MEMÓRIA DE PRÁCTICAS: EVALUACIÓN DE MODELOS

Biometría – Master en IARFID

Enric Bonet Cortés
Universidad Politécnica de Valencia

Realizar un programa que dado dos ficheros: scores clientes y scores impostores obtenga:

- Curva ROC
- FP(FN = X) y umbral
- FN(FP = X) y umbral
- FP = FN y umbral
- Área bajo la curva ROC
- D-Prime

En este primer ejercicio, se ha desarrollado un programa en el lenguaje Python, cuyo código se encuentra en el fichero "rocExercise.py". No obstante, aclarar que el primer parámetro del programa es una letra A o B (en mayúscula o en minúscula), que se utiliza para saber cuál de los dos sistemas de verificación se quiere evaluar, y el segundo parámetro es un numero en coma flotante que será utilizado para los tres puntos siguiente al cálculo de la curva ROC. Este programa cuenta con una opción -h o –help que permite visualizar bajo que parámetros se puede ejecutar. Así pues, para la ejecución de este programa **los ficheros con los scores deben de estar en la misma ruta que .py**.

Para cumplir los requisitos del ejercicio, lo primero que realiza este programa es un procesado de ambos ficheros de texto con los *scores* pertenecientes a los clientes y a los impostores almacenando estos *scores* en dos listas de Python diferentes, para luego juntarlos en una sola lista en forma de tuplas, de forma que estas tuplas almacenan en su primer valor dicho score, y en el segundo a quién pertenece, si a un cliente ("C") o a un impostor ("I").

Esta nueva lista, es de nuevo procesada para sacar de ella los valores sin repeticiones, creando así una lista con umbrales para dibujar la curva ROC.

La nueva lista es recorrida y para cada valor (umbral) se consultan que scores de la lista completa son "falsos negativos" y cuales "falsos positivos", para así, almacenar en una lista 'x', la tasa de falsos positivos, y en otra lista 'y' la tasa de verdaderos positivos (1 - FN) para cada umbral. Ambas listas, 'x' e 'y', se pasan a la función **plot()** de **matplotlib.pyplot**, y de esta forma se dibuja la curva ROC.

Tras esto, se calcula el área de la curva y se imprime por terminal, siguiendo la siguiente fórmula:

$$AreaROC = \frac{1}{n_c}\sum_{i=1}^{N}\frac{j}{i}, tal\ que: i, j \in C_j == S_i$$

En la fórmula anterior, nótese que C_j, es un cliente 'j' dentro de la lista de scores de los clientes, cuya posición en la lista completa de scores es 'i'. También hay que aclarar, que n_c hace referencia a la talla de la lista de scores de los clientes, y N es la talla de la lista completa.

Tras el cálculo del área de la curva, para los siguiente tres objetivos que propone en el ejercicio, se han extraído del propio cálculo de las FN y FP. Estas dos listas se utilizan junto al segundo

parámetro del programa, el número en coma flotante X, para averiguar cuál es el umbral y la tasa FN o FP respecto a un FP/FN = X (o cercano en caso de que no exista).

Poniendo como ejemplo FN(FP = X), se realiza una búsqueda de un valor de la lista FP igual o lo más cercano posible a X, y se recupera su índice. Con ese mismo índice, se imprime por pantalla tanto el valor que ocupa esa posición en la lista FN como ese mismo valor en la lista de umbrales.

Para el caso de FN = FP, simplemente se almacena una lista de distancias en las que se almacena por cada posición i, el valor absoluto de FN – FP en esa posición. De esta forma, resulta sencillo realizar una búsqueda del mínimo valor de esta resta, y con el indexar la lista de umbrales para imprimirla por pantalla. **Numpy**, otra librería de Python, realiza estas dos operaciones (búsqueda de un mínimo y su posición en una lista pasada como parámetro) en una sola función llamada **argmin**.

Por último, para calcula **d'**, primero se realiza el cálculo de la media aritmética de cada lista de scores (cliente e impostores), para posteriormente calcular también la desviación típica de cada conjunto. Con la varianza (potencia cuadrada de la desviación típica) y la media, se aplica la fórmula vista en clase para calcular la **d'** e imprimirla por pantalla:

$$d' = \frac{\mu_{clientes} - \mu_{impostores}}{\sqrt{\sigma^2_{clientes} + \sigma^2_{impostores}}}$$