PRÁCTICA 5 MACHINE LEARNING

Jorge Candia & Alejandro Valbuena

ERRORES: KNN bien, KMeans -> aunque tenga 5 grupos, el nº de clusters óptimos no es 5. Puedo tener por ejemplo 13 clusters y que se repitan el grupo asignado a un cluster en otro(s) cluster(s). La precisión del modelo no se calcula como en el resto de casos. En un modelo hipotético con 5 clusters para 5 grupos, el cluster 2 puede pertenecer al grupo 3, y así con todos. Esto es así ya que al asignar aleatoriamente los centroides al inicio, a cada uno se le asignará el grupo que le cuadre, no el de su índex.

Observación de los datos, preprocesado y elección del modelo

Como es costumbre, una vez más empezaremos con las tareas de preprocesado y observación de los datos. Una vez limpiado el dataset de NAs (no hay ninguno) y habiendo hecho un corrplot, donde vemos que la variable a explicar no tiene mucha correlación con ninguna variable explicativa, un str y un summary, se procede a normalizar los datos, tarea que requieren los dos algoritmos que vamos a emplear a continuación para no sesgar las predicciones. Como siempre, se separa el conjunto de datos en training y test. Esta vez no hay variables discretas, por lo que no hay que hacer ningún factor().

CASO 1 -> Algoritmo KNN

El algoritmo KNN (K Nearest Neighbors) trata de buscar los K puntos más cercanos a un punto concreto para poder inferir su valor. Este algoritmo de simple implementación pertenece al conjunto de técnicas del aprendizaje supervisado, y puede ser utilizado tanto para problemas de clasificación (este ejercicio), como de regresión.

El algoritmo KNN se encuentra bajo la categoría de métodos denominados Lazy Learning, ya que no entrena un modelo. No se optimizan unos pesos, sino que simplemente compara cómo de parecidos son los puntos que conocemos con los puntos nuevos, para obtener así una predicción.

Es decir, al predecir un nuevo dato, se calcula la distancia entre este y todas las observaciones de training set. Esto hace que por una parte tenga buen desempeño para datasets pequeños, pero si este es muy grande, requiera de mucha memoria y tiempo. El esfuerzo computacional ocurre al predecir nuevos datos, si se meten datos nuevos en el training set, no hay que recalcular nada.

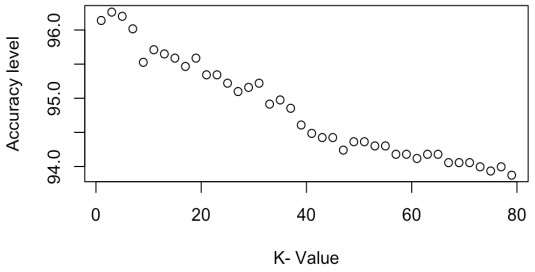
La forma en la que hemos empleado el algoritmo se muestra a continuación:

```
##### MÉTODO 1 -> KNN #####
k <- sqrt(nrow(train)) #Optimum value of k, generally square root of total => k = 62
knn.62 <-  knn(train=train, test=test, cl=train_labels, k=62) #CREACIÓN DEL MODELO
```

```r
confusionMatrix(knn.62 ,as.factor(test_labels)) #RESULTADOS => Precisión del 94%


#Precisión de múltiples modelos para múltiples Ks, para encontrar la K óptima
i=1                          # declaration to initiate for loop
k.optm=1                     # declaration to initiate for loop
for (i in seq(1,80,by=2)){
  knn.mod <-  knn(train=train, test=test, cl=train_labels, k=i)
  k.optm[i] <- 100 * sum(test_labels == knn.mod)/NROW(test_labels)
  k=i
  cat(k,'=',k.optm[i],'\n')       # to print % accuracy
}

#Plot de la precisión de todas las Ks calculadas
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
#K con máxima precisión
k.max = match(max(k.optm),k.optm) # K = 3
#Model for optimal k value
knn.best <-  knn(train=train, test=test, cl=train_labels, k=k.max)
#RESULTADOS DEL MODELO ÓPTIMO
confusionMatrix(knn.best ,as.factor(test_labels)) #Precisión  del 96% con K = 3
```

Finalmente, se obtiene que el valor óptimo de K (para el cual el modelo tiene una mayor precisión) es K = 3. Los resultados, excepcionales, son los siguientes:

```
Overall Statistics

               Accuracy : 0.9602
                 95% CI : (0.9495, 0.9691)
    No Information Rate : 0.8989
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7814
```



A la izquierda, el modelo considerado como óptimo. A la derecha, la precisión de cada modelo en función de k (secuencia de 1 a 80 cada 2 unidades).

CASO 2 -> Algoritmo KMeans

K-means es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en k grupos basándose en sus características. El algoritmo consta de tres pasos:

1.- Inicialización: una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.

2.- Asignación de los datos a los centroides: cada objeto de los datos es asignado a su centroide más cercano.
3.- Actualización centroides: se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.
En nuestro caso, tenemos el número de clusters predefinido, que el el número de grupos de figuras (5), por lo que eliminamos la parte no supervisada del estudio. La forma en la que hemos empleado el algoritmo se muestra a continuación:

```
##### MÉTODO 2 -> KMEANS #####
#IMPLEMENTACIÓN DEL ALGORITMO => 5 CLUSTERS, UNO POR CADA TIPO DE FIGURA
cluster_model <- kmeans(train,centers=5,nstart=5000)
#nstart es el máximo nº de actualización de centroides

#PREDICCIONES
cluster_predictor <- function(x, centers) {
  # compute squared euclidean distance from each sample to each cluster center
  tmp <- sapply(seq_len(nrow(x)),
                function(i) apply(centers, 1,
                                  function(v) sum((x[i, ]-v)^2)))
  max.col(-t(tmp))  # find index of min distance
}

prediction<- cluster_predictor(test, cluster_model[["centers"]])
#Resultados de las predicciones
table_pred<-table(test_labels, prediction)
confusionMatrix(as.factor(test_labels), as.factor(prediction))
```

Los resultados, de precisión muy pobre, son los siguientes:

	predictions				
targets	1	2	3	4	5
1	17	240	749	1	460
2	48	18	10	16	7
3	0	6	4	0	0
4	23	2	1	0	0
5	1	0	3	0	26

Cabe mencionar que cada vez que ejecutamos el algoritmo, sale una matriz de confusión distinta. Se cree que esto es debido a la asignación aleatoria de los centroides del inicio, que a pesar de poner un *nstart* de valor 5000, nunca se llega a un punto de estabilidad.

Comparativa de los métodos empleados

Como se ha ido comentando anteriormente, el desarrollo de esta propuesta mediante el algoritmo KNN obtiene una precisión casi absoluta, llegando al 96.02% para la k óptima. Por otra parte, la clusterización mediante KMeans no consigue llegar a clasificar correctamente

ni siquiera la mitad de las predicciones, un resultado bastante desastroso que se mejoraría con simplemente asignar a todas las figuras la clase 1, ampliamente mayoritaria en los datos estudiados.

Conclusión y recomendaciones del estudio

A pesar de que KNN haya obtenido una victoria aplastante, esto no quiere decir que un algoritmo sea mejor que otro. Para este caso de estudio en específico, KMeans no resulta de gran ayuda a priori, pero con otros datasets tendrá más eficiencia (además de que es un algoritmo de aprendizaje no supervisado).