Análisis Discriminante Lineal

Carlos Enrique Carleos Artime

March 4, 2025

Contents

1 Datos de ejemplo: iris

- Famoso conjunto de datos de Físher (1936).
- Medidas en centímetros de cuatro variables:
  - longitud de sépalos
  - anchura de sépalos

  - longitud de pétalos
  - anchura de pétalos

- Tamaño muestral: 150 registros en 3 grupos indicados por la variable «especie»:
  - 50 registros de *Iris setosa*
  - 50 registros de *Iris versicolor*
  - 50 registros de *Iris virginica*

- En R:
  - `iris # dataframe`
  - `?iris # ayuda`

- En Rcmdr:
  - Datos
  - Conjunto de datos en paquetes
  - Leer conjunto de datos en paquete adjunto
  - Paquete: datasets
  - Conjunto de datos: iris
  - Aceptar

2 Descripción univariante

```
pdf("/tmp/iris.pdf")
for (i in 1:4) boxplot (iris[,i] ~ iris$Species, main = names(iris)[i])
dev.off()
```

- *Iris setosa* se separa fácil del resto.
- Para las otras, de mejor a peor:
  - longitud/anchura de pétalos => descripción bivariante
  - longitud de sépalos
  - anchura de sépalos

3 Descripción bivariante

```
plot (Petal.Width ~ Petal.Length, iris, col=Species)
```

- La separación es bastante clara.
- Deberían producirse pocos errores de clasificación

4 Advertencia sobre evaluación de clasificadores

- Wikipedia
- Al usar técnicas de aprendizaje automático (de clasificación o de regresión) hay que tener en cuenta el problema del **sobreajuste**.
- Dada cualquier muestra de datos, es evidente que se puede construir un modelo que se ajuste perfectamente a ellos. Por ejemplo, cualquiera de las dos siguientes funciones clasificaría correctamente el 100% de los datos `iris`:

```
## buscar si el nuevo individuo coincide con alguno de los que ya existen;
## si no, clasificarlo como "setosa", por ejemplo:
clasificar1 <- function (long.sep, anch.sep, long.pet, anch.pet)
{
    for (i in 1:150)
        if (all (c (long.sep, anch.sep, long.pet, anch.pet) == iris [i, 1:4]))
          return (as.character (iris [i, "Species"]))
    "setosa"
}
## buscar al individuo más cercano, según la distancia euclídea,
## y asignarle su grupo
clasificar2 <- function (long.sep, anch.sep, long.pet, anch.pet)
  as.character (iris [which.min (apply (iris[,1:4], 1, function (fila)
                                        sum ((fila -
                                              c (long.sep, anch.sep,
                                                 long.pet, anch.pet))^2))),
                      "Species"])
## comprobación de que se aciertan todos
for (clasificar in  c (clasificar1, clasificar2))
    print (all (apply (iris[,1:4], 1,
                       function (f)     # "f" de "fila"
```

```
clasificar (f[1], f[2], f[3], f[4])) == iris$Species))
```

- Sin embargo, ¿cómo funcionarían esos clasificadores al usarlos sobre datos nuevos?
  - `clasificar1` clasificaría como "setosa" cualquier dato nuevo;
  - `clasificar2` para clasificar al dato nuevo usaría sólo el dato viejo más cercano, aunque estuviesen rodeados de datos de otro grupo.

  En ningún caso parecen clasificadores óptimos, aunque obtengan el 100% de aciertos sobre los datos originales. Están **sobreajustados**.
- Para dar una estimación más objetiva de la tasa de error de un modelo, se recurre a
  - método de retención
  - validación cruzada

4.1 Método de retención

Consiste en dividir los datos originales en dos partes:

Conjunto de Entrenamiento Suele ser el subconjunto más grande. Habitualmente, del 70% al 90% de los datos originales. El modelo clasificador se calcula usando sólo los datos de entrenamiento.

Conjunto de Validación Son los datos restantes. Se utilizan para probar el modelo generado a partir de los datos de entrenamiento. La tasa de error se calcula sobre dichas pruebas.

4.2 Validación cruzada

Si la aplicación del método de retención requiere poco tiempo de computación, se puede aplicar la idea de forma repetida, de forma que en cada repetición:

- Se muestrea un nuevo conjunto de entrenamiento.
- Se calcula una tasa de error.

Al final, se promedian todas las tasas de error obtenidas.

El muestreo puede ser determinista o aleatorio, siendo más habitual lo primero. Así, en una validación cruzada de orden k:

- Se dividen los datos originales en k grupos.
- Se repite con i desde 1 hasta k lo siguiente:
  - Se construye un clasificador con todos los grupos menos el i-ésimo.
  - Se usa ese clasificador sobre los datos del grupo i-ésimo.
  - Sea E_i la tasa de error obtenida con ese clasificador en el grupo i.
- Se calcula finalmente la tasa de error promedio: $\frac{1}{k}\sum_{i=1} E_i$

Valores habituales de k son $k = 5$ y $k = 10$.

4.3 Dejando uno fuera

Si la potencia de cálculo lo permite, la forma más completa de usar validación cruzada es tomar k igual al tamaño muestral. De esa forma, en cada iteración el clasificador se construye "dejando uno fuera". En inglés se usan las siglas LOO (leave one out) para referirse a este método.

5 Análisis discriminante

5.1 Cargar la biblioteca necesaria

Buscamos una función que realice análisis discriminante:

```
help.search ("discriminant")    # MASS::lda    Linear Discriminant Analysis
library (MASS)
```

5.2 Con retención (muestras de entrenamiento y de validación)

5.2.1 Cálculo del modelo a partir de la muestra de entrenamiento

```
entrena <- sample (150, 120)            # 120 = 80% de 150 para entrenar
modelo  <- lda (Species ~ ., iris)      # el punto «.» = resto de variables en «iris»
modelo  <- lda (Species ~ Petal.Length + Petal.Width + Sepal.Length + Sepal.Width,
                iris)                   # equivale a la anterior orden
modelo
```

5.2.2 Análisis de la salida

La salida obtenida incluye lo siguiente:

- Importancia de los ejes discriminantes

```
Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

  El primer eje explica más del 99% de la discriminación (variabilidad entre los grupos).

- Interpretación de los ejes

```
Coefficients of linear discriminants:
                    LD1         LD2
Petal.Length -2.2012117  0.93192121
Petal.Width  -2.8104603 -2.83918785
Sepal.Length  0.8293776 -0.02410215
Sepal.Width   1.5344731 -2.16452123
```

  - Primer eje (>99%)
    - Puntuaciones altas = pétalos pequeños y sépalos grandes.
    - Contrapone pétalos a sépalos.
  - Segundo eje (<1%)
    - Puntuaciones altas = pétalos largos y estrechos
    - Contrapone pétalos alargados a pétalos anchos

5.2.3 Clasificación en la muestra de validación

```
prediccion <- predict(modelo, iris[-entrena,])
names (prediccion)             # class  posterior  x
```

- «class» sirve para calcular la matriz de confusión (errores de clasificación)

```
> table (real = iris[-entrena,"Species"], predicho = prediccion$class)
            predicho
real         setosa versicolor virginica
  setosa         16          0         0
  versicolor      0          8         0
  virginica       0          0         6
```

Todas las muestras de validación se han clasificado bien. 100% efectividad.

- «posterior» sirve para detectar muestras dudosas, o cuantificar la credibilidad de la clasificación.

```
posteriores <- round (prediccion$posterior, 3)
## si fuesen muchos, podrían ordenarse así:
entropia <- function (p) -sum (ifelse (p==0, 0, p * log2(p))) # 0 en lugar de Nan
tail (posteriores [order (apply (posteriores, 1, entropia)), ])
```

  El peor caso es el «128», clasificada como «virginica» con sólo un 86,6% de seguridad (13,4% como «versicolor»).

- «x» son las coordenadas en el espacio de los ejes discriminantes:

```
plot (prediccion $ x, col = iris[-entrena,"Species"])
```

  - Se observa la importancia del primer eje en la separación de los grupos.

5.3 Con validación cruzada «dejando uno fuera»

Se puede pedir al comando «lda» que haga una validación cruzada exhaustiva, dejando uno fuera en cada iteración, es decir,

- se entrena el modelo con todos los datos menos uno;
- se predice la clasificación de éste y se compara con su clase real;
- se repite el proceso para todos los datos de la muestra original.

```
modelo  <- lda (Species ~ ., iris, CV=TRUE)  # validación cruzada
table (iris$Species, modelo$class)           # matriz de confusión
errores <- iris$Species != modelo$class      # individuos mal clasificados
iris [errores, ]                             # datos de esos individuos
modelo$posterior [errores, ]                 # seguridad en la clasificación
## representación destancando los errores
plot (Petal.Width ~ Petal.Length, iris, col=Species, pch=1+16*errores)
```