Trabajo Práctico 0: El algoritmo de umbralización de Kittler

Ph. D. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Computación
PAttern Recongition and MAchine Learning Group (PARMA-Group)

4 de abril 2024

Fecha de entrega: Domingo 21 de Abril.
Entrega: Un archivo .zip con el código fuente LaTeX o Lyx, el pdf, y un script en jupyter, debidamente documentado. El script en jupyter debe estar escrito usando pytorch. A través del TEC-digital.
Modo de trabajo: Grupos de 3 personas.

En el presente trabajo práctico se introducirá el problema de la clasificación a través del estudio del concepto de máxima verosimilitud. Se realizará un repaso de la teoría básica relacionada con los fenómenos aleatorios con distribución Gaussiana, para facilitar el análisis de la función de verosimilitud. Posteriormente se visitará el problema de la segmentación de imágenes desde un enfoque de máxima verosimilitud, donde se desarrollará el algoritmo de Kittler [**?**]. El estudiante implementará tal algoritmo y analizará los resultados respecto a los planteos teóricos introducidos previamente.

Trabajo realizado por:

- Marco Ferraro
- Jorge Monge

1. Implementación del algoritmo de Kittler

1. Implemente el algoritmo de Kittler, y realice una prueba con la imagen de entrada provista, aplicando posteriormente el umbral óptimo obtenido.

   a) **(20 puntos)** Implemente una funcion *calcular_momentos_estadisticos(T, p)* la cual reciba un umbral candidato T y una funcion de densidad p, y retorne todos los parametros de la función. Comente su implementacion con detalle en este informe.

   Respuesta:

Para la implementación de esta función, se segmento en varias subfunciones.

Las funciones calculate_p1 y calculate_p2 están diseñadas para calcular las probabilidades acumuladas a la izquierda y a la derecha de un umbral especificado en una distribución de probabilidad, respectivamente.

Ambas funciones toman tres parámetros: x, que representa un array de valores de la variable aleatoria, pdf, que es un array de la función de densidad de probabilidad correspondiente a los valores en x, y T, que es el umbral utilizado para acotar la función de densidad de probabilidad.

En caso de que el umbral T sea mayor que todos los valores en x, las funciones ajustarán automáticamente el índice para garantizar un cálculo adecuado de las probabilidades acumuladas.

```
def calculate_p1(x, pdf, T):
    index = (x > T).nonzero()[0].item()
    #index -= 1
    index = 1 if index < 0 else index
    x1 = x[:index]
    pdf1 = pdf[:index]

    p1 = pdf[:index].sum().item()
    return x1, pdf1, p1


def calculate_p2(x, pdf, T):

    index = (x > T).nonzero()[0].item()
    #index -= 1
    x2 = x[index:]
    pdf2 = pdf[index:]
    p2 = pdf[index:].sum().item()

    return x2, pdf2, p2
```

Listing 1: Función de Probabilidades

La función calculate_mu calcula el valor medio (o la esperanza) de una variable aleatoria representada por un conjunto de valores x y su correspondiente función de densidad de probabilidad pdf. Esta función acepta tres parámetros: x, que es un array de valores de la variable aleatoria, pdf, que es un array de la función de densidad de probabilidad asociada a los valores en x, y p, que representa la probabilidad total.

El cálculo del valor medio se realiza sumando el producto de cada valor en x con su respectiva probabilidad en pdf. Posteriormente, este valor se divide por p para normalizar el resultado.

La función devuelve el valor medio de la variable aleatoria normalizado por p.

```
def calculate_mu(x, pdf, p):
  mu = 0

  for i in range(len(x)):
    mu += (x[i].item() * pdf[i].item())

```

```
  return mu / p
```

Listing 2: Función de Medias

La función calculate_sigma calcula la varianza de una variable aleatoria representada por un conjunto de valores x y su correspondiente función de densidad de probabilidad pdf, dados el valor medio mu y la probabilidad total p. Esta función acepta cuatro parámetros: x, que es un array de valores de la variable aleatoria, mu, que representa el valor medio de la variable aleatoria, pdf, que es un array de la función de densidad de probabilidad asociada a los valores en x, y p, que representa la probabilidad total.

El cálculo de la varianza se realiza sumando el producto de cada probabilidad en pdf con el cuadrado de la diferencia entre cada valor en x y el valor medio mu. Posteriormente, este valor se divide por p para normalizar el resultado.

La función devuelve la varianza de la variable aleatoria normalizada por p.

```
def calculate_sigma(x, mu, pdf, p):
    variance = 0
    for i in range(len(x)):
        variance += pdf[i].item() * ((x[i].item() - mu) ** 2)
    return variance / p
```

Listing 3: Función de Varianzas

La función calculate_statistics calcula estadísticas descriptivas para una distribución de probabilidad dada, dividida en dos grupos según un umbral de separación T. Esta función acepta tres parámetros: T, que representa el umbral de separación para dividir la distribución de probabilidad en dos grupos, x, que es un array de valores de la variable aleatoria, y pdf, que es un array de la función de densidad de probabilidad correspondiente a los valores en x.

El primer paso de la función es dividir la distribución de probabilidad en dos grupos utilizando el umbral T. Luego, se calculan el valor medio (mean_1, mean_2) y la varianza (sigma_1, sigma_2) para cada grupo utilizando las funciones calculate_mu y calculate_sigma, respectivamente. Además, se calculan las probabilidades acumuladas para cada grupo (p1, p2) mediante las funciones calculate_p1 y calculate_p2.

La función devuelve seis valores: mean_1 y mean_2 representan los valores medios de los dos grupos respectivamente, sigma_1 y sigma_2 representan las varianzas de los dos grupos, y p1 y p2 son las probabilidades acumuladas de los dos grupos respectivamente.

```
def calculate_statistics(T, x, pdf):
  x1, pdf1, p1 = calculate_p1(x, pdf, T)
  x2, pdf2, p2 = calculate_p2(x, pdf, T)

  mean_1 = calculate_mu(x1, pdf1, p1)
  mean_2 = calculate_mu(x2, pdf2, p2)

  sigma_1 = calculate_sigma(x1, mean_1, pdf1, p1)
  sigma_2 = calculate_sigma(x2, mean_2, pdf2, p2)

```

```
11  return mean_1, mean_2, sigma_1, sigma_2, p1, p2
```

Listing 4: Función de Momentos Estadisticos

1) Diseñe al menos 2 pruebas unitarias donde verifique el funcionamiento correcto. Detalle en este documento el diseño de tales pruebas y los resultados, indicando si son los esperados.

Respuesta:

La función test_calculate_statistics constituye una prueba unitaria diseñada para evaluar la precisión de la función calculate_statistics. En esta prueba, se generan dos arreglos de valores fijos test_array_1 y test_array_2, cada uno con valores predefinidos. Se utiliza un umbral de separación (T) con un valor de 5 para dividir la distribución de probabilidad en dos grupos. Los valores esperados de la media y la varianza se calculan para cada arreglo, considerando la naturaleza rudimentaria de los cálculos de media (mu) y varianza (sigma). Posteriormente, se concatenan los arreglos y se obtiene el histograma y la función de densidad de probabilidad. Se invoca la función calculate_statistics con estos datos y se comparan los resultados con los valores esperados utilizando una tolerancia predefinida de 10. Es importante mencionar que debido a la naturaleza de los cálculos rudimentarios de mu y sigma, es probable que existan pequeñas diferencias entre los valores calculados y los esperados. Estas diferencias se tienen en cuenta al diseñar la prueba y al establecer la tolerancia.

```
def test_calculate_statistics(tolerance=10):
    test_array_1 = torch.tensor([2.0, 2.0, 2.0, 1., 3.])
    test_array_2 = torch.tensor([10., 10., 17., 13., 13., 13., 30])

    exp_mean_1 = test_array_1.mean().item()
    exp_var_1 = test_array_1.var().item()
    exp_mean_2 = test_array_2.mean().item()
    exp_var_2 = test_array_2.var().item()

    concatenated_tensor = torch.cat((test_array_1, test_array_2), dim=0)
    x_axis, histogram, p_gaussian = get_histogram(concatenated_tensor)
    mean_1, mean_2, sigma_1, sigma_2, p_1, p_2 = calculate_statistics(5,
    x_axis,p_gaussian)

    assert_values(mean_1, exp_mean_1, tolerance)
    assert_values(mean_2, exp_mean_2, tolerance)
    assert_values(sigma_1, exp_var_1, tolerance)
    assert_values(sigma_2, exp_var_2, tolerance)
```

Listing 5: Pruebas Unitarias

Adicionalmente, se utilizó otra prueba unitaria que sigue el mismo flujo, sin embargo inicializa los arreglos de forma aleatoria, teniendo las mismas consideraciones. Los valores son esperados, dentro del margen de tolerancia definido.

b) **(20 puntos)** Implemente la función *calcular_costo_J(T)* la cual calcule el costo del umbral candidato T. Comente su implementacion con detalle en este informe.

Respuesta:

La función calculate_cost calcula el costo asociado a una distribución de probabilidad dada, utilizando la información de las estadísticas calculadas mediante la función calculate_statistics. Esta función toma tres parámetros: t, que representa el umbral de separación, x, que es un array de valores de la variable aleatoria, y pdf, que es un array de la función de densidad de probabilidad asociada a los valores en x.

El costo se calcula utilizando la fórmula que involucra las estadísticas calculadas previamente, donde se ponderan los términos relacionados con la varianza y la entropía de la distribución. Es importante destacar que, debido a la naturaleza de los logaritmos naturales utilizados en el cálculo, si se produce una división por cero, el resultado del logaritmo será infinito negativo. En este caso, la función reemplaza el valor -inf con un número grande positivo, representado por np.finfo(float).max, para evitar problemas de cálculo y asegurar que el costo devuelto sea numéricamente estable.

```
def calculate_cost(t, x, pdf):
  _, _, sigma_1, sigma_2, p_1, p_2 = calculate_statistics(T=t, x=x, pdf=pdf)

  cost = 1 + 2*(p_1*np.log(sigma_1) + p_2*np.log(sigma_2)) - 2*(p_1*np.log(p_1)
     + p_2*np.log(p_2))

  if np.isinf(cost) and cost < 0:
        # Replace -inf with a large positive number
        cost = np.finfo(float).max
  return cost
```

Listing 6: Calculo de Costos

1) Diseñe al menos 2 pruebas unitarias donde verifique el funcionamiento correcto. Detalle en este documento el diseño de tales pruebas y los resultados, indicando si son los esperados.

   Respuesta:

   La función test_calculate_cost es una prueba unitaria diseñada para evaluar la función calculate_cost, la cual calcula el costo asociado a una distribución de probabilidad bajo diferentes umbrales de separación (T). En esta prueba, se inicializan dos arreglos de valores test_array_1 y test_array_2 utilizando la librería PyTorch. Luego, estos arreglos se concatenan y se genera un histograma junto con la función de densidad de probabilidad. La función calculate_cost se invoca dos veces con diferentes valores de T (umbral de separación) establecidos en 2 y 7, respectivamente. Se espera que cuando el umbral T sea menor (2 en este caso), el costo resultante sea mayor debido a la mayor separación entre las curvas de distribución. Se emplea una aserción para verificar que el costo calculado con T=2 sea efectivamente mayor que el costo calculado con T=7. En caso de que la aserción falle, se imprimirán los valores de los costos para ayudar en la depuración.

```
def test_calculate_cost():
    test_array_1 = torch.tensor([2.0, 2.0, 2.0, 1., 3.])
```

```
    test_array_2 = torch.tensor([10., 10., 17., 13., 13., 13., 30])

    concatenated_tensor = torch.cat((test_array_1, test_array_2), dim=0)

    x_axis, histogram, p_gaussian = get_histogram(concatenated_tensor)

    try:
        cost_1 = calculate_cost(2, x_axis, p_gaussian)
        cost_2 = calculate_cost(7, x_axis, p_gaussian)

        # Asserting that cost_1 is greater than cost_2
        assert cost_1 > cost_2
        print("Cost 1 should be greater than Cost 2")
        print("Cost 1: ", cost_1)
        print("Cost 2: ", cost_2)

    except AssertionError as e:
        print(f"Assertion Error: {e}")
```

Listing 7: Pruebas Unitarias

Asimismo, se utilizara una prueba con valores aleatorios, En esta prueba, se generan dos arreglos de tamaño 10 con valores aleatorios dentro de rangos específicos. La prueba sigue el flujo de la pasada, pero se espera que el costo del umbral donde se traslapan los datos sea peor que uno que se ubique en el valle.

c) **(20 puntos)** Basado en ambas funciones, implemente la función *calcular_T_optimo_Kittler(Imagen)* la cual retorne el T optimo para umbralizar la imagen recibida, además de la imagen umbralizada.

d) Aplique el algoritmo de Kittler en la imagen *cuadro1_005.bmp*, provista.

1) Grafique el histograma normalizado de la imagen de entrada provista.
Respuesta:

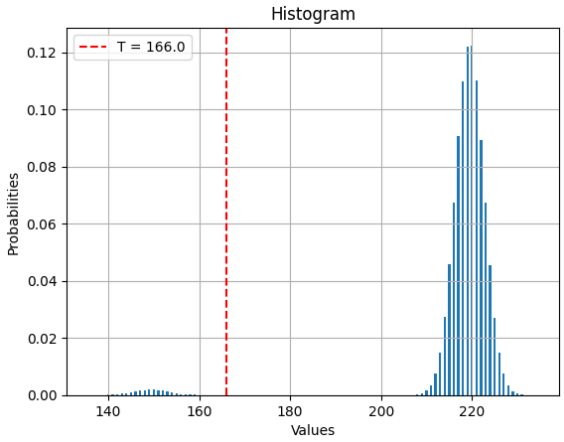

Figura 1: Histograma Normalizado con Umbral Calculado

2) Grafique la función $J(T)$, y documente el valor $T = \tau$ que logra el valor mínimo de $J(T)$, junto con las medias y varianzas de las dos funciones Gaussianas superpuestas. Son coherentes tales valores con el histograma graficado en el punto anterior?

a' El valor óptimo en el caso de esta imagen debe ser cercano a $\tau = 168$, con $\mu_1 = 149{,}45$, $\mu_2 = 219{,}49$ $\sigma_1^2 = 15{,}36$ y $\sigma_2^2 = 10{,}05$.

Respuesta:

Parameter	**Value**
Optimal T	166.0
T Cost	5.8210835273637835
Mean 1	149.45175371050092
Mean 2	219.49267965622414
Var 1	15.366404245530088
Var 2	10.054693964789035
$P(1)$	0.019073784351348877
$P(2)$	0.9809261560440063

Cuadro 1: Parametros Calculados

Después de analizar los resultados obtenidos para el umbral óptimo, observamos que estos son coherentes y están en línea con las expectativas establecidas en el

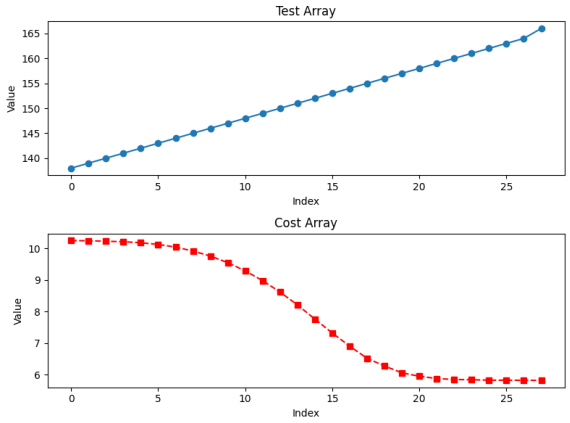

Figura 2: Graficas de Umbral y Costo

enunciado del problema. Además, los valores calculados se acercan satisfactoriamente a los valores esperados, con una pequeña diferencia en el valor de 'T'. Mientras que en el enunciado se espera un valor de 168 para 'T', la implementación calcula un valor ligeramente menor de 166.

Aunque existe esta discrepancia mínima en el valor de 'T', los resultados en general muestran una coherencia con las expectativas y proporcionan una base sólida para el análisis posterior. Es importante tener en cuenta que las pequeñas variaciones en los resultados pueden ser atribuibles a factores como la precisión de los cálculos numéricos y las diferencias en la implementación específica del algoritmo.

3) Lograría el umbral óptimo τ obtenido umbralizar satisfactoriamente la imagen de prueba? Umbralice la imagen de entrada provista y muestre los resultados.
4) Asigne con valor de 255 los pixeles del cuadrado (clase *foreground*), y 0 los del fondo (clase *background*).

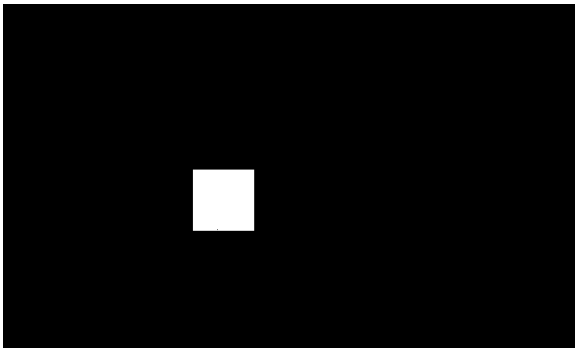

Figura 3: Imagen Umbralizada

e) (**25 puntos**) Pruebe la implementacion del algortimo de Kittler para detectar la actividad de voz humana en un audio, usando el audio de prueba provisto *contaminated_audio.wav*.

1) Grafique el histograma del audio leido, y argumente si es apropiado usar el algoritmo de Kittler o no.

Respuesta:

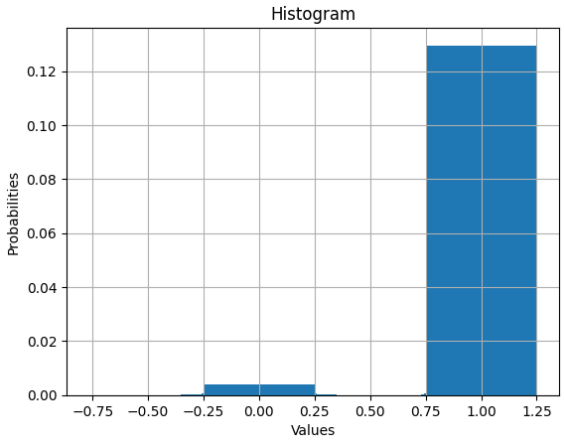

Figura 4: Histograma Audio

Notese que el tensor de audio tiene muchos valores continuos, a diferencia de la imagen que tienden a ser valores discretos. Esto se ve reflejado en el histograma que encapsula muchos.Es importante tener en cuenta que, a diferencia de las imágenes donde los valores de los píxeles tienden a ser discretos, los archivos de audio están compuestos por valores de amplitud que pueden ser continuos y variar en una escala amplia. Esto se refleja en el histograma del tensor de audio, donde muchos valores caen en el mismo bin debido a la naturaleza continua de los valores de amplitud. Mientras que en una imagen, el histograma puede representar la distribución de intensidades de los píxeles con más claridad, en el caso del audio, puede ser más desafiante interpretar la distribución de amplitudes debido a la presencia de valores continuos. valores en el mismo bin.

Además, es importante considerar que el algoritmo de Kittler, que se utiliza comúnmente en el procesamiento de imágenes para encontrar umbrales óptimos, puede no ser la mejor opción para aplicar directamente a datos de audio. Esto se debe a que los datos de audio tienden a ser continuos en naturaleza, lo que significa que no hay una separación clara entre las distintas clases o categorías de datos. En consecuencia, puede resultar desafiante encontrar un umbral adecuado que pueda separar efectivamente los datos en dos grupos distintos utilizando este algoritmo. Por lo tanto, es importante considerar alternativas o adaptar el algoritmo de Kittler según las características específicas de los datos de audio.

Otra acotación importante, es que para el calculo de estas métricas se uso una versión del audio acotado por cuestión de tiempo de procesamiento. Sin embargo el algoritmo funciona con el audio original y presenta un comportamiento similar.

2) Grafique la función $J(T)$, y documente el valor $T = \tau$ que logra el valor mínimo de $J(T)$, junto con las medias y varianzas de las dos funciones Gaussianas superpuestas. Son coherentes tales valores con el histograma graficado en el punto anterior?

Respuesta:

Parameter	Value
Optimal T	0.616851806640625
T Cost	-11.389923069013133
Mean 1	-0.0010341154268984616
Mean 2	0.9959984517028155
Var 1	0.003929538515831793
Var 2	3.379837501189842e-05
$P(1)$	0.7420892715454102
$P(2)$	0.25791072845458984

Cuadro 2: Values of Parameters

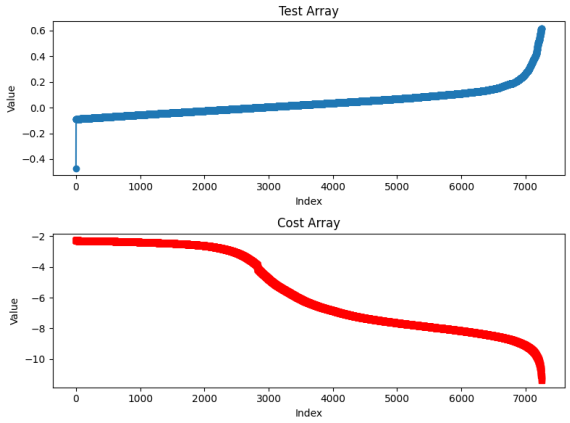

Figura 5: Umbral Optimo Y Costo en Audio

Analizando los resultados, y el umbral de separacion, vemos que los resultados tienen coherencia con el comportamiento del audio como tal.

3) Lograría el umbral óptimo τ obtenido partir satisfactoriamente el sonido de prueba en 3 segmentos (2 de actividades de audio y 1 segmento sin actividad)? Umbralice el audio provisto y muestre los resultados.

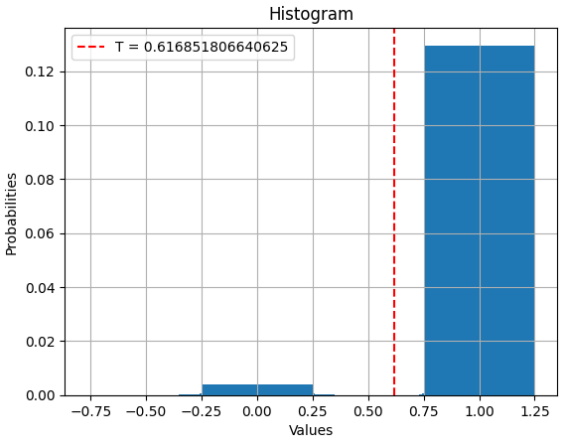

Figura 6: Histograma con Umbral

2. **(15 puntos)** La distancia de Bhattacharyya compara dos funciones de densidad de probabilidad $p(x)$ y $q(x)$:

$$D_{\text{JS}}(p, q) = -\ln\left(\sum_{x \in X} \sqrt{p(x)\, q(x)}\right)$$

a) Implemente la función *calcular_bhattacharyya_distance(p,q)*, para comparar las funciones de densidad estimada con el ajuste del modelo mixto Gaussiano con Kittler $p(x)$ y la aproximacion de la densidad con el histograma de los datos $q(x)$ para las dos pruebas realizadas con la imagen y el audio.

Respuesta:

```python
def bhattacharyya_distance(p, q):

    sqrt_product = torch.sqrt(p * q)
    sum_sqrt_product = torch.sum(sqrt_product)
    distance = -torch.log(sum_sqrt_product)

    return distance.item()
```

Listing 8: distancia de Bhattacharyya

Al ejecutar la función con ambos conjuntos de datos, observamos discrepancias de -8 y -10 respectivamente. Estas diferencias, aunque presentes, no se consideran significativas en términos absolutos. Más bien, sugieren que el algoritmo de distancia de Bhattacharyya determina que los conjuntos de datos son similares.

b) Explique la relación entre la distancia de Bhattacharyya y el proceso de estimación de los parámetros óptimos implementado en el algoritmo de Kittler ¿Que sucede cuando la distancia de Bhattacharyya entre el histograma de los datos y el modelo estimado crece o decrece ?

Respuesta:

La distancia de Bhattacharyya se utiliza en el algoritmo de Kittler para evaluar la similitud entre la distribución de los datos observados y el modelo estimado. Cuando esta distancia disminuye, indica un mejor ajuste del modelo a los datos, lo que implica una estimación más precisa de los parámetros óptimos. Por otro lado, un aumento en la distancia sugiere una discrepancia entre el modelo y los datos, lo que puede indicar la necesidad de ajustar el proceso de estimación de parámetros.

3. **(10 puntos extra)** Calcule el umbral óptimo con el algoritmo de Kittler, y umbralice la imagen de *trackedCell15.tif* provista documentando los resultados. Muestre la imagen umbralizada y el histograma de la misma.

   a) Usando la imagen, su histograma, y la matriz de confusion para la clase *foreground*, explique el porqué del resultado obtenido.

   b) Como modificaría el algoritmo de Kittler para mejorar el resultado de la umbralización? Puede usar recursos bibliográficos externos.

4. **(10 puntos extra)** Para tanto el audio y la imagen, implementa y pruebe la optimizacion con un paquete como *optuna* o *weights and biases*. Reporte los resultados y comentelos.

Referencias