CAPÍTULO 3: FASE DE ANÁLISIS

En este capítulo analizaremos por separado las dos posibilidades con que nos encontramos en un sistema basado en *fingerprints* que pueden provocar una detección errónea.

En el primer apartado se hará un análisis de falsa alarma, es decir, se analizará la probabilidad de que se produzca una detección de anuncio sin que debiera producirse porque la señal de entrada al sistema no se corresponde con ningún anuncio de nuestra base de datos. La llamaremos probabilidad de falsa alarma y la denotaremos por P_f.

En el segundo apartado analizaremos la probabilidad de que no se produzca una detección de anuncio cuando sí debería producirse. La llamaremos probabilidad de no alarma y la denotaremos por P_n.

Ya que ambas probabilidades dependerán del umbral escogido para considerar detección o no en función de la distancia de Hamming entre el *fingerprint* calculado en tiempo real y los que tenemos en la base de datos, en el tercer apartado de este capítulo justificaremos el umbral escogido.

3.1. ANÁLISIS DE FALSA ALARMA

En este apartado vamos a estudiar la probabilidad de falsa alarma (denotada por P_f), es decir, la probabilidad de que se produzca una detección de anuncio sin que debiera producirse. Esta probabilidad se dejará en función del umbral escogido, para posteriormente en el tercer apartado llegar a una decisión de compromiso entre la probabilidad de falsa alarma y la probabilidad de no alarma. Para este análisis, nos basamos en el artículo **"A Highly Robust Audio Fingerprinting System"** *(apartado 4.3, titulado False Positive Analisis)*.

Consideramos que dos trozos de 3 segundos extraídos de una señal de audio son similares o equivalentes si la distancia de Hamming (es decir, el numero de bits diferentes) entre los dos fingerprints correspondientes a cada uno de los trozos es inferior a un cierto umbral T. El valor de este umbral T determina directamente la probabilidad de falsa alarma o falsa detección (P_f), es decir, la tasa con la que dos señales de audio son consideradas iguales incorrectamente. Cuanto más pequeño es el valor de T, más baja será la probabilidad de que suceda una falsa alarma. Pero, por otra parte, un valor pequeño de T afectará negativamente a la probabilidad de no alarma cuando sí debería haber alarma (P_n), es decir, la probabilidad de que dos señales sean iguales, pero no sean identificadas como tal.

Analicemos la elección de este umbral de decisión T desde el punto de vista de falsa alarma. Para ello, asumimos que el proceso de extracción de fingerprints da como resultado un serie de bits aleatorios independientes e idénticamente distribuidos. El numero de bits erróneos tendrá pues una distribución binomial (n,p), donde n es el número de bits extraídos (n=229*32=7328 en 3 segundos) y p es la probabilidad de que un bit sea un "0" o un "1" (p=0.5).

$$P(x) = \binom{n}{x} p^x p^{n-x},$$

donde *x* es el número de bits erróneos, que puede tomar los valores 0,1,... *n*

Por tanto, la probabilidad de falsa alarma en función del umbral T escogido será:

$$P_f(T) = \sum_{x=T}^{n} P(x)$$

Puesto que n toma valores muy elevados en nuestra aplicación, la distribución binomial podemos aproximarla por una distribución normal con media μ=np y desviación típica de $\sigma = \sqrt{(np(1-p))}$.
Por tanto, dado un *fingerprints* correspondiente a un trozo de 3 segundos de audio F_1, la probabilidad de que otro *fingerprints* correspondiente a otro trozo de 3 segundos escogido al azar F_2 sea diferente en un número de bits inferior al umbral T=α·n, viene dada por la siguiente expresión:

$$P_f(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{(1-2\alpha)\sqrt{n}}^{\infty} e^{-\frac{x^2}{2}} dx = \frac{1}{2} erfc\left(\frac{(1-2\alpha)}{\sqrt{2}}\sqrt{n}\right)$$

donde α denota la tasa de error de bit (BER).

Sin embargo, en la práctica los sub-fingerprints tienen una alta correlación entre ellos a lo largo del tiempo. La existencia de esta alta correlación no sólo es debida a la inherente correlación que tiene una señal de audio en el tiempo, sino también al "overlap" tan alto utilizado en el proceso de extracción de los fingerprints. Cuanto más alta es la correlación, más grande es la desviación típica, tal y como mostramos con el siguiente argumento.

Asumimos una fuente binaria simétrica con un alfabeto {-1,1} tal que la probabilidad de que un símbolo x_i y un símbolo x_{i+1} sean iguales es igual a q. Entonces, tenemos que

$$E[x_i x_{i+k}] = a^{|k|},$$

donde $a = 2q - 1$. Si una fuente Z es la or-exclusiva de dos secuencias X e Y, entonces Z es simétrica y

$$E[z_i z_{i+k}] = a^{2|k|}.$$

Para un valor de N largo, la desviación estándar de la media $\overline{Z}_N$ sobre N muestras consecutivas de Z puede ser descrita aproximadamente por una distribución normal de media 0 y desviación típica igual a

$$\sqrt{\frac{1+a^2}{N(1-a^2)}}$$

Trasladando esto al caso de los bits de los *fingerprints*, un factor de correlación a entre dos secuencias de bits de *fingerprints* implica un incremento en la desviación estándar de la BER por un factor

$$\sqrt{\frac{1+a^2}{1-a^2}}$$

Para determinar la distribución de la BER con *fingerprints* reales, generamos una base de datos de 600 *fingerprints* escogidos aleatoriamente procedentes de 95 minutos de grabación de una emisora de radio. Para ello realizamos el código de Matlab "**probabilidad_eleccion_umbral.m"** (véase Anexo 2.5).

Este código se compone de dos partes claramente diferenciadas. En la primera parte se escogen *fingerprints* independientes procedentes de dos archivos de audio diferentes: 45_11_01_8000_16.wav *(archivo de audio de duración 64 minutos y 52 segundos)* y 46_11_01_8000_16.wav *(archivo de audio de duración 30 minutos y 20 segundos)*. Para véase el significado del nombre del archivo de audio se puede consultar el Anexo 1.

El hecho de separar la grabación en dos archivos es sólo para evitar problemas de memoria motivados por el excesivo tamaño que tendría un solo archivo de 95 minutos y que provocaría una parada en la ejecución del código. No obstante, antes de ejecutar las segunda parte del código, se agrupan todos los *fingerprints* procedentes de extracciones sobre ambos ficheros de audio en una sola variable matriz llamada en nuestro código "finger". Esta matriz contiene por cada fila todos los bits correspondientes a un *fingerprints*; por tanto, el tamaño de esta variable es de 600x7328.

Para hacer esta reagrupación de una matriz de 29x32, que contiene un *fingerprint*, en otra matriz de 1x7328, que contiene en una sola fila los bits de un *fingerprints*, para su posterior almacenamiento en la variable "finger", se utiliza la función de Matlab siguiente:

```
finger(k,:)=reshape(fing',1,229*32);
```

siendo "fing" la variable que contiene un *fingerprint* procedente de 3 segundos de audio.

Por su parte, para asegurar que vamos a tener 600 *fingerprints* independientes unos de otro, lo que hacemos es coger un trozo de audio de 3 segundos del archivo de audio de forma aleatoria y, a continuación, eliminamos los 3 segundos previos a ese trozo escogido, los 3 segundos pertenecientes al trozo escogido y los 3 segundos posteriores a dicho trozo. Así pues eliminamos 9 segundos del archivo de audio cada vez que calculamos un nuevo *fingerprint* asegurándonos que el siguiente que calculemos será totalmente independiente de los anteriormente calculados.

Una vez que con la primera parte del código hemos creado una base de datos de 600 *fingerprints* independientes entre sí, pasamos a ejecutar la segunda parte del código. En esta parte, a partir de dicha base de datos, se determina la BER entre todas las posibles parejas de *fingerprints* ($C_{600}^{2} = \binom{600}{2} = 179700$ parejas). Para este cálculo de la BER comparamos cada una de las filas de la variable "finger" con el resto de filas y vamos guardando en una variable el número de bits en que se diferencian.

Una vez terminada la ejecución de la segunda parte de este código, tenemos una variable que resume el número de veces que se ha producido una distancia de Hamming de *x* bits, siendo $x \in [0,7328]$. Para estudiar la desviación estándar de la BER a partir de esta variable, acudimos al código de Matlab **"estad.m"** (véase Anexo 2.6). Como resultado obtenemos que la desviación estándar de la distribución de la BER resultante es 0.0157, aproximadamente 3 veces más grande que los 0.0055 que uno esperaría para una secuencia de bits independientes e idénticamente distribuidos.

Para incorporar a la fórmula este hecho de que la desviación estándar de la BER es mayor, modificamos la fórmula de la probabilidad de falsa alarma incluyendo un factor de 3:

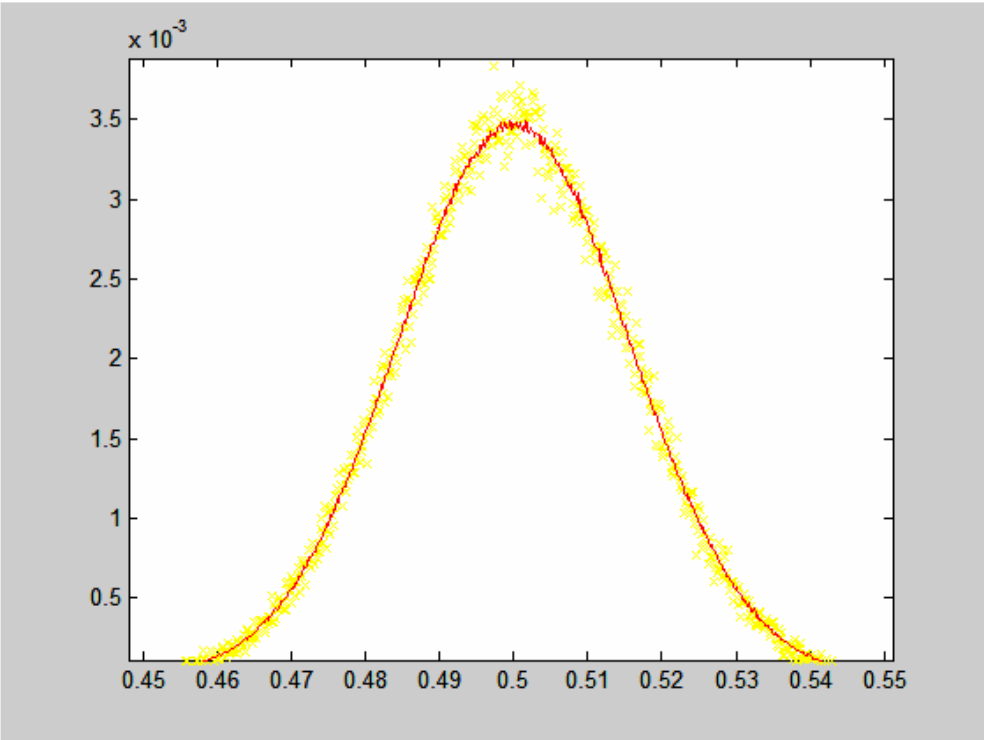
$$P_f = \frac{1}{2} erfc\left(\frac{(1-2a)}{3\sqrt{2}} \sqrt{n} \right)$$

A partir de la variable comentada anteriormente podemos hacer una gráfica con la función densidad de probabilidad (PDF) de la distribución de la BER medida. Para ello basta con ejecutar la siguiente línea del código:

```
plot((1:229*32)/229/32,n/93961,'yx');
```

siendo "n" la variable que contiene el número de veces que se ha dado una cierta distancia de Hamming entre dos *fingerprints*.
Con el segundo argumento de la función de Matlab **plot** indicamos que la gráfica de la BER medida experimentalmente la dibuje en amarillo y los puntos los represente con una cruz.

Sobre la misma figura dibujamos la función densidad de probabilidad (PDF) de una distribución normal con media 0.5 y desviación estándar 0.0157; para esto ejecutamos la función **plot_dist_normal**, implementada en el archivo **"plot_dist_normal.m"** (véase Anexo 2.9), obteniendo así la gráfica 1:



Gráfica 1. Comparación entre las funciones densidad de probabilidad de una distribución normal *N*(0.5,0.0157) y la BER obtenida experimentalmente.

Podemos observar que la PDF de la BER se aproxima mucho a la distribución normal.

Por tanto, podemos concluir de este análisis que el cálculo de la probabilidad de falsa alarma (P_f) lo podemos realizar aproximándolo por una función densidad de probabilidad de una distribución normal *N*(0.5,0.0157).

Pongamos un ejemplo práctico: si establecemos un valor para umbral de decisión tal que $\alpha = 0.35$ (lo cual significaría que de 7.328 bits, debe haber menos de 2564 bits diferentes entre los dos *fingerprints* para que se decida que pertenecen al mismo anuncio), haciendo uso de la última fórmula llegamos a que la probabilidad de falsa alarma que podemos esperar toma un valor muy bajo:

$$P_f = \frac{1}{2} erfc\left(\frac{(1-2\cdot 0.35)}{3\sqrt{2}}\sqrt{7328}\right) = \frac{erfc(6.4)}{2} = 3.6\cdot 10^{-20}$$

3.2. APROXIMACIÓN POR DISTRIBUCIÓN DE PROBABILIDAD DE WEIBULL

En el apartado anterior, "Análisis de falsa alarma", justificamos teóricamente que la distribución de probabilidad que sigue la tasa de error de bits entre dos *fingerprints* procedentes de trozos de audio diferentes se ajusta muy bien por a una distribución normal. En este apartado vamos a estudiar a que distribución se aproxima más la tasa de error de bits entre dos *fingerprints* que proceden del mismo anuncio, y que, por lo tanto, deberían dar lugar a una alarma.

Para este análisis realizamos los códigos de Matlab siguientes: **"prueba_eleccion_umbral_pdf.m"** (véase Anexo 2.7) y **"plot_distribuciones"** (véase Anexo 2.8).

En el primer código trabajamos con 12 señales diferentes. Cuatro de las cuales son el mismo anuncio grabado en diferentes momentos en la misma emisora:

22_23_01_8000_8
22_23_02_8000_8
22_23_03_8000_8
22_23_04_8000_8

Por su parte, las otras ocho señales son otros ocho anuncios cualesquiera escogidos de la base de datos que contiene los anuncios. Hemos escogido los siguientes:

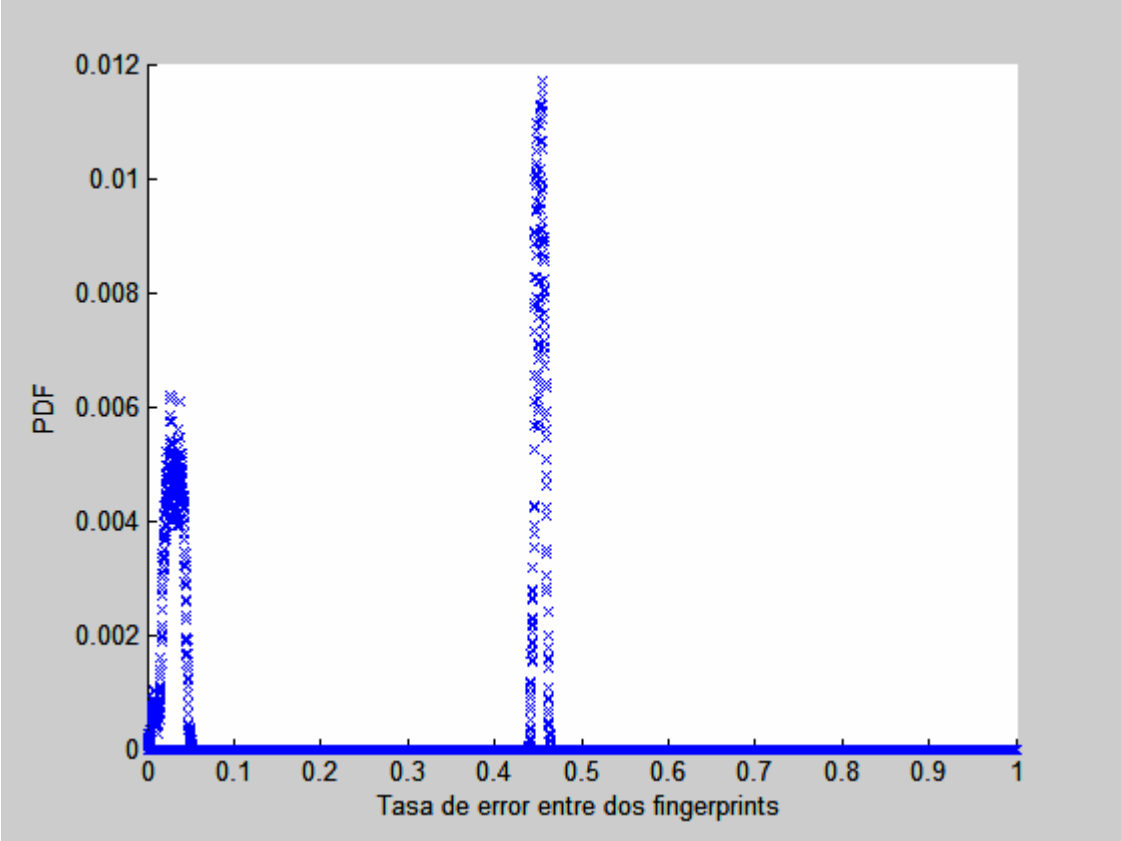
01_07_01_8000_8
08_19_01_8000_8
05_19_01_8000_8
05_23_01_8000_8
15_19_01_8000_8
16_19_01_8000_8
17_19_01_8000_8
18_23_01_8000_8

Se puede consultar de dónde y cómo se realizaron estas grabaciones de anuncios consultando en el Anexo 1 el significado de cada uno de los apartados que componen el nombre de un archivo de audio.

Lo que hacemos es repetir el mismo proceso con cada una de las 12 señales. El proceso consiste en coger 10000 trozos consecutivos de tres segundos (24000 muestras). Para cada trozo, se calcula su *fingerprint* y se halla la mínima distancia entre este y los *fingerprints* correspondientes al anuncio

22_23_01_8000_8 (el cual consideramos como anuncio de referencia). Una vez tenemos estas mínimas distancias calculadas, las guardamos sobre una variable global a todas las iteraciones del proceso que recoge el número de veces que se repite una determinada distancia entre dos *fingerprints.* Esta variable, una vez realizado el proceso con las 12 señales, nos permite ver la función densidad de probabilidad de la tasa de error de bits, es decir, el porcentaje de bits diferentes que hay entre un *fingerprints* calculado y el *fingerprints* perteneciente al anuncio de referencia más próximo en distancia.

El resultado de este código es la función densidad de probabilidad que vemos en la gráfica 2.



Gráfica 2. PDF resultado del código *prueba_eleccion_umbral_pdf.m*

En esta gráfica, primeramente podemos verificar lo que ya justificamos teóricamente con anterioridad sobre la función de distribución de probabilidad de la tasa de error de bits en el caso de falsa alarma. Es decir, vemos como la función de distribución que aparece en la parte central de la gráfica se puede aproximar por una distribución normal.

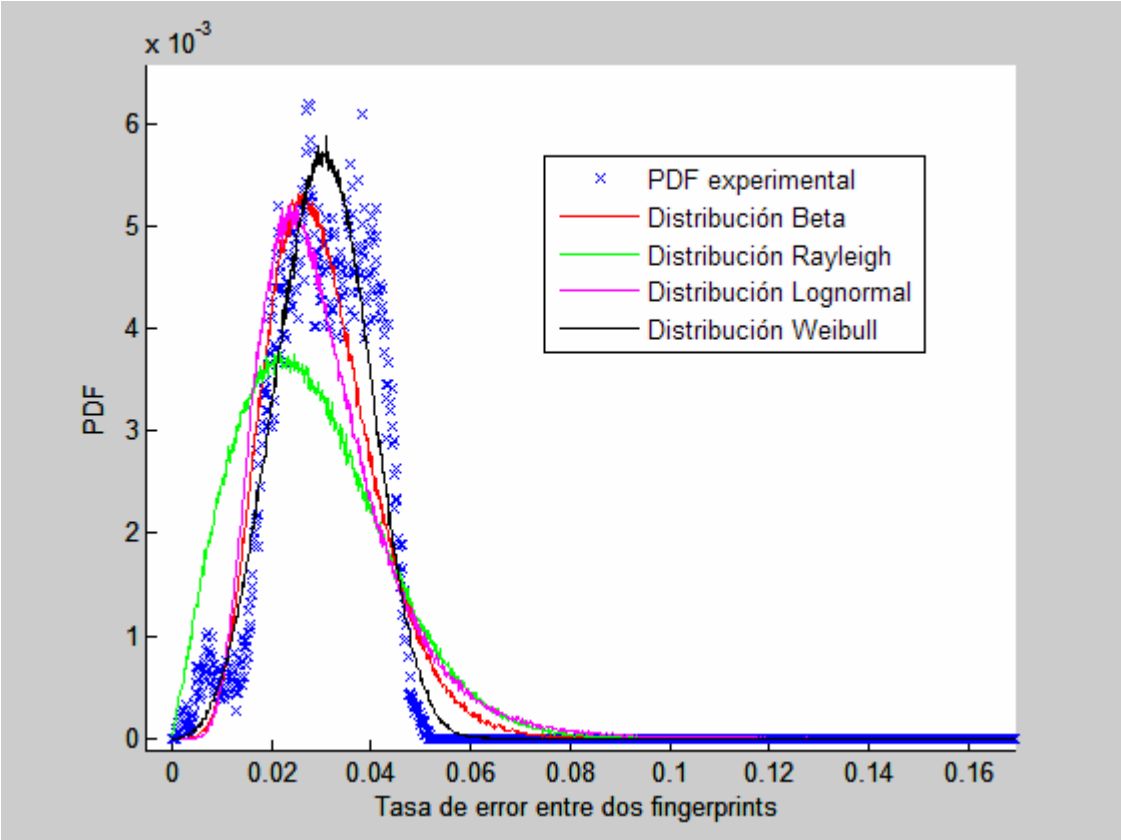
En segundo lugar, vemos claramente como un umbral de decisión de, por ejemplo, 0.35 nos permite detectar perfectamente cuándo un *fingerprints* procedente de la señal de entrada al sistema debe producir una alarma o no.

Centrándonos en la parte de la PDF correspondiente a una detección correcta (parte izquierda de la gráfica), acudimos al código de Matlab "***plot_distribuciones***" (véase Anexo 2.8).

Este código calcula, a partir de un vector de entrada que recoge las distancias de Hamming correspondientes a detecciones correctas (resultado del código "***prueba_eleccion_umbral_pdf.m***"), los parámetros que mejor aproximan la PDF correspondiente a este vector con las siguientes funciones de distribución de probabilidad:

- función de distribución de probabilidad Beta
- función de distribución de probabilidad Lognormal
- función de distribución de probabilidad Rayleigh
- función de distribución de probabilidad Weibull

Una vez obtenidos estos parámetros, pinta sobre una misma figura todas las PDF´s (incluida la correspondiente a detecciones correctas obtenida experimentalmente con anterioridad). El resultado se puede observar en la gráfica 3.



Gráfica 3. Distribuciones de probabilidad frente a la función densidad de probabilidad obtenida empíricamente

Para hacer la representación gráfica de todas estas funciones de distribución, una vez calculados los parámetros, se utilizan las siguientes sub-funciones:

- plot_dist_beta.m (véase Anexo 2.12)
- plot_dist_rayleigh.m (véase Anexo 2.13)
- plot_dist_lognormal.m (véase Anexo 2.14)
- plot_dist_weibull.m (véase Anexo 2.15)

Cada una de estas sub-funciones pinta una de las funciones de distribución; para ello, se les pasa como valores de entrada aquellos parámetros que mejor aproximan cada una de estas funciones de distribución de probabilidad a la PDF obtenida experimentalmente.

Observando la gráfica 3, vemos como la distribución de Weibull es la que más se aproxima a la función de distribución de probabilidad correspondiente a los casos en que el algoritmo de detección debería producir una alarma, es decir, una detección correcta.

Por tanto, podemos concluir de este análisis que el cálculo de la probabilidad de no producirse alarma cuando sí se debería producir (P_n) lo podemos realizar aproximándolo por una función densidad de probabilidad de una distribución de Weibull. Esta probabilidad dependerá directamente del umbral de decisión que escojamos, al igual que ocurría con la probabilidad de falsa alarma.

3.3. CONCLUSIÓN DE LOS ANÁLISIS

Del primer análisis, apartado "Analisis de falsa alarma", concluimos que la probabilidad de detección errónea (P_f) se puede calcular mediante la función densidad de probabilidad de una distribución normal cuyos parámetros (media y desviación típica) obtenemos experimentalmente.

Por su parte, del segundo análisis, apartado "Aproximación por distribución de probabilidad de Weibull", concluimos que la probabilidad de no detección cuando sí debería producirse (Pn) se puede calcular mediante la función densidad de probabilidad de una distribución de Weibull, cuyos parámetros se calculan también experimentalmente.

En este apartado vamos a establecer como umbral de decisión T, aquel valor que haga que el área por debajo de la cola de la función densidad de probabilidad de la distribución normal y el área por debajo de la cola de la función densidad de probabilidad de la distribución de Weibull sean iguales.

Para ver el cómputo de este valor realizamos el código de Matlab **"valor_umbral.m"** (véase Anexo 2.10). En este código trabajamos con la variable resultado de ejecutar el código **"prueba_eleccion_umbral_pdf.m"**. A partir de esta variable, obtenemos dos variables; una contiene los valores relativos al cálculo de la P_f y la otra los valores relativos al cálculo de la P_n. Con estas dos variables calculamos los parámetros estadísticos necesarios para las funciones densidad de probabilidad de la distribución normal y de la de Weibull que mejor se aproximan a nuestras funciones densidad de probabilidad experimentales.

Previo al cálculo de estos parámetros estadísticos, hay que convertir la información que contienen las dos variables con las que vamos a trabajar en el código. Cada una de las dos variables las tenemos que transformar en otras dos variables que contengan los valores de distancia de Hamming repetidos tantas veces como indica la variable original. Con estas dos nuevas variables, sí se pueden calcular los parámetros estadísticos necesarios gracias a las funciones de Matlab siguientes:

```
[a,b]=wblfit(vect1);
[c,d]=normfit(vect2);
```

Para hacer esta transformación de la información utilizamos la función **transforma_formato**, implementada en el archivo **"transforma_formato.m"** (véase Anexo 2.11)

A partir de las funciones densidad de probabilidad de las distribuciones normal y de Weibull calculamos los valores que toman P_f y P_n en función del valor de α. Mostramos los resultados en la siguiente tabla:

α	P_n	P_f
0.030	5.1007e-001	0
0.034	3.4666e-001	0
0.038	2.0492e-001	0
0.042	1.0251e-001	0
0.046	4.2129e-002	0
0.050	1.3789e-002	0
0.054	3.4777e-003	0
0.058	6.5286e-004	0
0.062	8.7971e-005	0
0.066	8.1925e-006	0
0.070	5.0690e-007	0
0.074	2.0002e-008	0
0.078	4.8243e-010	0
0.082	6.8069e-012	0
0.086	5.3735e-014	0
0.090	2.2204e-016	0
0.091	0	0
0.092	0	0
…	…	…

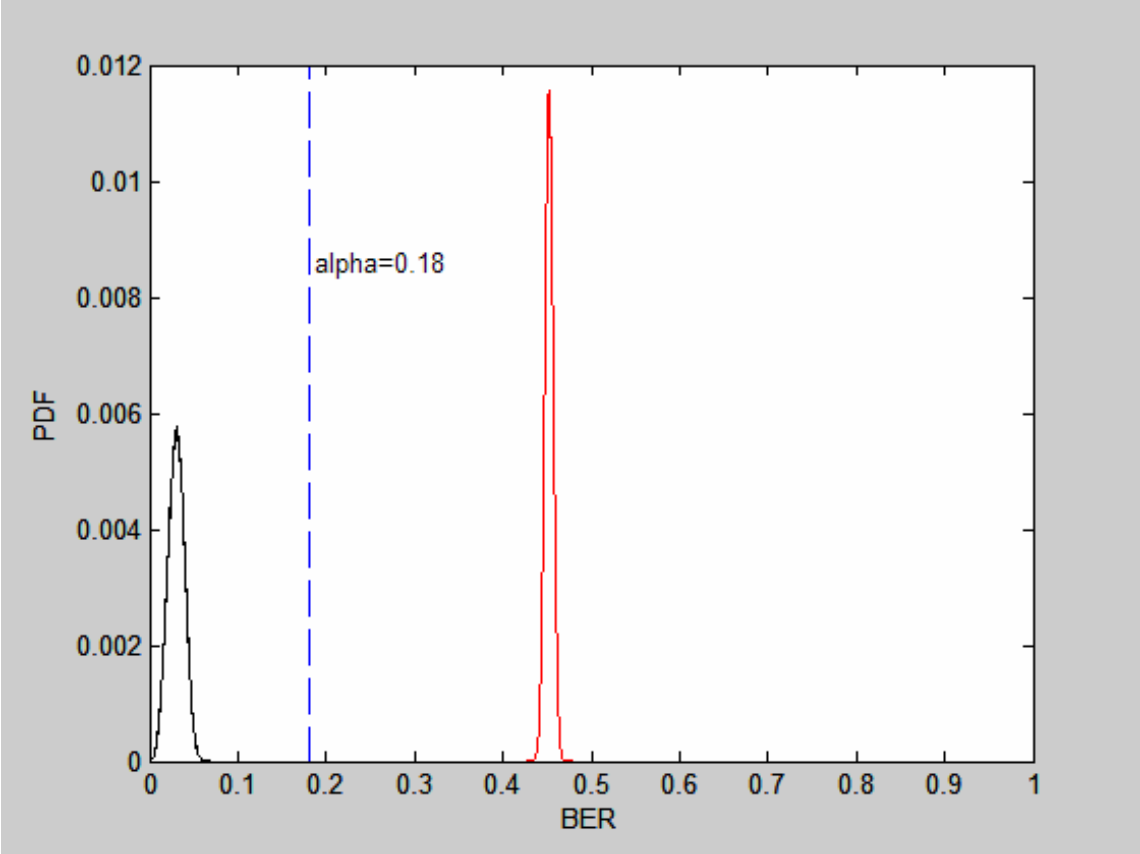
α	P_n	P_f
…	…	…
0.270	0	0
0.274	0	0
0.278	0	3.7506e-298
0.282	0	9.9718e-285
0.286	0	1.2943e-271
0.290	0	8.2017e-259
0.294	0	2.5374e-246
0.298	0	3.8328e-234
0.320	0	2.8267e-222
0.324	0	1.0179e-210
0328	0	1.7899e-199
0.342	0	1.5369e-188
0.346	0	6.4442e-178
0.350	0	1.3196e-167
0.354	0	1.3197e-157
0.358	0	6.4460e-148
0.362	0	1.5379e-138
0.366	0	1.7923e-129

Tabla 1. Probabilidad de falsa alarma y probabilidad de no alarma en función de alpha

Como podemos observar en los resultados de la tabla 1, la cola de la probabilidad de falsa alarma y la de la probabilidad de no alarma no se cortan. Esto implica que no podemos obtener ningún valor de α que iguale las áreas por debajo de las colas. Por tanto vamos a tomar como α el valor medio entre el valor de α a partir del cual P_n vale 0 y el valor de α a partir del cual P_f vale 0.

$$\left.\begin{aligned}\alpha_1 &= 0.091\\ \alpha_2 &= 0.274\end{aligned}\right\} \rightarrow \alpha = \alpha_1 + \frac{\alpha_2 - \alpha_1}{2} = 0.1825 \approx 0.18$$

En la gráfica 4 podemos ver las funciones densidad de probabilidad de las distribuciones normal y de Weibull y el umbral correspondiente a un $\alpha = 0.18$.



Gráfica 4. Funciones densidad de probabilidad de las distribuciones normal y de Weibull y un umbral correspondiente a $\alpha = 0.18$.

En conclusión de los dos análisis, vamos a considerar que dos trozos de 3 segundos extraídos de una señal de audio son similares o equivalentes si la distancia de Hamming (es decir, el numero de bits diferentes) entre los dos fingerprints correspondientes a cada uno de los trozos es inferior a un cierto umbral T igual a

$$T = \alpha \cdot n = 0.18 \cdot 7328 \approx 1319$$

Es decir, de los 7.328 bits que se comparan entre dos *fingerprints*, debe de haber menos de 1319 bits diferentes entre los dos para que se decida que pertenecen al mismo anuncio y, por lo tanto, se produzca una detección.

Con este valor de T, obtenemos las siguientes probabilidades de error:

- Probabilidad de falsa alarma: **P_f=0**
- Probabilidad de no alarma: **P_n=0**