

Analisis y Curacion clase 4: Práctico 2

30 de Mayo, 2019, Cohorte Alpha

Análisis exploratorio y curación de datos

- ▶ Gonzalez Nehuen
- ▶ Arja Adel
- ▶ Madoery Pablo

Practico 2: Entregar un Rmd donde se:

- ▶ Elija un dataset clasificado de su preferencia y area (domain expertise), aplique un metodo de clustering y/o mixtura de Gaussianas en el mismo.
- ▶ Investigue los resultados en el meta parametro K numero de cumulos e investigue posibles procesos de seleccion del mismo.
- ▶ Elabore un resumen, y seleccione un mejor valor segun el/los criterios aplicados, discuta el significado de los cumulos encontrados.
- ▶ Comente la influencia de la normalizacion de los datos en los resultados del clustering.

Solución

Utilizamos el método “k-Nearest Neighbour”, con el cual realizamos la clasificación de un data set de distintos tipos de vidrios.

El data set fue descargado de
<https://archive.ics.uci.edu/ml/datasets/glass+identification>.

Cargamos el data set en un data frame

Le ponemos los nombres a las columnas de acuerdo a la documentación del data set

```
### Cargamos el data set en un data frame  
glass_df = read.csv("glass.csv", header = FALSE)
```

```
### Le ponemos los nombres a las columnas de acuerdo a la documentación  
colnames(glass_df) <-
```

transformamos la columna de clasificación a string

hacemos un shuffle de las filas del data frame para no incurrir

en errores de inferencias triviales

```
### transformamos la columna de clasificación a string
glass_df$Type <- as.character(glass_df$Type)
### hacemos un shuffle de las filas del data frame para no
### en errores de inferencias triviales
glass_df <- glass_df[sample(nrow(glass_df)),]

### normalizamos los valores numéricos
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}
glass_df_n <- as.data.frame(lapply(glass_df[2:10], normalize))
```

definimos un conjunto de datos para entrenamiento y otro para tests

```
### definimos un conjunto de datos para entrenamiento y otro para tests
glass_df_n_train = glass_df_n[1:100,]
glass_df_n_test = glass_df_n[101:214,]
train_types = glass_df[1:100, "Type"]
test_types = glass_df[101:214, "Type"]
```

Realizamos “N_reps” repeticiones y calculamos la eficiencia promedio

(que guardamos finalmente en un vector v)

El gráfico muestra que no hay gran variación de la eficiencia en la clasificación cuando se varía el parámetro k.

```
### Realizamos "N_reps" repeticiones y calculamos la eficiencia
### (que guardamos finalmente en un vector v)
library(class)
v = rep(0, 15)
N_reps = 10
for (j in 1:N_reps){
  for (k in 1:15){
    data_test_pred <- knn(train=glass_df_n_train, test=glass_test)
    v[k] <- v[k] + sum(data_test_pred == test_types) / length(test_types)
  }
}
```

Calculamos la Crosstable para ver la eficiencia en la clasificación de cada clase y los errores cometidos.

```
### Calculamos la Crosstable para ver la eficiencia en la c
library(gmodels)
CrossTable(x=test_types, y=data_test_pred, prop.chisq = FALSE)
```

```
##
##
##      Cell Contents
## |-----|
## |                      N |
## |      N / Row Total |
## |      N / Col Total |
## |      N / Table Total |
## |-----|
##
##
## Total Observations in Table:  114
##
##
```


