César Isaí García Cornejo
cesar.cornejo@cimat.mx
Ciencia de Datos

Tarea 1

Semestre 3

CIMAT

Ejercicio 1
Considere tres poblaciones Poisson, con parámetros $\lambda_1 = 10$, $\lambda_2 = 15$, $\lambda_3 = 20$ respectivamente.

1. Establezca la regla óptima de clasificación, basandose en una sola observación, x.
2. Calcule la probabilidad de error asociado a esta regla óptima.
3. Escriba un programa (R o Python) para validar, vía simulación, el nivel de error encontrado en el inciso anterior.

Solución:

1. Por el siguiente Teorema construimos la regla óptima para la clasificación de interes.

   Teorema:

   Supongamos que $Y \in \mathcal{Y} = \{1, ..., K\}$. La regla óptima es

   $$\begin{aligned} g(x) =& argmax_k \mathbb{P}(Y = k|X = x), \\ =& argmax_k \pi_k f_k(x) \end{aligned} \tag{1}$$

   donde

   $$\mathbb{P}(Y = k|X = x) = \frac{f_k(x)\pi_k}{\sum_r f_r(x)\pi_r},$$

   $\pi_r = \mathbb{P}(Y = r)$, $f_r(x) = f(x|Y = r)$ y $argmax_k$ significa "el valor de k que maximiza la expresión."

   De esta forma, notamos que las tres poblaciones están *dispersas* en $\mathbb{N} \cup \{0\}$. Como la media de cada población es su propio parámetro entonces las tres poblaciones están ordenadas en la recta real. Con la única observación $x \in \mathbb{R}$ y considerando las barreras de decisión (suponiendo que $\pi_1 = \pi_2 = \pi_3 = 1/3$)

   $$\mathbb{P}(X = x|Y = 1) = \mathbb{P}(X = x|Y = 2), \quad \mathbb{P}(X = x|Y = 2) = \mathbb{P}(X = x|Y = 3)$$

   Manipulando tenemos

   $$\begin{aligned} \mathbb{P}(X = x|Y = 1) &= \mathbb{P}(X = x|Y = 2), \\ \frac{\lambda_1^x}{x!}e^{-\lambda_1} &= \frac{\lambda_2^x}{x!}e^{-\lambda_2}, \\ \left(\frac{\lambda_1}{\lambda_2}\right)^x &= e^{\lambda_1 - \lambda_2} \\ x \log\left(\frac{\lambda_1}{\lambda_2}\right) &= \lambda_1 - \lambda_2, \\ x &= \frac{\lambda_1 - \lambda 2}{\log(\lambda_1) - \log(\lambda_2)} \\ x &\approx 12{,}33 \end{aligned}$$

   Análogamente,

   $$\begin{aligned} \mathbb{P}(X = x|Y = 2) &= \mathbb{P}(X = x|Y = 3), \\ x &\approx 17{,}38 \end{aligned}$$

De está forma, podemos construir $g(x) \to \mathcal{Y}$ partiendo el intervalo como

$$g(x) = \begin{cases} 1 & x \leq 12{,}33 \\ 2 & 12{,}33 < x \leq 17{,}38 \\ 3 & x \geq 17{,}38 \end{cases}$$

como $x \in \mathbb{N} \cup \{0\}$ se obtiene finalmente

$$g(x) = \begin{cases} 1 & x \leq 12 \\ 2 & 13 \leq x \leq 17 \\ 3 & x \geq 18 \end{cases} \tag{2}$$

la regla de clasificación óptima.

2. Calculemos ahora la probabilidad de error $\mathbb{P}(g(X) \neq Y)$ de la función $g(X)$ asociada.

$$\begin{aligned} \mathbb{P}(g(X) \neq Y) &= 1 - \mathbb{P}(g(X) = Y) \\ &= 1 - \mathbb{P}(g(X) = 1|Y = 1)\pi_1 - \mathbb{P}(g(X) = 2|Y = 2)\pi_2 - \mathbb{P}(g(X) = 3|Y = 3)\pi_3 \\ &= 1 - \frac{1}{3}\left(\mathbb{P}(X \leq 12|Y = 1) + \mathbb{P}(13 \leq X \leq 17|Y = 2) + \mathbb{P}(X \geq 18|Y = 3)\right), \\ &= 0.3414 \end{aligned}$$

donde las cuentas se obtuvieron computacionalmente con el código adjunto.

3. Para poder obtener vía simulación una probabilidad nos basaremos en la ley de grandes números. Es decir tomaremos la proporción de errores de clasificación obtenidos bajo m iteraciones. Seleccionando una de las tres poblaciones posibles con sus respectivos parámetros y simulando una variable poisson, podemos entonces usar $g(x)$ para clasificar la realización x en la simulación. Tomando m relativamente grande, este debe converger a la probabilidad de error de la regla de clasificación.

   Dicha simulación se implemento en Python. Para obtener una elección al azar de las clases usamos el comando

```
    r = random.choice(clases) #Seleccionar clase
```

   Luego con scipy se simula la Poisson con su respectivo parámetro. Así podemos entonces usar la regla de clasificación para contar los errores que produce, mostrado en el condicional que adjunta a una lista una variable booleana que es True si está mal clasificado. Finalmente cuenta el promedio de errores en dicha lista.

```
for i in range(m):
    r = random.choice(clases)
    x = poisson.rvs(lamda[r-1])
    if r == 1:
        error.append(x >= 13)
    elif r == 2:
        error.append( (x <= 12) or (x>= 18))
    elif r == 3:
        error.append((x <= 17))

print("La proporción de errores es: ", sum(error)/len(error))
```

   El resultado obtenido por está simulación nos dice que

$$\mathbb{P}(g(X) \neq Y) \approx 0{,}3421 \tag{3}$$

que es bastante similar al obtenido analíticamente. Enfatizo que el enlace al código detallado se encuentra aquí

Ejercicio 2
Descargue la base de datos llamada Wine.

1. Basados en las características químicas de los vinos, construya un clasificador para determinar el origen de los mismos.
2. ¿ Cuántos vinos están mal clasificados por este clasificador?

Solución:

1. Vemos de la base de datos que tenemos 14 columnas de las cuales la primera es la columna de la clase a la que pertenece cada vino. Luego las restantes 13 columnas serán las que pertenecen a los atributos de cada vino. En la Fig 1 mostramos la cabeza de dicha base de datos.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

Figura 1: Primeros datos en wine

Para hacer un modelo de clasificación optamos por construirlo con el clasificador de Bayes Gaussiano. Primeramente, se requiere partir la base de datos en la tradicional particion de datos de prueba y datos de entrenamiento. Para ello y con la paqueteria sklearn podemos hacer la partición con el siguiente comando

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
    random_state=1)
```

Ahora, implementamos el modelo naive Bayes Gaussiano con el comando

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

lo que concluye la construcción del modelo.

2. Para medir la precisión del modelo construido hacemos uso de la predicción que tienen los datos de prueba y comparamos con las verdaderas clasificaciones. Esto se obtiene con el comando siguiente

```
# making predictions on the testing set
y_pred = gnb.predict(X_test)
```

Imprimendo el resultado dado vinos mal clasificados por el modelo propuesto

```
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
    y_pred)*100)
```

se obtiene que la proporción de vinos mal clasificados es de 98.61 % que lo hace un excelente clasificador.

El código en detalle se encuentra aquí.