Ejercicio 2 Señales y Sistemas II parte 2

Pía Contreras - Valentina Norambuena - Camilo Ramírez
Profesores: Manuel Duarte - Marcos Orchard
Auxiliar: Albert Rudolph; Ayudantes: Diego L. - José M.

P1: PCA, Score Plot y Test de Hotteling

En primer lugar, dado el archivo "P2_data.mat", se reemplazó los datos que en su interior contenían Nan por la mediana de la matriz, para que de tal forma, no variara el contenido principal de la matriz, ya que, si por ejemplo, se hubiese reemplazado los Nan con 0, se podría haber alterado el resultado final, con resultados no reales, ya que no se sabía si el número a reemplazar era ciertamente 0, pero si se reemplazaba por la mediana se podían evitar errores mayores ya que los datos no tendrían mayor desviación estándar.

Al normalizar los datos mediante la ecuación:

$$\widehat{X_{ij}} = \frac{x_{ij} - \overline{x}}{\sigma_j} \tag{1}$$

Donde x_{ij} es la matriz dada con los datos Nan reemplazados por la mediana cuyas dimensiones son 205x26, $\overline{x}$ es el promedio de la matriz x_{ij} cuyas dimensiones son 1x26, y finalmente, σ_j es la desviación estándar de la matriz x_{ij} cuyas dimensiones están dadas por 1x26.

Así se obtiene la normalización de los datos, y la matriz obtenida, es una matriz $\widehat{X_{ij}}$ cuyas dimensiones son 205x26. Luego, se calculó la matriz de varianza-covarianza empírica, para ello, se utilizó la siguiente fórmula:

$$S = \frac{1}{n-1}\widehat{X}^T\widehat{X} \tag{2}$$

De donde se tomó como X la matriz normalizada anteriormente obtenida, de ello, al operar, se obtuvo la matriz de varianza-covarianza empírica, de dimensiones 26x26.

Posteriormente, se calculó los valores y vectores propios de la matriz de varianza-covarianza empírica, donde se obtuvo que las dos últimas columnas de la matriz obtenida correspondes a los 2 mayores valores propios, por lo que se tomó como x al vector propio asociado al mayor valor propio y se tomó como y al vector propio asociado al segundo mayor valor propio.

Se construyó la matriz de proyección, como $P = [xy]$ cuyas dimensiones son 26x2. Luego se construyó la matriz Y, como $Y = \widehat{X_{ij}} * P$ cuyas dimensiones están dadas por 205x2.

Se ubicó en el eje de las abscisas el primer vector columna de la matriz Y y en el eje de las ordenadas se ubicó el segundo vector columna de la matriz Y. Al graficar se obtiene la Figura 1:

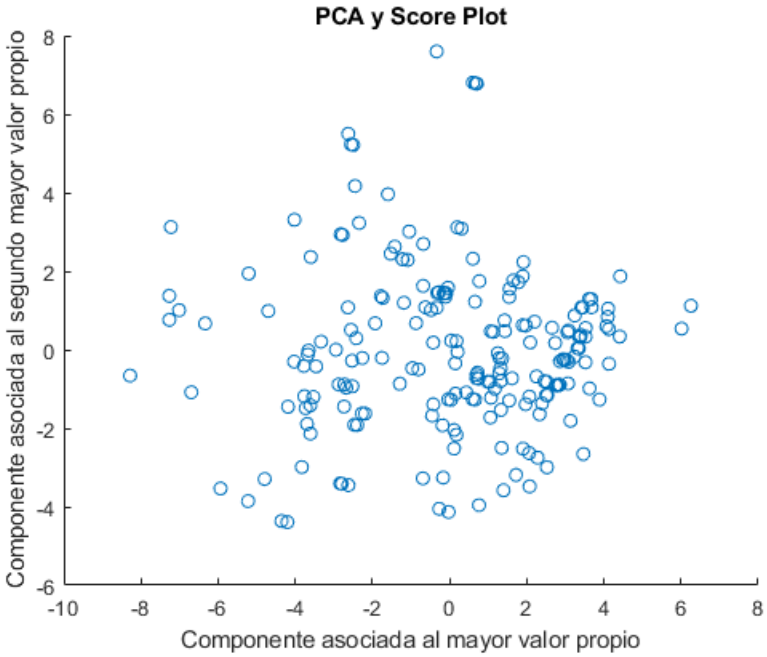

Figura 1: Proyección datos sobre las dos primeras componentes principales.

En la segunda parte de la preguntar, dirigida a calcular el Test de Hotelling, se determinó en una primera instancia la forma cuadrática de la elipse de control en base a las dos primeras componentes principales. Para realizar lo descrito, se siguió la siguiente ecuación:

$$\|x_t\|^2 = \frac{x_{t1}^2}{\sigma_1^2} + \frac{x_{t2}^2}{\sigma_2^2} = x_t P \lambda_a^{-1} P^T x_t^T \tag{3}$$

Entonces, se obtuvo la forma cuadrática como una matriz z como un loop tal que es un vector columna que contiene en cada fila la forma cuadrática correspondiente. El vector resultante es de dimensiones 205x1.

Se recorrió el vector, y junto con la parametrización de una elipse, se graficó cada elipse asociada a cada distancia (cada fila). Para cada fila del vector columna, se obtuvo el valor a y el valor b como:

$$a_i = (vp_1 * \text{vectorcolumna}(i))^{\frac{1}{2}} \qquad b_i = (vp_2 * \text{vectorcolumna}(i))^{\frac{1}{2}}$$

Parametrizando, con vector=[0: 0.01: 2π], es decir, un vector que va desde 0 a 2π a un paso de 0.01 :

$$x_i = a_i * cos(\text{vector})$$
$$y_i = b_i * sin(\text{vector})$$

Como el vector columna tenía dimensiones de 205x1, entonces en el loop, se obtuvo 205 curvas de nivel, cada una asociada a la distancia correspondiente en el vector columna.

Así al graficar las curvas de nivel se obtiene la Figura 2:

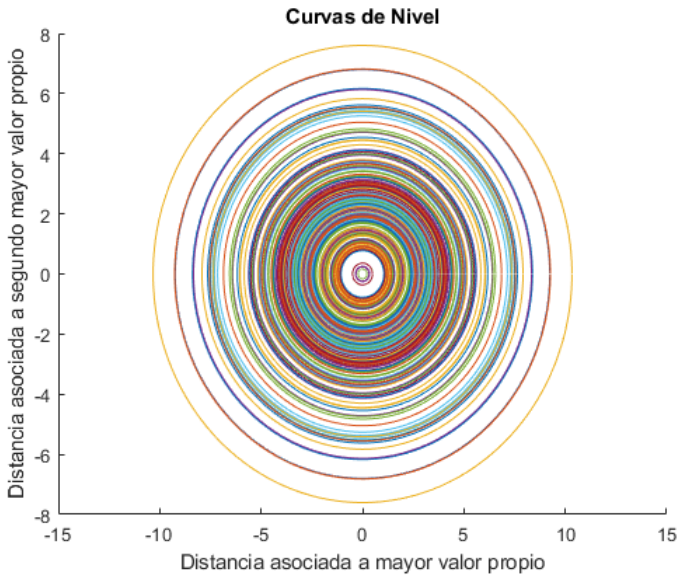

Figura 2: Curvas de Nivel.

Se puede observar además, que para cada punto en la Figura 1 existe una curva de nivel asociada, lo cual se puede observar en la Figura 3:

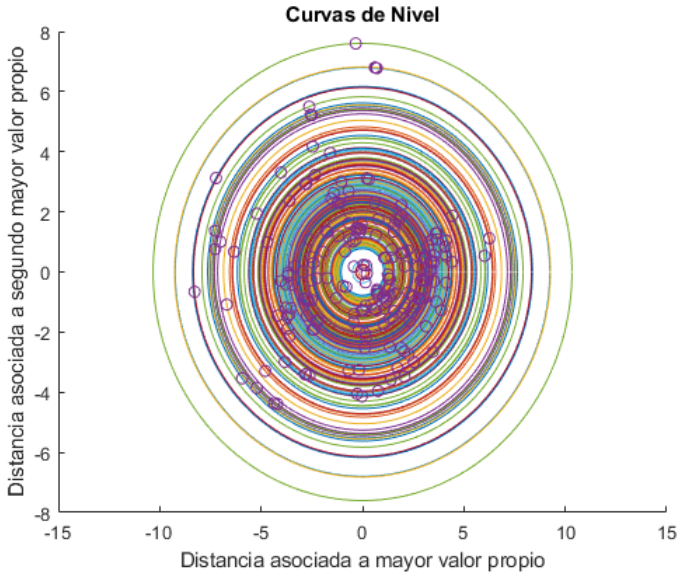

Figura 3: Curvas de Nivel y Scatter plot

Finalmente, se plantea el Test de Hotteling para dos umbrales, 95 % y 99 %. Para ello, lo primero que se hizo fue obtener los coeficientes de Fisher, de donde se obtuvo:

$$F1_{95} = 3{,}888 \qquad F2_{95} = 3{,}041$$
$$F1_{99} = 6{,}763 \qquad F2_{99} = 4{,}713$$

Se definió además, un vector cuya primera componente era el mayor valor propio obtenido y su segunda componente era el segundo mayor valor propio. Se tomó como variable n la cantidad de datos del vector columna de la parte anterior, es decir 205 datos, y como variable a, ésta tomó el valor de dos puesto que se están estudiando las primeras dos componentes más relevantes.

Se definió ν tanto para datos que formaban parte del training set como para datos que no formaban parte del training set. Con cada ν correspondiente, es decir, para el umbral del 95 % y 99 %, se definió el valor de a y de b como la raíz de el valor propio asociado por el ν respectivo.

Se parametrizó la elipse para cada umbral como:

$$x_i = a_i * cos(\text{vector})$$
$$y_i = b_i * sin(\text{vector})$$

Donde vector corresponde a vector=[0: 0.01: 2π]. Finalmente, en el mismo esquema se graficó ambos umbrales para datos que no están en el training set. El gráfico obtenido se muestra en la Figura 4:

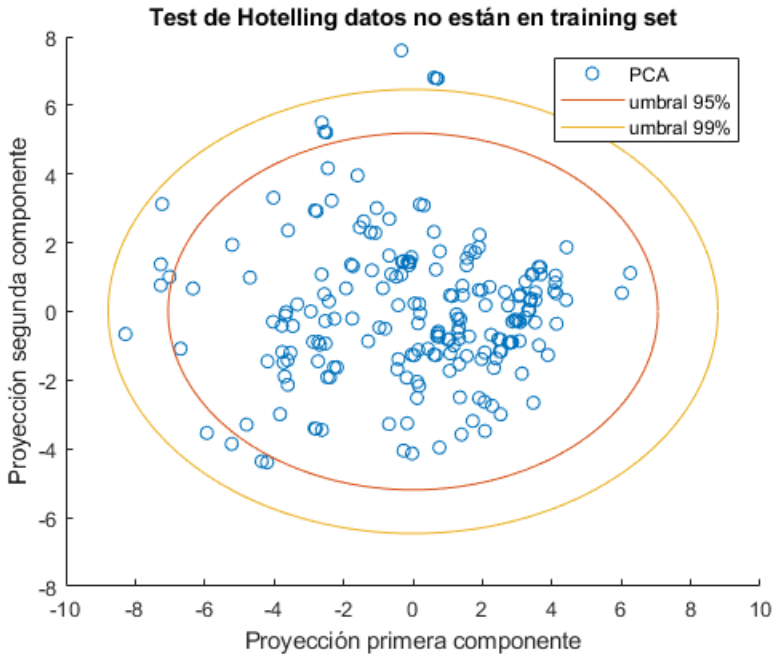

Figura 4: Test Hotteling

Mientras que el test de Hotelling obtenido para datos que si están en el training set está dado por la Figura 5:

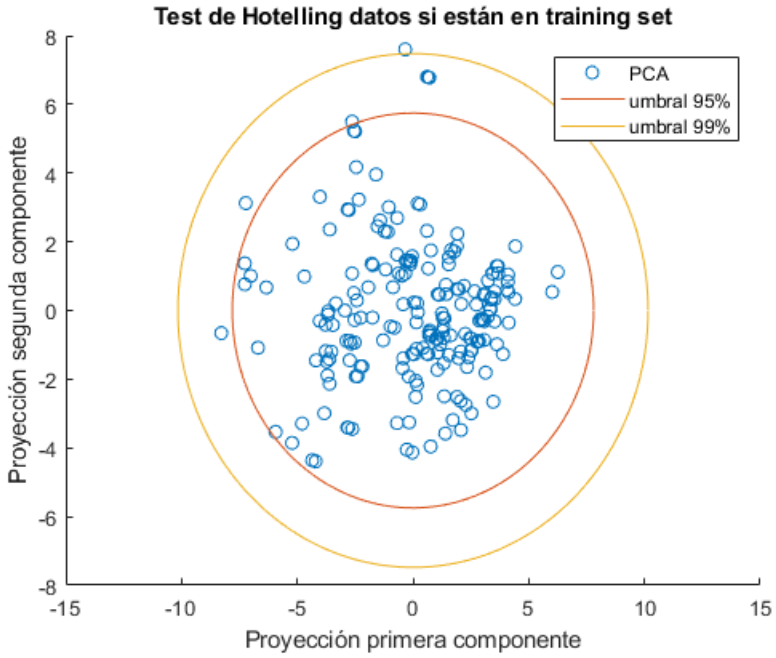

Figura 5: Test de Hotteling para datos en el training set

Se puede observar que la diferencia entre éstos dos gráficos, es decir las figuras 4 y 5, que para el segundo la mayoría de datos se concentra en el umbral de 95 %, mientras que para el test de Hotelling, en la que los los datos no están en el training set, la mayoría de los datos se concentra en la elipse correspondiente al 99 %.

P2: Detección - Planteamiento teórico

Un problema de detección consiste en tomar una decisión entre dos o más clases, a partir de la observación de una variable aleatoria x. En particular, para el problema de detección binario, se debe decidir entre dos hipótesis H_0 y H_1. De este modo, dada una observación y, esto puede verse como:

$$\text{Aceptar } H_0 \longrightarrow y \text{ pertenece a la clase } \omega_0$$
$$\text{Aceptar } H_1 \longrightarrow y \text{ pertenece a la clase } \omega_1$$

Este ejercicio apunta a determinar si un paciente tiene un tumor benigno (clase ω_0) o maligno (clase ω_0), para lo cual debe realizar el siguiente procedimiento:

1. Se busca caracterizar el conjunto de las dos decisiones recién descritas $\Omega = \{\omega_0, \omega_1\}$, mediante la observación de las variables $y_i(t),\ i = \{1, \ldots, N\}$, con N la cantidad de características, por ejemplo, variables o índices asociados al estado de salud del paciente obtenidos mediante exámenes médicos, como el nivel de glicemia, linfocitos, presión arterial, etc. Dichas características se miden a lo largo del tiempo, dando origen a mediciones en $t = \{1, \ldots, T\}$, con T la cantidad de observaciones. Una vez ya medidas las diferentes características, se busca asociarlas con la enfermedad a detectar, en otras palabras, dadas las características y_i medidas en conjunto, se determina la probabilidad de que el paciente padezca o no cáncer.

2. Sea θ_{ji} el valor real de la medición de la característica i que corresponde a la clase j y $n_{ji}(t)$ el ruido asociado la medición de esa característica i que corresponde a la clase j. Si se considera que $n_{ji}(t) \sim N(0, \sigma_{ji}^2)$ e $y_i(t) = \theta_{ji} + n_{ji}(t)$. Se caracteriza probabilísticamente las i características:

$$y_i(t) = \begin{cases} \theta_{0_i} + n_{0_i}(t), & \text{si } y_i \text{ pertenece a la clase } \omega_0 \\ \theta_{1_i} + n_{1_i}(t), & \text{si } y_i \text{ pertenece a la clase } \omega_1 \end{cases} \quad i = \{1, \ldots, N\}$$

   Con $\theta_{0_i},\ \theta_{1_i}$ las mediciones reales de la característica i que se asocian a las clases ω_0 u ω_1, respectivamente. Incluyendo el ruido de las mediciones, la distribución de cada medición es:

$$y_i(t) \sim \begin{cases} N(\theta_{0_i}, \sigma_{0_i}^2), & \text{si } y_i \text{ pertenece a la clase } \omega_0 \\ N(\theta_{1_i}, \sigma_{1_i}^2), & \text{si } y_i \text{ pertenece a la clase } \omega_1 \end{cases} \quad i = \{1, \ldots, N\}$$

   A nivel multidimensionall, se define $Y(t) = [y_1(t), y_2(t), ..., y_N(t)]^T$ como el vector aleatorio de N características medidas en el instante t, que sigue una distribución normal multivariante dependiente de la hipótesis que sea correcta. Esto es:

$$Y(t) \sim \begin{cases} N_{\mathcal{N}}(\vec{\theta_0}, \Sigma_0), & \text{si } Y \text{ pertenece a la clase } \omega_0 \\ N_{\mathcal{N}}(\vec{\theta_1}, \Sigma_1), & \text{si } Y \text{ pertenece a la clase } \omega_1 \end{cases}$$

En donde $\vec{\theta}$ es un vector de N dimensiones que representa el valor real de las mediciones y Σ es la matriz de covarianzas de las mediciones asociadas a esa clase:

$$\Sigma_0 \in \mathbb{R}^{n\times n}, \text{ con: } a_{ij} = \begin{cases} 0, & \text{si } i \neq j \\ \sigma_{0_i}^2 & \text{si } i = j \end{cases}$$

$$\Sigma_1 \in \mathbb{R}^{n\times n}, \text{ con: } a_{ij} = \begin{cases} 0, & \text{si } i \neq j \\ \sigma_{1_i}^2 & \text{si } i = j \end{cases}$$

3. En caso de que se reciba un conjunto de características independientes con $\vec{Y}(t) = [y_1(t), ..., y_N(t)]^T$, el teorema de Bayes dice que la probabilidad de una clase:

$$P(\Omega|Y) = \frac{f(Y|\Omega)P(\Omega)}{f(Y)}$$

En donde la distribución de probabilidad condicionales del tipo $f(Y|\Omega)$ y la distribución $f(Y|\Omega)$ son, debido a ser la agrupación de características independientes:

$$f(Y|\Omega) = \prod_{i=1}^{N} f(y_i|\Omega) \qquad\qquad f(Y) = \prod_{i=1}^{N} f(y_i)$$

De la distribución de las características, la densidad de probabilidad individual dada las clases $f(y_i|\Omega)$ es:

$$f(y_i|\Omega = \omega_0) = f_{y_i|\Omega}(y_i|\omega_0) = \frac{1}{\sigma_{0i}\sqrt{2\pi}}\exp\left(- \frac{(y_i - \theta_{0_i})^2}{2\sigma_{0i}^2}\right)$$
$$f(y_i|\Omega = \omega_1) = f_{y_i|\Omega}(y_i|\omega_1) = \frac{1}{\sigma_{1i}\sqrt{2\pi}}\exp\left(- \frac{(y_i - \theta_{1_i})^2}{2\sigma_{1i}^2}\right)$$

Entonces la expresión cerrada para las probabilidades de clase $\Omega = \omega_0$:

$$\begin{aligned} f(Y|\Omega = \omega_0) &= \prod_{i=1}^{N} f(y_i|\omega_0) \\ &= \prod_{i=1}^{N} \frac{1}{\sigma_{0i}\sqrt{2\pi}}\ \exp\left(- \frac{(y_i - \theta_{0_i})^2}{2\sigma_{0i}^2}\right) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}}\ \exp\left(\sum_{i=1}^{N} - \frac{(y_i - \theta_{0_i})^2}{2\sigma_{0i}^2}\right) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}}\ \exp\left(\sum_{i=1}^{N} - \frac{(y_i - \theta_{0_i})^T\ (y_i - \theta_{0_i})}{2\sigma_{0i}^2}\right) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}}\exp\left(- \frac{1}{2}(\vec{y} - \vec{\theta}_0)^T\Sigma_0^{-1}(\vec{y} - \vec{\theta}_0)\right) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}}\exp\left((\vec{y} - \vec{\theta}_0)^T\hat{\Sigma}_0^{-1}(\vec{y} - \vec{\theta}_0)\right) \end{aligned}$$

Con $|\Sigma_0|$ el determinante de la matriz Σ_0 Y $\hat{\Sigma}_0 = -2\Sigma_0$. Análogamente para la clase ω_1:

$$f(Y|\Omega = \omega_1) = \frac{1}{(2\pi)^{n/2}|\Sigma_1|^{1/2}} \exp\left((\vec{y} - \vec{\theta}_1)^T \hat{\Sigma}_1^{-1} (\vec{y} - \vec{\theta}_1) \right)$$

Luego la probabilidad de una clase es:

$$P(\Omega|Y) = \begin{cases} \frac{f(Y|\omega_0)P(\omega_0)}{\prod_{i=1}^{N} f(y_i)}, & \text{si } y_i \text{ pertenece a la clase } \omega_0 \\ \\ \frac{f(Y|\omega_1)P(\omega_1)}{\prod_{i=1}^{N} f(y_i)}, & \text{si } y_i \text{ pertenece a la clase } \omega_1 \end{cases}$$

4. Para formalizar el problema de minimización de riesgo para detectar la clase, se siguen los siguientes pasos para determinar si se elige la clase ω_0 u ω_1:

   (a) La función de costos r_{ij} asociado a elegir la clase j cuando realmente es la clase i es:

   $$r_{ij} = L_{ij} P(\Omega|Y) = L_{ij} \frac{f(Y|\omega_i)P(\omega_i)}{f(Y)}$$

   Donde L_{ij} es el costo asociado a elegir j cuando $Y \in \omega_i$. Luego el costo total asociado a elegir la clase j, independiente de la clase real de la medición es:

   $$r_j = \sum_{i=0}^{C-1} r_{ij} = \sum_{i=0}^{C-1} L_{ij} \frac{f(Y|\omega_i)P(\omega_i)}{f(Y)} \qquad C\text{: Cantidad total de clases}$$

   (b) En el caso de la decisión binaria con dos clases y las probabilidades ya despejadas:

   $$r_0 = L_{00} \frac{f(Y|\omega_0)P(\omega_0)}{f(Y)} + L_{10} \frac{f(Y|\omega_1)P(\omega_1)}{f(Y)}$$
   $$r_1 = L_{01} \frac{f(Y|\omega_0)P(\omega_0)}{f(Y)} + L_{11} \frac{f(Y|\omega_1)P(\omega_1)}{f(Y)}$$

   En donde se observa que el término $f(Y)$ es común a todos los sumandos.

   (c) Si se asume que los costos de no equivocarse[1] son 0, esto es $L_{00} = L_{11} = 0$ y que la distribución de densidad asociada a una medición posible Y es $f(Y) > 0$, entonces se pueden redefinir los costos como:

   $$\hat{r}_0 = L_{10} f(Y|\omega_1)P(\omega_1)$$
   $$\hat{r}_1 = L_{01} f(Y|\omega_0)P(\omega_0)$$

   En donde los costos L_{10} y L_{01} se dejan como valores a determinar de acuerdo al umbral de decisión. Dada la temática del problema, se realiza la suposición que el costo asociado a decidir que es un tumor benigno cuando realmente es maligno es mayor costoso que en el caso inverso, esto es $L_{10} > L_{01}$.

[1] Lo que no es necesariamente cierto.

(d) El criterio de decisión basado en la minimización de riesgos es:

$$\text{Decisión} : \begin{cases} H_0, & \text{si } \hat{r}_0 < \hat{r}_1 \\ H_1 & \text{si } \hat{r}_0 > \hat{r}_1 \end{cases}$$

$$\Leftrightarrow \hat{r}_0 \underset{H_1}{\overset{H_0}{\lessgtr}} \hat{r}_1$$

$$\Leftrightarrow L_{10} f(Y|\omega_1) P(\omega_1) \underset{H_1}{\overset{H_0}{\lessgtr}} L_{01} f(Y|\omega_0) P(\omega_0)$$

$$\Longrightarrow \frac{f(Y|\omega_1)}{f(Y|\omega_0)} \underset{H_1}{\overset{H_0}{\lessgtr}} \frac{L_{01} P(\omega_0)}{L_{10} P(\omega_1)}$$

5. Finalmente, el criterio de decisión expresado en función de la densidad de probabilidad conjunta de las observaciones de las N características y_i que distribuyen normal con ruido $n_{ji}(t)$ respecto a una variable θ_{ji} con j la clase real, mediante la asociación $\vec{y} = (y_1, \ldots, y_N)^T$ y $\vec{\theta}_j = (\theta_{j1}, \ldots, \theta_{jN})^T$, considerando las probabilidades de clase $P(\omega_0)$ y $P(\omega_1)$ constantes y los costos asociados a equivocarse de clase L_{01} y L_{10} constantes[2], es:

$$\frac{f(Y|\omega_1)}{f(Y|\omega_0)} \underset{H_1}{\overset{H_0}{\lessgtr}} \frac{L_{01} P(\omega_0)}{L_{10} P(\omega_1)}$$

$$\frac{(2\pi)^{n/2}|\Sigma_0|^{1/2}}{(2\pi)^{n/2}|\Sigma_1|^{1/2}} \frac{\exp\left((\vec{y} - \vec{\theta}_1)^T \hat{\Sigma}_1^{-1} (\vec{y} - \vec{\theta}_1)\right)}{\exp\left((\vec{y} - \vec{\theta}_0)^T \hat{\Sigma}_0^{-1} (\vec{y} - \vec{\theta}_0)\right)} \underset{H_1}{\overset{H_0}{\lessgtr}} \frac{L_{01} P(\omega_0)}{L_{10} P(\omega_1)}$$

$$\frac{\exp\left((\vec{y} - \vec{\theta}_1)^T \hat{\Sigma}_1^{-1} (\vec{y} - \vec{\theta}_1)\right)}{\exp\left((\vec{y} - \vec{\theta}_0)^T \hat{\Sigma}_0^{-1} (\vec{y} - \vec{\theta}_0)\right)} \underset{H_1}{\overset{H_0}{\lessgtr}} \frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} \frac{L_{01} P(\omega_0)}{L_{10} P(\omega_1)} = \gamma$$

Si σ_{ji} son i.i.d, entonces $\frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} = 1$.

[2]Dependientes del umbral de decisión

P3: Implementación

En este problema se implementa computacionalmente la solución teórica expuesta en el problema 2, el código completo y ejecutable se encuentra en el fichero `P3_E2-2_EL4003.ipynb`, un *Jupyter Notebook* adjunto a esta resolución. La identificación de cada porción de código está dada por su número de bloque (*chunk*) y de línea en el respectivo bloque (la notación [C. n; l. i] corresponde al bloque n y línea i de código).

Los datos a trabajar corresponden a un procesado de la base de datos *Breast Cancer Winsconsin (Diagnostic) Data Set*[3], estos se entregan en dos ficheros:

- `Clasificador_data.mat`, el cual contiene una tabla de 30 columnas (atributos) de valores numéricos, donde cada cual corresponde a alguna de las características visuales obtenidas de imágenes digitales de los núcleos celulares de una muestra de masa mamaria. La tabla contiene 596 tuplas, una tupla por paciente.
- `Clasificador_label.mat`, el cual contiene la misma cantidad de tuplas que la tabla del fichero anterior, cada cual relacionada según su índice. Respecto a los atributos, esta tiene una columna única, la cual corresponde a una etiqueta de clase `B` en caso que el paciente corresponda a alguien con células benignas y `M` en caso de un paciente con células malignas. Ambas clases las representa el caracter correspondiente.

  Es importante identificar que en el contexto de clasificación, la clase `B` corresponde a una medición típica, es decir, hipótesis nula H_0 y la clase `M` corresponde a una medición atípica, es decir, hipótesis alternativa H_1.

Previo a la implementación de las funciones requeridas, se realiza un breve preprocesamiento de datos. Entendiendo que la información necesaria para procesar los mismos se encuentra disgregada en los ficheros mencionados previamente y que es necesario mantener la cohesión entre los datos con su respectiva etiqueta, es que toda la información une en una única tabla, mas específicamente se crea un nuevo tipo de dato de la librería `numpy` (`dtype` llamado `labeled_dtype`) que permite generar arreglos con elementos que siguen la estructura:

```
(data, label)
```

Según se define en [C. 1; l. 14]. `data` corresponde a un arreglo de datos numéricos (tamaño 30 tipo `float`) asociados a un determinado paciente, mientras que `label` corresponde a un caracter. La mezcla se realiza en el bucle en [C. 1; l. 18].

Función auxiliar `BM_split`

A modo de facilitar el trabajo posterior, se define la función auxiliar `BM_split` en [C. 2; l. 3] cuya función es particionar un *dataset* (`x`) en dos arreglos `B_subset` y `M_subset`, donde cada cual corresponde a un subconjunto del *dataset* cuyos datos son exclusivamente datos clase `B` y `M` respectivamente, esta separación se realiza en el bucle programado en [C. 2; l. 18].

[3] `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`

El único parámetro de la función `BM_split` es:

- `x`: Arreglo de *numpy* con `dtype` = `labeled_dtype` correspondiente al *dataset* preprocesado o algún subconjunto de este.

Mientras que retorna la tupla `(B_subset, M_subset)`, donde:

- `B_subset`: Subconjunto de `x` parte de la partición de datos de `x` de datos de clase `B` únicamente.
- `M_subset`: Subconjunto de `x` parte de la partición de datos de `x` de datos de clase `M` únicamente.

(a) Implementación de la función `Database`

Esta función se encuentra implementada en el bloque 3 del archivo adjunto y realiza una partición de los datos de un *dataset* con la estructura resultante del preprocesamiento (`x`) según un muestreo aleatorio estratificado de los datos, dentro de los parámetros de la función se encuentran:

- `x`: Arreglo de *numpy* con `dtype` = `labeled_dtype` correspondiente al *dataset* preprocesado.
- `train`: Número correspondiente al porcentaje de `x` con el cual se formará el subconjunto de datos de entrenamiento, valor por defecto: `60`, tiene como restricción que `train` > 0.
- `test`: Número correspondiente al porcentaje de `x` con el cual se formará el subconjunto de datos de prueba, valor por defecto: `40`, tiene por restricción que `test` > 0.

Una restricción adicional es que en conjunto `train`+`test` ≤ 100, notar que no necesariamente `train` y `test` suman 100. El retorno de la función por su parte, corresponde a `(train_set, test_set)`, es decir, una tupla en donde:

- `train_set`: Subconjunto de `x` de entrenamiento, tiene un tamaño del `train` % del tamaño de `x`.
- `test_set`: Subconjunto de `x` de prueba, tiene un tamaño del `test` % del tamaño de `x`.

Lo importante de lo conjuntos de salida, es que son disjuntos, es decir, no tienen tuplas en común, los datos se distribuyen en estos de forma aleatoria, y las proporciones de clases `B` y `M` en `train_set` y `test_set` son las mismas que en `x`, esta decisión de diseño se fundamenta en eliminar la posibilidad de generación de sesgos en el entrenamiento.

(b) Implementación de la función `Parametros`

Función implementada en el bloque 4 del archivo adjunto que a partir de un *dataset* `X` obtiene los parámetros necesarios para realizar la detección de clase `M` según los resultados planteados en el problema 2, donde estos corresponden a $\vec{\theta}_0$, $\vec{\theta}_1$, $\hat{\Sigma}_0$ y $\hat{\Sigma}_1$.

Los parámetros con subíndice 0 corresponden a aquellos calculados a partir del subconjunto de clase `B` de `X`, mientras que aquellos con subíndice 1 son aquellos obtenidos a partir del subconjunto de clase `M` de `X`, la partición mencionada se realiza con la función `BM_split`.

Notar que para $i \in \{0, 1\}$, θ_i es un vector, para retornar este, basta devolver un arreglo con sus coordenadas, pero por otro lado, para $\hat{\Sigma}_i$, como esta es una matriz diagonal tal que $\hat{\Sigma}_i = -2\Sigma_i$ y $\Sigma_i = \text{diag}(\sigma^2_{i_1}, \sigma^2_{i_2}, \ldots, \sigma^2_{i_{m-1}}, \sigma^2_{i_m})$ donde m corresponde a la cantidad de atributos, entonces para retornar la información suficiente para operar $\hat{\Sigma}_i$, basta entregar un arreglo que contenga los valores de la diagonal.

El único parámetro de la función `Parametros` corresponde a:

- `X`: Arreglo de *numpy* con `dtype` = `labeled_dtype`, corresponde al subconjunto de datos de entrenamiento, es decir a `train_set` resultante de la aplicación de la función `DataBase`.

Mientras que el retorno de la función [C. 3; l. 33] corresponde a:

```
((B_promedio, B_std), (M_promedio, M_std))
```

En donde cada una de las salidas está dada por:

- `B_promedio`: Arreglo de valores tal que `B_promedio[`$i-1$`]` $= \theta_{0_i}$, donde cada valor del arreglo se calcula usando el promedio, es decir, `numpy.mean(...)`, ver línea [C. 4; l. 29].
- `M_promedio`: Arreglo de valores tal que `M_promedio[`$i-1$`]` $= \theta_{1_i}$, donde cada valor del arreglo se calcula de forma análoga a los de `B_promedio`, ver línea [C. 4; l. 30].
- `B_std`: Arreglo de valores tal que `B_std[`$i-1$`]` $= \sigma_{0_i}$, donde cada valor del arreglo se calcula usando raíz del estimador insesgado de la varianza, es decir, `numpy.std(..., ddof=1)`, donde el parámetro `ddof=1` indica que la suma de diferencias cuadráticas entre los valores y su promedio se divide por $m-1$ y **no** por m, ver línea [C. 4; l. 31].
- `M_std`: Arreglo de valores tal que `M_std[`$i-1$`]` $= \sigma_{1_i}$, donde cada valor del arreglo se calcula de forma análoga a los de `B_std`, ver línea [C. 4; l. 32].

En toda la formulación descrita i refiere al atributo o columna, por esto los vectores devueltos por la función tienen todos tamaño 30.

(c) Implementación de la función de detección

La función de detección, llamada `Detector`, se encuentra implementada en el bloque 5 del archivo adjunto y considera como entrada el conjunto de pruebas en suma con los parámetros generados por la función de `Parametros` y un umbral definido arbitrariamente por el usuario devolviendo un vector de ceros y unos donde cada valor se asocia a un dato de los ingresados como conjunto de prueba tomando valor 0 en caso que el detector lo asocie a la clase `B` y 1 en caso que el detector lo asocie a la clase `M`.

Los parámetros de la función `Detector` corresponden a:

- `parametros`: Tupla de tuplas de arreglos de números con la forma del retorno de la función `Parametros`, es decir: `((B_promedio, B_std), (M_promedio, M_std))`.
- `test_set`: Arreglo de arreglos de valores numéricos (sin etiqueta de clase) del subconjunto de datos de prueba obtenidos, es equivalente a la aplicación de `DataBase(datos)[1]["data"]`, donde `datos` corresponde al *dataset* completo posterior al preprocesamiento.
- `umbral`: Corresponde al número real γ según la formulación del problema anterior.

Mientras que su retorno corresponde a:

- `resultados`: Arreglo de valores enteros 0 o 1, donde cada cual está asociado a un dato presente en `test_set` siendo 0 si el detector entrega la hipótesis nula o 1 si lo hace con la hipótesis alternativa.

El mecanismo de realización de la detección corresponde a calcular la expresión resultante de la parte 5. del problema 2, es decir, calcular la siguiente expresión para cada tupla (ver código en [C. 5; l. 20]) y compararla con el valor `umbral`:

$$\frac{\exp\Big((\vec{y}-\vec{\theta_1})^{\mathsf{T}}\hat{\Sigma}_1^{-1}(\vec{y}-\vec{\theta_1})\Big)}{\exp\Big((\vec{y}-\vec{\theta_0})^{\mathsf{T}}\hat{\Sigma}_0^{-1}(\vec{y}-\vec{\theta_0})\Big)} = \frac{\exp\Big(\sum_{i=1}^{m}\Big[-\left(y_i-\theta_{0_i}\right)^2/\left(2\sigma_{0_i}^2\right)\Big]\Big)}{\exp\Big(\sum_{i=1}^{m}\Big[-\left(y_i-\theta_{1_i}\right)^2/\left(2\sigma_{1_i}^2\right)\Big]\Big)}$$

Como se puede ver en el código de [C. 5; l. 22] el numerador de la expresión anterior se almacena como `M_exp` y se calcula equivalente a:

$$\exp\left(\sum_{i=1}^{m}\left[-\frac{(\texttt{test\_set[obs\_index]}[i-1]\ \texttt{-}\ \texttt{M\_promedio}[i-1])^2}{2\cdot(\texttt{M\_std}[i-1])^2}\right]\right)$$

Mientras que el denominador se almacena como `B_exp` como se ve en el código [C. 5; l. 23] y se calcula equivalente a:

$$\exp\left(\sum_{i=1}^{m}\left[-\frac{(\texttt{test\_set[obs\_index]}[i-1]\ \texttt{-}\ \texttt{B\_promedio}[i-1])^2}{2\cdot(\texttt{B\_std}[i-1])^2}\right]\right)$$

Donde `obs_index` indica el índice correspondiente al de la tupla de datos que se está clasificando. Notar que tanto `B_exp` como `M_exp` toman valores con órdenes de magnitud altamente variables, el extremo de esto es que pueden superar el límite de la representación numérica de punto flotante por lo que pueden tomar valor `0` en caso que el orden de magnitud sea muy pequeño y por otro lado valor `np.inf` (repreentación en *numpy* de ∞) en caso que el orden de magnitud sea muy elevado. Así, el valor de la expresión `M_exp/B_exp` en caso que `B_exp` $\to 0$ se tiene que se hace ∞.

Lo anterior implica que para obtener variaciones significativas en el resultado de la detección en cuanto a las tasas de falsos positivos y de detección, es necesario modificar significativamente el orden de magnitud de γ.

Finalmente, como es posible ver en [C. 5; l. 26], el valor asociado a determinada tupla corresponde a 1 en caso que `M_exp/B_exp` $>$ `umbral` y a 0 en caso que `M_exp/B_exp` $\leq$ `umbral`.

(d) Funciones para tasa de falsos positivos y detección conjunto a ROC

Función `Performance` para el cálculo de α y β

La función que permite calcular la tasa de falsos positivos (α) y de detección (β), llamada `Performance`, se encuentra implementada en el bloque 6 del archivo adjunto y considera como entrada un vector con la clase real de los datos de prueba (`clase_real`) y otro con el resultado de la función `Detector` a los mismos datos de prueba (`clase_predicha`), entregando finalmente las tasas pedidas.

Siguiendo la línea de lo anterior, los parámetros de la función `Performance` corresponden a:

- `clase_real`: Arreglo de etiquetas de clase (caracteres `B` o `M`) del subconjunto de datos de prueba obtenido, es equivalente a la aplicación de `DataBase(datos)[1]["label"]`, donde `datos` corresponde al *dataset* completo posterior al preprocesamiento.
- `clase_predicha`: Arreglo de valores 0 o 1 según resulta de la aplicación de `Deteccion` sobre los datos asociados a `clase_real`.

Por otro lado el retorno de la función es `(alpha, beta)`, donde:

- `alpha`: Número real que indica la tasa de falsos positivos o nivel de significancia del test ($\alpha \in [0, 1]$) asociado al par `clase_real` y `clase_predicha`.
- `beta`: Número real que indica la tasa de detecciones ($\beta \in [0, 1]$) asociada al par `clase_real` y `clase_predicha`.

Para realizar el cálculo de los dos valores de retorno, tal como se ve en el bucle de [C. 6; l. 12], se realiza un conteo del total de datos de clase real `B` (`M_total`) y `M` (`B_total`), además de registrar la cantidad de falsos positivos (valores con clase real `B` detectados como `M`, i.e., valor 1 en `clase_predicha`) [`falsos_positivos`] y la cantidad total de detecciones correctas (valores con clase real `M` y detectados como tal) [`detecciones_correctas`]. Con lo anterior, el cálculo de α y β se realiza según lo mostrado en el código de [C. 6; l. 25] y [C. 6; l. 26] respectivamente, esto es:

$$\alpha = \texttt{falsos\_positivos}/\texttt{B\_total}$$
$$\beta = \texttt{detecciones\_correctas}/\texttt{M\_total}$$

Función para la gráfica de ROC (`PlotROC`)

La función final a implementar llamada `PlotROC`, se encuentra en el bloque 7, corresponde a aquella que a partir de básicamente un *dataset* genera una curva ROC y grafica la misma. Para realizar la gráfica se entrena con un conjunto de entrnamiento y se genera un arreglo de umbrales sobre los cuales se itera y ejecuta la función `Detector` (ver [C. 7; l. 30]), sobre los resultados de esta ejecución se utiliza la función `Performance` para obtener α y β en función de γ ($\alpha(\gamma)$ y $\beta(\gamma)$, ver [C. 7; l. 31]) con lo que se generan los vectores `alphas` y `betas` que contienen los valores de estos parámetros en función de distintos umbrales, estos vectores permiten graficar la curva ROC.

Respecto al arreglo de umbrales, este contiene a los mismos logarítmicamente espaciados, es decir $\log(\gamma_{i+1}) - \log(\gamma_i)$ para todo i válido según el tamaño del arreglo. El motivo de esto corresponde a lo explicado en la parte (c) de implementación de la función de detección, donde γ varía significativamente entre órdenes distintos órdenes de magnitud, superando incluso los límites de representación de punto flotante, para generar el arreglo, se utilizan valores límites mínimo y máximo (parámetros `min_val` y `max_val` respectivamente) así como una cantidad de pasos entre los extremos (parámetro `steps`) según se ve en [C. 7; l. 23].

La función `PlotROC` tiene una mayor cantidad de parámetros que las otras funciones, estos son:

- `dataset`: Único que **no** tiene valor por defecto, corresponde al arreglo de *numpy* con tipo de dato `dtype` = `labeled_dtype` que contiene el *dataset* posterior al preprocesamiento inicial.
- `min_val`: Valor numérico que representa el límite inferior del arreglo de umbrales, valor por defecto: `10**(-20)` (número 10^{-20}).
- `max_val`: Valor numérico que representa el límite superior del arreglo de umbrales, valor por defecto: `10**200` (número 10^{200}).
- `steps`: Cantidad de pasos en el arreglo de umbrales, es decir, corresponde a la cantidad total de valores distintos con los que se itera el cálculo de `Performance`, a mayor valor, mayor detalle se obtiene en la curva ROC, valor por defecto: `10000`.
- `margen`: Parámetro visual del gráfico resultante, corresponde al margen numérico que se le da a los límites de los ejes de la gráfica al momento de visualizar la misma; este implica que tanto en el eje x como en y, los límites son $0 -$ `margen` y $1 +$ `margen` (ver código [C. 7; l. 90]), valor por defecto: `0.05`.
- `significancia`: Parámetros que en caso de no ser `None` es numérico e indica superponer una cruz roja al gráfico que indica el punto de operación para el nivel de significancia ingresado, es decir, el valor de α que mejor se aproxime dentro de los computados a `significancia` y β asociado [$\beta(\alpha)$], valor por defecto `None`.
- `get_info`: Parámetro booleano que indica si se requiere mostrar en consola valores útiles para calcular los costos de un nivel de significancia en particular según el desarrollo de la parte teórica (problema 2), estos son $P(\omega_0)$, $P(\omega_1)$ (calculados en base a conteo), junto a $|\Sigma_0|$ y $|\Sigma_1|$. En caso que `significancia` no sea `None`, también muestra en consola los valores de α, β y γ asociados a la significancia ingresada. Valor por defecto: `False`.
- `save`: Booleano que indica a la función si guardar el gráfico o no, este es guardado con el nombre `name` (descrito a continuación), en formato `.svg`. Valor por defecto: `False`.

- `name`: Parámetro que en caso de no ser `None` es un *string* que indica el nombre del fichero con que se guarda el gráfico (en caso que `save=True`). Valor por defecto: `None`.
- `train_per`: Valor numérico que indica el porcentaje de datos que se toman como datos de entrenamiento, el resto de datos se toma como prueba. Tal como se ve en [C. 7; l. 17], la partición de `dataset` se hace con `DataBase(dataset, train_per, 100-train_per)`. Valor por defecto: `60`.

Es importante notar que esta función no tiene retorno, sino que muestra el gráfico, y en caso de ser requerido, hace un *print* de valores en consola.

La figura 6 muestra la curva ROC generada (curva azul) al usar `train_per=60`.

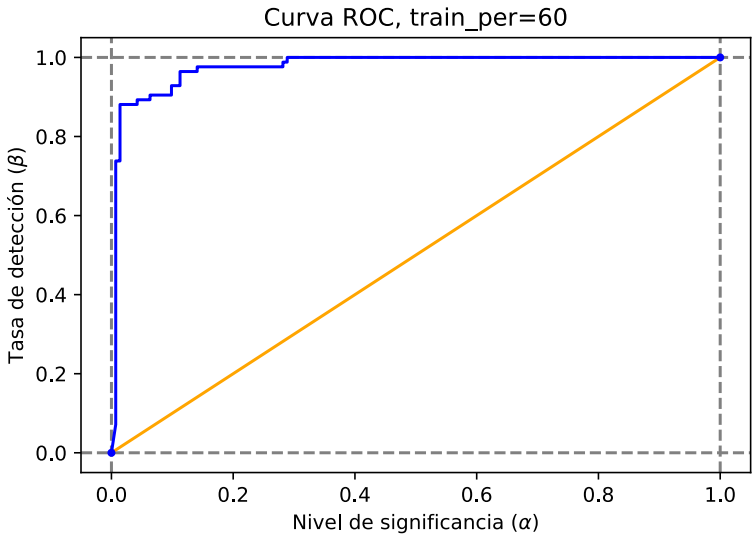

Figura 6: Curva ROC para un porcentaje de entrenamiento de 60 %

Como es posible notar, la curva ROC es bastante cercana a una ideal, por lo que el desempeño del clasificador se considera muy satisfactorio.

Respuestas a interrogantes planteadas

(1) Se pide responder si es que se observan cambios al modificar el porcentaje de datos usados en entrenamiento y test.

Para responder esto, se generan las curvas ROC considerando parámetros `train_per` igual a `10`, `20`, `40` y `80`, estas se encuentran en la figura 7.

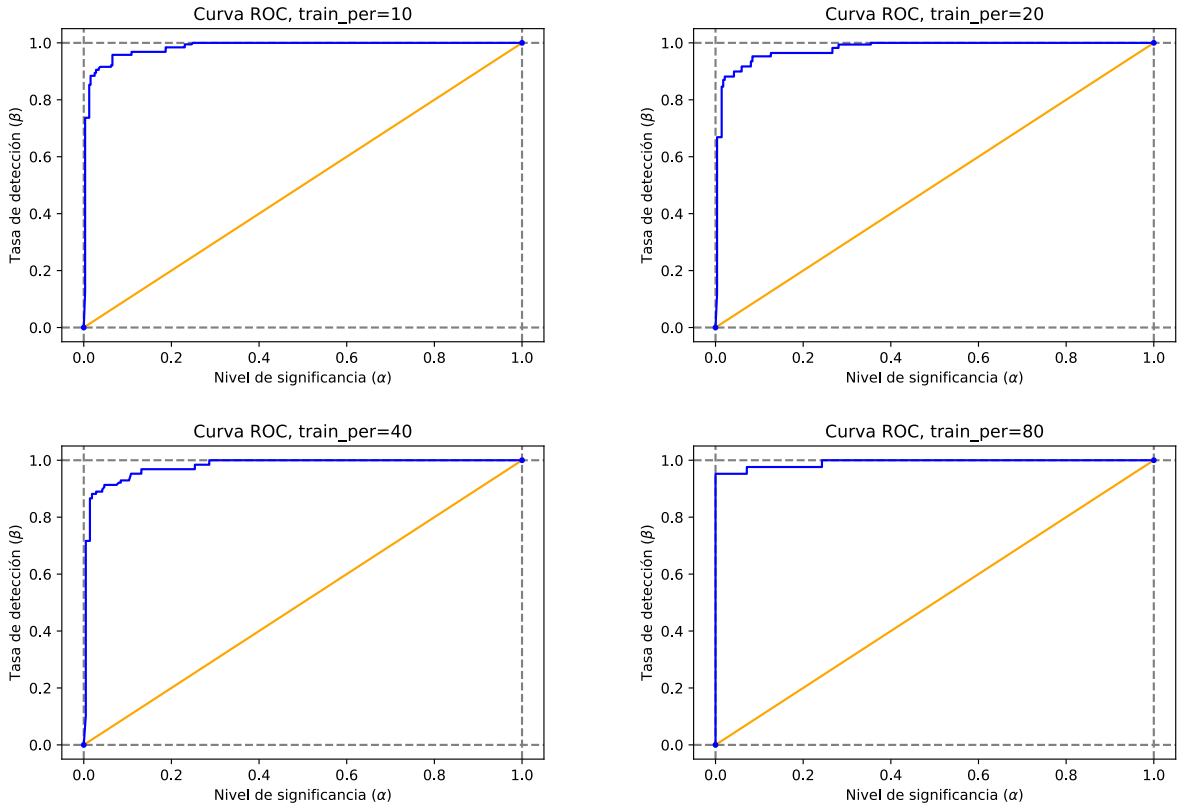

Figura 7: Curvas ROC ante distintos parámetros de `train_per`

Como es posible notar, las curvas no varían significativamente, a excepción del caso donde `train_per=80`, pero esto se explica por una limitada cantidad de datos de *testing* que no permite obtener pares $(\alpha(\gamma), \beta(\gamma))$, mas que por variaciones en el desempeño del modelo. Notar que incluso en el caso donde se trata con un 10 % de datos de entrenamiento y 90 % de datos de prueba la curva tiene una forma muy deseable, esto da a entender que la implementación del detector es bastante robusta.

(2) Se pide responder cuántos parámetros se necesita para describir el detector implementado.

La respuesta a esta preguntas se puede ver en los parámetros de la función `Detector` [parte (c)], sin considerar los datos de prueba o entrada, estos corresponden a un umbral y a las salidas de la función `Parametros`, así, se puede decir que para describir completamente el detector, es necesario conocer:

- El vector $(\theta_{0_1}, \theta_{0_2}, \ldots, \theta_{0_{m-1}}, \theta_{0_m})$, donde θ_{0_j} corresponde al estimador de la media (promedio) para los valores del atributo j-ésimo para datos de clase `B`.
- El vector $(\theta_{1_1}, \theta_{1_2}, \ldots, \theta_{1_{m-1}}, \theta_{1_m})$, donde θ_{1_j} corresponde al estimador de la media (promedio) para los valores del atributo j-ésimo para datos de clase `M`.
- El vector $(\sigma_{0_1}, \sigma_{0_2}, \ldots, \sigma_{0_{m-1}}, \sigma_{0_m})$, donde σ_{0_j} corresponde a la raíz del estimador insesgado de la varianza para los valores del atributo j-ésimo para los datos de clase `M` (este permite describir Σ_0).
- El vector $(\sigma_{1_1}, \sigma_{1_2}, \ldots, \sigma_{1_{m-1}}, \sigma_{1_m})$, donde σ_{1_j} corresponde a la raíz del estimador insesgado de la varianza para los valores del atributo j-ésimo para los datos de clase `B` (este permite describir Σ_1).
- El valor umbral γ, este es el único que no depende de los datos de entrenamiento, sin embargo, los valores $\alpha(\gamma)$ y $\beta(\gamma)$ si dependen de estos.

(3) Se pide utilizar la curva ROC para terminar un punto de operación óptimo para el detector, además de explicitar cuál es el costo asociado a un proceder incorrecto en el detector.

Considerando la naturaleza del problema con el que se trabaja (detección de células malignas), se entiende que el costo de una falla de detección es significativamente mayor que el de un falso positivo (que no necesariamente tiene bajo costo), por lo mismo se busca un punto que tenga β alto dentro de un valor de α tolerable, así, es como se escoge $\alpha = 15\,\%$, así, en la figura 8 se puede ver el resultado de ejecutar:

```
PlotROC(datos,significancia=0.15,get_info=True,save=True,name='ROC_sig.svg')
```

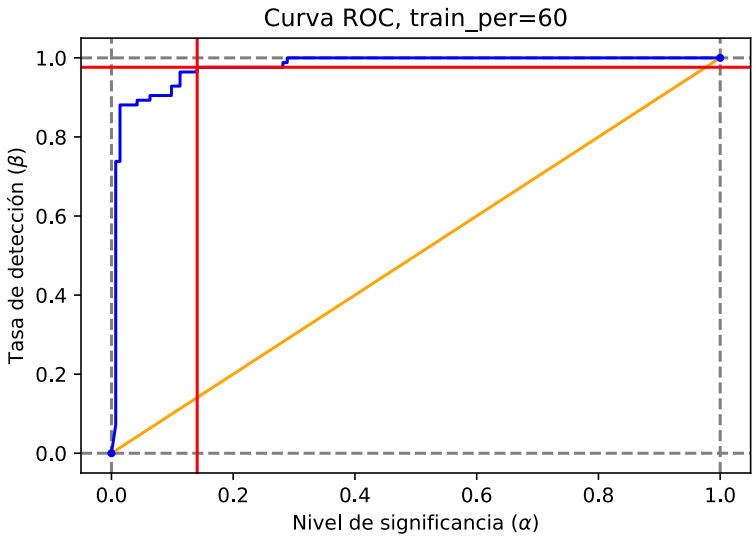

Figura 8: Curva ROC con cruz marcando el punto de operación asociado a $\alpha = 15\,\%$

El texto impreso por la función corresponde a:

```
Valor de alpha significativo: 0.14084507042253522
Valor de beta significativo: 0.9761904761904762
Valor gamma asociado: 0.008124394043810454
Probabilidad de la clase B: 0.6264705882352941
Probabilidad de la clase M: 0.3735294117647059
Determinante de Sigma_0: 3.1272987769520877e-09
Determinante de Sigma_1: 7.467420525756814e-09
```

Así, siguiendo el desarrollo explicitado en la parte teórica, se puede ver que:

$$\frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}}\frac{L_{01}P(\omega_0)}{L_{10}P(\omega_1)} = \gamma \Leftrightarrow L_{10} = L_{01}\cdot\frac{1}{\gamma}\frac{|\Sigma_0|^{1/2}P(\omega_0)}{|\Sigma_1|^{1/2}P(\omega_1)}$$

Donde reemplazando valores numéricos, se tiene que:

$$\left.\begin{aligned}\gamma &\approx 8{,}1244\cdot 10^{-3}\\ P(\omega_0) &\approx 0{,}62647\\ P(\omega_1) &\approx 0{,}37353\\ |\Sigma_0| &\approx 3{,}1273\cdot 10^{-9}\\ |\Sigma_1| &\approx 7{,}4674\cdot 10^{-9}\end{aligned}\right\} \Rightarrow \frac{1}{\gamma}\frac{|\Sigma_0|^{1/2}P(\omega_0)}{|\Sigma_1|^{1/2}P(\omega_1)} \approx 133{,}59 \Rightarrow L_{10} \approx 134\cdot L_{01}$$

Lo anterior permite explicitar que al considerar que el costo de una falla de detección es 134 veces mayor que el costo de un falso positivo, entonces se opera el detector con una tasa de falsos positivos de 14,1 % y una tasa de detección del 97,6 %.