Procesamiento digital de señales
Universidad de Antioquia
2023-1
Proyecto final

Filtros adaptativos

César Augusto Montoya Ocampo (1036681523)

El principio de operación de un filtro adaptativo son sus características variables en el tiempo autoajustables. Un filtro adaptativo usualmente toma la forma de un filtro FIR, con un algoritmo de adaptación que continuamente actualiza los coeficientes del filtro, tal que la señal de error se minimiza de acuerdo a cierto criterio. La señal de error usualmente es derivada de alguna manera desde diagrama de flujo de la aplicación, tal que sea una medida de qué tan cerca el filtro está del óptimo. [1]

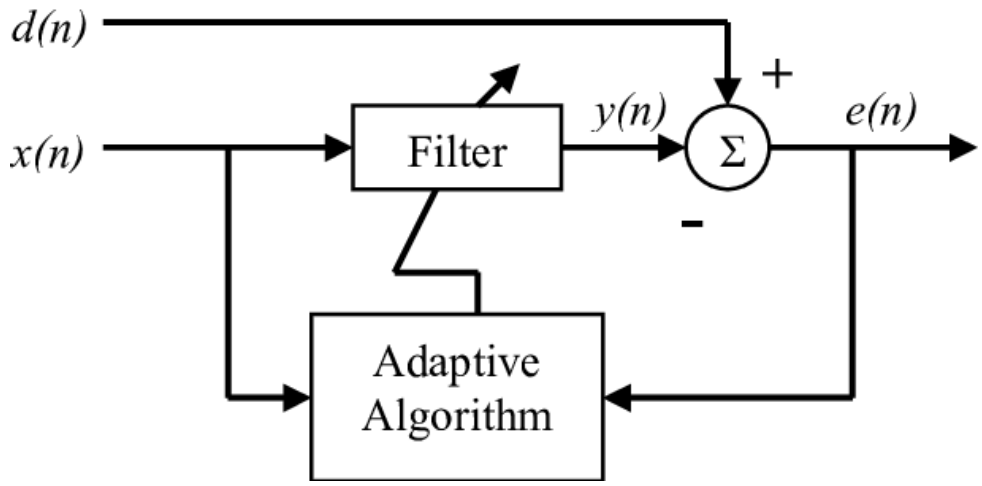

Figure 1: Diagrama de bloques de un filtro adaptativo genérico.

LMS (Least Mean Squares)

El algoritmo de los mínimos cuadrados medios es extremadamente popular por su simplicidad y baja complejidad computacional. El algoritmo está basado en el método de gradiente descendiente, pero es aún más simple ya que solo ejecuta una iteración por muestra, y además, calcula solo un estimado del vector gradiente por cada muestra, los coeficientes del filtro se calculan mediante la siguiente ecuación.

$$w_{n+1}(i) = w_n(i) + 2\mu e_n x_{n-i} \tag{1}$$

La variable w son los coeficientes del filtro, que se actualizan con cada iteración del algoritmo, la variable x es la señal de entrada del filtro, mientras que y es la salida del filtro, la cual se utiliza para calcular el error en cada iteración e ir reduciendo su valor. El parámetro μ es una constante que regula la estabilidad y rapidez de convergencia, para calcular el valor del error se usa:

$$e_n = d_n - \sum_{i=0}^{N-1} w(i) \cdot x_{n-i} \tag{2}$$

1. Escriba un programa que cargue el archivo `voices.wav` el cual contiene un audio con diversas voces, grafique la señal, nombrando los ejes de manera adecuada y poniendo un título respectivo a la gráfica, finalmente reproduzca el audio.

2. Escriba un programa que cargue el archivo `rock.wav` el cual contiene una pista musical, grafique la señal, nombrando los ejes de manera adecuada y poniendo un título respectivo a la gráfica, finalmente reproduzca el audio.

3. Realice la tarea de combinar los dos audios, asegurándose de que terminen al mismo tiempo sin alterar la longitud del audio más largo, reproduzca el resultado ¿Logra distinguir ambos audios?

4. Grafique el espectro de los audios por separado ¿Qué filtro utilizaría para intentar separar la voz del audio?

5. Según su respuesta al numeral anterior implemente un filtro FIR con esas características, grafique la forma de onda y el espectro del resultado del filtro, reproduzca el audio filtrado ¿Fue posible remover la música usando este filtro? Para implementar el filtro puede utilizar una de las siguientes funciones.

```
def low_pass_filter(fs, cutoff_hz, roll_off, ripple_dB):
    nyq_rate = fs / 2.0
    width = roll_off/nyq_rate
    N, _ = kaiserord(ripple_dB, width)
    taps = firwin(N, cutoff_hz/nyq_rate, pass_zero="lowpass")
    return taps

def high_pass_filter(fs, cutoff_hz, roll_off, ripple_dB):
    nyq_rate = fs / 2.0
    width = roll_off/nyq_rate
    N, _ = kaiserord(ripple_dB, width)
    taps = firwin(N, cutoff_hz/nyq_rate, pass_zero="highpass")
    return taps

# Para aplicar el filtro se puede usar lfilter:
taps = low_pass_filter(44100, 1500, 200, 60)
fir_filtered = lfilter(taps, 1.0, x)
```

6. Utilice un filtro LMS para filtrar la señal combinada, como parámetro `x` use la señal combinada, como parámetro `d` use la señal de voz individual, y como parámetro `mu` use 0.1, grafique el espectro de las voces originales, y del resultado de los dos filtros para poder comparar, reproduzca el resultado, según el espectro, y el audio resultante ¿Qué resultado se parece más al original? ¿Fue posible aislar las voces? ¿Por qué algunas voces resultaron distorsionadas y otras no? Puede ayudarse de la siguiente función para implementar el filtro LMS.

```
def lms(x, d, N=4, mu=0.1):
    nIters = min(len(x), len(d)) - N
    # Copia de la senal
    u = np.zeros(N)
    # Coeficientes del filtro
    w = np.zeros(N)
    # Salida del filtro
    y = np.zeros(nIters)
    # Error estimado
    e = np.zeros(nIters)
    for n in range(nIters):
        # Tomar una copia de la senal
        u[1:] = u[:-1]
        u[0] = x[n]
        # Calcular la salida del filtro
        y[n] = np.dot(u, w)
        # Calcular el error estimado
        e_n = d[n] - y[n]
        # Actualizar los pesos del filtro
        w = w + 2*mu * e_n * u
        e[n] = e_n
    return y
```

7. Para poder observar mejor el comportamiento del filtro, aplique el filtro al audio combinado para los valores de μ de 0.001, 0.01, 0.1 y 0.2. Determine de qué manera influye este parámetro en el algoritmo.

Conclusiones

Tómese el tiempo de escribir conclusiones generales de la práctica realizada, estas deben ser basadas en lo trabajado, y en caso de referenciar algún artículo externo deberá citarlo.

Referencias

[1] "EEE305 - Digital Signal Processing". https://www.staff.ncl.ac.uk/oliver.hinton/eee305/