Notas de R para Machine Learning

Edwin Uchupe Pipa, Oswaldo Ruesta Morales, Victor Ruiz Ccori

Introducción al Machine Learning

El Machine Learning es parte de la inteligencia artificial, en la cual, `los ordenadores adquieren la capacidad de aprender`, sin la necesidad de ser programados explícitamente.

Esta centrado en el desarrollo de programas informt́icos, a los que se les puede enseñar a crecer y cambiar cuando se exponen a nuevos datos

`A menudo, los algoritmos de Machine Learnig se clasifican en supervisados y no supervisados.`

1. Aprendizaje supervisado: En resumidas cuentas, pueden aplicar lo aprendido en el pasado a los nuevos datos. En este trabajo se hablará de

   - Regresión Lineal.
   - Regresión Logística.
   - Árboles de decisión.

2. Aprendizaje no supervisado: Estos algoritmos pueden extraer `inferencias` de conjuntos de datos. En este trabajo se hablará de

   - K-Means
   - KNN (K-Nearest Neighbors)
   - Algoritmo EM

Además, en los siguiente programas usaremos distintas librerías

```
> install.packages("HSAUR")
> install.packages("tools")
> install.packages("rpart")
> install.packages("rpart.plot")
> install.packages("e1071")
> install.packages("mclust")
> install.packages("fpc")
> install.packages("class")
> install.packages("UsingR")
> install.packages("gmodels")
> install.packages("psych")
> install.packages("mclust")
```

Regresión Lineal

Es un modelo matemático, que se usa cuando se desea aproximar una dependencia entre una variable dependiente, varias variables independientes, y un término aleatorio.

Un ejemplo de regresión lineal simple:

```
> data(iris)
> lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients
>
> # Se usa la funcion lsfit, para encontrar el ajuste de minimos cuadrados
>
> plot(iris$Petal.Length, iris$Petal.Width, pch=21, bg=c("red","green3","blue")
+      [unclass(iris$Species)], xlab="Petal length", ylab="Petal width")
> abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")
>
> # Luego, con plot() se grafica la dispersion, y con abline la recta ajustada
>
> summary(lm(Petal.Width ~ Petal.Length, data=iris))
```

Árboles de decisión

En simples cuentas, es una forma analítica (además de gráfica) de representar los sucesos que surgen ante una decisión, y ayudan a elegir la más acertada en cuanto a la probabilidad. Para usar los árboles en R se necesitan las librerías `rpart` y `rpart.plot`, para lo cual es necesario instalarlas (si es que no se tienen ya), y cargarlas en el espacio de trabajo

```
> library("rpart")
> library("rpart.plot")
```

`rpart` sirve para realizar estos árboles de decisión, mientras que `rpart.plot` sirve para graficar estos árboles

Un ejemplo sencillo de un árbol de decisión podría ser el siguiente:

```
> s <- sample(150,100)
> Entrenamiento_iris <- iris[s,]
> Test_iris <- iris[-s,]
> Arbol <- rpart(Species ~ .,Entrenamiento_iris)
> Arbol
```

Hasta aquí lo único que se ha hecho es un algoritmo que usa el conjunto de `datos iris` para crear el árbol de decisión. Con estos datos, el algoritmo aprende las reglas que crean el árbol.

Luego, se realiza la predicción y se mide la precisión del árbol en cuestión.

```
> p <- predict(Arbol,Test_iris,type = "class")
> table(Test_iris[,5],p)
```

Lo único que faltaría, sería graficar el árbol

```
> rpart.plot(Arbol)
```

K-Means

`K-Means` es un método de agrupamiento, el cual construye una partición de un conjunto de n observaciones en k grupos, cada observación pertenece al grupo cuyo valor medio es más cercano

Se presenta aquí un ejemplo de como se puede ejecutar el `K-means` en R. Para este ejemplo se van a necesitar las librerías y el siguiente código y mostrar una tabla con las especies en iris y la clasificación de estas

```
> library(e1071)
> library(mclust)
> library(fpc)
>
> km <- kmeans(iris[,1:4], 3)
> layout(matrix(1:1, ncol = 1))
> plot(iris[,1], iris[,2], col=km$cluster)
> points(km$centers[,c(1,2)], col=1:3, pch=8, cex=2)
```

Con esto se grafica el modelo inicial.

```
> table(km$cluster, iris$Species)
> sampleiris <- iris[sample(1:150, 40),]
> distance <- dist(sampleiris[,-5], method="euclidean")
> cluster <- hclust(distance, method="average")
> plot(cluster, hang=-1, label=sampleiris$Species)
```

Se crea ahora la variable `resultado`

```
> resultado <- cmeans(iris[,-5], 3, 100, m=2, method="cmeans")
> plot(iris[,1], iris[,2], col=resultado$cluster)
> points(resultadot$centers[,c(1,2)], col=1:3, pch=8, cex=2)
> result$membership[1:3,]
```

Escribimos una tabla de las espacies con la variable `result$membership` (los primeros 3 valores)

```
> table(iris$Species, resultado$cluster)
>
> mc <- Mclust(iris[,1:4], 3)
> plot(mc, what=c('classification'), dimens=c(3,4))
> table(iris$Species, mc$classification)
```

Esta nueva tabla simplemente ilustra la dependencia de las especies con la variable `mc` creada anteriormente

```
> cluster <- dbscan(sampleiris[,-5], eps=0.6, MinPts=4)
> plot(cluster, sampleiris)
> plot(cluster, sampleiris[,c(1,4)])
```

Los puntos de aviso en el clúster 0 son outliers no asignados. Se finaliza con una tabla que contiene los resultados finales

```
> table(cluster$cluster, sampleiris$Species)
```

Otros ejemplo de `K-Means:`

```
> require(graphics)
```

Se presenta en este caso, un ejemplo de 2 dimensiones, Se combinan 100 valores aleatorios de la distribucion normal en una matriz de 2 columnas,

```
> x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
+            matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
>
> colnames(x) <- c("x", "y")
```

y se les llama x, y y se usa la funcion `kmeans`

```
> cl <- kmeans(x, 2)
> plot(x, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 8, cex = 2)
```

Se escribe la funci ón `ss` que devuelve la suma de cuadrados

```
> ss <- function(x) sum(scale(x, scale = FALSE)^2)
```

Se ajusta los centros de agrupación a cada observación

```
> fitted.x <- fitted(cl);  head(fitted.x)
> resid.x <- x - fitted(cl)
>
> cbind(cl[c("betweenss", "tot.withinss", "totss")],
+       c(ss(fitted.x), ss(resid.x),    ss(x)))
> stopifnot(all.equal(cl$ totss,        ss(x)),
+           all.equal(cl$ tot.withinss, ss(resid.x)),
+           all.equal(cl$ betweenss,    ss(fitted.x)),
+           all.equal(cl$ betweenss, cl$totss - cl$tot.withinss),
+           all.equal(ss(x), ss(fitted.x) + ss(resid.x))
+ )
>
> kmeans(x,1)$withinss
```

Este es un cluster trivial. Es aquí donde el aleatorio comienza a ayudar, pues los cluster son demasiados

```
> (cl <- kmeans(x, 5, nstart = 25))
> plot(x, col = cl$cluster)
> points(cl$centers, col = 1:5, pch = 8)
```

KNN (K- Nearest Neighbors)

Es un método de clasificación no paramétrico, que sirve para estimar el valor de la función de densidad de probabilidad de que un elemento pertenezca a una clase a partir de la información que se le proporciona.

Se crea una variable `entrenamiento (train)`, y otra `test`, con las que luego se usará la función `knn`, luego de crear el factor `cl`, y por último se usa la función `atributes`, que muestra tanto los niveles, como la clase y la probabilidad de cada uno.

```
> library(class)
> train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
> test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
> cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
> knn(train, test, cl, k = 3, prob=TRUE)
> attributes(.Last.value)
```

Para el trabajo, su usará la `tabla iris` de la librería `UsingR`

```
> data("iris")
> str(iris)
> table(iris$Species)
> head(iris)
> iris
> set.seed(9850)
> runif(5)
> gp <- runif(nrow(iris))
> gp
```

Hasta aquí, se hacen valores aleatorios de la distribución uniforme, con la función `runif` y se asignan a `gp`.

Con el siguiente código, se está haciendo una función `normalize`, que resta el valor mínimo del vector, y lo divide entre el rango.

```
> iris <- iris[order(gp),]
> str(iris)
> head(iris)
> head(iris, 10)
> str(iris)
> summary(iris[, c(1, 2, 3, 4)])
> normalize <- function(x){
+  return((x - min(x))/(max(x) - min(x)))
+ }
```

Se crea la variable `iris_n`, que usa la función `as.data.frame`, que usa las 4 primeras filas de `iris` y la función `normalize`, además de las variables `iris_train, iris_test, iris_train_target` e `iris_test_target`

```
> normalize(c(1, 2, 3, 4, 5))
> normalize(c(10, 20, 30, 40, 50))
> iris_n <- as.data.frame(lapply(iris[,c(1, 2, 3, 4)], normalize))
> str(iris_n)
> summary(iris_n)
> str(iris)
> iris_train <- iris_n[1:139, ]
> iris_train <- iris_n[1:129, ]
> iris_test <- iris_n[130:150, ]
> iris_train_target <- iris[1:129, 5]
> iris_test_target <- iris[130:150, 5]
```

Lo unico que resta es usar nuevamente la función `knn`, con las variables mostradas, y mostrar todo en una tabla con `iris_test_target`

```
> ml <- knn(train = iris_train, test = iris_test, cl = iris_train_target, k = 13)
> ml
> table(iris_test_target, ml)
```