O3–O4–O5	–

La información para realizar el dendograma se recoge en la siguiente tabla:

Grupo	**Grupo**	**Distancia de unión**
{O4}	{O5}	0.8
{O1}	{O2}	0.9
{O1,O2}	{O3}	1.9
{O1,O2,O3}	{O4,O2}	2.0

7.2. Algoritmos basados en el concepto de densidad

La idea clave de estos métodos es se supone que los grupos están definidos en áreas de gran concentración (densidad) de datos. Las zonas pocos densas separan los grupos. En esta sección vamos a estudiar dos de los algoritmos más importantes de este tipo: el DBSCAN y el DPC.

7.2.1. DBSCAN

DBSCAN fue introducido en [EKSX96] con el objetivo de poder descubrir clusters con forma arbitraria y además poseer un buena eficiencia en grandes bases de datos. Este algoritmo se basa en el concepto de densidad y sólo requiere dos hiperparámetro de entrada, existiendo técnicas para determinar los valores apropiados.

Considérese que disponemos de un conjunto de datos $X = \{x_i\}_{i=1}^n$ que queremos agrupar. Dado el parámetro ε definimos el concepto de densidad del punto x_i por

$$\rho_i = |B(x_i, \varepsilon) \cap X| \tag{7.1}$$

donde $B(x_i, \varepsilon)$ es la bola abierta de centro x_i y radio ε y $|\cdot|$ es el cardinal de un conjunto. La densidad ρ_i es el número de puntos que contiene el entorno de x_i de radio ε.

El DBSCAN clasifica los puntos en:

- El punto x_i es un **punto núcleo** si $\rho_i \geq$ `minPts`. Además los puntos de su entorno, $B(x_i, \varepsilon) \cap X$, son **directamente alcanzables** desde él.
- El punto x_j es un **punto alcanzable** desde x_i si existe una secuencia de puntos núcleos $q_1, \ldots, q_m$ cumpliendo $q_1 = x_i, q_m = x_j$, todos son puntos núcleos y q_{i+1} es directamente alcanzable desde q_i.
- El punto x_r es **ruido** si no es alcanzable desde ningún punto.

La figura 7.2 ilustra esta definición. Se ha considerado el radio ε para definir las bolas abiertas, lo que conduce a que los puntos A, B y C tengan respectivamente la densidad $\rho_A = 3$, $\rho_B = 2$ y $\rho_C = 1$. Si fijamos el parámetro `minPts=3` todos los puntos son núcleos exceptuando los puntos B y C. El punto B es accesible desde A mediante el camino marcado. Este camino consta de puntos núcleos que son directamente accesibles desde su predecesor. El punto C no se puede acceder desde ningún punto (núcleo) por lo que es ruido.

Vamos a hacer una observación a estas definiciones. La relación alcanzable no es simétrica. Esto es debido a que por definición solo se puede alcanzar directamente desde puntos núcleos. Por ejemplo el punto B de la figura 7.2 es alcanzable desde A pero A no es alcanzable desde B. En el análisis cluster los grupos introdu-

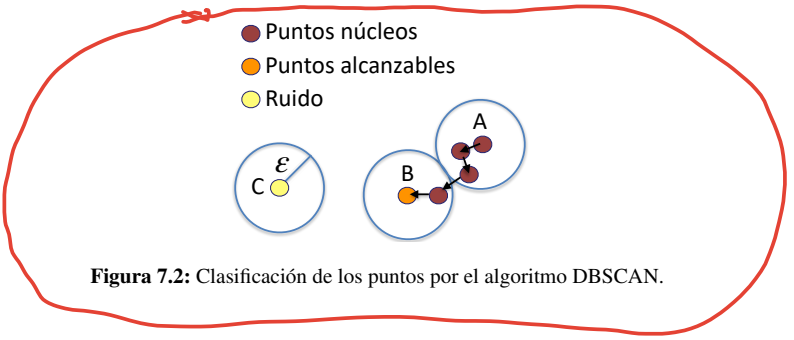

Figura 7.2: Clasificación de los puntos por el algoritmo DBSCAN.

cen una noción de *conexión*, dos puntos que están en la mismo grupo deben estar conectados y esta operación es simétrica. Por este motivo debemos hacer que el concepto de *alcanzable* sea también simétrico. Por eso introducimos el siguiente concepto para reemplazar el concepto de directamente conectado.

Definición 8 (densamente conectados) *Dos puntos x_i y x_j están conectados densamente si existe un punto $\widetilde{x}$ de modo que tanto x_i como x_j son alcanzables directamente desde $\widetilde{x}$. La figura 7.3 ilustra este concepto. Los puntos x_i, x_j están en el mismo entorno de $\widetilde{x}$ siendo este un punto núcleo por lo que los puntos x_i, x_j están conectados densamente.*

Figura 7.3: Ilustración del concepto de conectado densamente.

Cada cluster está caracterizado por estas dos condiciones:

`C1` Los elementos de un cluster solo están densamente conectados a elementos del propio grupo.

`C2` Dos elementos cualesquiera de un cluster son densamente alcanzables.

Ahora vamos a desarrollar un algoritmo para hacer efectiva la determinación de los clusters atendiendo a las anteriores dos características. El algoritmo se inicializa con dos hiperparámetros: i) ε que define el tamaño del entorno y ii) `minPts` que es el número mínimo de puntos para considerar un punto núcleo. El algoritmo comienza por un punto arbitrario p que no haya sido visitado. Si se trata de un punto ruido no estará en ningún cluster, en caso contrario comenzamos la construcción del cluster al que pertenece. Para ello hay que analizar si su entorno contiene algún punto núcleo.

Tabla 7.3: El algoritmo DBSCAN

Paso 0. (**Inicialización**). Elegir los parámetros ε y `minPts`.

- (**Cálculo de las densidades**). Calcular la densidad de cada punto ρ_i empleando la ecuación (7.1).
- (**Inicialización de conjuntos**). Inicializar el conjunto de puntos núcleo

$$X_N := \{i : \rho_i \geq \texttt{minPts}\} \tag{7.2}$$

y el conjunto de puntos etiquetados $X_e = \{\emptyset\}$.

Paso 1. Para todo $p \in X_N - X_e$ hacer

Tomar $X_e = X_e \cup \{p\}$.

Inicializa un nuevo cluster como $C = B(p, \varepsilon) \cap X$

While $C - X_e \neq \{\emptyset\}$

Toma un $q \in C - X_e$, haz $X_e = X_e \cup \{q\}$

Si $q \in X_N \Rightarrow C = C \cup (B(q, \varepsilon) \cap X)$

Paso 2. Asigna a los datos no etiquetados $\{1, 2, \ldots, n\} - X_e$ la etiqueta de outlier

---

La propiedad `C2` indica que todos los puntos del cluster al que pertenecen son densamente alcanzables, por tanto, todo punto núcleo de su entorno $q \in B(p, \varepsilon)$ está en el cluster y además los puntos $B(q, \varepsilon)$ pertenecen al cluster. Este nuevo conjunto permite densamente acceder a nuevos puntos del cluster. Para ello hay que llevar un control de si se ha visitado o no un punto. El proceso continúa hasta construir completamente un clúster densamente conectado. Entonces, un nuevo punto no visitado se visita y procesa con el objetivo de descubrir otro clúster o ruido.

Ventajas e inconvenientes

Las principales ventajas se pueden sintetizar en:

- DBSCAN no necesita que el modelador le indique el número de cluster K a formar, a diferencia del $K-$means y el FCM dónde si es necesario.
- A diferencia del $K-$ means puede encontrar clusters no esféricos, de formas arbitrarias.
- Como incorpora específicamente el concepto de outliers es insensible a los mismos.
- DBSCAN requiere solamente dos parámetros.

Las principales desventajas se pueden sintetizar en:

- El orden de los datos en la base de datos afecta al agrupamiento de los puntos no núcleo y no ruido. Dependiendo de su disposición pueden ser asignados a cluster diferentes. El DBSCAN no es determinista para este tipo de puntos.

- Cuando aumenta el número de dimensiones de los datos x_i, la variabilidad de la distancia disminuye exponencialmente con el número de dimensiones, haciendo que la distancia se vuelva menos discriminativa. (Es más difícil decir qué datos están realmente lejos y cerca de uno dado). Este efecto se le conoce con el nombre de la **Maldición de la dimensionalidad** (*Curse of Dimensionality*). Como el algoritmo DBSCAN, al igual que el K-means, se basa en la distancia euclídea su rendimiento decae en problemas con altas dimensiones en los datos.

- DBSCAN no puede agrupar bien grupos con densidades muy dispares, ya que en esos casos no se pueden ajustar correctamente los parámetros ε y `minPts`.

Ejercicio. Considérese la figura 7.4. Esta figura muestra 16 puntos dispuestos sobre una cuadrícula de lado unidad. Considérese entornos de diámetro $\varepsilon = 1{,}1$. Vamos a calcular la densidad de los puntos. Si nos fijamos en el entorno dibujado del punto 7 se observa que los puntos que contiene un entorno son los puntos localizados a su derecha e izquierda y arriba o abajo. Los puntos que se encuentran en su diagonal se escapan del entorno. Se pide:

1. ¿Cuanto vale la densidad de los puntos? Para calcular la densidad tenemos que contar el número de puntos que contiene cada entorno. Fijandonos en la disposición geométrica de los puntos se puede comprobar los valores mostrados en la siguiente tabla:

Nodos	Valor de la densidad
6,7,10,11	5
5,9,2,3,8,12	4
13,1,4,16	3

2. ¿Cual sería el resultado de aplicar el DBSCAN con `minPts`=6? Se observa que no existe ningún punto núcleo por lo que el algoritmo etiquetará a todos los nodos como ruido.

3. ¿Cual sería el resultado de aplicar el DBSCAN con `minPts`=5? El resultado será un cluster formado por todos los nodos menos los vértices de las esquinas $1, 4, 13$ y 16 ya que estos nodos no son accesibles desde nodos núcleo. No están densamente conectados. Estos cuatro puntos serían etiquetados como ruido.

4. ¿Cual sería el resultado de aplicar el DBSCAN con `minPts`=4? Todos los puntos están densamente conectados por lo que formaría un único cluster con todos los puntos.

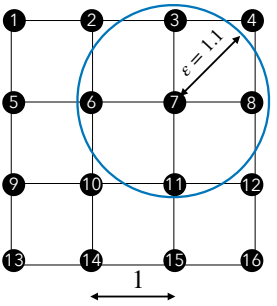

Figura 7.4: Disposición de puntos sobre una cuadrícula.

7.2.2. Algoritmo basado en picos de densidad

Alex Rodríguez y Alessandro Laio ([RL14]) propusieron el llamado algoritmo **clustering basado en picos de densidad** en inglés *Density-Peak Clustering* (DPC). Este algoritmo considera que los centros de los clusters tienen mayor densidad que sus vecinos, y también que están a una distancia relativamente grande de cualquier otro punto con mayor densidad. En una primera etapa, este método localiza los centros de los clusters, y en una segunda usa una estructura de entornos para asignar el cluster a los puntos restantes. El DPC puede detectar clusters con formas no convexas. [LWY18] remarcó las siguientes dos ventajas esenciales de DPC:

1. El algoritmo es simple y eficiente, y puede encontrar rápidamente los picos de alta densidad (centros de cluster).
2. El DPC es adecuado para el análisis cluster para gran cantidad de datos porque los objetos se asignan a los clusters en una sola iteración basándose en asignar el mismo cluster que su vecino más cercano con mayor densidad.

La caracterización de los centros se basa en los siguientes dos simples suposiciones ([LT18]):

- **Hipótesis 1.** Los centros de los clusters están rodeados de vecinos con menor densidad.
- **Hipótesis 2.** Los centros de los clusters están a una distancia relativamente grande de otros puntos con mayor densidad que ellos.

La formulación matemática de estas dos hipótesis requiere la definición de dos magnitudes para cada objeto i a ser agrupado : i) su **densidad** ρ_i y la ii) **distancia** δ_i del punto i al punto más cercano con mayor densidad. Sea $d_{ij} = d^2(x_i, x_j)$ la distancia euclídea entre los puntos x_i y x_j. La densidad ρ_i, dependiendo del tamaño n de objetos a agrupar, se puede calcular de dos formas:

- *Bases de datos grandes*. La siguiente expresión es usada

$$\rho_i = \sum_{j \neq i} \chi\left(d_{ij} - d_c\right), \chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{7.3}$$

  que calcula el número de puntos vecinos de i que se encuentran a una distancia inferior a d_c unidades. El parámetro d_c es especificado por el modelador y es un umbral que define la *vecindad*. Notar que este concepto es el mismo que el empleado por el DBSCAN pero denotando $d_c = \varepsilon$.

- *Bases de datos de tamaños moderados*. La densidad se calcula a partir de una funciones conocidas con el nombre de *núcleos gaussianos*:

Se usa todo el conjunto

$$\rho_i = \sum_{j \neq i} \exp\left[-\left(\frac{d_{ij}}{d_c}\right)^2\right] \tag{7.4}$$

En las expresiones (7.3) y (7.4) es necesario especificar la distancia de corte d_c para determinar las densidades ρ_i. Originalmente se propuso calcular este valor utilizando los cuantiles de la distribución de distancias entre los elementos:

$$d_c = D_q \tag{7.5}$$

donde D_q es cuantil de orden q de la distribución de distancias $D = \{d_{i,j} : i < j\}$ y el parámetro p es una proporción fijada por el modelador. La ventaja de utilizar el parámetro p en lugar de d_c es que p puede interpretarse en el conjunto de todas los problemas, mientras que el valor d_c varía de un problema a otro, dependiendo de la escala de los datos $\{x_i\}$.

Una vez calculadas las densidades, la siguiente etapa consiste en calcular la distancia mínima de cada punto i al punto j de mayor densidad que él. Esta distancia se define por:

$$\delta_i = \min_{j:\rho_j > \rho_i} \left(d_{ij}\right) \tag{7.6}$$

Una vez obtenidos ρ_i y δ_i, se construye un gráfico de decisión, a veces denominado grafo de decisión, con el fin de seleccionar los centros (ver Figura 7.5), que serán puntos más separados del resto del gráfico, ya que tienen una densidad mayor que sus vecinos, y también están separados de los otros candidatos. En este gráfico se observa que los centros de los clusters tienen un valor ρ_i y δ_i mayor que el resto de puntos que no son centros y están situado en la esquina superior derecha (Figura 7.5.(a)). El procedimiento requiere la intervención humana para seleccionar los centros manualmente. Una heurística para encontrar los centros automáticamente es definir $\gamma_i = \rho_i \delta_i$ y elegir los centros como aquellos puntos con los valores más altos de γ_i.

Tabla 7.4: El algoritmo DPC

Paso 0. (**Inicialización**). Elegir el parámetro distancia de corte d_c.

Paso 1. (**Calculo de distancias entre puntos**). Calcular la matriz de distancias (d_{ij}).

Paso 2. (**Calculo de densidades**). Calcular ρ_i para cada objeto i usando la fórmula (7.3) o (7.4).

Paso 3. (**Calculo de distancias mínimas a puntos de mayor densidad**). Calcular δ_i para cada objeto i empleando la fórmula (7.6)

Paso 4. (**Gráfico de decision**). Dibuja el grafo de decisión y selecciona los centros de los clusters.

Paso 5. (**Asignación**). Asignar los objetos restantes al mismo cluster que su vecino más cercano de mayor densidad.

Una vez encontrados los centros de los clusters, cada punto restante se asigna al mismo cluster que su vecino más cercano de mayor densidad. La asignación de clusters se realiza en un solo paso, a diferencia de otros algoritmos de clustering en los que se optimiza una función objetivo de forma iterativa.

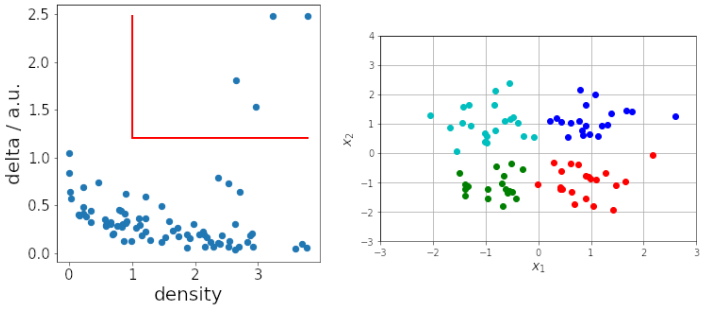

Figura 7.5: Gráfico de decisión en el método DPC.

7.3. Algoritmos $K-$means basados en núcleos

7.3.1. Motivación

El algoritmo $K-$means es adecuado para identificar grupos que están **separados linealmente** como los que se muestra en la figura 7.6 del lado de la izquierda. En esta situación existe un hiperplano (en el caso de datos bidimensionales, como en el ejemplo, el hiperplano se trata de una recta) que **los separa**. Si los datos tu-

vieran la distribución como los mostrados en la parte derecha de la figura 7.6 el algoritmo $K-$means no sería capaz de identificar los dos grupos. Cuando se produce esta disposición geométrica se dice que los datos **NO** pueden ser separados linealmente, esto es, no se puede dibujar una recta que deje a un lado los cuadrados y al otro los círculos.

Vamos a formalizar matemáticamente este concepto. Para definir un hiperplano hacemos uso del **producto escalar** de dos vectores $< x, y >= \sum_{i=1}^{P} x_i y_i$. Un hiperplano en un espacio euclídeo de dimensión n viene definido por

$$f(x) =< \alpha, x > +\beta$$

donde $\alpha \in \mathbb{R}^n$ y $\beta \in \mathbb{R}$. Por ejemplo en el caso bidimensional nos queda la expresión de la recta $f(x_1, x_2) = \alpha_1 x_1 + \alpha_2 x_2 + \beta$.

Siguiendo con el ejemplo de la figura 7.6, decimos que el hiperplano separa al grupo de círculos y cuadrados si se cumple las condiciones:

$$< \alpha, x > +\beta \geq 0 \text{ para todo } x \text{ círculo} \tag{7.7}$$
$$< \alpha, y > +\beta \leq 0 \text{ para todo } y \text{ triángulo} \tag{7.8}$$

o si se satisfacen las mismas desigualdades pero intercambiando $\leq$ por $\geq$.

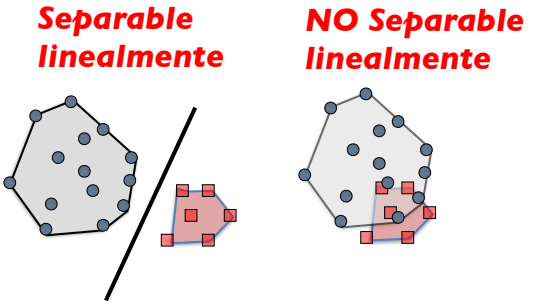

Figura 7.6: Grupos linealmente separables.

El algoritmo $K-$means con núcleos busca resolver la problemática de trabajar con grupos que no están separados linealmente. La idea que lo sustenta está ilustrada en la figura 7.7. Lo que hacen es proyectar los datos en un espacio en los que los datos si estén separados linealmente y trabajar entonces con el algoritmo $K-$means en él. Este espacio se denomina **espacio de características**.

Si pensamos que en el espacio de características se deben separar linealmente los puntos deben existir hiperplanos y por tanto tiene que tener un producto escalar. Además, debe ser diferente en alguna manera del espacio euclídeo $n-$ dimensional original. Los espacios vectoriales de funciones (en lugar de tener puntos con n componentes tienen funciones) poseen dimensión infinita. Estos espacios de funciones con su respectivo producto escalar son ejemplos de **espacios de Hilbert**.

Además cuando se dispone de un producto escalar este define una norma mediante la relación:

$$\|x\| = \sqrt{< x, x >}$$

y a su vez esta norma define una distancia de esta manera:

$$d(x, y) = \|x - y\| = \sqrt{< x - y, x - y >}$$

lo que va a permitir poder aplicar el algoritmo de $K-$means.

Los llamados **espacios de Hilbert con núcleo reproductor** (RHKS) son los espacios adecuados de trabajo. Estos espacios están completamente definidos por unas funciones $K(x, y)$ denominadas **núcleos de Mercer** y el RHKS asociado a ella se denota por $\mathcal{H}_K$. Ejemplos de núcleos de Mercer son los siguientes:

1. *Lineal.*
$$K(x_i, x_j) = x_i^T \cdot x_j. \tag{7.9}$$

2. *Polinomial de grado p.*
$$K(x_i, x_j) = (1 + a x_i^T \cdot x_j)^b, \quad a \in \mathbb{R}, b \in \mathbb{N}. \tag{7.10}$$

3. *Gaussiano.*
$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) \quad \sigma \in \mathbb{R}. \tag{7.11}$$

Para que el algoritmo de $K-$means quede completamente definido en el espacio de características (ver figura 7.7) se tiene que definir la aplicación de proyección $\Phi(x)$ en $\mathcal{H}_K$ y definir el producto escalar en él.

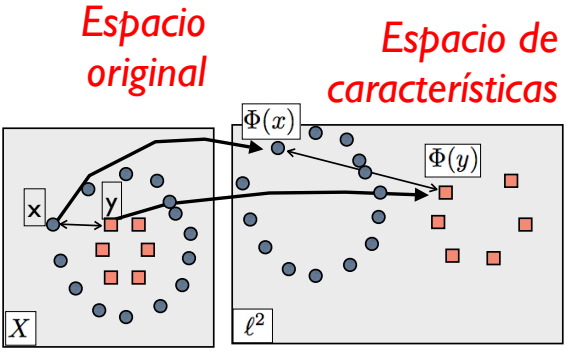

Figura 7.7: Espacio original y espacio de características.

7.3.2. Algoritmo de las $K-$medias en el espacio de características

La aplicación de proyección viene definida por:

$$\begin{array}{rcl} \Phi : X & \rightarrow & \ell^2 \\ x & \mapsto & \Phi(x) = \left(\sqrt{\lambda}_j \phi_j(x)\right)_{j\in\mathbb{N}} \end{array}$$

donde las funciones ϕ_j son conocidas solo implícitamente. La definición formal de estas funciones está recogida en la sección 7.3.2 pero aquí omitimos la parte matemática. Por tanto la aplicación Φ sólo se conoce implícitamente y permite asignar datos de un espacio finito $X \subset R^P$ a un espacio de dimensión infinita denominado el *espacio de características* y denotado por ℓ^2. Realizar particionamientos separables linealmente (como lo realiza el algoritmo de las $K-$ medias) en el *espacio de características* conduce a particionamientos no separables linealmente en el espacio de datos originales. El llamado **truco del núcleo** permite, sin necesidad de conocer explícitamente la aplicación Φ, calcular productos escalares en el espacio de características mediante la relación:

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle \tag{7.12}$$

Este resultado nos permite calcular distancias en ℓ^2

$$\begin{array}{l} \|\Phi(x) - \Phi(y)\|^2 = \langle \Phi(x) - \Phi(y), \Phi(x) - \Phi(y) \rangle \\ = \langle \Phi(x), \Phi(x) \rangle + \langle \Phi(y), \Phi(y) \rangle - 2 \langle \Phi(x), \Phi(y) \rangle \\ \qquad = K(x,x) + K(y,y) - 2K(x,y) \end{array} \tag{7.13}$$

como una función de los datos originales y replantear el algoritmo de $K-$medias a través de núcleos de Mercer.

Por ejemplo, si consideramos un núcleo lineal $K(x,y)$ dado en la ecuación (7.9), el truco del núcleo (7.13) conduce a:

$$\begin{array}{rcl} \|\Phi(x) - \Phi(y)\|^2 & = & K(x,x) + K(y,y) - 2K(x,y) \\ & = & x^T \cdot x + y^T \cdot y - 2x^T \cdot y = \|x - y\|^2. \end{array}$$

Este resultado muestra que la distancia en el espacio de características asociada a un núcleo lineal equivale a la norma euclídea en el espacio original de los datos.

Nota. Hay un abuso de notación que no debe llevar a confunsión al lector. Por un lado K representa el número de clusters en el algoritmo de $K-$means y por otro $K(x,y)$ denota una determinada función núcleo.

Si X es el conjunto de datos originales, entonces transformamos X en ℓ^2 mediante la aplicación Φ y podremos considerar K centros en ℓ^2 ($v_i^\Phi \in \ell^2$ con $i = 1, \cdots, K$), siendo $V^\Phi = \left\{v_1^\Phi, \cdots, v_K^\Phi\right\}$ el conjunto de centroides en el espacio ℓ^2.

Para describir el algoritmo introducimos las siguientes definiciones.

Definición 9 (Regiones de Voroni) *Dado un conjunto de centroides* $V^{\Phi} = \left\{v_1^{\Phi}, \cdots, v_K^{\Phi}\right\}$, *llamaremos región de Voroni para* v_i^{Φ} *al conjunto de elementos de* ℓ^2 *cuyo centroide más próximo es* v_i^{Φ}. *Esto es,*

$$R_i^{\Phi} = \left\{x^{\Phi} \in \ell^2 \mid i = arg\ min_j \|x^{\Phi} - v_j^{\Phi}\|\right\}. \tag{7.14}$$

Definición 10 (Conjunto de Voroni) *Dado un conjunto de centroides* $V^{\Phi} = \left\{v_1^{\Phi}, \cdots, v_K^{\Phi}\right\}$, *definimos el conjunto de Voroni* π_i^{Φ} *al conjunto de todos los vectores en el espacio original* $x \in X$ *en el que* v_i^{Φ} *es el punto más cercano a su imagen* $\Phi(x)$, *formalmente:*

$$\pi_i^{\Phi} = \left\{x \in X \mid i = arg\ min_j \|\Phi(x) - v_j^{\Phi}\|\right\}. \tag{7.15}$$

Empleando las definiciones (7.14) y (7.15) el algoritmo de $K-$medias basado en núcleos se describe en la tabla 7.5.

Tabla 7.5: El algoritmo de K-medias basado en núcleos de Mercer

---

Paso 0. (**Inicialización**). Proyectar el conjunto de n datos originales en el espacio ℓ^2 mediante la aplicación no lineal Φ. Inicializar el conjunto de centroides $V^{\Phi} = (v_1^{\Phi}, \cdots, v_K^{\Phi},)$ con $v_i^{\Phi} \in \ell^2$

Paso 1. (**Actualización de clusters**). Calcular para cada v_i^{Φ} el conjunto π_i^{Φ}

Paso 2. (**Actualización de centroides**). Calcular los nuevos centroides $v_i^{\Phi} \in \ell^2$

$$v_i^{\Phi} = \frac{1}{|\pi_i^{\Phi}|} \sum_{x \in \pi_i^{\Phi}} \Phi(x) \tag{7.16}$$

Paso 3. (**Criterio de paro**). Chequear si ha habido algún cambio en los centroides. Si es que si, ir al paso 1. En caso contrario el algoritmo ha encontrado los mejores K-grupos.

Paso 4. Trasladar la solución encontrada al espacio original X.

Debido a que la aplicación $\Phi(x)$ no es conocida explícitamente, no se puede calcular directamente la actualización de los centroides (ver ecuación (7.16)) y se debe recurrir al truco del núcleo (7.13). Escribiendo cada centroide en espacio de características ℓ^2 como la siguiente combinación lineal

$$v_j^{\Phi} = \sum_{h=1}^{n} \eta_{jh} \Phi(x_h) \tag{7.17}$$

donde $\eta_{jh} = \frac{1}{|\pi_j^{\Phi}|}$ si $x_h \in \pi_j^{\Phi}$ y en caso contrario 0.

Operando, se obtiene:

$$\|\Phi(x_i) - v_j^{\Phi}\|^2 = \|\Phi(x_i) - \sum_{h=1}^{n} \eta_{jh}\Phi(x)\|^2$$
$$= k_{ii} - 2\sum_{h=1}^{n} \eta_{jh}k_{ih} + \sum_{r=1}^{n}\sum_{s=1}^{n} \eta_{jr}\eta_{js}k_{rs},$$

donde $k_{ji} = K(x_j, x_i)$ es la llamada **matriz de Gram**.

Ejercicio Calcular la matriz de Gram de un núcleo Gaussiano con $\sigma = 1$ y datos $x_1 = (0,0)$ $x_2 = (1,0)$ y $x_3 = (1,1)$.

$$k_{ii} = \exp(-\|x_i - x_i\|^2) = \exp(0) = 1$$
$$k_{1,2} = \exp(-\|x_1 - x_2\|^2) = \exp(-((0-1)^2 + (0-0)^2)) = \exp(-1)$$
$$k_{1,3} = \exp(-\|x_1 - x_3\|^2) = \exp(-((0-1)^2 + (0-1)^2)) = \exp(-2)$$
$$k_{2,3} = \exp(-\|x_2 - x_3\|^2) = \exp(-((1-1)^2 + (0-1)^2)) = \exp(-1)$$

Usando la simetría de la matriz, obtenemos:

$$\mathbf{K} = \begin{pmatrix} 1 & e^{-1} & e^{-2} \\ e^{-1} & 1 & e^{-1} \\ e^{-2}1 & e^{-1} & 1 \end{pmatrix}$$

APÉNDICE A: Fundamentos matemáticos de los espacios de Hilbert con núcleo reproductor

Esta sección aborda los fundamentos matemáticos de los espacios de Hilbert, permitiendo formular la función de proyección Φ y establecer el resultado teórico (Teorema de Mercer) que sustenta el truco del núcleo. Se puede omitir su lectura.

Los espacios de Hilbert han permitido generalizar los espacios euclídeos a espacios de dimensión infinita. Los espacios de Hilbert se definen del siguiente modo.

Definición 11 (Espacio prehilbertiano) *Un espacio prehilbertiano es un espacio vectorial* $(H, +, \cdot)$ *dotado de una aplicación*

$$\langle \cdot, \cdot \rangle : H \times H \mapsto \mathbb{R},$$

denominada **producto escalar** *o* **interno***, que cumple las siguientes propiedades:*

i) Simetría. $\langle x, y \rangle = \langle y, x \rangle$ *para todo* $x, y \in H$.

ii) Sesquilinealidad.

$$\begin{aligned} \langle ax, y\rangle &= a\langle x, y\rangle \text{ para todo } x, y \in H \text{y para todo } a \in \mathbb{R}. \\ \langle x + y, z\rangle &= \langle x, z\rangle + \langle y, z\rangle \text{ para todo } x, y, z \in H. \end{aligned}$$

iii) Definida positiva.

$$\langle x, x\rangle \geq 0 \text{ para todo } x \in H.$$

Como todo producto interno induce mediante la expresión $\|x\| = \sqrt{\langle x, x\rangle}$ una norma, los espacios prehilbertianos son casos particulares de espacios normados. Un caso especial son los denominados espacios de Hilbert.

Definición 12 (Espacio de Hilbert) *Un espacio prehilbertiano H se dice que es de Hilbert si el espacio normado $(H, \|\cdot\|)$ con la norma inducida por $\|x\| = \sqrt{\langle x, x\rangle}$ es completo. Esto es, si toda sucesión de Cauchy converge a un elemento del espacio H.*

Los espacios de Hilbert con núcleos reproductores ([Aro50]) representa un marco conceptual para abordar múltiples problemas. En esta sección se emplearán para representar funciones de las cuales se conocen un conjunto discreto de datos. En esta sección describimos los principales resultados teóricos siguiendo la exposición de [GH10].

Definición 13 (Espacio de Hilbert con núcleo reproductor) *Un espacio de Hilbert de funciones H definidas sobre un dominio compacto X es un espacio de Hilbert con núcleo reproductor* RKHS[1]*) si todos los funcionales evaluación ${\ell^2}_x : H \mapsto \mathbb{R}$ definidos por ${\ell^2}_x(f) = f(x)$ son acotados en H. En otras palabras, para cada $x \in X$ existe una constante $M_x > 0$ cumpliendo*

$$|f(x)| \leq M_x \|f\| \text{ para toda función } f \in H, \tag{7.18}$$

donde $\|\cdot\|$ es la norma en el espacio de Hilbert.

Definición 14 (Núcleo de [Mer09]) *Sea X un espacio métrico y $K : X \times X \mapsto \mathbb{R}$ una función continua y simétrica. Si asumimos que K es definida positiva, esto es, para cualquier conjunto $\{x_1, \cdots, x_n\} \subset X$ la matriz $K|_x$ cuyos componentes son $(K|_x)_{ij} = K(x_i, x_j)$ es definida positiva, entonces K es un núcleo de Mercer.*

El teorema de Moore-Aronszajn ([Aro50]) establece una relación biunívoca entre núcleos y espacios RKHS. Para cada espacio RKHS de funciones en X existe un único núcleo reproductor K definido positivo. Por otro lado cualquier espacio RKHS puede ser caracterizado por un núcleo de Mercer.

[1] Se ha emplearemos el acrónimo inglés RKHS de *Reproducing Kernel Hilbert Space* por estar ampliamente difundido en la literatura.

Teorema 1 (Generación de RKHS, [Aro50]) *Sea X un conjunto compacto. Sea $K : X \times X \mapsto \mathbb{R}$ un núcleo de Mercer. Sea $x \in X$, definimos $K_x : X \mapsto \mathbb{R}$ por $K_x(y) = K(x, y)$. Entonces para cada núcleo de Mercer K existe un único espacio RKHS $(\mathcal{H}_K, \langle , \rangle_{\mathcal{H}_K})$ de funciones en X cumpliendo:*

1. *Para todo $x \in X$ se cumple que $K_x \in \mathcal{H}_K$.*
2. *La envoltura lineal de $\{K_x : x \in X\}$ es densa en $\mathcal{H}_K$.*
3. *Para todo $f \in \mathcal{H}_K$ se tiene $\langle K_x, f\rangle_{\mathcal{H}_K} = f(x)$.*

El recíproco de este teorema también se cumple. Dado un RKHS existe un único núcleo de Mercer que lo caracteriza.

Nota 1 (Generación de $\mathcal{H}_K$ a partir de K) *Sea $\mathcal{H}'$ el conjunto de todas las combinaciones lineales finitas de la forma $f(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$ donde $N \in \mathbb{N}$, $x_i \in X$ y $\alpha_i \in \mathbb{R}$ equipado con el producto interno*

$$\langle f, g\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\beta_j K(x_i, x_j) = \alpha^T K\,|_x\, \beta \tag{7.19}$$

donde $f(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$ y $g(x) = \sum_{j=1}^{N} \beta_j K(x_i, x)$. Entonces $\mathcal{H}_K$ es la completitud de $\mathcal{H}'$ con el producto interno asociado. Esto es, se deben añadir a $\mathcal{H}'$ los límites de todas las sucesiones de Cauchy.

Operadores definidos por núcleos

A continuación definiremos una nueva caracterización de los espacios RKHS basada en funciones propias de operadores integrales.

Definición 15 (Operador integral) *Sea $L^2_\nu(X)$ el espacio de las funciones cuyo cuadrado es integrable en X y donde ν es una medida de Borel. Sea $K : X \times X \mapsto \mathbb{R}$ una función continua. Definimos el operador integral (lineal) $L_K : L^2_\nu(X) \mapsto C(X)$ donde $C(X)$ es el conjunto de las aplicaciones continuas en X por*

$$L_K(f)(x) = \int_X K(x, y) f(y) d_\nu(y), \tag{7.20}$$

Esta función está bien definida y la función K se denomina núcleo del operador L_K

Definición 16 (Funciones y valores propios de L_K) *Diremos que ϕ_j es una función propia de L_K y el valor λ_j es su correspondiente valor propio si cumple*

$$L_K(\phi_j) = \lambda_j \phi_j \tag{7.21}$$

es decir,

$$\phi_j(x) = \frac{1}{\lambda_j} \int_X K(x,y)\phi_j(y) d_\nu(y), \tag{7.22}$$

Teorema 2 (Base[2] ortonormal de $L^2_\nu(X)$) *Sea K un núcleo de Mercer entonces el sistema formado por las funciones propias $\{\phi_1, \phi_2, \cdots, \}$ de L_K constituye una base ortonormal de $L^2_\nu(X)$. Esto es $\|\phi_j\|_{L^2_\nu(X)} = 1$ y $\langle \phi_i, \phi_j \rangle_{L^2_\nu(X)} = 0$ si $i \neq j$ y además para cualquier $f \in L^2_\nu(X)$ esta se expresa como $f = \sum_{j=1}^{\infty} \langle f, \phi_j \rangle \phi_j$. La sucesión $\{\lambda_j\}$ es finita o $\lambda_j \to 0$ cuando $j \to \infty$.*

Teorema 3 (Base ortonormal de $\mathcal{H}_K$) *Sea X un conjunto compacto y sea $K : X \times X \mapsto \mathbb{R}$ un núcleo de Mercer. Sea ν una medida de Borel no degenerada[3] en X. Sea $\{\phi_j\}$ el conjunto de funciones propias de L_K y $\{\lambda_j\}$ el correspondiente conjunto de valores propios. Entonces se cumple:*

1. *Si $\lambda_j > 0$ entonces ϕ_j es una función continua y pertenece al espacio RKHS $\mathcal{H}_K$.*
2. *El sistema $\{\sqrt{\lambda_j}\phi_j : \lambda_j > 0\}$ es una base ortonormal de $\mathcal{H}_K$.*

El anterior teorema, debido a que $\mathcal{H}_K$ es independiente de la medida de ν, nos permite caracterizar $\mathcal{H}_K$ del siguiente modo:

- $\dim(\mathcal{H}_K) = +\infty$

$$\mathcal{H}_K = \left\{ f \in L^2_\nu(X) : f = \sum_{j=1}^{\infty} a_j \sqrt{\lambda_j}\phi_j \text{ con } (a_j) \in \ell^2 \right\} \tag{7.23}$$

  donde ℓ^2 es el espacio vectorial de las sucesiones (a_j) que cumplen que $\sum_{j=1}^{\infty} a_j^2 < \infty$.

- $\dim(\mathcal{H}_K) = d$

$$\mathcal{H}_K = \left\{ f \in L^2_\nu(X) : f = \sum_{j=1}^{d} a_j \sqrt{\lambda_j}\phi_j \text{ con } (a_1, \cdots, a_d) \in \mathbb{R}^d \right\} \tag{7.24}$$

En el espacio $\mathcal{H}_K$ el producto escalar se define:

$$\langle f, g \rangle_{\mathcal{H}_K} = \left\langle \sum_{j=1}^{d} a_j \sqrt{\lambda_j}\phi_j, \sum_{i=1}^{d} b_i \sqrt{\lambda_i}\phi_i \right\rangle_{\mathcal{H}_K} = \tag{7.25}$$

$$\sum_{j=1}^{d}\sum_{i=1}^{d} a_j b_i \left\langle \sqrt{\lambda_j}\phi_j, \sqrt{\lambda_i}\phi_i \right\rangle_{\mathcal{H}_K} = \sum_{j=1}^{d} a_j b_j \tag{7.26}$$

[3] Una medida en X es no degenerada si todo subconjunto no vacío de $U \subset X$ cumple $\nu(U) > 0$.

donde $d = \dim(\mathcal{H}_K)$.

Nota 2 (Expresión del producto escalar) *En muchas ocasiones se considera que* $f = \sum_{j=1}^{d} a'_j \phi_j$ *y* $g = \sum_{j=1}^{d} b'_j \phi_j$*, en este caso el producto escalar se expresa:*

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_{j=1}^{d} \frac{a'_j b'_j}{\lambda_j} \tag{7.27}$$

Considérese la siguiente aplicación:

$$\begin{array}{rcl} \Phi : X & \to & \ell^2 \\ x & \mapsto & \Phi(x) = \left(\sqrt{\lambda}_j \phi_j(x) \right)_{j \in \mathbb{N}} \end{array}$$

esta aplicación sólo se conoce implícitamente y permite asignar datos de un espacio finito $X \subset R^P$ a un espacio de dimensión infinita denominado el *espacio de características* y denotado por ℓ^2. Realizar particionamientos lineales (como lo realiza el algoritmo de las $K-$ medias) en el *espacio de características* conduce a particionamientos no lineales en el espacio de datos originales. El siguiente teorema nos va a permitir calcular el producto interno en el conjunto de puntos proyectados en el *espacio de características* sin necesidad de conocer explícitamente la aplicación Φ, este hecho se conoce como *el truco del núcleo.*

Teorema 4 (Teorema de Mercer, [Mer09]) *Sea* X *un conjunto compacto y* ν *una medida no degenerada de Borel en* X *y* $K : X \times X \mapsto \mathbb{R}$ *un núcleo de Mercer. Sea* $\{\phi_j\}_{j \geq 1}$ *y* $\{\lambda_j\}_{j \geq 1}$ *respectivamente las correspondientes funciones y valores propios de* L_K*. Entonces para todo* $x, y \in X$

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(y) = \langle \Phi(x), \Phi(y) \rangle \tag{7.28}$$

donde las series convergen absolutamente para cada $(x, y) \in X \times X$ *y uniformemente en* $X \times X$*.*

APÉNDICE B: Reducción de la dimensionalidad mediante análisis de componenentes principales

Motivación e idea intuitiva

Supongamos que un determinado objeto viene representado por el vector $\mathbf{x} = (x_1, \ldots, x_P)$ y otro objeto por el el vector $\mathbf{y} = (y_1, \ldots, y_P)$. Si quisiéramos calcular la distancia euclídea entre ambos objetos calcularíamos la expresión:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{P} (x_i - y_i)^2} \tag{7.29}$$

El número de operaciones que realizamos es $P - 1$ sumas, P restas y P potencias, por tanto esta operación es de orden $\mathcal{O}(P)$. Si el número de variables P del problema fuera muy grande el coste computacional pudiera a llegar a ser muy elevado. Por ejemplo si aplicásemos un algoritmo jerárquico tendríamos que calcular la distancia entre todos los pares de objetos y por tanto tendríamos que aplicar $\frac{(n-1)n}{2}$ veces la fórmula de la distancia (¿Por qué?). En este ejemplo el coste computacional puede llegar a ser prohibitivo.

Planteemos sobre un ejemplo sencillo el problema de reducción de la dimensionalidad. Supongamos que partimos de la matriz (tabla) de datos

$$X = \begin{bmatrix} 1 & 0 & 3 & 4 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \tag{7.30}$$

En este ejemplo tenemos $n = 3$ objetos y $P = 4$ variables. Nos planteamos definir una única variable que resuma (represente) las cuatro anteriores. El modo más sencillo es considerar una transformación lineal de las variables. Este cálculo se expresa matricialmente por:

$$X \cdot \boldsymbol{\phi} = \begin{bmatrix} 1 & 0 & 3 & 4 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} = \begin{bmatrix} \phi_1 + 3\phi_3 + 4\phi_4 \\ -\phi_1 + 2\phi_3 + \phi_4 \\ \phi_1 + \phi_2 - \phi_4 \end{bmatrix} \tag{7.31}$$

Los parámetros ϕ_i se denominan en inglés *loadings* y dan idea sobre qué peso tiene cada variable en la transformación. Por ejemplo, si consideramos los *loadings* $\phi_1 = \phi_2 = \phi_3 = \phi_4 = \frac{1}{4}$ correspondería a tomar la media de las variables .

De la misma manera nos podemos plantear considerar una segunda transformación lineal de las variables originales, para ello introducimos un segundo subíndice en los parámetros ϕ y expresamos matricialmente las dos transformaciones:

$$X \cdot \boldsymbol{\phi} = \begin{bmatrix} 1 & 0 & 3 & 4 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \\ \phi_{31} & \phi_{32} \\ \phi_{41} & \phi_{42} \end{bmatrix} \tag{7.32}$$

Por ejemplo, podemos considera la media de las dos primeras columnas y la media de las dos últimas columnas, mediante la expresión:

$$X \cdot \boldsymbol{\phi} = \begin{bmatrix} 1 & 0 & 3 & 4 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{7}{2} \\ -\frac{1}{2} & \frac{3}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \tag{7.33}$$

La anterior expresión transforma nuestras cuatro variables en dos. Cada transformación se denomina **componente principal** y los nuevos valores (en columna) $Z = X \cdot \phi$ que toman los datos mediante esta transformación se denominan **scores**.

La cuestión esencial es saber cómo calcular los coeficientes ϕ_{ij} de las componentes principales. Una idea intuitiva nos la proporciona la siguiene situación. Supongamos que se está realizando un análisis cluster y en los datos hay una variable que siempre toma el mismo valor (por ejemplo tenemos una variable que toma el valor 1 si el sexo del individuo corresponde con hombre y todos los sujetos experimentales de la muestra son hombres). La varianza de esta variable es 0. Esta variable no aporta ninguna información a la hora de determinar los grupos ya que toma el mismo valor en todos ellos. No podemos discriminar por si es hombre o no porque todos los individuos lo son. Este ejemplo muestra que una elección adecuada de los anteriores coeficientes es aquella que preserva la máxima varianza de las variables originales. Este es el criterio aplicado por el PCA para determinar los coeficientes. Los *loadings* de la primera componente principal se obtienen como solución al problema de optimización:

$$\begin{array}{ll} \text{maximizar} & \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{P} X_{ij}\phi_{j1}\right)^2 \\ \text{sujeto a:} & \sum_{j=1}^{P}\phi_{j1}^2 = 1 \end{array} \tag{7.34}$$

La función objetivo se deriva de maximizar la suma de la varianza del primer *score* ($\sum_{i=1}^{n}\text{Var}(Z_{i1})$) y la restricción aparece para poder comparar (en igualdad de condiciones) entre todas las combinaciones lineales.

La segunda componente se elige de la misma manera pero una vez descontado el efecto de la primera componente principal. Así sucesivamente.

Aplicaciones del PCA

El algoritmo PCA puede utilizarse por sí solo, o puede servir como técnica de limpieza o preprocesamiento de datos utilizada antes de aplicar un segundo algoritmo de aprendizaje automático. Cuando se utiliza el PCA como parte del preprocesamiento, el algoritmo se aplica para:

1. **Reducción de la dimensionalidad del problema.** Reducir el número de dimensiones en el conjunto de datos, como se ha mostrado en el ejemplo anterior que se pasa de cuatro a dos variables.
2. **Eliminar el ruido de los datos.** Dado que el PCA se calcula encontrando los componentes que explican la mayor cantidad de varianza, capta la señal de los datos y omite el ruido.

Por sí solo, el PCA se utiliza en una gran variedad de casos de uso, por ejemplo:

1. **Visualizar datos multidimensionales.** La proyección de datos multidimensionales en espacios bidimensionales y tridimensionales permiten su visualización. Esta es una herramienta para la visualización de datos multidimensionales.
2. **Comprimir información.** El análisis de componentes principales se utiliza para comprimir la información con el fin de almacenarla y transmitirla de forma más eficiente. Por ejemplo, puede utilizarse para comprimir imágenes sin perder demasiada calidad, o en el procesamiento de señales.

¿Cómo se calculan las PCA?

El método más difundido para su cálculo está basado en la **descomposición de valores propios** de la matriz de varianza-covarianza. Este método consta de los siguientes pasos:

1. **Estandarización de las variables**. Estandarizamos cada variable para que tenga media 0 y una varianza de 1. Abusamos de la notación y denotamos estos datos nuevamente por X.
2. **Cálculo de la matriz de varianza-covarianza** La matriz de varianza-covarianza es la matriz cuadrada, de dimensiones $P \times P$, donde P es el número de variables de nuestra base de datos, definida en este caso por:

$$S = \frac{1}{n-1} X^T X \tag{7.35}$$

donde $\cdot^T$ indica la matriz transpuesta.

3. **Calcular la descomposición de valores propios de la matriz de varianza-covarianza.** Calculamos los vectores propios v_i (vectores unitarios) y sus valores propios asociados λ_i (escalares por los que multiplicamos el vector propio) de la matriz de varianza-covarianza. Esto es, hay que encontrar los $i = 1, \ldots, P$ pares número-vector (λ_i, v_i) cumpliendo:

$$Sv_i = \lambda_i v_i \tag{7.36}$$

   Estos vectores propios son los *loadings*, i.e $\phi_{\cdot i} = v_i$.

4. **Ordena los vectores propios** desde el valor propio más alto hasta el más bajo. El vector propio v_1 corresponde con el valor propio λ_1 más alto y él es la primera componente principal. Así sucesivamente, $\lambda_1 \geq \lambda_2 \ldots, \lambda_P$.

5. **Seleccione el número de componentes principales.** Los valores propios informan de la cantidad de varianza recogida en las componentes principales. La expresión:

$$\frac{\lambda_j}{\sum_{i=1}^{P} \lambda_i} \tag{7.37}$$

   representa la **proporción de varianza explicada** (PVE) por la $j-$ésima componente principal. Se puede multiplicar por 100 para expresarla en forma de porcentaje.

   Se seleccionan las m primeras componentes principales. El número óptimo m de componentes principales es subjetivo y depende del problema. Por lo general, nos fijamos en la cantidad acumulada de varianza explicada por la combinación de componentes principales

$$100 * \sum_{j=1}^{m} \frac{\lambda_j}{\sum_{i=1}^{P} \lambda_i} \tag{7.38}$$

   y elegimos el número de componentes donde el porcentaje acumulado recogido en la expresión (7.38) sea significativo en el contexto del problema. Valores de $m \leq 3$ permiten una representación gráfica de los *scores*, ya sea utilizando dos o tres dimensiones.

¿Cuáles son las ventajas e inconvenientes de la técnica PCA?

Las principales ventajas del PCA son:

- **Fácil de realizar.** El PCA se basa en métodos básicos del álgebra lineal. Estas técnicas están ampliamente difundidas y se recogen en el software dedicado al aprendizaje automático.

- **Acelera algoritmos de aprendizaje automático.** Los algoritmos de aprendizaje automático convergen más rápidamente cuando se entrenan con componentes principales en lugar de los datos originales. Esto es debido fundamentalemene a que el número de variables del problema de entrenamiento se reduce.

- **Palía algunos de los problemas de la alta dimensionalidad de los datos.** La alta dimensionalidad de los datos hace que los algoritmos basados en la regresión se sobreajusten fácilmente. Al utilizar el PCA se elimina ruido de los datos y se reduce la dimensión del conjunto de datos de entrenamiento, evitando así el sobreajuste de los modelos de regresión.

Las desventajas prinicipales del PCA:

- **Baja interpretabilidad de los componentes principales**. Los componentes principales son combinaciones lineales de las características de los datos originales, pero esta ponderación de variables no es fácil de interpretar. Por ejemplo, después de calcular los componentes principales es difícil saber cuáles son las variables más importantes del conjunto de datos original.

- **Equilibrio entre la pérdida de información y la reducción de la dimensionalidad.** La pérdida de información es un peaje que hay que pagar por emplear el PCA. El modelador debe establecer un equilibrio entre la reducción de la dimensionalidad y la pérdida de información.

Bibliografía

[Aro50] N. Aroszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[BEF84] J.C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2-3):191–203, 1984.

[Cox57] D. R. Cox. Note on grouping. *Journal of the American Statistical Association*, 52:543–547, 1957.

[DDW02] E. Dimitriadou, S. Dolnicar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1)(1):137–160, 2002.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[GG92] A. Gersho and R.M. Grey. *Vector quantizacion and signal compression*. Kuwer Academic, Boston, 1992.

[GH10] J. González-Hernández. *Representing Functional Data in Reproducing Kernel Hilbert Spaces with Applications to Clustering, Classification and Time Series Problems*. PhD thesis, Department of Statistics, Universidad Carlos III, Getafe, Madrid, 2010.

[HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, December 2001.

[HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[KR90] L. Kaufman and P. Rousseeuw. *Finding groups in data: An introduccion to cluster*. Wiley, New York, 1990.

[LT18] Z. Li and Y. Tang. Comparative density peaks clustering. *Expert Systems with Applications*, 95:236–247, 2018.

[LWY18] R. Liu, H. Wang, and X. Yu. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. *Information Sciences*, 450:200–226, 2018.

[Mac67] J. MacQueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1. L. M. Le Cam and J. Neyman (Eds.). Berkeley, CA: University of California Press*, pages 281–297, 1967.

[MC85] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the numbers of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[MC12] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[Mer09] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A*, 209:415–446, 1909.

[Mil80] G.W. Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Phichometrika*, (45):181–204, 1980.

[RL14] A. Rodríguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[Spa80] *Cluster analysis algorithms for data reduction and classification of objects*. Wiley, New York, 1980.

[SS63] R. Sokal and P.H.A. Sneath. *Principles of numerical taxonomy*. San Francisco: W.H. Freeman., 1963.

[SS71] A.J. Scott and M.J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, (27):387–398, 1971.

[Ste03] D. Steinley. Local optima in k-means clustering: What you don't know may hurt you. *Psychological Methods*, 8(3):294–304, 2003.

[Sym81] M.J. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, (37):35–43, 1981.

[TSK16] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[TWH01] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusterin a data set via the gap statistic. *Journal of the Royal Statistical Society B*, (63):411–423, 2001.

[VCH10] L. Vendramin, R.J.G.B. Campello, and E.R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209–235, 2010.

[Win87] M.P. Windham. Parameter modification for clustering criteria. *Journal of Classification*, (4):191–214, 1987.

[XC07] J. Xia and M. Chen. A nested clustering technique for freeway operating condition classification. *Computer-Aided Civil and Infrastructure Engineering*, 22(6):430–437, 2007.