Modelos de Mercadotecnia

Métodos de agrupación: Conglomerados, Segunda Parte

Jorge de la Vega Góngora

Maestría de Mercadotecnia,
Instituto Tecnológico Autónomo de México

Sesión 6

Introducción

- En esta segunda parte veremos algunos métodos para tipos de datos específicos
- Se ampliarán los métodos a técnicas de clasificación, en particular a árboles de clasificación

Determinación del número de clusters

Determinación óptima del número de clusters I

- Como se ha mencionado, sería ideal no determinar de antemano el número de clusters. Se sugiere hacer una evaluación adecuada para determinar cuál debería ser el número óptimo de clusters a considerar. Hay tres métodos que son comunes:
  - Método del codo
  - Método de la silueta
  - estadística gap
- Revisaremos aquí sólo los dos primeros casos. El tercero se puede consultar en el artículo de Tibishirani, Walther, Hastie (2001).

Método de codo

1. Se evalúa el algoritmo de conglomerados (e.g. k-medias) para diferentes valores de k.
2. Para cada k se calcula $ESS(k)$[a].
3. Graficar $ESS(k)$ vs k
4. Si se forma un codo, es donde se especifica el número de clusters óptimo.

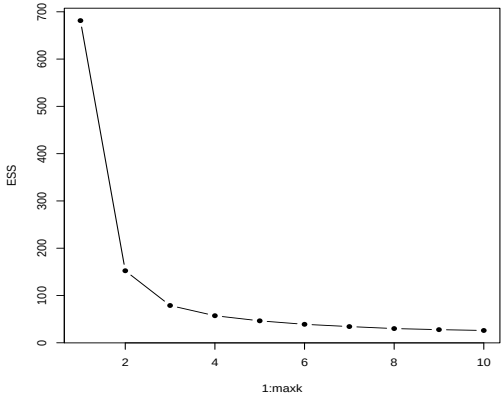
[a] Notar que $ESS(k) = ESS$ no se refiere a los componentes de ESS, que denotamos con ESS_k.

Ejemplo. [Cálculo para `iris`]

A continuación evaluamos el procedimiento indicado

```
data("iris")
iris2 <- iris
iris2$Species <- NULL
maxk <- 10 #número de clusters a considerar
ESS <- sapply(1:maxk,function(k)kmeans(iris2,k,nstart=10)$tot.withinss)
plot(1:maxk,ESS, type="b",pch=16)
```

Método de codo II



Método de la silueta I

- A veces el método del codo es un poco ambiguo en la determinación del número óptimo de clusters. Por eso es conveniente tener más de una opción.
- El *análisis de silueta* mide qué tan bien una observación es conglomerada y estima la distancia promedio entre conglomerados. La gráfica de la silueta muestra una medida de que tan cerca cada punto en un cluster está de puntos en los clusters vecinos.

Algoritmo para silueta

Para cada observación i, el ancho de la silueta s_i se calcula de la siguiente manera:

1. Se calcula la disimilaridad promedio a_i entre i y todos los puntos del cluster al que pertenece el item i.
2. Para todos los otros clusters C a los que no pertenece i, se calcula la disimilaridad promedio $d(i, C)$ de i a todos los items de C. Se define $b_i = \text{mín}_C\, d(i, C)$, y se interpreta como la disimilaridad entre i y el cluster más cercano al que no pertenece i.
3. El ancho de la silueta del item i se define como $s_i = \frac{b_i - a_i}{\text{máx}\{a_i, b_i\}}$

Podemos interpretar el ancho de la silueta de la siguiente manera:

Método de la silueta II

- Los items con un valor grande de s_i (casi 1) están bien agrupados
- Los items con un valor pequeño de s_i (casi 0), significa que los items quedan entre dos clusters.
- Items con valores negativos de s_i están mal agrupados.
- En el paquete `cluster` se puede usar la función `silhouette`.
- En el paquete `factoextra` la función `fviz_nbclust` genera las gráficas de codo y de silueta.

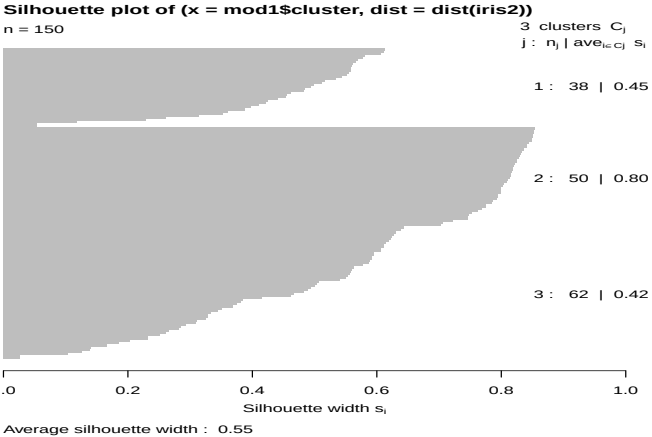
Ejemplo. [silueta para `iris`]

```
library(cluster)
library(factoextra)

Loading required package: ggplot2
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

mod1 <- kmeans(iris2,3)
plot(silhouette(mod1$cluster,dist(iris2)))
```

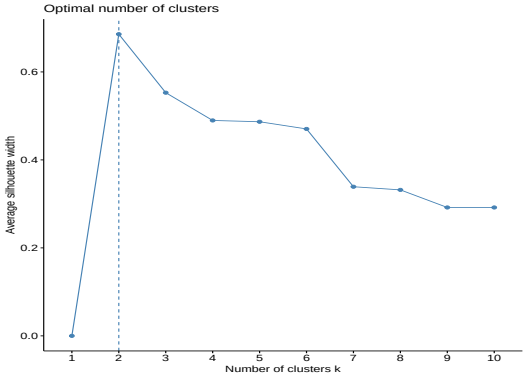
Método de la silueta III



Otra alternativa utilizando `factoextra`. Se calcula el agrupamiento para diferente número k. Se calcula la silueta promedio de los clusters de acuerdo al numero considerado k. La silueta promedio mide la calidad del agrupamiento, con una alt valor de la silueta promedio indicando una buena agrupación. La k óptima es la que maximiza la silueta promedio sobre los valores de k.

Método de la silueta IV

```
library(factoextra)
fviz_nbclust(iris2, pam, method = "silhouette")
```



Métodos para casos especiales

Análisis monotético (MONA) I

- Especialmente para vectores binarios, correspondientes a indicadoras de atributos. Por ejemplo, los datos en `animals`:

```
library(cluster)
data(animals)
head(animals)
```

```
    war fly ver end gro hai
ant   1   1   1   1   2   1
bee   1   2   1   1   2   2
cat   2   1   2   1   1   2
cpl   1   1   1   1   1   2
chi   2   1   2   2   2   2
cow   2   1   2   1   2   2
```

En este conjunto de datos cada renglón corresponde a un animal y las columnas son los siguientes atributos:

- `war` si es de sangre fría o caliente
- `fly` si vuela o no vuela
- `ver` si es vertebrado o invertebrado.
- `end` si está en peligro o no
- `gro` si vive en grupos o no
- `hai` si tiene cabello o no.

- El procedimiento para este caso es el siguiente:
  1. Imputar la matriz si hay datos faltantes (usualmente los métodos de agrupamiento no aceptan casos con datos faltantes).

2. Dividir cada cluster de acuerdo a la variable con la mayor medida de asociación a las otras variables: X_J. Un cluster es para todos los casos que tienen $X_{lJ} = 1$ y el otro cluster es para los que tienen $X_{mJ} = 0$.
   Se define la asociación entre dos variables f y g como $A_{fg} = |\det(Cont_{fg})|$ donde $Cont_{fg} = \begin{pmatrix} a_{fg} & b_{fg} \\ c_{fg} & d_{fg} \end{pmatrix}$ es la tabla de contingencia de las variables f y g y a_{fg} es la frecuencia de unos en ambas variables condicional al cluster que se va a dividir. La asociación total de una variable f es $A_f = \sum_{g \neq f} A_{fg}$. La variable h que satisface $A_h = \max_f A_f$ se selecciona para dividir el cluster.
3. Repetir el paso 2 hasta que cada cluster consista de objetos que tengan idénticos valores para todas las variables

Ejemplo. [Ejemplo de MONA para los datos de animales]

Análisis monotético (MONA) III

```
animales <- animals -1          # se cambia las indicadoras a ser 0s y 1s.
animales[is.na(animales)] <- 0 # imputación (se puede mejorar)
head(animales)
```

```
    war fly ver end gro hai
ant   0   0   0   0   1   0
bee   0   1   0   0   1   1
cat   1   0   1   0   0   1
cpl   0   0   0   0   0   1
chi   1   0   1   1   1   1
cow   1   0   1   0   1   1
```

```
mona1 <- mona(animales)
mona1
```

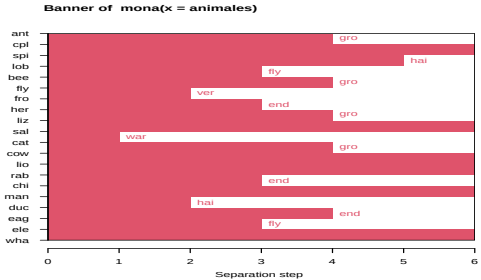
```
mona(x, ..) fit;  x of dimension 20x6
Order of objects:
 [1] ant cpl spi lob bee fly fro her liz sal cat cow lio rab chi man duc eag ele wha
Variable used:
 [1] gro  NULL hai  fly  gro  ver  end  gro  NULL war  gro  NULL NULL end  NULL hai  end  fly  NULL
Separation step:
 [1] 4 0 5 3 4 2 3 4 0 1 4 0 0 3 0 2 4 3 0

Available components:
[1] "data"      "hasNA"     "order"     "variable"  "step"      "order.lab" "call"
```

Gráfica de Banner I

- Para mostrar las divisiones sucesivas de izquierda a derecha se utilizar la *gráfica de Banner*, que se construye como se indica:
  - La longitud de una barra es el número de pasos necesarios para hacer una división.
  - Junto a cada barra, se muestra cuál es la variable que causa la división.
  - Una barra que continúa en el margen derecho, indica un cluster que no se puede dividir.

```
plot(mona1)
```



En la primera división se usa la variable `war` para separar {ant, cpl, spi, lob, bee, fly, fro, her, liz, sal} y {cat, cow, lio, rab,chi,man, duc, eag,ele,wha}, y continuamos así.

- El algoritmo de k-modas es similar al de k-medias, pero utilizando la moda de los datos. Esto abre la posibiildad de aplicar a datos categóricos en lugar de datos numéricos. Adicionalmente, se pueden considerar datos faltantes.
- La principal referencia para este médtodo es el artículo de Huang.
- El algoritmo está implementado en el paquete `klaR` en `R`

Ejemplo I

El siguiente análisis se hace con los datos de la Encuesta Nacional de los factores determinantes del embarazo adolescente (ENFaDEA. En el sitio se pueden encontrar todos los microdatos para el análisis.

- Se apicará un análisis sencillo para mostrar cómo se aplica el algoritmo.
- Primero se convierten todas las variables de interés en caractéres, y se reemplazan los datos faltantes reales por datos faltantes “falsos” (sustituyéndolos por etiquetas). De esta manera estos datos se consideran otro grupo más en el conjunto de datos.

```
library(tidyverse)  # Para poder manipular los datos con facilidad
library(dplyr)
library(foreign)      # Para importar archivos de SPSS
library(klaR)   # k-modas para datos categoricos
library(factoextra)
options(width = 150)

datos <- read.spss("../data/ENFADEA_2017_FINAL_29_11_2019/ENFADEA_2017_FINAL_29_11_2019.SAV",
                   use.value.labels = T,to.data.frame = T,trim.factor.names = F,use.missings = T,
                   duplicated.value.labels = "condense")
head(datos[,1:14])
```

```
  NCUESTI              ENT_IND MUN_IND LOC_IND AGEBI MIND VIVIND N_HOGAR_IND TELEG              NOM_ENT RENHOG NCUESTH             res_ind P101_D
1     315           Guanajuato      15      39  0000   NA      6  Uno de uno      1 ANA BEATRIZ              4    2565 Entrevista completa      18
2    2711      Estado de México     121       1  2278    6      3  Uno de uno      1 VERONICA                 4   28459 Entrevista completa      16
3    2701      Estado de México     121       1  0746    6      3  Uno de uno      1 KARINA                   2   29201 Entrevista completa      27
4    1583 Coahuila de Zaragoza      35       1  4406   55      4  Uno de uno      1 BIANCA                   2   12820 Entrevista completa      21
5    1528      Estado de México     124      41  0000   NA     66  Uno de uno      1 CAROLINA                 2   30523 Entrevista completa      16
6     559           Guanajuato      15      34  0000   NA     36  Uno de uno      1 LIZBETH                  2   10556 Entrevista completa      21
```

Ejemplo II

- Para propósitos del ejemplo, separamos un subconjunto de preguntas, por ejemplo, todas las que van de P101 a P109 como ejemplo.
- Todas las variables se convierten a caractéres y se sustituyen los valores NA por el śimbolo '_'

```
H1 <- datos %>%
        dplyr::select(starts_with("P10")) %>%
        mutate_all(as.character) %>%
        replace(., is.na(.), "--")
```

- Consideramos 5 posibles grupos. Es importante fijar la semilla aleatoria cada vez, para tratar de obtener los mismos resultados

Ejemplo III

```
set.seed(100)
m2 <- kmodes(H1, 2, weighted = F)
set.seed(100)
m3 <- kmodes(H1, 3, weighted = F)
set.seed(20)
m4 <- kmodes(H1, 4, weighted = F)
set.seed(100)
m5 <- kmodes(H1, 5, weighted = F)
m5$modes # Ejemplo de resultado
```

```
  P101_D P101_M P101_A P102 P104                      P105_N P105_G     P106           P107 P108_A P108_B P108_C P108_D P109_A P109_B P109_C P109_D
1      2      8   1993   24   Sí Preparatoria o bachillerato      3 Católica Poco religiosa     Sí     No     No     Sí     Sí     --     --     Sí
2     23      7   1996   20   No  Licenciatura o profesional      3 Católica Poco religiosa     Sí     Sí     Sí     Sí     Sí     Sí     Sí     Sí
3     21      2   1993   24   Sí Preparatoria o bachillerato      3 Católica Poco religiosa     Sí     Sí     No     No     Sí     Sí     --     --
4      1      8   1996   20   Sí                  Secundaria      3 Católica Poco religiosa     No     No     No     No     --     --     --     --
5     16     10   1996   21   Sí                  Secundaria      3 Católica Poco religiosa     Sí     No     No     No     Sí     --     --     --
```

```
m5$withindiff # medida de similaridad de los grupos
```

```
[1] 5247 6291 2239 3605 2891
```

Clasificación

Problema general de clasificación I

- A diferencia del análisis de conglomerados, la clasificación asume que ya se tiene conocimiento de los grupos que hay y se tienen algunas observaciones de los grupos.
- Consideren un vector de atributos observables $\mathbf{x}_i \in \Omega \subset \mathbb{R}^p$ para un objeto o individuo i que se sabe que debe pertenecer a uno de g grupos o poblaciones Π_i posibles.
- Ω es el espacio muestral, y se puede representar como una partición de regiones R_i correspondientes a las diferentes poblaciones:

$$\Omega = R_1 \cup R_2 \cup \cdots \cup R_g R_i \cap R_j = \emptyset \quad \forall i \neq j$$

Problema general de clasificación

Dada una observación i con atributos $\mathbf{x}_i$, encontrar una regla de clasificación que defina regiones R_i, usando la información $\mathbf{x}_i$, *junto con lo que sabemos de las poblaciones* Π_j, para ubicar la población de pertenencia de i, con la máxima precisión posible.

- Este pronóstico puede tener un margen de error, usualmente vinculado a que los atributos pueden darse en más de una región R_i.
- Lo anterior supone que las poblaciones Π_j son conocidas de antemano.

Problema general de clasificación II

- El problema de clasificación usualmente se considera como un ejemplo de *aprendizaje supervisado*, en donde un conjunto de *inputs* (o variables independientes o predictores) $\mathbf{x}$, tienen influencia sobre uno o más *outputs* (o variables dependientes, o respuestas) Π_j. Entonces los *inputs* son usados para predecir el valor de los *outputs*[1].
- Las reglas de clasificación o asignación se desarrollan "aprendiendo" de la muestra $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ de la que sabemos de qué población proviene. ¿Cómo sabemos esto de algunas observaciones y no de otras? Por diferentes razones:
  - por conocimiento incompleto del desempeño futuro.
  - conocimiento completo requiere la destrucción total del producto.
  - Información costosa o no disponible.

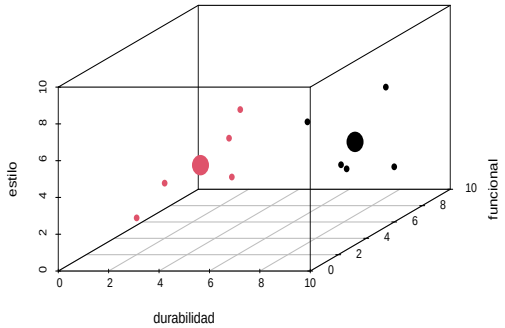
[1] El lenguaje de *inputs* y *outputs* es mucho más común en *Machine Learning*.

Ejemplo I

- Supongan que Ben & Frank quiere saber si ciertos armazones serán comercialmente aceptables. Hacen un estudio de mercado para evaluar sus nuevos armazones en tres características: durabilidad, desempeño y estilo, con una escala que va de 0 (muy mal) a 10 (excelente). Los resultados de las tres variables se muestran a continuación con su representación gráfica. Los puntos grandes representan las medias de cada grupo de compra.
- Se desea conocer qué características de un nuevo producto son útiles para diferenciar a los compradores de los no-compradores.

```
library(scatterplot3d)
ByF <- data.frame(
        durabilidad = c(8,6,10,9,7,5,3,4,2,2),
        funcional = c(9,7,6,4,8,4,7,5,4,2),
        estilo = c(6,5,3,4,2,7,2,5,3,2),
        compra  = c(1,1,1,1,1,2,2,2,2,2)
)
graf <- scatterplot3d(ByF[,1:3], color = ByF$compra, pch = 16, xlim = c(0,10), ylim = c(0,10), zlim = c(0,10))
graf$points3d(rbind(colMeans(ByF[ByF$compra == 1, -4]),
colMeans(ByF[ByF$compra == 2, -4])),pch = 16,col = c(1,2), cex = 3)
```

Ejemplo II



- Noten que ya se sabe quiénes compraron y quiénes no.

kNN (k Vecinos cercanos) I

- Este es uno de los algoritmos más simples para agrupamiento. Se define una distancia sobre el conjunto de observaciones con p atributos.Se calculan la matriz de distancias y se fija k como el número de vecinos más cercanos en la población. Finalmente se asigna la categoría basada en la que tiene la mayoría de vecinos cercanos en el caso que se está clasificando.
- Para clasificar observaciones nuevas, necesitamos *entrenar* al conjunto para determinar la regla de decisión que se usará

```
library(class)
data(iris)
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
idx <- seq(1,nrow(iris))
train_idx <- sample(idx,100)
test_idx <- setdiff(idx,train_idx)
x_train <- iris[train_idx,-5]
x_test <- iris[test_idx,-5]
y_train <- iris[train_idx,5]
y_test <- iris[test_idx,5]
```

kNN (k Vecinos cercanos) II

- El resultado que se obtiene es la clasificación que corresponde a los casos que no fueron incluidos en el conjunto de entrenamiento, que forman el conjunto de prueba. Al comparar los resultados de la predicción con los resultados reales, se puede calcular la *tabla de confusión*

```
m1 <- knn(x_train,x_test,y_train,k = 3, prob = F,use.all = T)
table(m1,y_test)  # Tabla de confusión.
```

```
            y_test
m1           setosa versicolor virginica
  setosa         18          0         0
  versicolor      0         19         0
  virginica       0          0        13
```

Árboles de decisión/clasificación (CART)

- Los árboles de decisión son diagramas en forma de árboles, en donde cada decisión es un *nodo* y éstas decisiones se pueden anidar. La predicción final se realiza en una *hoja* del árbol, o *nodo terminal*.
- CART: *Classification and Regression Trees* (Breiman, et. al. 1984)
  - **Respuesta categórica**: *Classification Tree*. Tratan de predecir las probabilidades de las clases generadas por la variable de respuesta categórica en las hojas del árbol. Por ejemplo, probabilidades de lluvia, probabilidades de default, preferencias de géneros cinematográficos, etc.
  - **Respuesta Continua**: *Regression Tree*. Tratan de predecir un valor medio para la respuesta en las hojas del árbol, como cantidad promedio de lluvia o la tasa esperada de default en crédito.
- Los modelos CART son un ejemplo de modelo no paramétrico de clasificación, y permite mucho mayor flexibilidad que los modelos paramétricos.

Construcción del árbo de decisión I

- Se considera un nodo de un árbol de decisión *puro* si todos los puntos asociados con el nodo tienen el mismo valor de la variable dependiente.
- Sólo los nodos impuros son los que se deben dividir.
- Hay varias métricas que se usan para medir la impureza de un nodo. Entre ellas:
  - Coeficiente de Gini: $Gini(t) = 1 - \sum_{i=1}^{2} w_i(t)^2$
  - Entropía: $En(t) = -\sum_{i=1}^{2} w_i(t) \log_2 w_i(t)$

  donde $w_i(t)$ es la fracción de casos que pertenecen a la clase i en el nodo t.
- Se calcula la impureza de una división como el promedio ponderado de las impurezas de los nodos involucrados en la división, y donde el peso del nodo hijo es proporcional al número de observaciones en ese nodo.

Ejemplo CART: datos de yogurt I

Consideremos los siguientes datos. Queremos determinar una regla simple para determinar si una persona comprará yogurt griego. Se tienen las medidas de 15 adultos

```
yogurth <- data.frame(sexo = factor(c(1,0,0,1,0,0,1,0,0,0,1,0,1,1,1)),
                      civil = factor(c("s","c","s","c","d","c","d","s","c","s","c","d","d","s","s")),
                      ingreso = factor(c("A","M","B","A","M","B","A","M","B","M","A","M","M","M","M")),
                      compra = factor(c(0,0,0,0,1,0,0,1,0,1,1,1,1,1,0)))
yogurth
```

	sexo	civil	ingreso	compra
1	1	s	A	0
2	0	c	M	0
3	0	s	B	0
4	1	c	A	0
5	0	d	M	1
6	0	c	B	0
7	1	d	A	0
8	0	s	M	1
9	0	c	B	0
10	0	s	M	1
11	1	c	A	1
12	0	d	M	1
13	1	d	M	1
14	1	s	M	1
15	1	s	M	0

En el siguiente ejemplo ponemos la restricción de que el número mínimo de observaciones en las hojas del árbol son 5 observaciones:

Ejemplo CART: datos de yogurt II

```
library(tree)

Registered S3 method overwritten by 'tree':
 method     from
 print.tree cli

arbol <- tree(compra ~ ., data = yogurth, split = "gini", mincut = 3)
summary(arbol)


Classification tree:
tree(formula = compra ~ ., data = yogurth, split = "gini", mincut = 3)
Variables actually used in tree construction:
[1] "ingreso"
Number of terminal nodes:  2
Residual mean deviance:  1.134 = 14.74 / 13
Misclassification error rate: 0.2 = 3 / 15

arbol

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 15 20.730 0 ( 0.5333 0.4667 )
  2) ingreso: A,B 7  5.742 0 ( 0.8571 0.1429 ) *
  3) ingreso: M 8  8.997 1 ( 0.2500 0.7500 ) *

plot(arbol)
text(arbol,digits=2)
```

Ejemplo CART: datos de yogurt III