Aprendizaje no supervisado: clustering y reducción de la dimensionalidad

10th April 2019

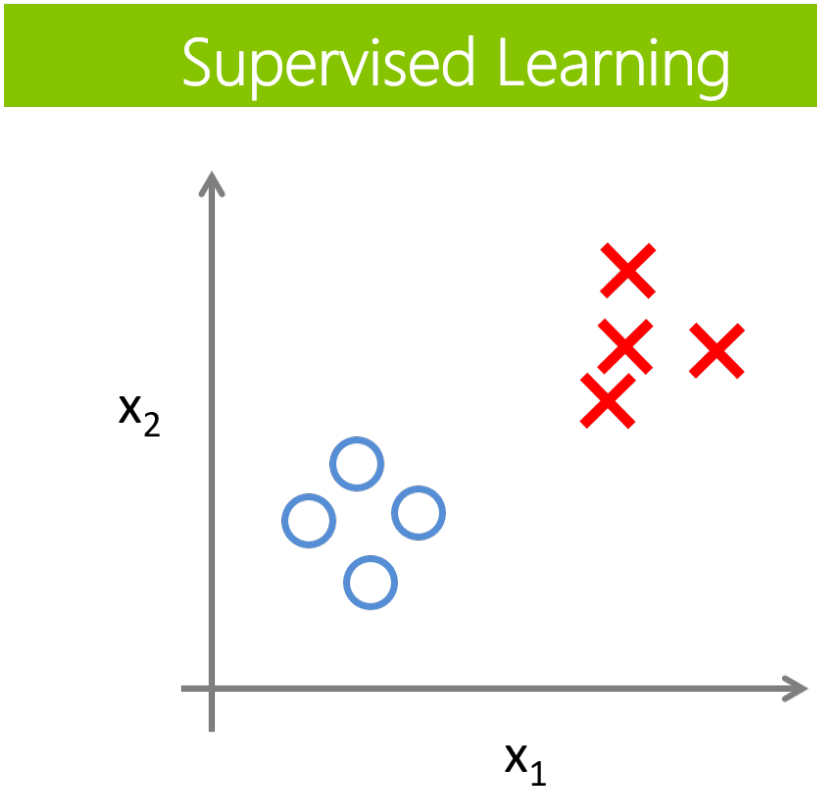
Outline

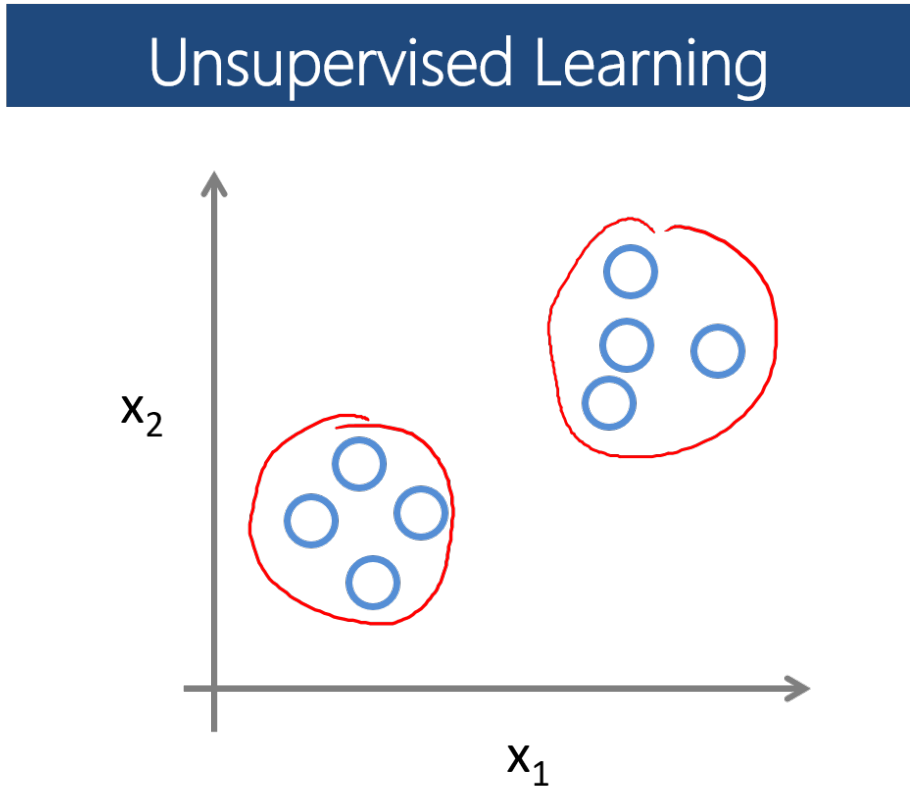
Introducción

¿Aprendizaje no supervisado?

Dado un input de *features* $\{x_1, x_2, \ldots, x_m\}$, El aprendizaje no supervisado consiste en encontrar patrones subyacentes en los datos, sin ningún tipo de constraint externo (ésta es la principal diferencia con el aprendizaje supervisado). Además, permite encuentrar subgrupos naturales, que presentan propiedades similares entre si, y comprimir los datos a lo largo de los patrones encontrados para reducir la dimensionalidad del problema.

Supervisado versus No Supervisado





Ejemplos

NETWORK NEUROSCIENCE

an open access journal

Check for updates

Citation: Rasero, J., Pellicoro, M., Angelini, L., Cortes, J. M., Marinazzo, D., & Stramaglia, S. (2017). Consensus clustering approach to group brain connectivity matrices. *Network Neuroscience*, *1*(3), 242–253. https://doi.org/10.1162/netn_a_00017

DOI: https://doi.org/10.1162/netn_a_00017

METHODS

Consensus clustering approach to group brain connectivity matrices

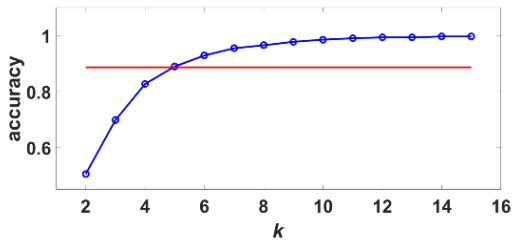
Javier Rasero[1,2,3], Mario Pellicoro[2], Leonardo Angelini[2,3,4], Jesus M. Cortes[1,5], Daniele Marinazzo[6], and Sebastiano Stramaglia[2,3,4]

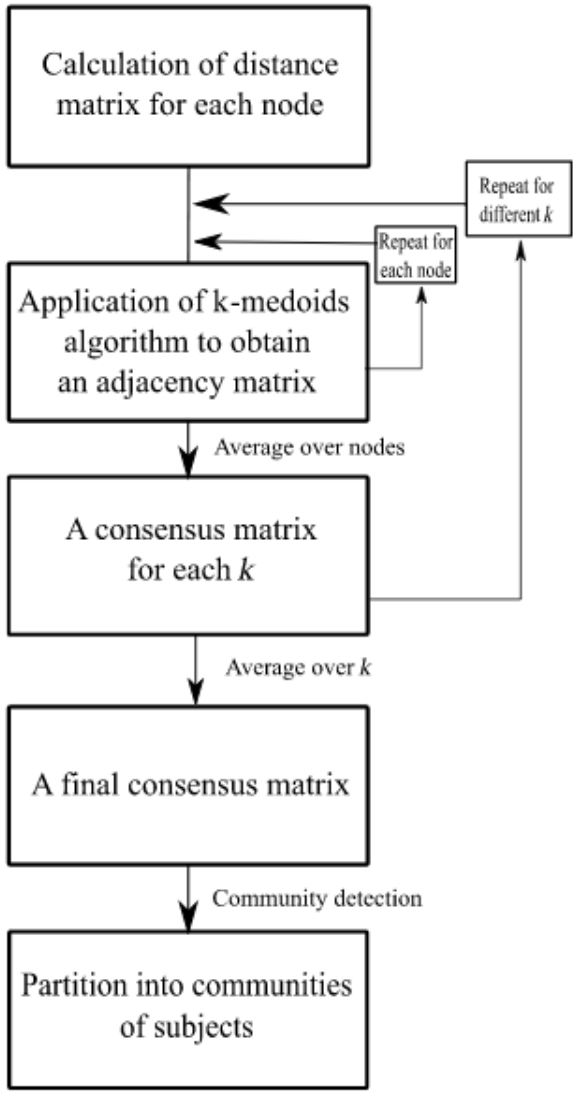
[1]Biocruces Health Research Institute. Hospital Universitario de Cruces, Barakaldo, Spain
[2]Dipartimento di Fisica, Università degli Studi Aldo Moro, Bari, Italy
[3]Istituto Nazionale di Fisica Nucleare, Sezione di Bari, Italy
[4]TIRES-Center of Innovative Technologies for Signal Detection and Processing, Università degli Studi Aldo Moro Bari, Italy
[5]Ikerbasque, the Basque Foundation for Science, Bilbao, Spain
[6]Faculty of Psychology and Educational Sciences, Department of Data Analysis, Ghent University, Ghent, Belgium

Keywords: Unsupervised learning, Consensus clustering, Resting fMRI, Structural DTI

ABSTRACT

A novel approach rooted on the notion of *consensus* clustering, a strategy developed for community detection in complex networks, is proposed to cope with the heterogeneity that characterizes connectivity matrices in health and disease. The method can be summarized as follows: (a) define, for each node, a distance matrix for the set of subjects by comparing the connectivity pattern of that node in all pairs of subjects; (b) cluster the distance matrix for each node; (c) build the consensus network from the corresponding partitions; and (d) extract groups of subjects by finding the communities of the consensus network thus obtained. Different from the previous implementations of consensus clustering, we thus propose to use the consensus strategy to combine the information arising from the connectivity patterns of each node. The proposed approach may be seen either as an exploratory technique or as an unsupervised pretraining step to help the subsequent construction of a supervised classifier. Applications on a toy model and two real datasets show the effectiveness of the proposed methodology, which represents heterogeneity of a set of subjects in terms of a weighted network, the consensus matrix.





Ejemplos

SCIENTIFIC REPORTS

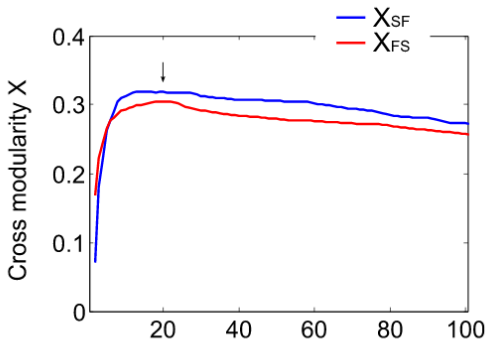
OPEN

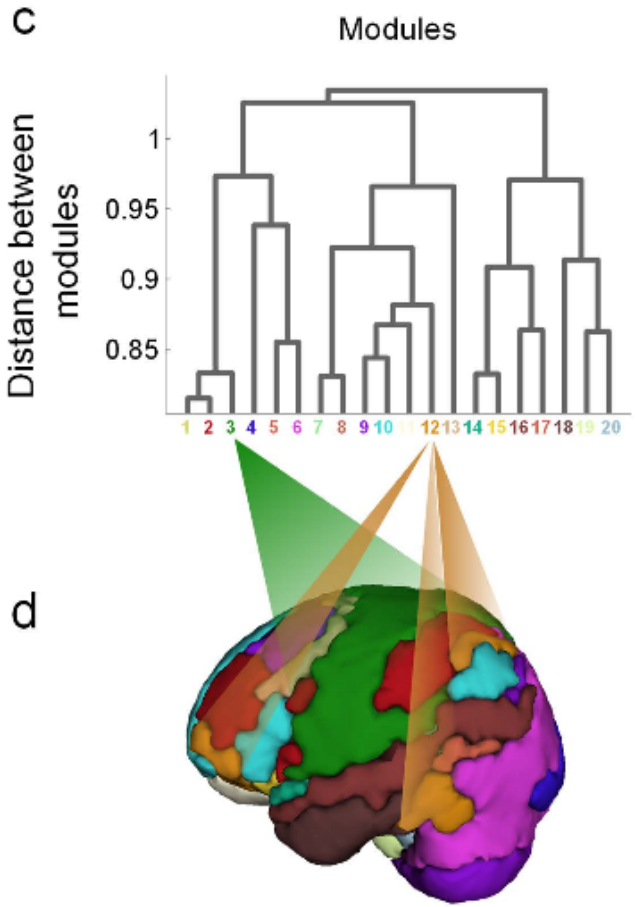
Received: 28 November 2014
Accepted: 23 April 2015
Published: 03 June 2015

A novel brain partition highlights the modular skeleton shared by structure and function

Ibai Diez[1,*], Paolo Bonifazi[2,*], Iñaki Escudero[1,3], Beatriz Mateos[1,3], Miguel A. Muñoz[4], Sebastiano Stramaglia[1,5,6,+] & Jesus M Cortes[1,6,7]

Elucidating the intricate relationship between brain structure and function, both in healthy and pathological conditions, is a key challenge for modern neuroscience. Recent progress in neuroimaging has helped advance our understanding of this important issue, with diffusion images providing information about structural connectivity (SC) and functional magnetic resonance imaging shedding light on resting state functional connectivity (rsFC). Here, we adopt a systems approach, relying on modular hierarchical clustering, to study together SC and rsFC datasets gathered independently from healthy human subjects. Our novel approach allows us to find a common skeleton shared by structure and function from which a new, optimal, brain partition can be extracted. We describe the emerging common structure-function modules (SFMs) in detail and compare them with commonly employed anatomical or functional parcellations. Our results underline the strong correspondence between brain structure and resting-state dynamics as well as the emerging coherent organization of the human brain.





Clustering

Objetivo

Partir las observaciones en subgrupos (clusters), tal que la similaridad entre las observaciones pertenecientes al mismo cluster es mayor que aquéllas en diferentes clusters.

Tipos de clustering

1. Hard Clustering, en el que cada observación pertenece a un cluster.
2. Soft Clustering, que da una probabilidad de pertenencia de las observaciones a cada cluster.

Elementos de un clustering

- Matriz de (dis)similaridad. Suele ser representada por una matriz de distancias D de tamaño $N \times N$, donde N como siempre es el número de observaciones.
- Para cada feature, tenemos una métrica de similaridad entre cada par de observaciones $d_j(x_{ij}, x_{i'j})$.
- La similaridad de dos observaciones viene dado entonces por:

$$D(x_i, x_{i'}) = \sum_{j=1}^{m} w_j d_j(x_{ij}, x_{i'j}) \tag{1}$$

- Los algoritmos de clustering se diferencian en la elección de la métrica que define la matriz D.

Algoritmos de clustering (en scikit)

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Clustering: Optimización de distancias

La pertenencia de las observaciones a los clusters se obtiene minimizando las distancia de los puntos pertenecientes a un mismo cluster (within cluster distance)

$$W \propto \sum_{k=1}^{K} \sum_{i \in k} \sum_{i' \in k} d(x_i, x_{i'}) \tag{2}$$

o maximizando la distancia entre puntos de diferente cluster (between cluster distance)

$$B \sim \sum_{k=1}^{K} \sum_{i \in k} \sum_{i' \ni k} d(x_i, x_{i'}) \tag{3}$$

K-means

- Se basa en la distancia euclidea entre las observaciones

$$d(x_i, x_{i'}) = \sum_{j=1}^{m} (x_{ij} - x_{i'j})^2 = |x_{ij} - x_{i'j}|^2 \tag{4}$$

- La distancia de las observaciones dentro del cluster es

$$W \propto \sum_{i \in k} |x_i - \mu_k|^2 \tag{5}$$

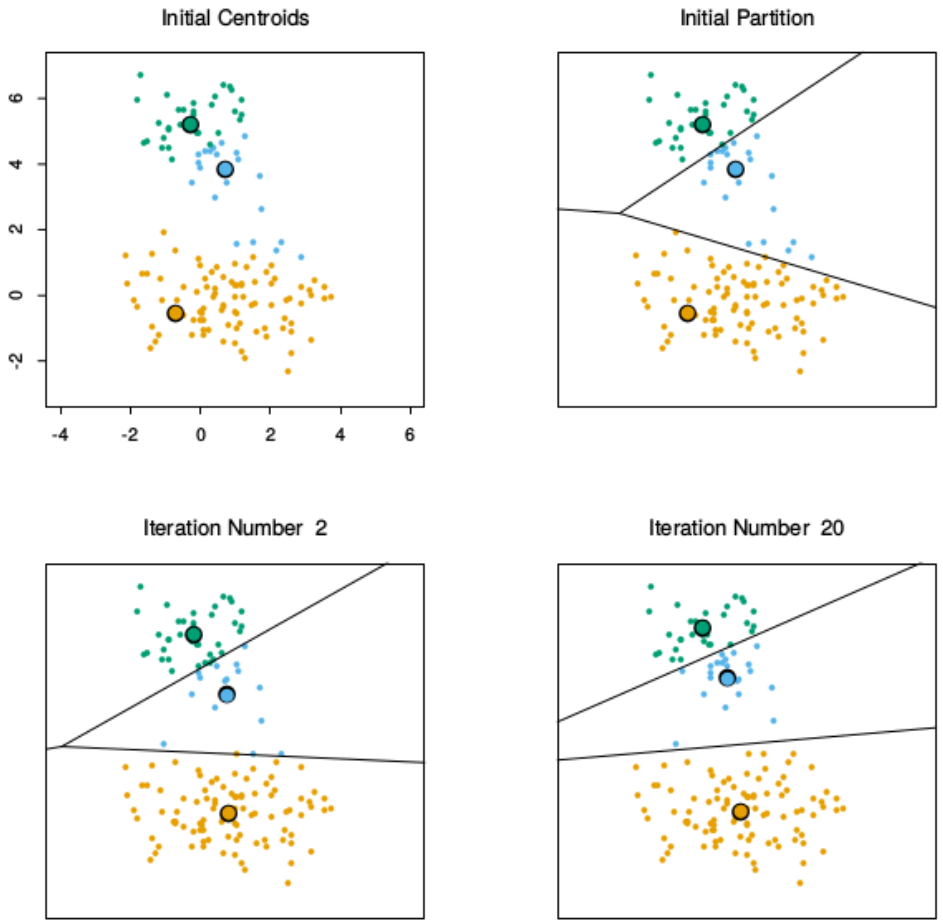
  donde μ_k define las coordenadas del centroide de dicho cluster K.
- K-means busca minimizar esta cantidad asignando cada observación al cluster K dado por su centroide más cercano.

K-means

Este algorimo se implementa de la siguiente manera:

1. Se eligen arbitráreamente los k centroides $\mathcal{C} = (\mu_1, \mu_2, \ldots \mu_k)$.
2. Para cada $i \in 1, \ldots, k$, se define el cluster $\mathcal{C}_i$ como el conjunto de puntos más próximos a μ_i.
3. Para cada $i \in 1, \ldots, k$, se actualizan los centros tomando la media de los puntos pertenecientes a cada cluster $\mu_i = \frac{1}{N_{c_i}} \sum_{i \in \mathcal{C}_i} x_i$.
4. Se repiten los dos pasos anteriores hasta que no haya más cambios.

En cada paso, W se va reduciendo, aunque puede pasar que caigamos en un mínimo local debido a la elección del punto inicial. Por ello, es conveniente correr el algoritmo con varios puntos iniciales diferentes y elegir la solución con el menor valor de W.



En scikit, se puede encontrar en **cluster.KMeans**

Gaussian Mixtures

- Puede verse como un soft K-means.
- La probabilidad total de cada observación viene dada por

$$\mathcal{L} = \prod_i p(x_i) \tag{6}$$

$$p(x_i) \propto \sum_k \alpha_i p(x_i|\mu_k, \Sigma_k)$$

$$\propto \sum_k \alpha_i \mathcal{N}_i(\mu_k, \Sigma_k) \tag{7}$$

- Es decir, que cada cluster viene representado por un centroide μ_k y una matriz de covarianza Σ_k.
- Cada observación es asignada una probabilidad de pertenencia a cada cluster como

$$p(k|x_i) = \mathcal{N}_i(x_i|\mu_k, \Sigma_k) \tag{8}$$

- Cada observación es asignada al cluster con probabilidad mayor.

Gaussian Mixtures

La forma de calcular μ y Σ (y por tanto las probabilidades de pertenencia a cada cluster) es parecido a k-means, maxímizando en este caso $\mathcal{L}$

1. Se toma un valor inicial para μ_k, Σ_k y α_k
2. Se calcula un nuevo $p(k|x_i)$ y nuevo $\mathcal{L}$

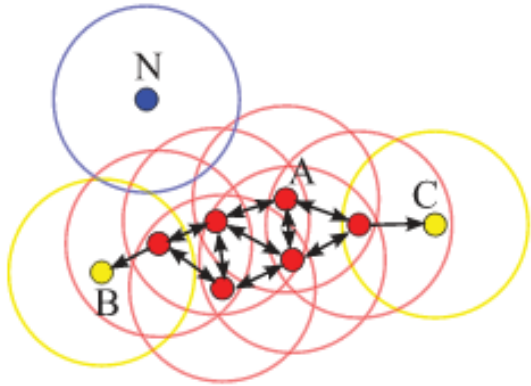
$$\mu_k \rightarrow \mu_k = \frac{\sum_i p(k|x_i) x_i}{\sum_i p(k|x_i)} \tag{9}$$

$$\Sigma_k \rightarrow \Sigma_k = \frac{\sum_i p(k|x_i)(x_i - \mu_k)^T (x_i - \mu_k)}{\sum_i p(k|x_i)} \tag{10}$$

En scikit, se puede encontrar en **mixture.GaussianMixture**

DBSCAN

- Considera los clusters como áreas de alta densidad separadas por áreas de baja.
- La densidad está definida por el número de *minPts* y y el radio ϵ.
- Un core point es un punto dentro de un objeto con más de *minPts*
- Border points son aquéllos puntos conectados con algún core point, pero no forma parte de un cluster.
- Noise point son aquellos no conectados con ningún punto core



DBSCAN

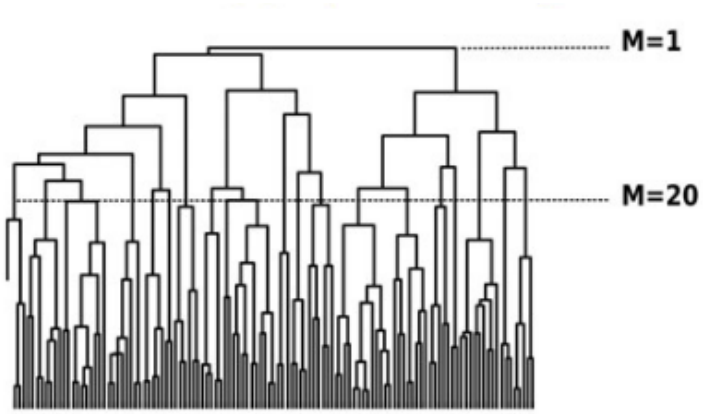
El algoritmo funciona de la siguiente forma:

1. Calcular dentro del radio ϵ los vecinos de cada punto.
2. Identificar como core points aquéllos que tengan mas de *minPts* vecinos.
3. Identificar los core points como un cluster.
4. Asignar cada border point al cluster vecino.
5. Los noise points se quedan como tal.

En scikit: **cluster.DBSCAN**

Hierarchical Clustering

- A diferencia de K-means, hierarchical clustering no requiere elección de antemano del número de clusters.
- Basado en la métrica de disimilaridad entre grupo de observaciones, produce clusters en multi-escala.



- Puede ser aglomerativo (bottom-up) o divisivo (top-down)

Agglomerative Clustering

- Empieza con cada observación representando un solo clúster.
- Se escoge la métrica de la distancia (euclidea, coseno, manhattan...)
- Se van uniendo observaciones con la distancia más pequeña al cluster según el *linkage*:
  1. Ward, la diferencia entre distancias de puntos (la varianza) dentro de cada cluster.

$$D \equiv min \sum_{k} \sum_{i \in c_k} \sum_{i' \in c_k} d_{ii'} \tag{11}$$

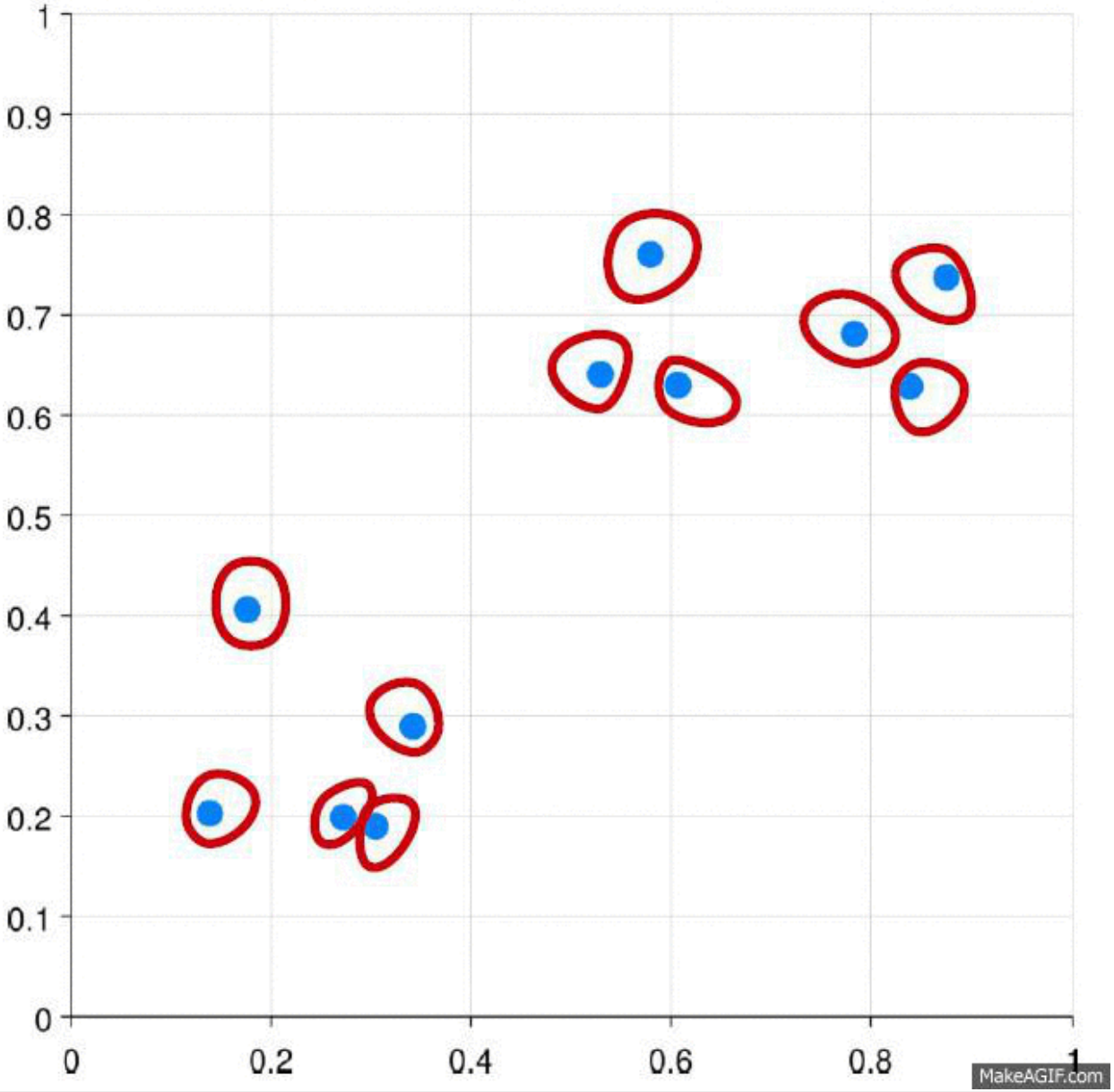
  2. Average, la distancia media entre par de clusters

$$D(c_1, c_2) \equiv max \frac{1}{N_{c_1}} \frac{1}{N_{c_2}} \sum_{i \in c_1} \sum_{i' \in c_2} d_{ii'} \tag{12}$$

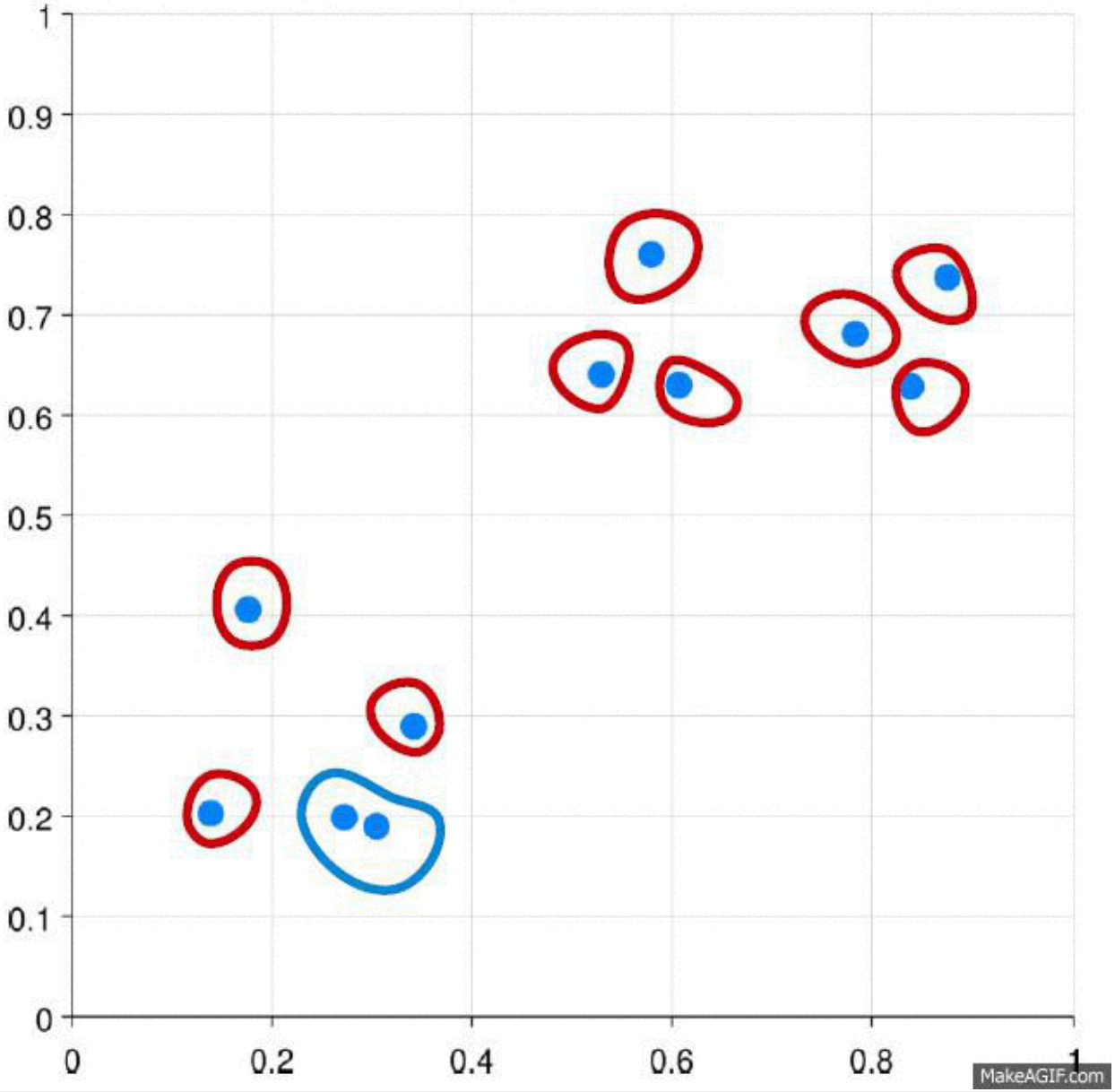
  3. Complete, la maxima distancia entre observaciones de pares de clusters

$$D(c_1, c_2) \equiv max_{i \in c_1, i' \in c_2} d_{ii'} \tag{13}$$

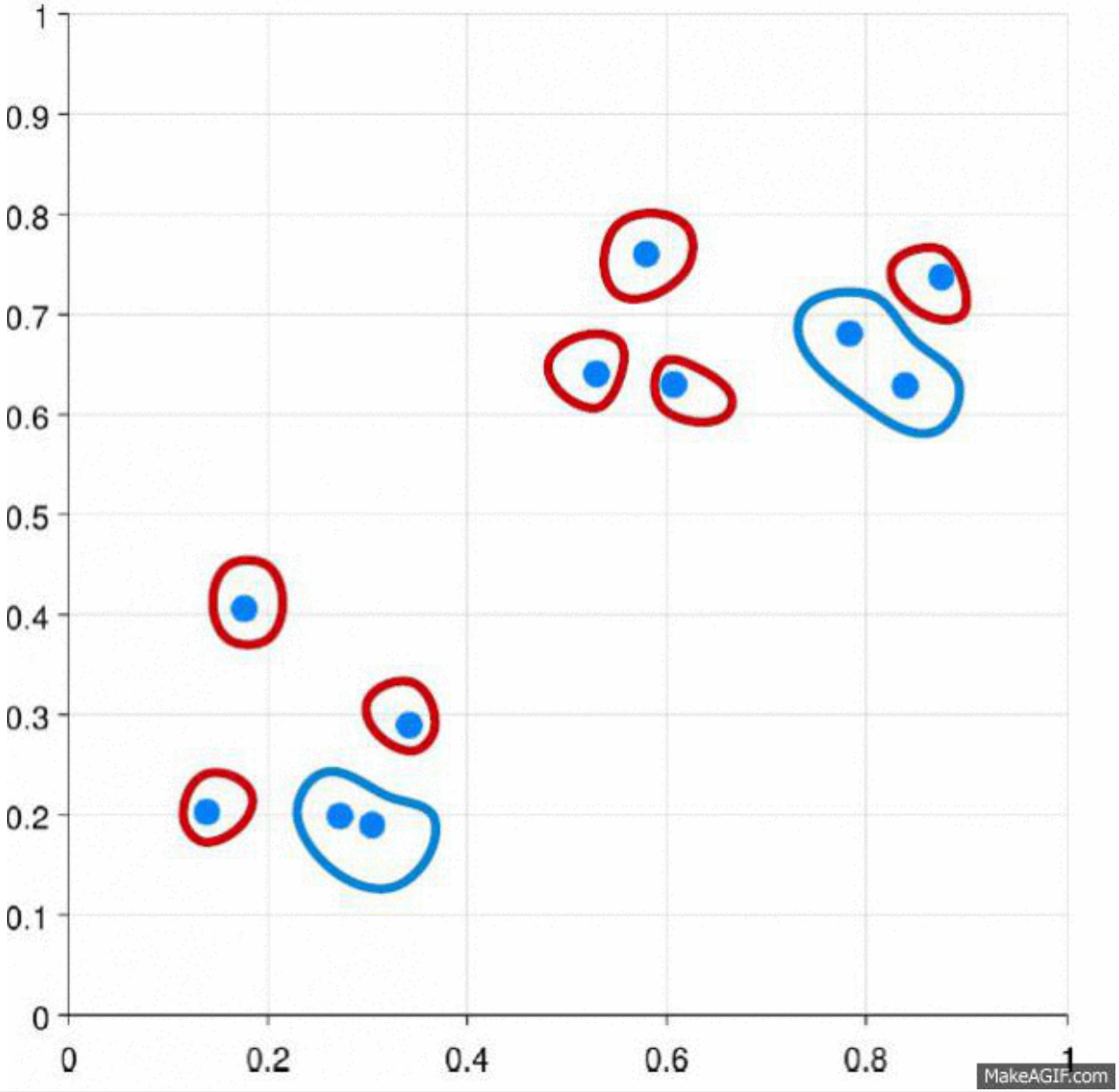
Agglomerative Clustering



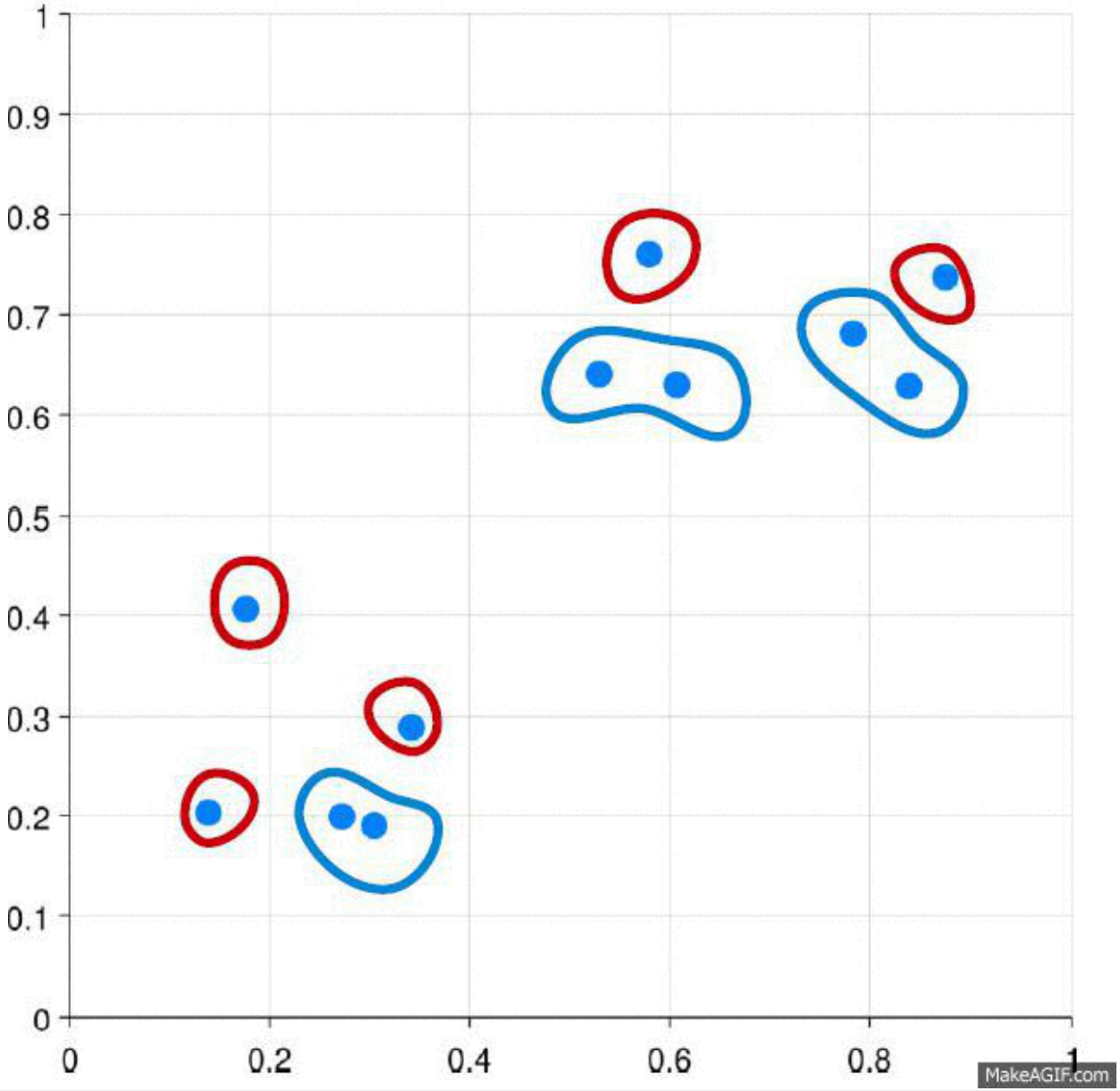
Agglomerative Clustering



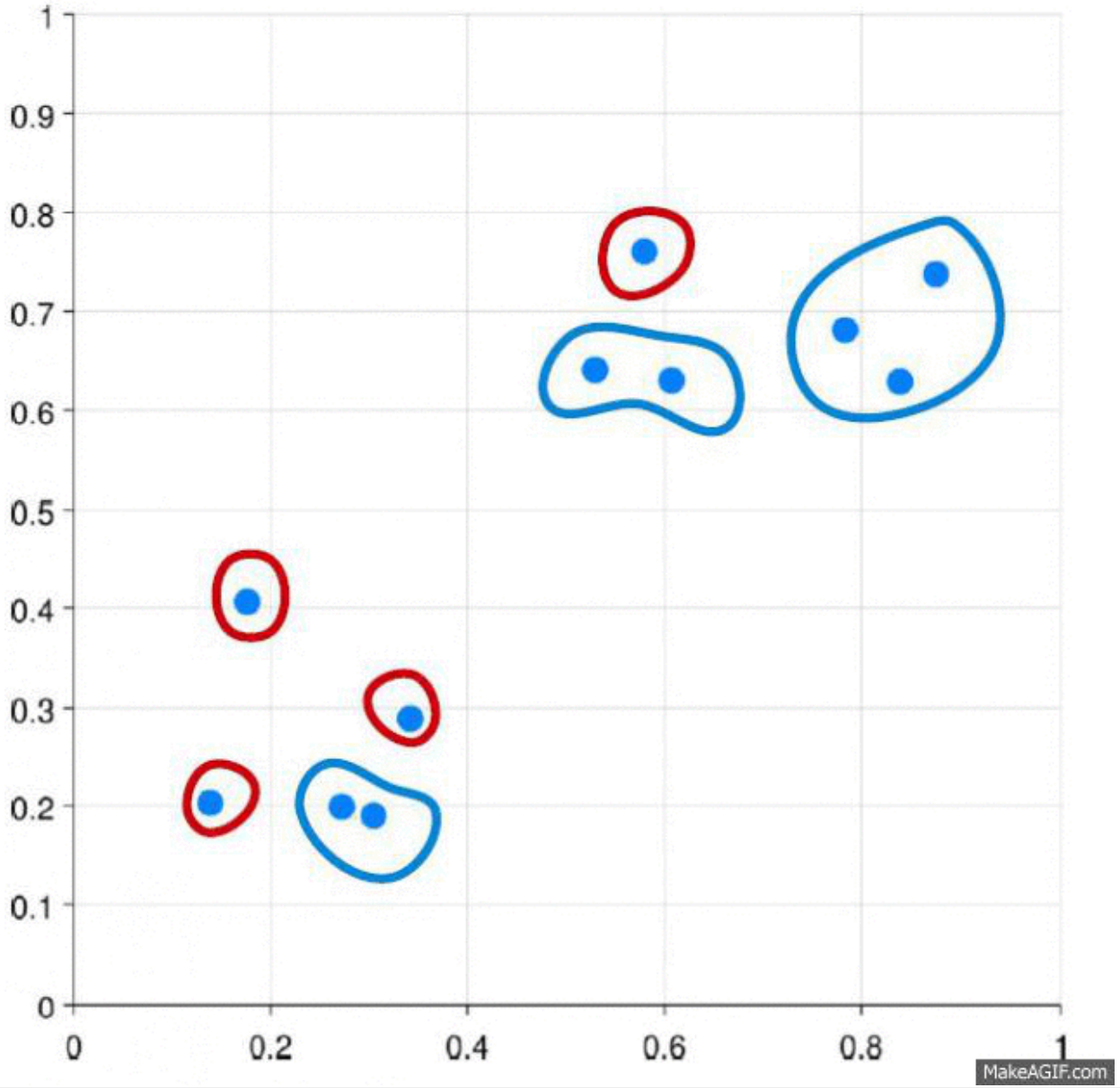
Agglomerative Clustering



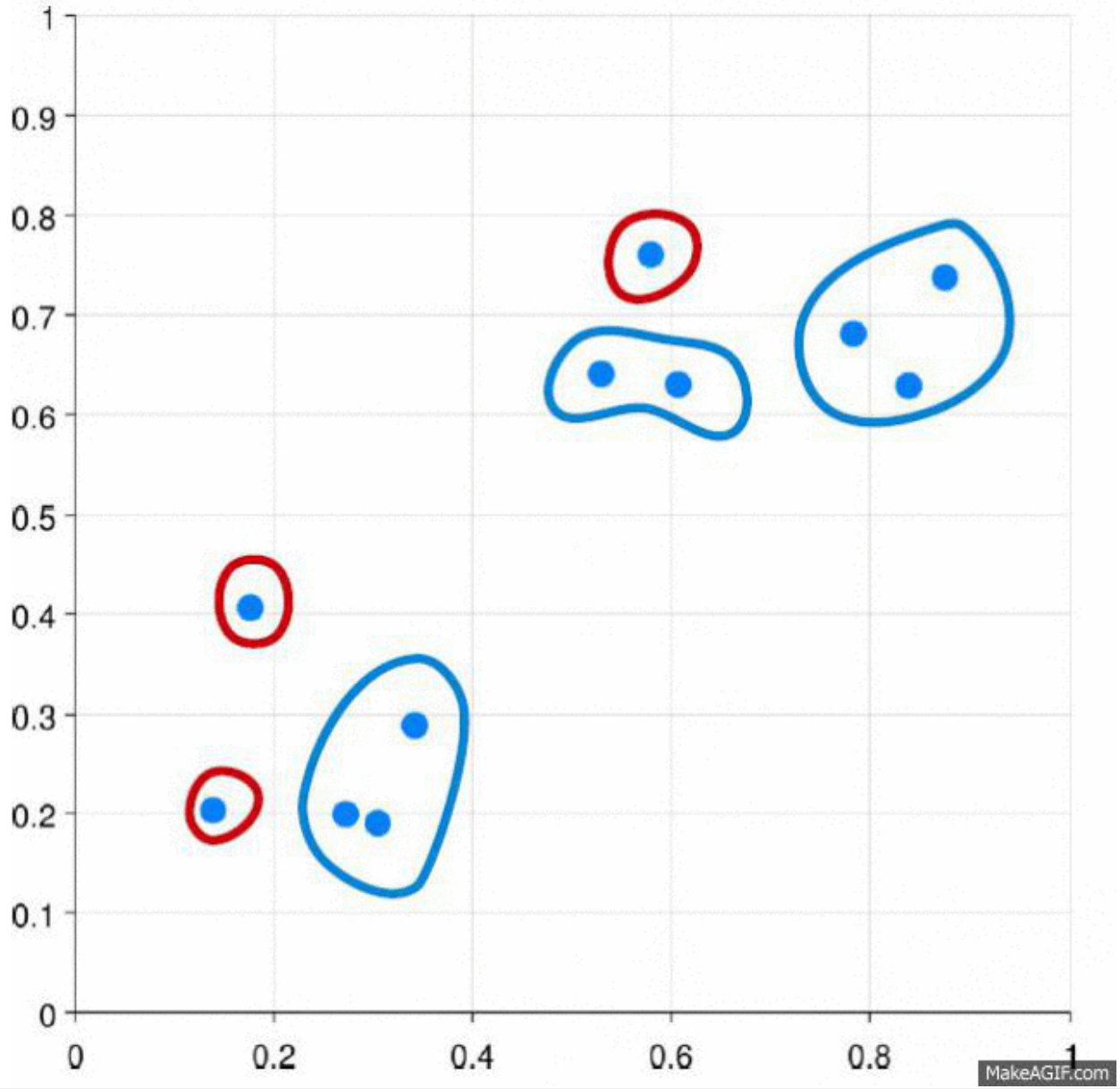
Agglomerative Clustering



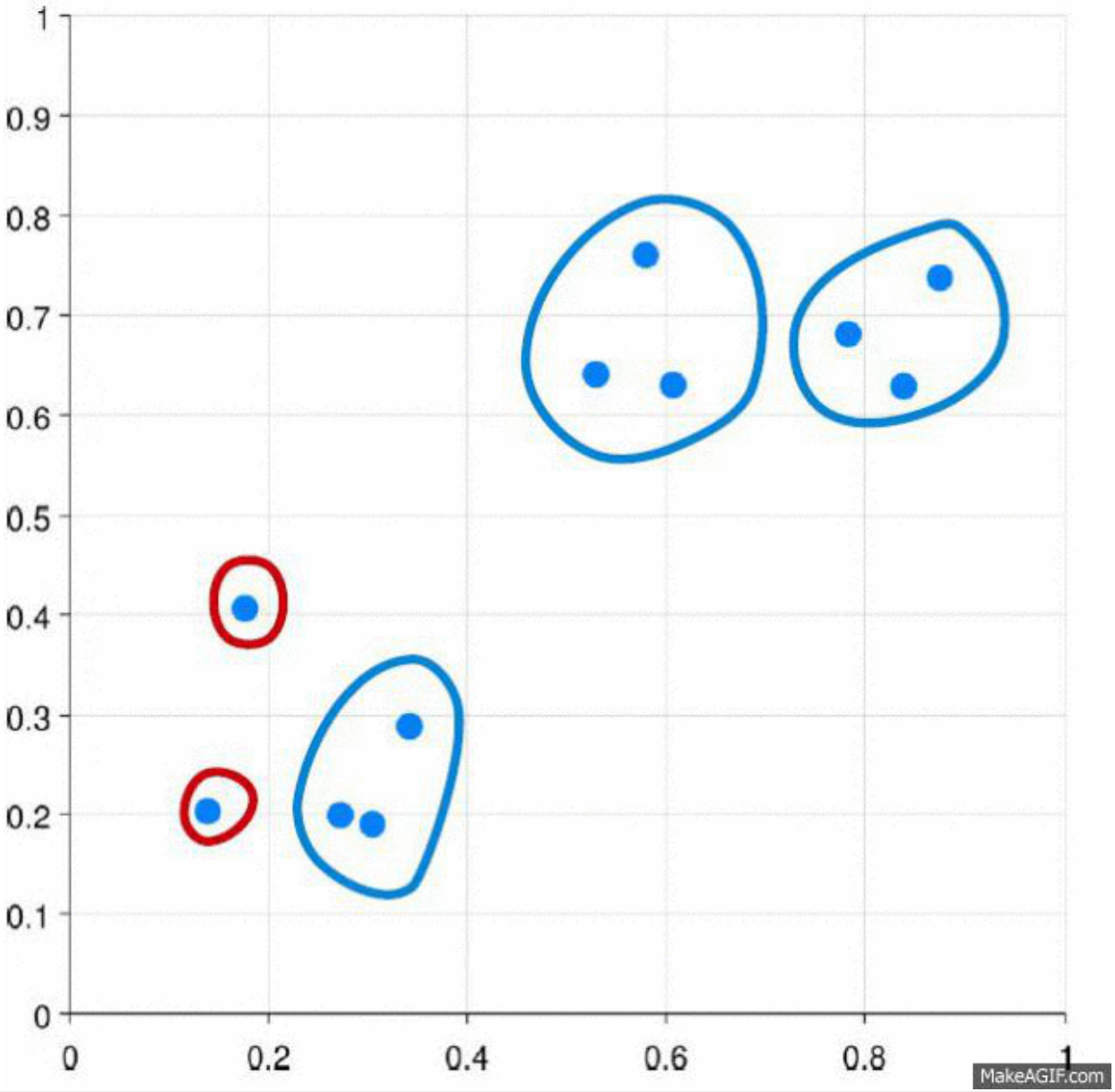
Agglomerative Clustering



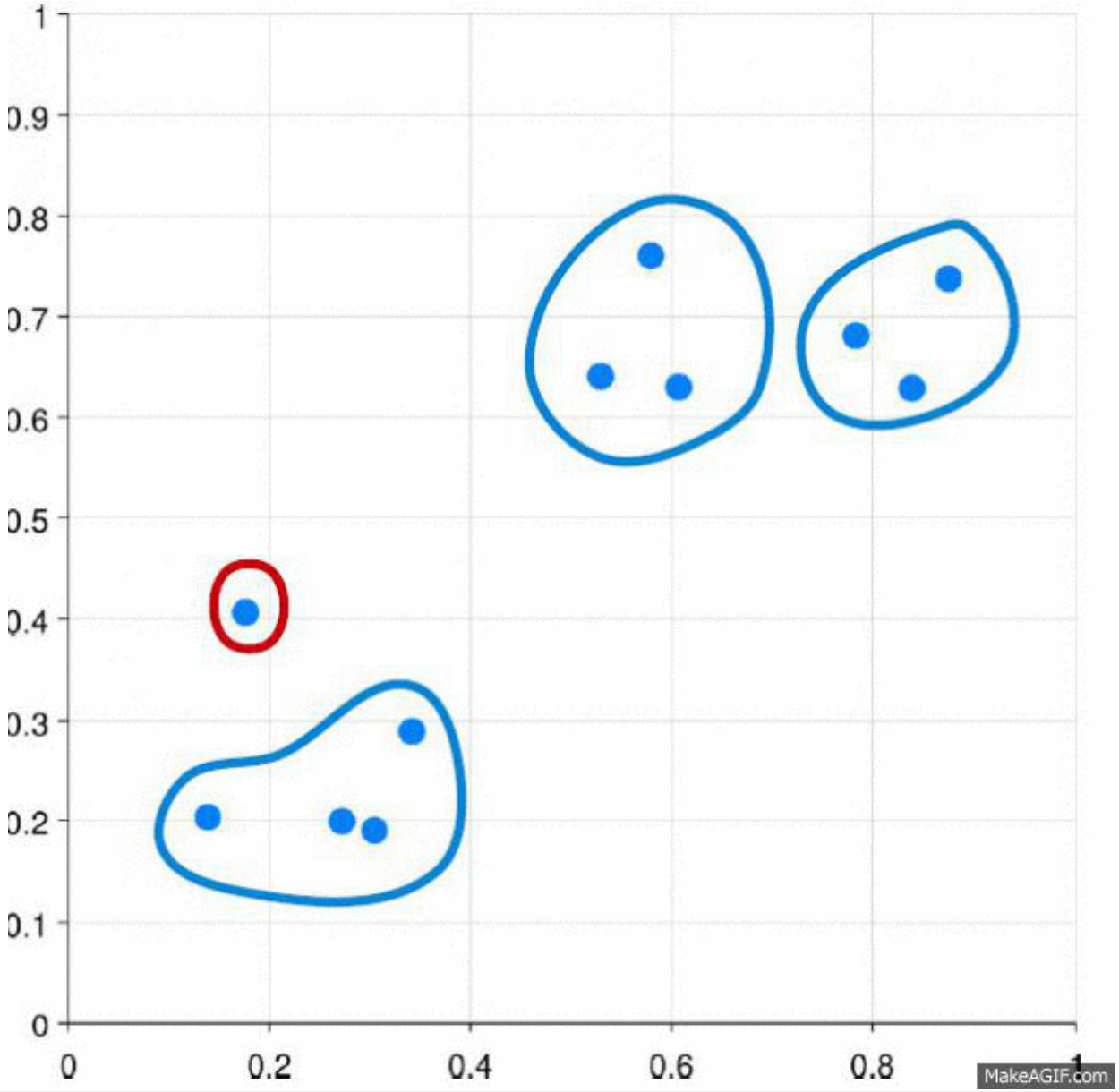
Agglomerative Clustering



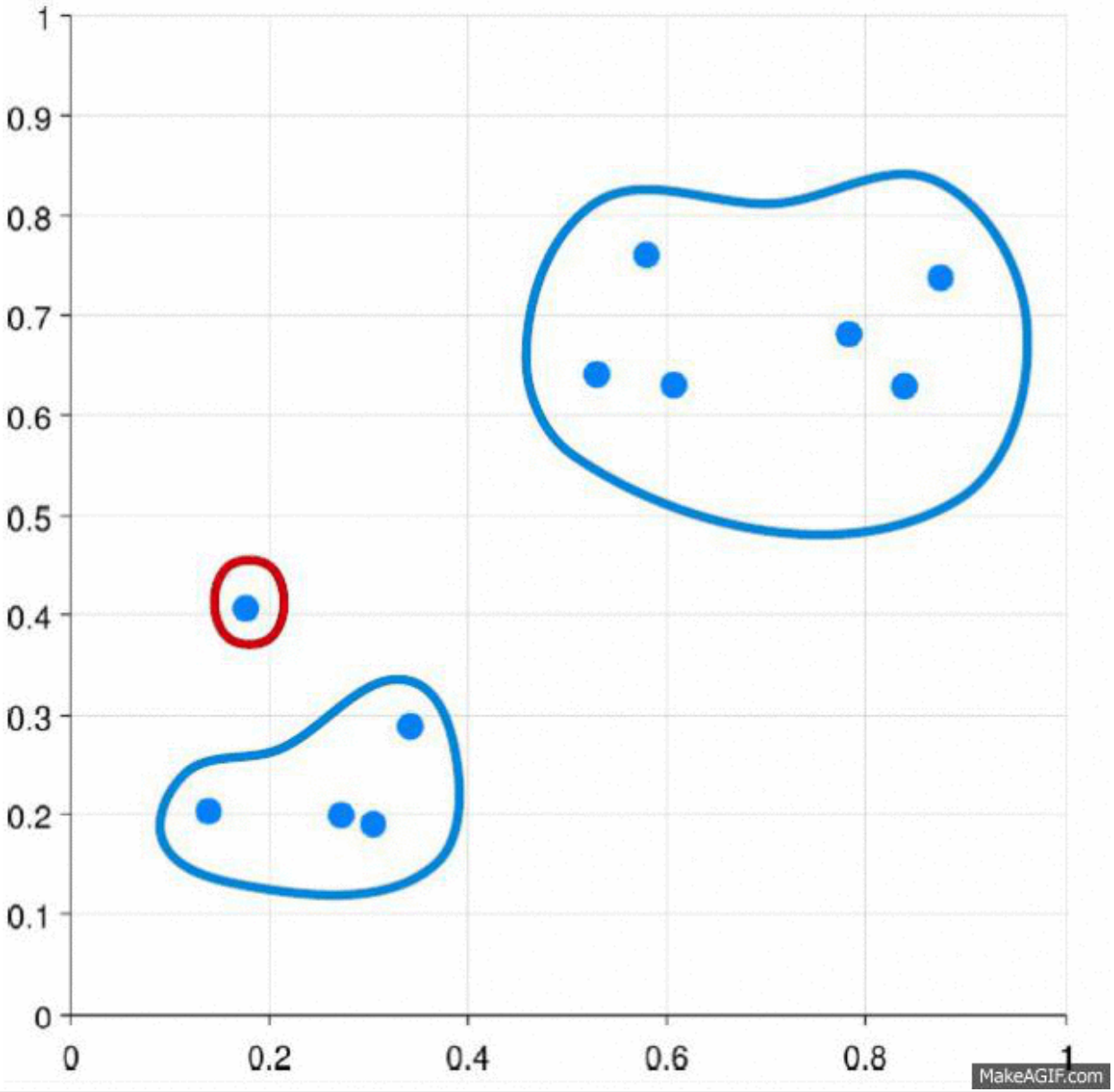
Agglomerative Clustering



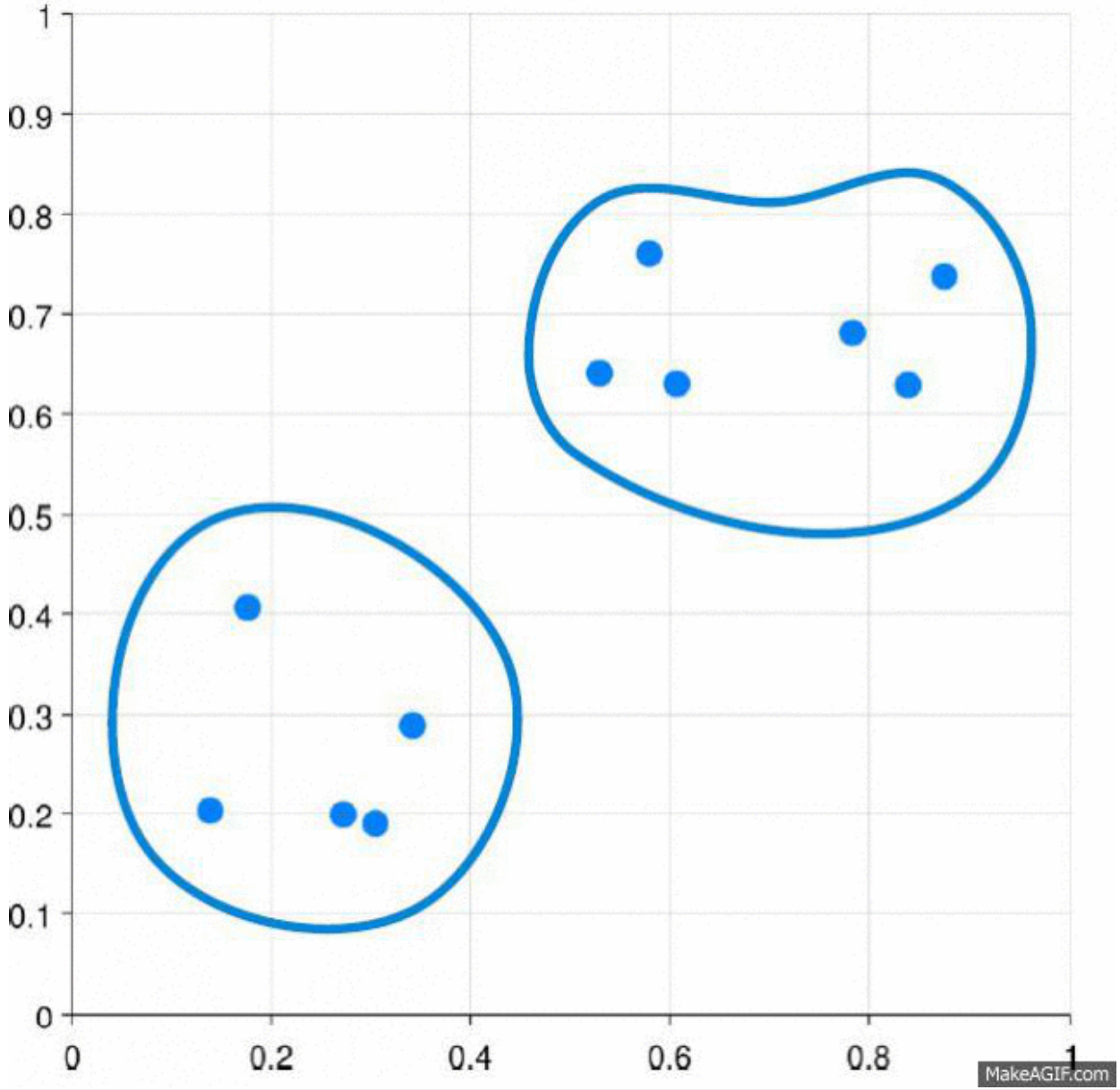
Agglomerative Clustering



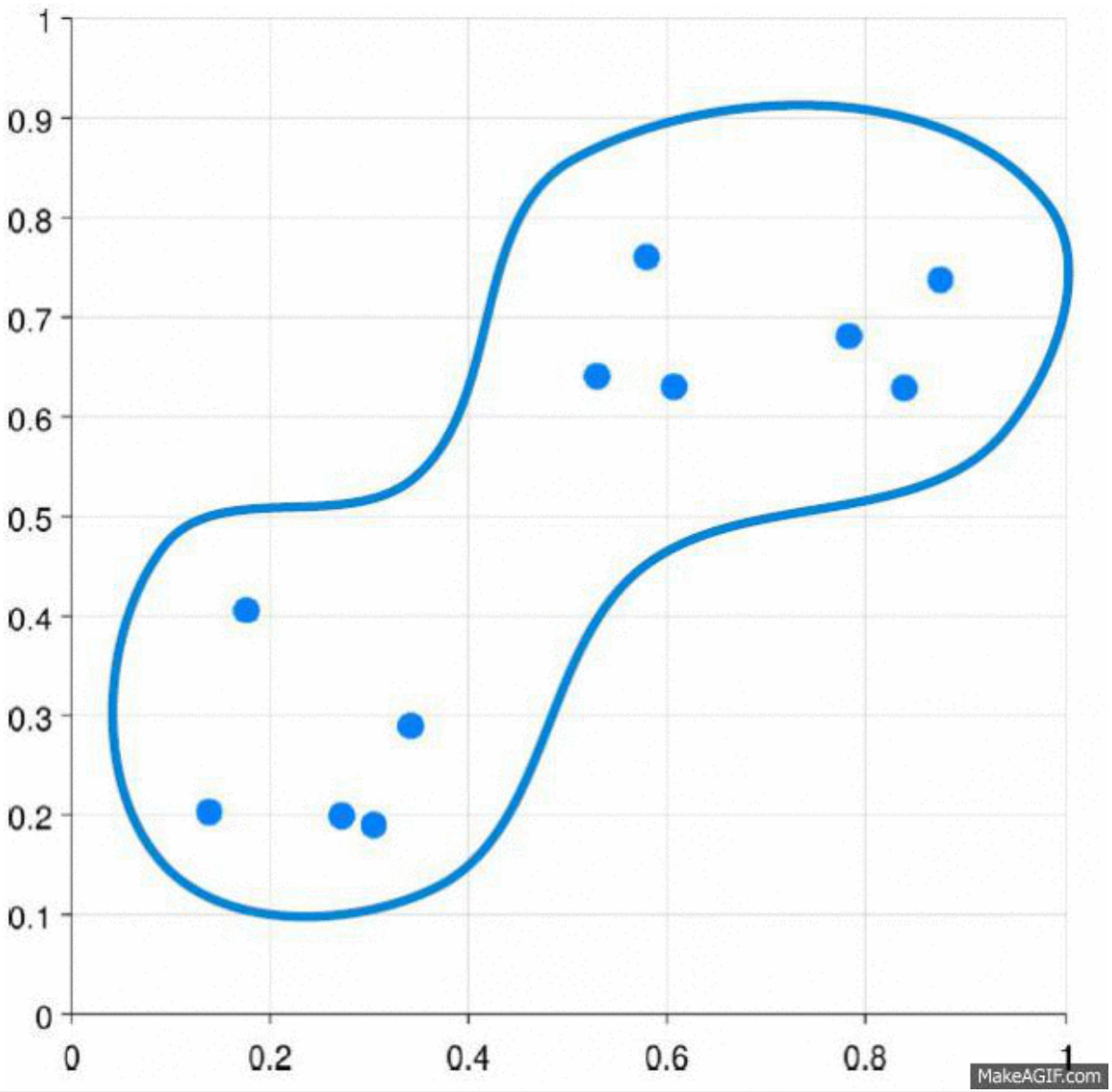
Agglomerative Clustering



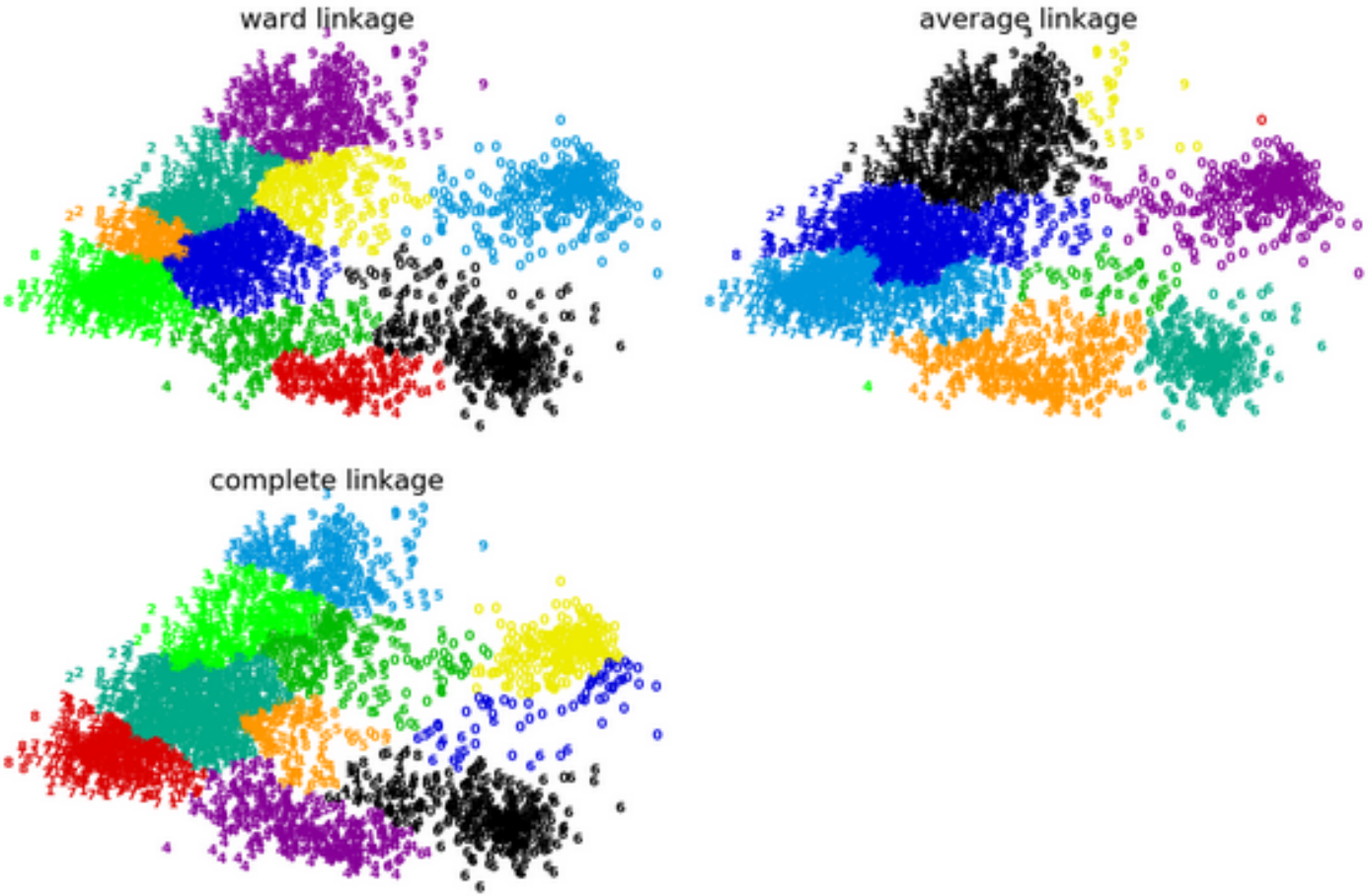
Agglomerative Clustering



Agglomerative Clustering



Agglomerative Clustering



En scikit: **cluster.AgglomerativeClustering**

Métricas

Si sabemos los labels, algunas métricas son

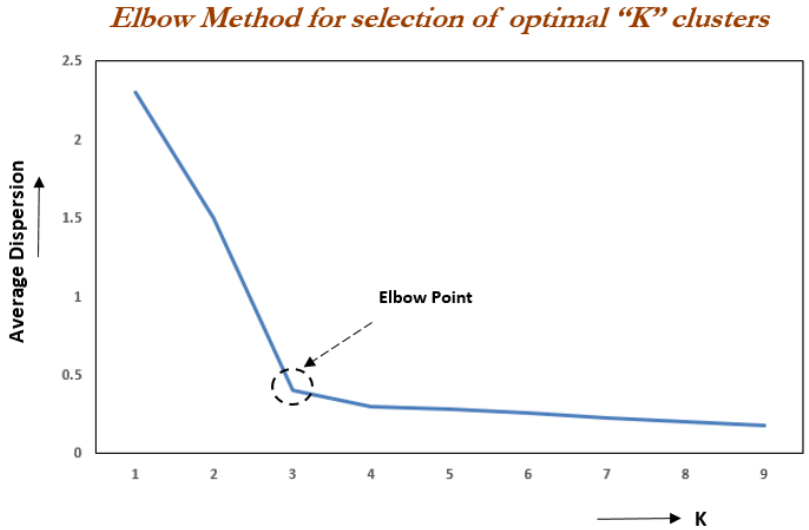
- Adjusted Rand index, que mide la similaridad entre dos clusterings considerando todos los pares y contanto aquellos que son asignados al mismo cluster tanto en las predicciones como en el verdadero. En scikit: **metrics.adjusted_rand_score**
- Adjusted Mutual Information (AMI), que mide el agreement entre clustering predicho y los labels conocidos midiendo la información mutua y ajustándolo por chance. En scikit: **metrics.adjusted_mutual_info_score**
- Homogeneidad, que mide si cada cluster contiene sólo miembros de una sola clase. En scikit: **metrics.homogeneity_score**
- Completitud, que mide si todos los miembros de una sola clase son asignados al mismo cluster. En scikit: **metrics.completeness_score**

¿Cuántos clusters coger?

Si no sabemos los labels, lo que tenemos que medir es la calidad de las particiones obtenidas. Básicamente, para este caso, el número de clusters a escoger es desconocido.

Elbow method

1. Usar un método de clustering con diferentes k's
2. Para cada k, calcular la distancia total dentro.
3. Plotear la curva para los diferentes número de k.
4. El punto en el que la pendiente cambia (el "codo"), suele dar la mejor indicación del número de clusters



¿Cuántos clusters coger?

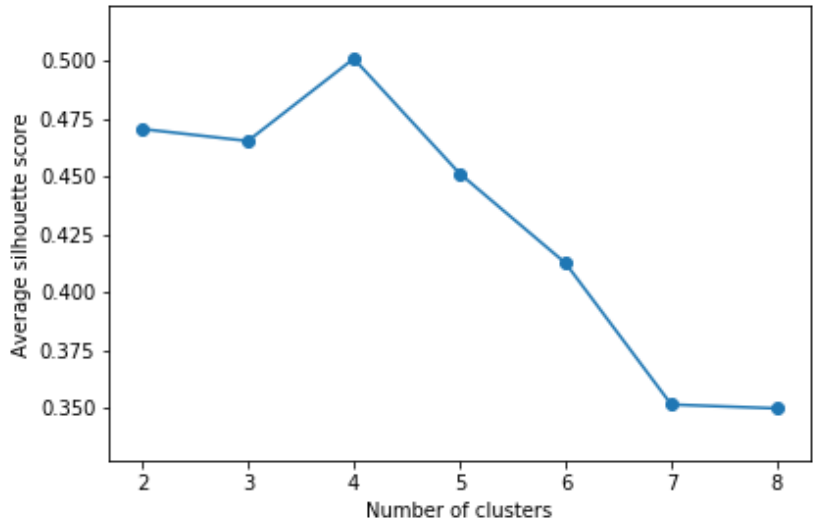
Si no sabemos los labels, lo que tenemos que medir es la calidad de las particiones obtenidas. Básicamente, para este caso, el número de clusters a escoger es desconocido.

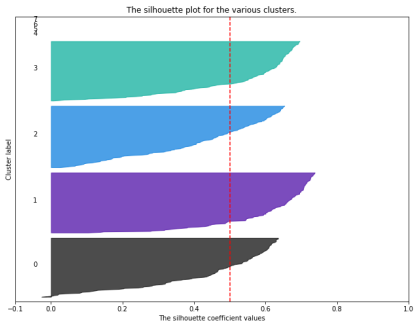
Silhouette method

1. Usar un método de clustering con diferentes k's
2. Para cada k y punto i, calcular la métrica silhouette s_i

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \qquad (14)$$

3. Tomar el k donde la media de los silhouette es máxima.





Reducción de la dimensionalidad

Motivación

- Habíamos dicho que uno de los problemas más comunes y graves en machine learning es el del Overfitting, ya que arruina el poder de generalización de nuestro modelo.
- Este problema suele estar relacionado con un exceso de complejidad, asociado a una dimensionalidad muy alta, que en muchos casos sólo aporta información redundante.
- Existen por tanto dos formas (pueden ser complementarias) de atacar el problema de la alta dimensionalidad:
  1. Quedarnos sólo con aquellas features más relevantes (**feature selection**)
  2. Encontrar un subset de nuevas variables, combinación de las originales, manteniendo la misma información original (**reducción de la dimensionalidad**)

Motivación

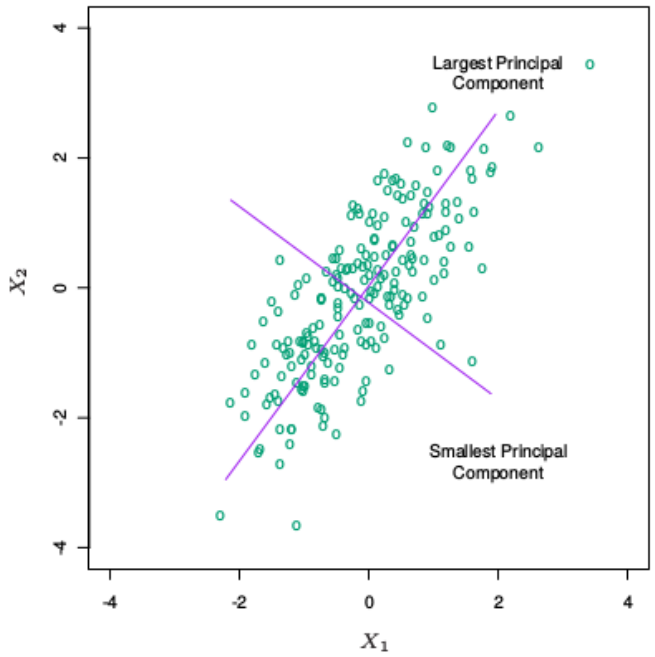
Además, las técnicas de reducción de la dimensionalidad permiten:

- Comprimir los datos y reducir espacio de almacenamiento
- Liberar demanda de poder computacional
- Usar algoritmos no apropiados para altas dimensiones
- Visualizar mejor los resultados

PCA

- Probablemente, la técnica de reducción de la dimensionalidad más usada
- Convierte un conjunto de features posiblemente correlacionadas en una serie de features (**componentes principales**) no correlacionadas
- Las componentes principales son ordenadas según la información total que retienen de los datos originales
- El número de componentes principales diferentes son $min(N-1, m)$

PCA



- La primera PC representa una línea que ajusta distancia mínima a ella
- La segunda PC representa una línea que ajusta distancia mínima a ella y que es perpendicular a la primera PC.
- Las componentes principales son entonces una serie de direcciones que ajustan la distancia mínima a ellas y son ortogonales entre si

Matemática de la PCA

- Suponemos que existe una función f que aproxima las observaciones:

$$f(\lambda) = \mu + V_q \lambda \tag{15}$$

donde $\mu = (\mu_1, \ldots, \mu_m)^T$, $V_q = (v_1, \ldots, v_p)^T$ y $\lambda = (\lambda_1, \ldots, \lambda_p)^T$, $\sum_{q=1}^{p} |v_q|^2 = 1$.

- La diferencia con lo observado se puede calcular a partir de los residuos

$$RSS = \sum_{i=1}^{N} |x_i - \mu - V_q \lambda_i|^2 \tag{16}$$

- Minimizando para μ y λ_i nos da

$$\mu = < x > \tag{17}$$

$$\lambda_i = V_q^T (x_i - < x >) \tag{18}$$

Matemática de la PCA

- Sustituyendo en los residuos da

$$RSS = \sum_{i}^{N} \left| (x_i - < x >) - H_q (x_i - < x >) \right|^2 , \tag{19}$$

  con $H_q = V_q V_q^T$
- La matrix H_q se conoce como la matriz de proyección, que mapea x_i en un subespacio p y lo devuelve al espacio original.

- La ecuación anterior se puede escribir como

$$RSS = 2(\sum_{i=1}^{N} |x_i - < x >|^2 - |V_q^T(x_i - < x >)|^2)$$
$$= 2(\sum_{i=1}^{N} |x_i - < x >|^2 - |(\sum_{i} V_q^T(x_i - < x >))|^2 + |Var(V_q x_i)|^2)$$

- Por lo tanto, para minimizar RSS, significa que tenemos que maximizar la suma de las varianzas a lo largo de las p direcciones y minimizar los términos cruzados (la covarianza)

Matemática de la PCA

- Esto es similar a encontrar los autovalores de la matriz de covarianza, que a veces quede ser muy ineficiente.
- Más eficientemente, mediante singular value decomposition (SVD)

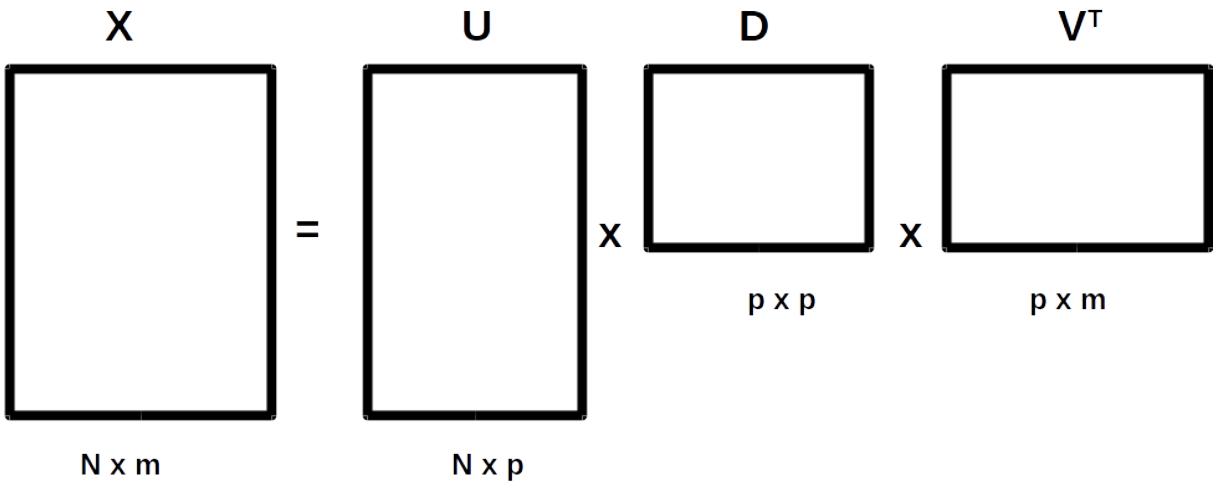
SVD

- **Centramos o estandarizamos** la matriz de features X con dimensiones $N \times m$.
- Se construye la decomposición en valores singulares de X como

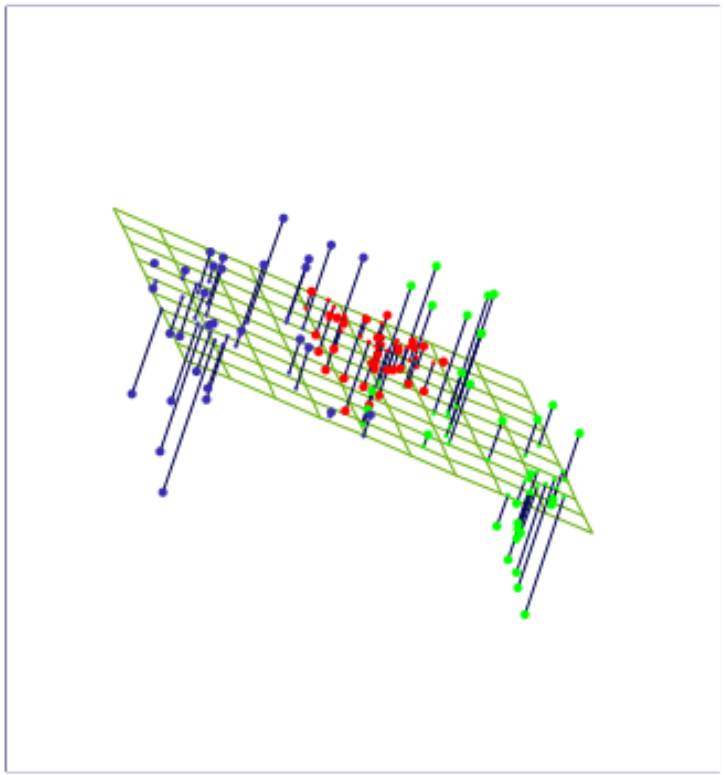
$$X = UDV^T \tag{20}$$

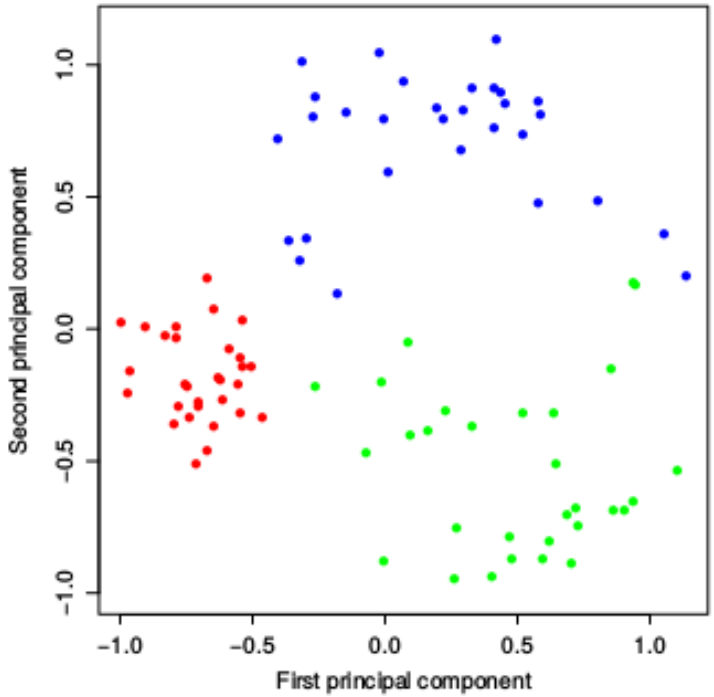
- U es una matriz ortogonal $N \times p$, cuyas columnas se conocen como vectores singulares izquierdos, las columnas V^T como vectores singulares derechos y D es la matriz diagonal $p \times p$ con elementos $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$.

PCA



- Las columas de U representan los vectores principales, ortogonales entre si y cuya combinación linear permite reconstruir los datos originales.
- D es diagonal y muestra la importancia (mayor varianza) de cada componente principal.
- Las columnas de V^T muestran la relación entre los features y las componentes principales.





En scikit está implementado por **decomposition.PCA**

Factor Analysis

- Asume que los datos X se pueden descomponer de la siguiente manera

$$X = WH + M + E \tag{21}$$

donde H es una matriz de factores gaussianos latentes, W el peso de estos factores latentes, M es un factor que cambia el origen de coordenadas y E representa el ruido, también gaussiano, con media 0 y covarianza Ψ, i.e. , $\epsilon \sim \mathcal{N}(0, \Psi)$, donde Ψ = diag$(\psi_1, \ldots, \psi_m)$.

- Asumiendo $p(x) = \mathcal{N}(\mu, WW^T + \Psi)$, los coeficientes de W pueden ser calculados mediante maximum Likelihood Estimation (MLE).
- Puede producir resultados similares a PCA, pero a diferencia de ésta, sus factores latentes no tienen por qué ser ortogonales.

En scikit está implementado por **decomposition.FactorAnalysis**

Non-negative Matrix Factorization

- Método de descomposición útil para los casos en que los datos de los que partimos sean positivos.
- Descompone la matrix X en el producto de dos matrices W y H, de tal forma que

$$X \approx WH \tag{22}$$

- W y H se obtienen minimizando las distancia con X

$$d = \frac{1}{2}||X - WH||^2 = \frac{1}{2}\sum_{i,j}\left(X_{ij} - \sum_{p} W_{ip}H_{pj}\right) \tag{23}$$

- A diferencia de la PCA, la representación de un vector se hace superponiendo las componentes, sin substracción.
- Pueden ser eficientes para la representación de imágenes y texto.

En scikit está implementado por **decomposition.NMF**

Resumen

Para realizar clustering

- el módulo **cluster**: **KMeans, DBSCAN, AgglomerativeClustering**
- **mixture**: **GaussianMixture**

Para ver la cualidad de los clusters, en el módulo **metrics**:

- Si no sabemos la estructura verdadera: **silhouette_score**
- Si sabemos la estructura verdadera: **adjusted_rand_score, adjusted_mutual_info_score, homogeneity_score, completeness_score**

Para realizar reducción de la dimensionalidad, el módulo **decomposition**:

- análisis por componentes principales: **PCA, FactorAnalysis, NMF**