

Lec8: Unsupervised Learning I

Isidoro Garcia Urquieta

2021

Agenda

- ▶ Aprendizaje No Supervisado
- ▶ Clustering
- ▶ K-means
- ▶ Hierarchical Clustering

Aprendizaje Estadístico

El aprendizaje estadístico concierne dos tipos:

- 1) **Aprendizaje Supervisado:** Existe una variable endógena que gobierna el aprendizaje. Esto genera la siguiente función a estimar:

$$Y = f(X) + \epsilon$$

Donde $X : X_1, \dots, X_p$ son todas las variables explicativas que se nos ocurran, Y es la variable supervisora y ϵ es un error aleatorio.

- 2) **Aprendizaje No Supervisado:** No existe una variable objetivo. Tenemos X y queremos entender como se relacionan entre ellas. Típicamente relacionado a método de reducción de dimensionalidad.

Aprendizaje No supervisado

Existen dos tipos de objetivos en el aprendizaje no supervisado:

1. **Clustering:** Se trata de dividir a las observaciones en grupos (clusters) tal que las observaciones dentro de cada grupo sean muy similares y disimilares a observaciones en otros grupos.
2. **Factor Models:** Se trata de *reducir dimensionalidad* a partir de una X para entender los factores principales.

En ambos, estamos buscando simplificar cómo vemos el conjunto de las X_s . A pesar de que no buscamos predicción, buscamos modelos *estables* a nuevas observaciones.

Clustering

Por qué hacer clustering?

- ▶ Un resumen: Mostrar representaciones de los datos interesantes que reflejen la varianza de toda la base.
- ▶ Descubrimiento: Encontrar sub grupos interesantes dentro de todo el set de observaciones.

En las técnicas de clustering, buscamos encontrar las membresías a unos subgrupos **desconocidos** a partir de algoritmos.

Noten como esto hace que el unsupervised learning sea un reto y con algunos toques más artísticos. La razón es porque no tenemos una y que corrobore que hicimos un buen o mal trabajo.

Clustering Ejemplos prácticos

- ▶ Un buscador o plataforma quiere encontrar grupos de productos similares (Para ofrecerlos a los usuarios): Netflix haciendo recomendaciones de películas similares a las que ya viste.
- ▶ Sentimiento de las personas en Twitter u otra red social: Tenemos una base de X de palabras, tweets y queremos resumir esta información en un score de sentimiento.
- ▶ Encontrar los temas de los que habla un documento de manera automatizada.
- ▶ User personas (Arquetipos): Encontrar grupos de usuarios similares que ayuden a generar entendimiento más profundo de los usuarios.

Tipos de Clustering

1. **Métodos basados en modelos mixture (Mixture models):** Se asume que cada x_{ip} viene de 1 de los K mixture components. Donde definimos la probabilidad de pertenecer a este grupo como $p_k(x)$ para $k \in \{1, \dots, K\}$.
2. **Métodos basados en heurísticas:** Son modelos agnosticos a K , se construyen modelos jerárquicos que dividan mejor a las observaciones.

Empecemos con 1) K-means

K-means

K-mean es un método de clusterización no paramétrico que, dándole el número de grupos K deseado, asigna a cada observación a **un** grupo en particular. El algoritmo crea grupos mutuamente excluyentes:

1. $k_1 \cup k_2 \cup \dots \cup k_k = \{1, \dots, n\}$
2. $k_k \cap k'_k = \emptyset$ para $k_k \neq k'_k$

Sea la base x_{ip} y k grupos $k \in \{1, \dots, K\}$. Para cada k buscamos $p(k)$:

$$p(k) = \pi_1 p_1(x) + \dots + \pi_k p_k(x)$$

Donde π_k es la probabilidad incondicional de pertenecer al grupo k . Es decir, estamos buscando la probabilidad de que una observación pertenezca a k condicional en sus características x

K-means

Adicionalmente, definimos la media de cada grupo k como $\mu_{k_j} = E[x_i | k_j]$.

Así, para cada nueva x con pertenencia a los grupos k desconocida calculamos:

$$E[x] = p_1(x)\mu_1 + \dots + p_k(x)\mu_k$$

Función objetivo

Hasta ahora dijimos que queremos estimar probabilidad de pertenecer a cada grupo k **desconocido** condicional en las características x . Con eso, podemos calcular la media de cada grupo con μ_k (centroides).

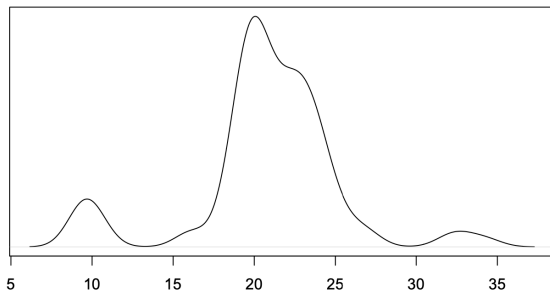
Pero, que buscamos? Partir los datos en k clusters tal que la intravarianza, sumada para todos los clusteres, sea mínima:

$$\min_{\{k_1, \dots, k_k\}} \frac{1}{N_k} \sum_{i \in k} \sum_{j=1}^p (x_{ij} - \hat{\mu}_k)^2$$

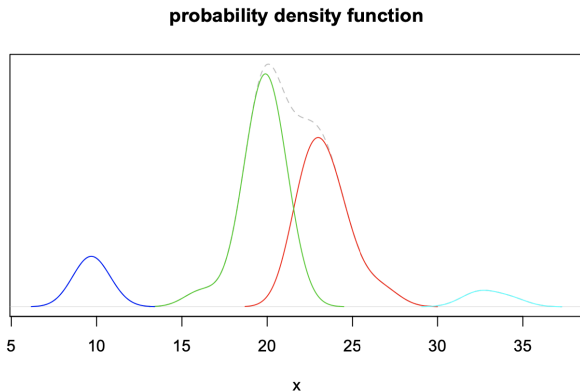
Eso no parece tan difícil. El problema es que aún tenemos que encontrar k . Para una base de tamaño n , tenemos K^n formas de partir los datos!!!

Ejemplo

probability density function



Ejemplo



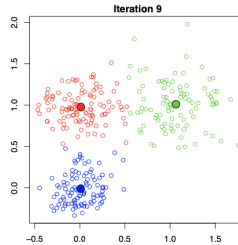
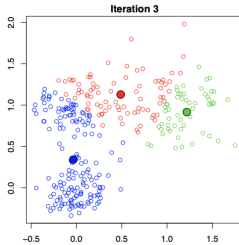
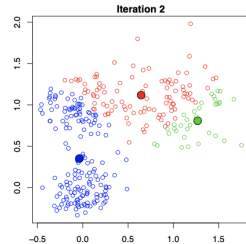
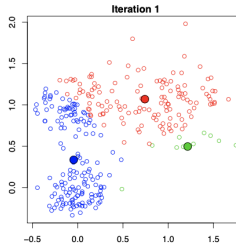
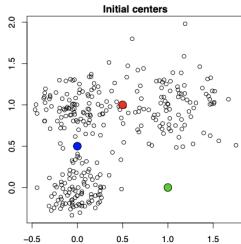
K-means algoritmo

Pasos:

1. Asigna aleatoriamente cada observación
2. Calcula los centroides $\hat{\mu}_k$ para las p variables en la base
3. Asigna a cada observación al centroide al que este más cercano.
4. Repite hasta que la asignación de cada observación deje de cambiar.

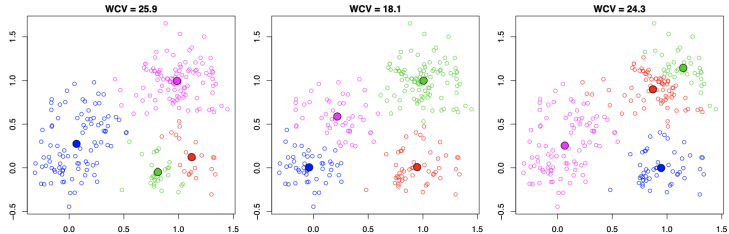
Problema: La solución depende de la iniciación aleatoria. Es decir, el algoritmo encuentra un óptimo local en lugar de un óptimo global. Por ello, es mejor estimar K-means varias veces con inicios aleatorios distintos.

Ejemplo



Ejemplo II

Acá vemos para distintos intentos. El segundo da mejor separación.



K-means en R

La función `kmeans(x,centers,nstart)` se utiliza para estimar K-means. Donde `x` es una `data.frame` **númeroico**, `centers = K` y `nstart` es el número de inicios aleatorios.

```
> grp = kmeans(x=mydata, centers=3, nstart=10)
```

```
K-means clustering with 3 clusters of sizes 28, 31, 31
```

```
Cluster means:
```

	x	y
1	1.0691704	-0.99099545
2	-0.2309448	-0.04499839
3	0.4987361	1.01209098

```
Clustering vector:
```

```
[1] 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2  
[39] 1 1 3 3 3 3 3 3 3 3 3 3 2 2 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
```

```
Within cluster sum of squares by cluster:
```

```
[1] 12.332683 4.911522 3.142067  
(between_SS / total_SS = 80.5 %)
```


K-means en R

← → ↺

kmeans (stats)

R Documentation

K-Means Clustering

Description

Perform k-means clustering on a data matrix.

Usage

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)
```

Arguments

<code>x</code>	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
<code>centers</code>	either the number of clusters, say <code>k</code> , or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in <code>x</code> is chosen as the initial centres.
<code>iter.max</code>	the maximum number of iterations allowed.
<code>nstart</code>	if <code>centers</code> is a number, how many random sets should be chosen?
<code>algorithm</code>	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm.
<code>object</code>	an R object of class "kmeans", typically the result of <code>ob <- kmeans(...)</code> .
<code>method</code>	character: may be abbreviated. "centers" causes <code>fitted</code> to return cluster centers (one for each input point) and "classes" causes <code>fitted</code> to return a vector of class assignments.
<code>trace</code>	logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
<code>...</code>	not used.

Details

The data given by `x` are clustered by the k-means method, which aims to partition the points into `k` groups such that the sum of squares from points to the assigned cluster centres is minimized. At the minimum, all cluster centres are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre).

The algorithm of Hartigan and Wong (1979) is used by default. Note that some authors use *k-means* to refer to a specific algorithm rather than the general method: most commonly the algorithm given by MacQueen (1967) but sometimes that given by Lloyd (1957) and Forgy (1965). The Hartigan-Wong algorithm generally does a better job than either of those, but trying several random starts (`nstart > 1`) is often recommended. In rare cases, when some of the points (rows of `x`) are extremely close, the algorithm may not converge in the "Quick-Transfer" stage, signalling a warning (and returning `ifault = 4`). Slight rounding of the data may be advisable in that case.

For ease of programmatic exploration, `k=7` is allowed, notably returning the center and `withinss`.

Except for the Lloyd-Forgy method, `k` clusters will always be returned if a number is specified. If an initial matrix of centres is supplied, it is possible that no point will be closest to one or more centres, which is currently an error for the Hartigan-Wong method.

Value

`kmeans` returns an object of class "kmeans" which has a `print` and a `fitted` method. It is a list with at least the following components:

<code>cluster</code>	A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
----------------------	--

Cómo escoger K

La selección de K es altamente subjetiva. Por ello es importante seguir algo de intuición en la decisión de K .

No obstante, podemos hacer un algoritmo parecido a los modelos predictivos:

1. Elige un set de K candidatas $K_1 < K_2 < \dots < K_m$
2. Utiliza AICc/BIC para decidir el mejor approach.

En el caso de K-means. Los parámetros estimados son $K \times p$. Dado que el BIC no asume distribuciones, es normalmente preferido. No obstante, no son una regla fija para elegir K .

Hierarchical Clustering

Una ventaja y desventaja de K-means es que estima **exactamente** los K grupos que le pedimos.

- ▶ Si elegimos bien K o es lo que necesita el caso de negocio/policy, es ventaja.
- ▶ Si no tenemos un prior de K puede ser problemático.

Hierarchical Clustering es una alternativa que no requiere un modelado previo ni una K .

- ▶ Produce tree-based representaciones de los datos.
- ▶ Esto genera clusteres anidados (grandes clusteres, pequeños clusteres dentro de esos)
- ▶ Observaciones similares igualmente terminarán en los mismos clusteres.

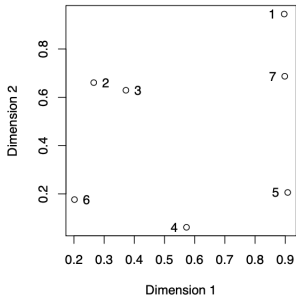
Hierarchical Clustering

Algo muy importante es que Hierarchical Clustering es un método **aglomerativo** (bottom-up).

- ▶ Aglomerativo:
- ▶ Todas las observaciones empiezan siendo un cluster (tienes n clusters).
- ▶ Fusiones a los dos grupos más similares de manera iterativa hasta que terminas con un sólo grupo.
- ▶ Divisivo (top-down):
- ▶ Empiezas con un cluster para n observaciones
- ▶ Separas en dos grupos maximizando la inter-varianza y minimizando la intravarianza (Ojo! aquí no tenemos y para eso).

Los métodos aglomerativos son más sencillos. Empecemos ahí.

Hierarchical Clustering



Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\};$

Step 2: $\{1\}, \{2, 3\}, \{4\}, \{5\}, \{6\}, \{7\};$

Step 3: $\{1, 7\}, \{2, 3\}, \{4\}, \{5\}, \{6\};$

Step 4: $\{1, 7\}, \{2, 3\}, \{4, 5\}, \{6\};$

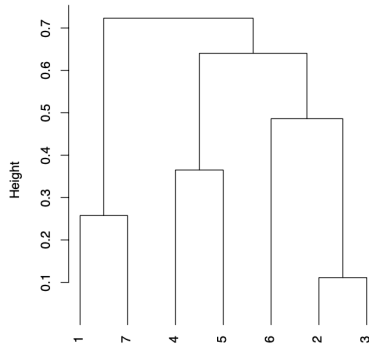
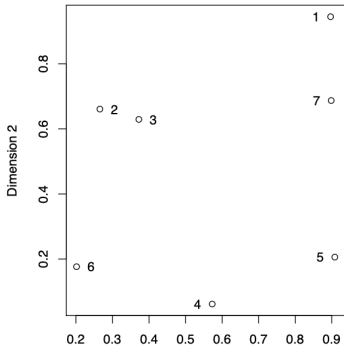
Step 5: $\{1, 7\}, \{2, 3, 6\}, \{4, 5\};$

Step 6: $\{1, 7\}, \{2, 3, 4, 5, 6\};$

Step 7: $\{1, 2, 3, 4, 5, 6, 7\}.$

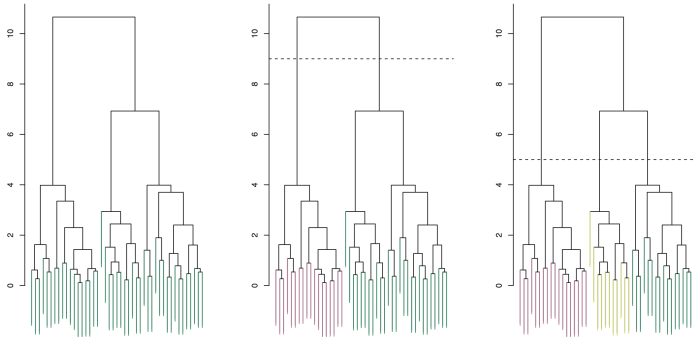
Hierarchical Clustering

Esto se puede representar como un árbol constuído de abajo hacia arriba:



Donde cortamos el árbol

Noten como cada nodo terminal de este dendograma es una observación. Por ende, tenemos que decidir en donde cortarlo para encontrar los números de clusters K . Una ventaja de esto es que una corrida te puede dar todas las opciones de K posibles.



Algoritmo Hierarchical Clustering

Formalicemos el algoritmo. Pasos:

1. Escoges una medida de similitud (i.e. Distancia Euclideana).
2. Empiezas con todas las observaciones en su cluster (n clusteres).
3. Calculas $\binom{n}{2}$ métricas de distancia.
4. Encuentra el par más similar y fusionalo en un cluster. La métrica de similitud es la altura de corte del dendograma
5. Repite 3 y 4 con los $i - 1$ clusteres restantes.

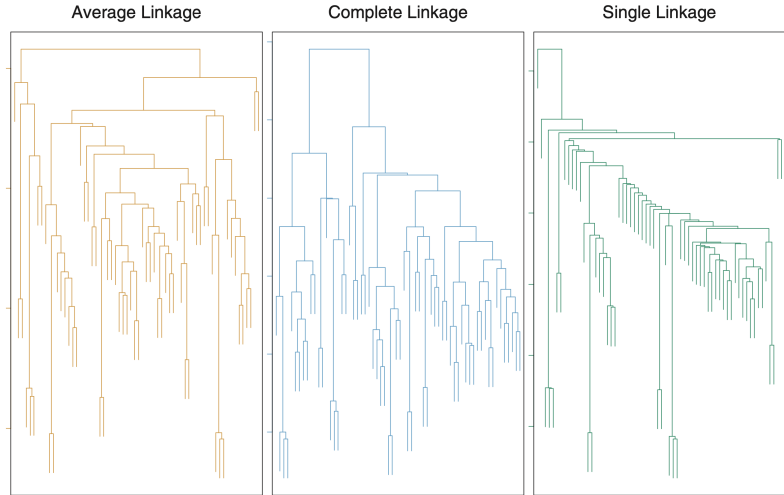
Retos: Cómo defines la distancia entre 2 clusteres cuándo uno o ambos tienen más de una observación? Definimos *linkage*

Linkage

<i>Linkage</i>	<i>Description</i>
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>largest</i> of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>smallest</i> of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>average</i> of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable <i>inversions</i> .


En general $Average = Complete > Single > Centroid$

Linkage: Impacto en el dendograma



Hierarchical Cluster en R

La función `hclust` genera Hierarchical Clustering. `dist(X)` para distancia eucldeana. Despues se utiliza `cuttree(h,x)` para cortar al nivel del dendograma.



Hierarchical Clustering

Description

Hierarchical cluster analysis on a set of dissimilarities and methods for analysing it.

Usage

```
hclust(d, method = "complete", members = NULL)
```

R3 method for class 'hclust'

```
plot(x, labels = NULL, hang = F, check = TRUE,
     axes = TRUE, frame.plot = FALSE, ann = TRUE,
     main = "Cluster Dendrogram",
     sub = NULL, xlab = NULL, ylab = "height", ...)
```

Arguments

d	a dissimilarity structure as produced by <code>dist</code> .
method	the agglomeration method to be used. This should be (as unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (=UPGMA), "mcquitty" (=WPGMA), "median" (=WRGMC) or "centroid" (=UPGMC).
members	NULL or a vector with length size of <code>d</code> . See the 'Details' section.
x	an object of the type produced by <code>hclust</code> .
hang	The fraction of the plot height by which labels should hang below the rest of the plot. A negative value will cause the labels to hang down from 0.
check	logical indicating if the <code>x</code> object should be checked for validity. This check is not necessary when <code>x</code> is known to be valid such as when it is the direct result of <code>hclust()</code> . The default is <code>check=TRUE</code> , as invalid inputs may crash R due to memory violation in the internal C linking code.
labels	A character vector of labels for the leaves of the tree. By default the row names or row numbers of the original data are used. If <code>labels = FALSE</code> no labels at all are plotted.
axes , frame.plot , ann	logical flags as in <code>plot.default</code> .
main , sub , xlab , ylab , plot	character strings for <code>title</code> . <code>sub</code> and <code>xlab</code> have a non-NULL default when there's a <code>treePee11</code> .
...	Further graphical arguments. E.g. <code>cex</code> controls the size of the labels (if plotted) in the same way as <code>text</code> .

R Documentation