Trabajo práctico Integrador

Linear predictive coding

1. Introducción

Una forma de producir síntesis de voz artificial es imitando el proceso de generación del habla humana. Considerando el modelo de resonancia de la Figura 1, la producción de habla (y otros sonidos) se puede pensar como una fuente de señal que excita un sistema que resonará más en ciertas frecuencias que en otras, dependiendo la configuración que tenga el tracto vocal para cada fonema particular. Uno de los métodos más conocidos para resolver este problema es el denominado LPC (Linear Predictive Coding), el cual permite representar la señal de voz mediante un conjunto reducido de parámetros que capturan sus características acústicas esenciales.

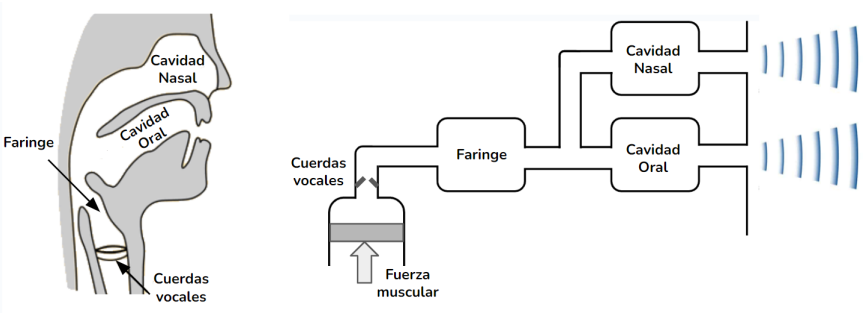

Figura 1: Modelo de resonancia del sistema de generación de habla. A la izquierda modelo anatómico. A la derecha, modelo idealizado.

1.1. Modelo

El modelo utilizado con LPC para la generación habla es el que se muestra en la Figura 2 [2]. Allí, el sistema LTI $H(z)$ es utilizado para modelar la resonancia del tracto vocal para un determinado fonema, asumiendo condiciones de estacionariedad. El método supone dos clases de sonidos dependiendo la fuente de excitación: el habla *sonora*, para un sonido vocálico como "aaaa", "eeee", etc, y el habla *sorda*, para sonidos de consonantes como "ssss", "ffff", etc. Para el habla sonora la fuente de excitación debe ser un tren de impulsos periódico normalizado $U(n) = \sum_{k=-\infty}^{\infty} \sqrt{T_p f_s}\, \delta(n - kN_p)$ de periodo T_p, siendo $f_p = 1/T_p$ el "*pitch*" o frecuencia fundamental de la excitación, fs la frecuencia de muestreo del audio digitalizado y $N_p \approx T_p f_s$ el periodo en tiempo discreto. Por su parte, el habla sorda recibe como entrada un proceso de ruido blanco gaussiano $U(n) \sim N(0, 1)$, que resulta más adecuado para este caso. El sistema LTI utilizado para modelar a $H(z)$ en este problema es del tipo *all-pole* (de solo polos) [3], como se describe en la ecuación (1), caracterizado por los coeficientes a_1, a_2, ..., a_P y la ganancia G. Asumiendo la entrada $U(n)$ como un proceso aleatorio ESA, la salida del sistema representa un proceso autorregresivo $X(n)$ de orden P que puede expresarse con su ecuación en diferencias indicada en (2).

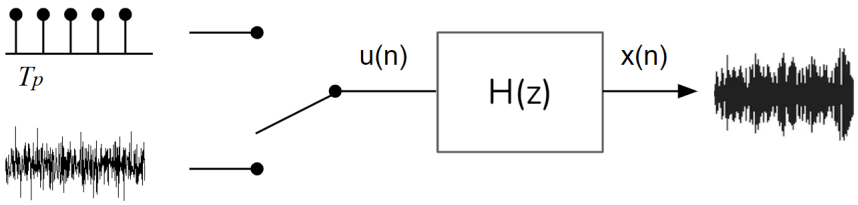

Figura 2: Modelo de generación del habla para LPC.

$$H(z) = \frac{X(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{P} a_k z^{-1}} \tag{1}$$

$$X(n) = \sum_{k=1}^{P} a_k X(n-k) + G\,U(n) \tag{2}$$

1.2. Estimación de parámetros del modelo

Nos interesa encontrar el conjunto de coeficientes a_1, a_2, ..., a_P y el factor de ganancia G (de la ecuación 1) de modo que éstos se ajusten lo mejor posible al modelo. Una posible solución consiste en resolver las ecuaciones de Yule-Walker [4]. De la ecuación (2) se puede comprobar que $R_X(k) = \sum_{i=1}^{P} a_i R_X(k-i) + G^2\delta(k)$, donde $\delta(k)$ es un impulso discreto. Para $k = 0$, la autocorrelación se puede reescribir para determinar la ganancia de acuerdo a la ecuación (4). Para para $k \neq 0$, expresando $R_X(k)$ de forma matricial, donde $\mathbf{a} = [a_1\, a_2 \,...\, a_P]^T$, $\mathbf{r} = [R_X(1)\, R_X(2) \,...\, R_X(P)]^T$ y R como se define en (5), se pueden estimar los parámetros de $H(z)$ utilizando (3) para los coeficientes y (4) para la ganancia.

$$\widehat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{r} \tag{3}$$

$$\widehat{G} = \left(R_X(0) - \sum_{i=1}^{P} a_i R_X(i) \right)^{1/2} \tag{4}$$

$$\mathrm{R} = \begin{bmatrix} R_X(0) & R_X(-1) & \dots & R_X(-P+1) \\ R_X(1) & R_X(0) & \dots & R_X(-P+2) \\ \vdots & \vdots & \ddots & \vdots \\ R_X(P-1) & R_X(P-2) & \dots & R_X(0) \end{bmatrix} \tag{5}$$

1.3. Detección de pitch

En la sección precedente vimos cómo hallar los coeficientes del sistema que modela la resonancia del tracto vocal, pero para el caso de fonemas sonoros, necesitamos también identificar la frecuencia de pitch del fonema. De la misma señal a codificar, puede estimarse el pitch y guardar esa información para que en la reconstrucción se sintetice una señal con la misma entonación que la original. Para ello se deberá aplicar un algoritmo de detección de pitch (PDA, Pitch Detection Algorithm). Nosotros utilizaremos el método de *Autocorrelación*, el cual requiere estimar la autocorrelación del residuo $R_e(k) = \mathbb{E}[e(n)e(n+k)]$, donde $e(n) = X(n) - \sum_{i=1}^{P} a_i X(n-i)$. Dado que estamos suponiendo una señal sonora, la autocorrelación de este residuo conserva la información de periodicidad para estimar el pitch midiendo

la diferencia de tiempo (discreto) entre el máximo pico de potencia $\hat{R}_e(0)$ y el segundo pico de mayor potencia $\hat{R}_e(k_{max})$ y de ahí determinar el periodo fundamental T_p (teniendo en cuenta la frecuencia de muestreo f_s) de acuerdo a la siguiente ecuación:

$$f_p = \frac{1}{T_p} = \frac{f_s}{|k_{max}|} \tag{6}$$

El pitch estimado será válido solo para señales del tipo sonoras, ya que las señales sordas no se modelan con una excitación periódica, sino con ruido blanco. Esto significa que deberá identificarse si ese tramo de señal de habla pertenece al tipo sonoro o sordo. Para éste trabajo práctico, utilizaremos un método sencillo que consiste en normalizar la correlación del residuo $r_e(k) = R_e(k)/\sigma_e^2$ y evaluar el segundo máximo $r_e(k_{max})$ comparándolo con un umbral α, como se observa en al Figura 3. Si se supera dicho umbral, se asumirá que el segundo pico más alto tiene suficiente potencia como para considerar una excitación periódica. Caso contrario asumiremos el residuo es ruido blanco. El valor de este umbral será un parámetro más a calibrar.

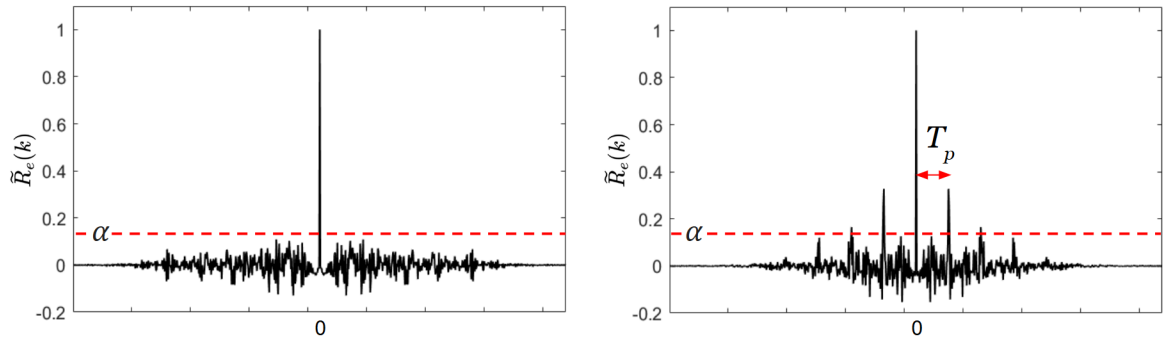

Figura 3: Estimación del pitch con el método de Autocorrelación. Izquierda: para señal sorda. Derecha: para señal sonora. $T_p = k_{max}/f_s$ es el período en tiempo continuo.

1.4. Codificación y decodificación

En la Figura 4 se representan esquemas generales de proceso de codificación y decodificación LPC. Si bien la señal de habla se puede considerar un proceso aleatorio **no estacionario**, es posible asumir que es localmente estacionario. Esto significa que podemos segmentar la señal en muchas ventanas de corta duración (decenas de milisegundos), como se observa en la Figura 4 (a). Esta segmentación suele realizarse con una ventana móvil que suaviza los bordes del segmento, por ejemplo una ventana de Hamming. Además, para que las transiciones entre segmentos sean más naturales, las ventanas suelen estar solapadas en tiempo. El bloque de codificación LPC, realiza el proceso de estimación de los coeficientes **a**, ecuación (3), la ganancia G, ecuación (4) y el pitch o frecuencia fundamental f_p, ecuación (6).

Por otra parte, en la Figura 4 (b) se puede ver el proceso de decodificación. En este caso se recibe una trama de datos que contiene los coeficientes, la ganancia y el pitch, asociados a cada ventana de tiempo. Estos datos se usan para regenerar la señal del habla aplicando una excitación de impulsos periódicos de frecuencia f_p o ruido blanco, según sea el caso, a un sistema IIR como el de la ecuación (2). Finalmente la señal completa se reconstruye sumando los segmentos regenerados y ventaneados (suavizados), con el mismo solapamiento y ubicación temporal con el que fue segmentada.

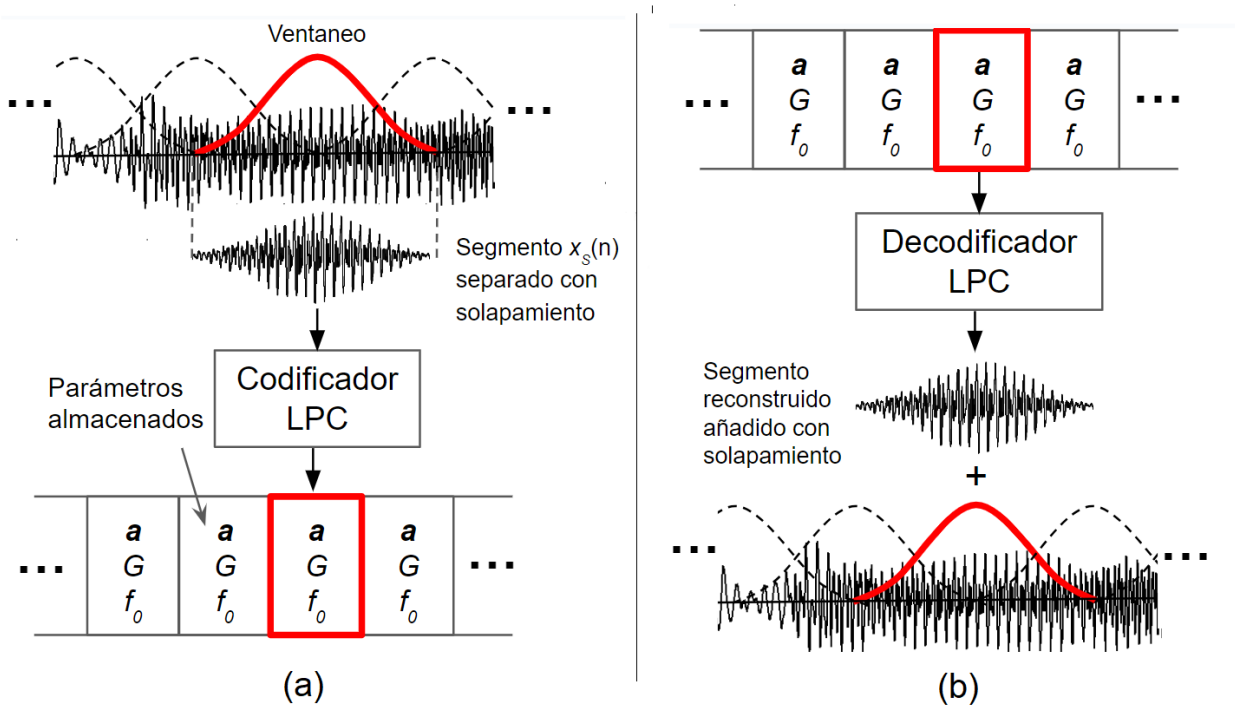

Figura 4: (a) Segmentación y codificación. (b) Decodificación y reconstrucción.

2. Desarrollo

Ejercicio 1

(a) Defina una función con prototipo `param_lpc(xs, P)` donde `xs` es la señal y `P` el orden del modelo. La función debe retornar los coeficientes LPC y la ganancia G. Para cada uno de los audios de los archivos "`a.wav`", "`e.wav`", "`s.wav`", "`sh.wav`" (disponibles en el campus), estime el los parámetros del modelo LPC suponiendo órdenes $P = \{5, 10, 30\}$.

(b) Grafique la respuesta temporal de los audios originales y el periodograma de cada uno (en decibéles) superpuesto a la PSD paramétrica estimada de acuerdo al modelo de la ecuación (7) (para el rango $\omega \in [0, \pi)$). Ayuda: para obtener $S_U(\omega)$ puede simular el periodograma de $U(n)$, considerando una frecuencia de 200 Hz para los fonemas sonoros ("`a`" y "`e`").

$$S_X(\omega) = \frac{G^2}{|1 - \sum_{k=1}^{P} a_k e^{-j\omega k}|^2} S_U(\omega) \tag{7}$$

Ejercicio 2

(a) Implemente la función `pitch_lpc(xs, a, alpha, fs)` que reciba un segmento de señal `xs` (que suponemos es aproximadamente estacionario), el vector de coeficientes `a`, el umbral de detección `alpha`, la frecuencia de muestreo `fs` y retorne la frecuencia de pitch estimada. En caso de no detectar una frecuencia, debe retornar 0.

(b) De los archivos disponibles, seleccione al menos una vocal y una consonante para sintetizarlas mediante sus coeficientes LPC, con un orden P que considere apropiado, y una duración igual al audio original que se está emulado. Para el caso de la vocal, utilizar la función `pitch_lpc()` para estimar la frecuencia de pitch. Compare cada fonema sintético con el original graficando superpuestos sus periodogramas.

Ejercicio 3

En este ejemplo se busca realizar la segmentación de una señal de audio completa, la estimación de los parámetros LPC en cada segmento y la reconstrucción. Tenga en cuenta que puede ajustar los parámetros (orden P, ancho de cada segmento y umbral de pitch), para lograr una reproducción óptima. Considere como audios de prueba los archivos "`audio_01.wav`", "`audio_02.wav`", "`audio_03.wav`" y "`audio_04.wav`".

(a) Complete el código la función principal del script "`tpi.py`" para ejecutar el proceso completo de codificación y decodificación LPC. Considere la frecuencia de pitch para de cada segmento reconstruido sea la estimada por la función "`pitch_lpc()`". Ajuste los parámetros hasta que la reproducción del audio reconstruido sea lo más fiel posible.

(b) Repita el punto anterior, pero esta vez la reconstrucción debe hacerla con uan frecuencia de pitch fija igual a 200 Hz.

(c) Repita el punto anterior pero en este caso sin ninguna frecuencia de pitch, es decir, sintetizando todos los segmentos con ruido blanco.

(d) Repita nuevamente la ejecución del programa pero esta vez usando como frecuencia de pitch la frecuencia generada por la función "`pitch_sintetico(i)`" (donde `i` debe corresponderse con el número de segmento).

3. Conclusiones

Como conclusiones, elabore un resumen breve y conciso comentando características que considere relevantes del método propuesto en este trabajo y los resultados obtenidos, así como dificultades encontradas y cómo fueron abordadas.

4. Normas y material entregable

EL trabajo práctico debe subirse al campus en formato ZIP de nombre `TPI_GXX.ZIP` (donde `XX` es el número de grupo), dentro del cual se debe incluir el Informe y el Código. También deben cumplirse la siguientes condiciones:

- **Informe**: debe ser conciso y mostrar los resultados solicitados con los comentarios y suposiciones hechas para cada ejercicio. El informe debe entregarse en formato PDF (**no se aceptarán otros formatos**) de nombre `TPI_GXX.PDF` . No incluir líneas de código en el mismo PDF, ya que el código debe entregarse aparte.
- **Código**: Los archivos de código utilizados deben ser en formato `.py` (o `.m` si prefiere usar Matlab/Octave) de nombre `TPI_GXX_EJX.PY` .
- **No deben incluirse los archivos de audio** en la entrega.
- Los conceptos y desarrollos involucrados en el TP podrán ser evaluados en cualquiera de las instancias de examen que establezca el docente.

Referencias

[1] Steven M. Kay. Intuitive probability random processes using MATLAB. Springer 1951.

[2] Rabiner, L.R., R.W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[3] Makhoul, J., "Linear Prediction: A Tutorial Review,ÏEEE Proceedings, Vol. 63, pp. 561-580, 1975.

[4] S. Haykin, Adaptive filter theory, Prentice-Hall, 1996.