

K-Vecinos más cercanos (KNN)

Maisy Samai Vázquez Sánchez

2022-06-01

Introducción

k-vecinos más cercanos es un método para clasificar casos basándose en sus similitudes a otros casos. El aprendizaje automático se desarrolló como una forma de reconocer patrones de datos sin la necesidad de una coincidencia exacta con otros casos ya almacenados.

Librerías

```
#Se carga la librería  
library(MASS)
```

Base de datos de iris, precargada en R.

```
# Cargar los datos iris  
Z<-as.data.frame(iris)  
colnames(Z)
```

```
## [1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

1.Se define la matriz de datos y la variable respuesta, con las clasificaciones.

```
x<-Z[,1:4]  
y<-Z[,5]
```

2.Se definen las variables y las observaciones

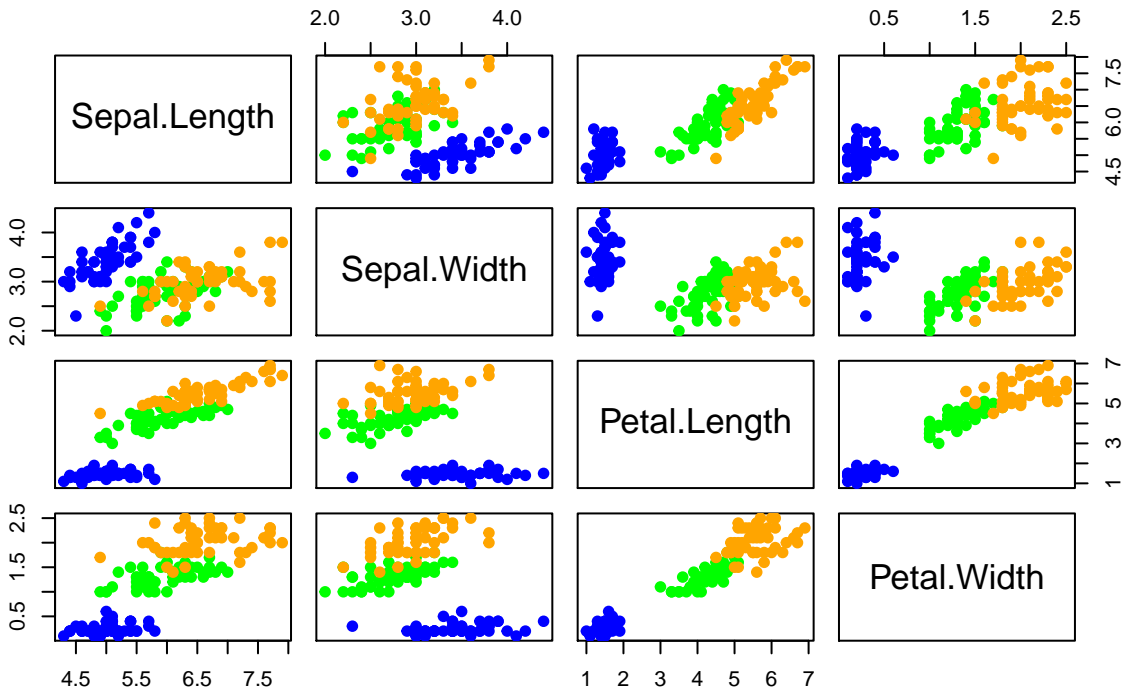
```
n<-nrow(x)  
p<-ncol(x)
```

3.Se hace un gráfico scatter plot

```
# Creacion de un vector de colores  
col.iris<-c("blue","green","orange")[y]
```

```
pairs(x, main="Data set Iris, Setosa(azul), Versicolor(verde), Virginica(naranja)", pch=19,col=col.iris)
```

Data set Iris, Setosa(azul), Versicolor(verde), Virginica(naranja)



Método k-vecinos más próximos

```
#Se carga la librería  
library(class)
```

1. Se fija una “semilla” (para obtener los mismos valores).

```
set.seed(1000)
```

Creación de los ciclos

En este caso será un ciclo de $k=1$ hasta $k=20$.

```
# Inicialización de una lista vacía de tamaño 20  
knn.class<-vector(mode="list",length=20)  
knn.tables<-vector(mode="list", length=20)
```

```
# Clasificaciones erróneas  
knn.mis<-matrix(NA, nrow=20, ncol=1)
```

```
for(k in 1:20){
  knn.class[[k]]<-knn.cv(x,y,k=k)
  knn.tables[[k]]<-table(y,knn.class[[k]])
  # la suma de las clasificaciones menos las correctas
  knn.mis[k]<- n-sum(y==knn.class[[k]])
}
```

```
knn.mis
```

```
##      [,1]
## [1,]    6
## [2,]    7
## [3,]    6
## [4,]    6
## [5,]    5
## [6,]    4
## [7,]    5
## [8,]    5
## [9,]    4
## [10,]   5
## [11,]   4
## [12,]   6
## [13,]   5
## [14,]   3
## [15,]   4
## [16,]   5
## [17,]   4
## [18,]   3
## [19,]   3
## [20,]   4
```

```
# Número óptimo de k-vecinos
which(knn.mis==min(knn.mis))
```

```
## [1] 14 18 19
```

2.Se visualizan los resultados que nos arrojó el ciclo con el error más bajo.

```
knn.tables[[14]]
```

```
##
## y          setosa versicolor virginica
## setosa      50         0         0
## versicolor  0         48         2
## virginica   0         1        49
```

```
knn.tables[[18]]
```

```
##
## y          setosa versicolor virginica
## setosa      50         0         0
## versicolor  0         48         2
## virginica   0         1        49
```

```
knn.tables[[19]]
```

```
##  
## y          setosa versicolor virginica  
## setosa      50         0         0  
## versicolor  0         48         2  
## virginica   0         1         49
```

En los tres casos nos da el mismo resultado donde setosa clasifica de manera perfecta, en el caso de versicolor 48. Y virginica sólo una la clasifica como versicolor.

```
# Se señala el k mas eficiente:  
k.opt<-14
```

```
knn.cv.opt<-knn.class[[k.opt]]
```

3.Se visualiza la tabla de contingencia con las clasificaciones buenas y malas:

```
knn.tables[[k.opt]]
```

```
##  
## y          setosa versicolor virginica  
## setosa      50         0         0  
## versicolor  0         48         2  
## virginica   0         1         49
```

La cantidad de observaciones mal clasificadas:

```
knn.mis[k.opt]
```

```
## [1] 3
```

Esto quiere decir que de 100 flores, 2 no están bien clasificadas.

```
# Error de clasificacion (MR)  
knn.mis[k.opt]/n
```

```
## [1] 0.02
```

4.Ahora se crea un gráfico identificando las clasificaciones correctas y erróneas.

```
# Grafico de clasificaciones  
col.knn.iris<-c("indianred1","black")[1*(y==knn.cv.opt)+1]  
pairs(x, main="Clasificación kNN de Iris",  
      pch=19, col=col.knn.iris)
```

Clasificación kNN de Iris

