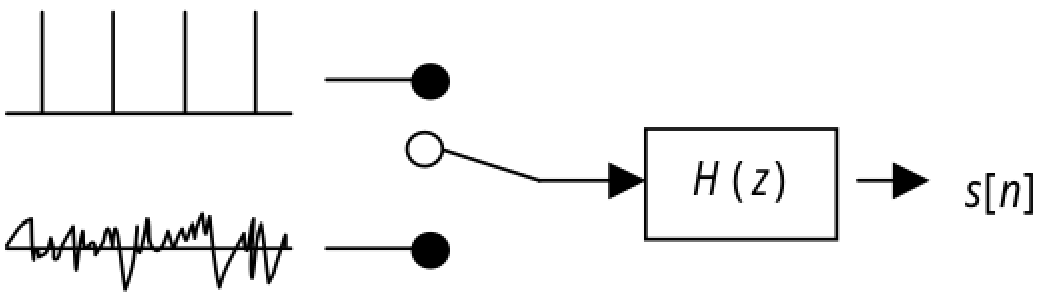
Procesamiento del habla

Modelo del habla

Para empezar a hablar de procesamiento del habla tenemos que tener un modelo de cómo se produce el habla. El modelo que usamos es este: tenemos una excitación que es la entrada a un sistema, esta excitación decimos que es un tres de deltas si es fonema voiced, o es ruido si es unvoiced. El filtro $H(z)$ es lo que distingue entre fonemas. Por ejemplo, si es una 'a' el fonema, la entrada es el tren de deltas y el $H(z)$ es el filtro correspondiente a la 'a'. Si es una 'f' la entrada es la de abajo y el $H(z)$ es el filtro de la 'f'.

- Excitación -> filtro -> señal de audio
- Voiced: tren de deltas
- Unvoiced: ruido
- $H(z)$ distinto para cada fonema



Lo importante de esto es que lo que nos da la información acerca de qué fonema se dice es básicamente el filtro H.

Entonces si tenemos una señal de audio $s[n]$ en la que queremos hacer reconocimiento, tenemos que de alguna forma obtener $H(z)$.

Cepstrum

Según lo del grafico anterior, la señal de audio $s[n]$ es la excitación $x[n]$ pasada por el filtro $h[n]$.

$$s(n) = x(n) * h(n)$$

Para hacer la desconvolución, separar $x(n)$ de $h(n)$ obviamente tenemos que trabajar con el espectro.

$$S(\omega) = X(\omega).H(\omega)$$

En frecuencia tenemos un producto, pero esto sigue siendo difícil de separar. Entonces lo que hacemos es aplicar el logaritmo al espectro, y esto resulta en una suma, que es mucho más fácil de separar.

$$|S(\omega)| = |X(\omega)|.\,|H(\omega)|$$

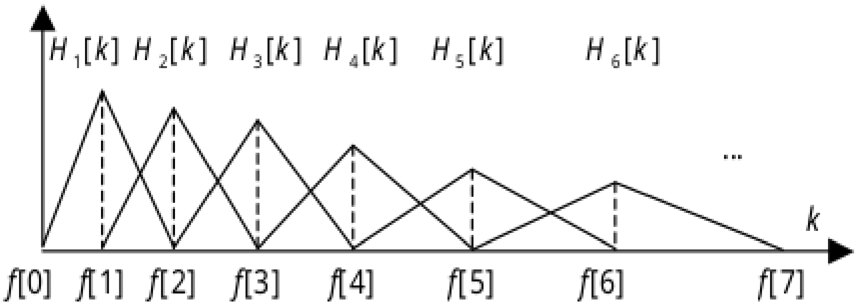
$$\log|S(\omega)| = \log|X(\omega)| + \log|H(\omega)|$$

Ahora que ya tenemos esta suma, si aplicamos otra vez una transformada o antitransformada, (según la versión), vemos que la parte correspondiente a H queda en la parte más baja, mientras que la de X queda en los coeficientes más altos. Entonces podemos simplemente tomar los primeros N coeficientes de $\hat{s}$ y nos quedamos con los coeficientes de $\hat{h}$. Con esto entonces ya tenemos la información del filtro H. Si bien podemos convertir este $\hat{H}$ a $H(\omega)$ o incluso a $h[n]$, en realidad no hace falta para el reconocimiento del habla y podemos usar los coeficientes Cepstrum directamente.

$$s(n) \rightarrow DFT \Rightarrow Mel \Rightarrow log \Rightarrow IDFT|FFT \rightarrow \hat{s}(n)$$

MFCC

- La sensibilidad del oído no es la misma para los distintos rangos de frecuencias
- Un valor "promediado" para cada rango:



$$s(n) \rightarrow DFT \Rightarrow Mel \Rightarrow log \Rightarrow DCT \rightarrow \hat{s}(n)$$

$$F(m) = \log\left\{\sum_{k=0}^{N-1}|S(k)|^2 H_m(k)\right\},\ 0 \leq m \leq M$$

$$\hat{s}(n) = \sum_{m=0}^{M-1} F(m)\cos\left(\pi n(m+\frac{1}{2})/M\right),\ 0 \leq n \leq M$$

Clasificación

- Etiquetar a los MFCC con el fonema correspondiente

Aprendizaje supervisado

- Set de entrenamiento: $x_1, \ldots, x_N$ **y** las etiquetas $z_1, \ldots, z_N \in 1, \ldots, K$

1. Maximum likelihood:

$$\mu_{ML}^{k} = \frac{\sum_{z_i=k} x_i}{\#(z_i = k)}$$

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{ML})(x_i - \mu_{ML})^T$$

- Igual Σ para todo $k \Rightarrow$ separación lineal entre clases (LDA)

1. Obtener k que maximice $P(k|x) = \frac{P(x|k)P(k)}{P(x)} \Rightarrow P(x|k)P(k)$

$$P(x|k) = \mathcal{N}(x; \mu_{ML}^{k}, \Sigma_{ML})$$

$$P(k) = \frac{\#(z_i = k)}{N}$$

Aprendizaje NO supervisado

- Set de entrenamiento: $x_1, \ldots, x_N$ **sin** las etiquetas $z_1, \ldots, z_N$

Algoritmo EM

Asigna una probabilidad a cada x de pertenecer a cada clase k

1) **Inicializar** μ_k, σ_k^2, y $\pi_k = P(k)$ para cada clase (p.ej.: K-Means)

2) **Paso E**: Calcular $\gamma_k(x) = P(z = k|x)$: probabilidad de que la muestra x pertenezca a la clase k.

$$\gamma_k(x) = P(z = k|x) = \frac{P(x|z = k)P(z = k)}{P(x)}$$

$$\gamma_k(x) = \frac{\pi_k\, \mathcal{N}(x; \mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l\, \mathcal{N}(x; \mu_l, \Sigma_l)}$$

3) **Paso M**: Reestimar μ_k, σ_k^2, y $\pi_k = P(k)$ para cada clase.

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma_k(x_i) x_i}{\sum_{i=1}^{N} \gamma_k(x_i)}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} \gamma_k(x_i)(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} \gamma_k(x_i)}$$

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(x_i)$$

4) Calcular **log-likelihood**

$$P(x_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \sigma_k^2)$$

$$LL = \log \prod_{i=1}^{N} P(x_i) = \sum_{i=1}^{N} \log P(x_i)$$

5) Repetir 2, 3, 4 hasta que el log-likelihood no cambie.

Cadenas de Markov

- Estados: $1, \ldots, N$
- Transiciones: probabilidades a_{ij}

Experimento: generar secuencia de estados $Q = q_1, \ldots, q_T$

- $P(qt|q_{t-1}, q_{t-2}, \ldots q_1) = P(q_t|q_{t-1})$: la probabilidad de pasar de un estado a otro NO DEPENDE de todo la secuencia de estados previa
- $a_{ij} = P(q_t = j | q_{t-1} = i)$: la probabilidad de pasar de un estado a otro es constante en el tiempo

Cadenas de Markov Ocultas (HMM)

- En cada estado se genera una observación $\Rightarrow$ distribución de probabilidad
- $Q = q_1, q_2, \ldots q_T$: secuencia de estados
- $Y = y_1, y_2, \ldots, y_T$: secuencia de observaciones

- $P(y_t|Q, Y_{-t}) = P(y_t|q_t = j)$: la observación generada en un estado sólo depende de cuál es el estado actual $\Rightarrow$ $P(y_t|q_t = j) = b_j(y_t)$

Modelo: $\lambda = \{\{i\}, \{a_{ij}\}, \{b_j(y_t)\}\}$, con $i, j = 1, \ldots, N$, $t = 1, \ldots, T$

Algoritmo de Viterbi

- Dado λ y una secuencia de observaciones $Y = y_1, \ldots, y_T$, hallar la secuencia de estados óptima $Q^* = q_1^*, \ldots, q_T^*$

$$Q^* = \underset{\forall\, Q}{\operatorname{argmax}}\, P(Q|Y)$$

$\phi_t(j) = \max\limits_{\forall\, Q_{t-1}} P(Q_{t-1}, q_t = j, Y_t)$: probabilidad del camino óptimo hasta t que genera la secuencia de observaciones $Y_t = y_1, \ldots, y_t$, terminando en el estado j.

1) **Inicialización**:

$$\phi_1(i) = b_i(y_1)\pi_i$$

$$\psi_1(i) = 0$$

2) **Recursión**:

$$\phi_t(j) = b_j(y_t) \max_{1 \leq i \leq N} a_{ij}\phi_{t-1}(i)$$

$$\psi_t(j) = \underset{1 \leq i \leq N}{\operatorname{argmax}}\, \phi_{t-1}(i) a_{ij}$$

Camino óptimo $S^* = s_1^*, \ldots, s_t^*$ con $s_t^* = j \Rightarrow \psi_t(j)$ nos da s_{t-1}^*, el estado anterior a j.

$\phi_t(j)$: probabilidad de ese camino.

3) **Backtracking**:

Primero el último estado:

$$q_T^* = \underset{1 \leq j \leq N}{\operatorname{argmax}}\, \phi_T(j)$$

Después:

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

Entrenamiento de la HMM

- Dado Y, estimar $\lambda = \{\{i\}, \{a_{ij}\}, \{b_j(y_t)\}\}$, con $i, j = 1, \ldots, N$, $t = 1, \ldots, T$

Algoritmo Baum-Welch

- Basado en EM

1) **Inicialización** de μ_k, Σ_k (generales, para todas las observaciones) y $\{a_{ij}\}$

2) **Paso E**:

Recursión backward

$\alpha_t(j) = P(y_1, \ldots, y_t, q_t = j)$: probabilidad de la secuencia de observaciones hasta t y que el estado actual sea j

$\alpha_t(j) = \sum_{i=1}^{N} b_j(y_t) a_{ij} \alpha_{t-1}(i)$

Recursión forward

$\beta_t(i) = P(y_t + 1, \ldots, y_T | q_t = i)$: probabilidad de que se dé la secuencia de observaciones desde $t+1$ hasta el final, dado que el estado actual es i

$\beta_t(i) = \sum_{j=1}^{N} b_j(y_{t+1}) a_{ij} \beta_{t+1}(j)$

$\gamma_t(i) = P(q_t = i | Y)$: probabilidad de que el estado en el instante t sea i, dada la secuencia de observaciones

$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$

$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | Y)$: probabilidad de estar en el estado i en el instante t y pasar al estado j, dada la secuencia de observaciones

$\xi_t(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j) b_j(y_{t+1}) a_{ij}}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$

3) **Paso M**: reestimar los parámetros

$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) y_t}{\sum_{t=1}^{T} \gamma_t(j)}$: media para el estado j: promedio ponderado de las observaciones con las probabilidades de que cada observación se dé en el estado j

$\Sigma_j = \frac{\sum_{t=1}^{T} (y_t - \mu_j)^T (y_t - \mu_j) \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$: varianza para el estado j: varianza ponderada con las probabilidades de que cada observación se dé en el estado j

$a_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}$: probabilidad de transición del estado i al j: suma de las probabilidades de pasar del estado i al estado j en todos los instantes, dividido por la probabilidad total de transición desde el estado i hacia otro estado

4) Calcular **log-likelihood**

$$P(X) = \sum_{i=1}^{N} \alpha_{t_0}(i) \beta_{t_0}(i)$$

$$LL = \log P(X)$$

5) Repetir 2, 3, 4 hasta que el log-likelihood no cambie.

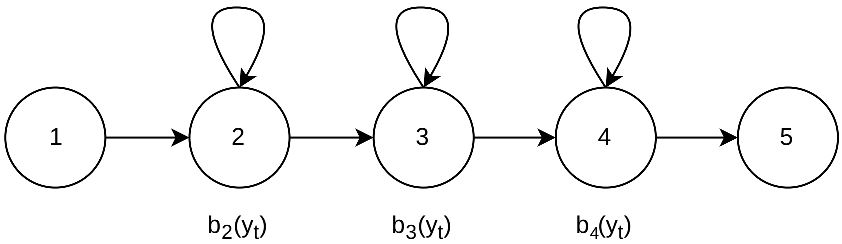
Aplicación al reconocimiento de habla

1) Set de entrenamiento: frases grabadas y sus transcripciones fonéticas

2) Parametrización: coeficientes MFCC por cada ventana

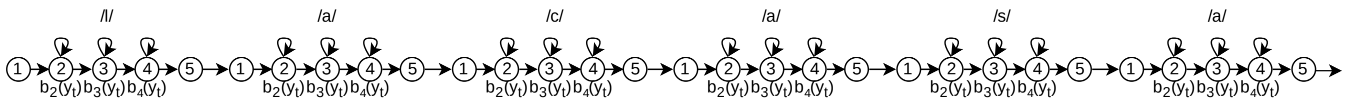
3) HMM:

- 3 estados emisores por fonema + 2 estados no emisores (inicial y final) para unirlos:



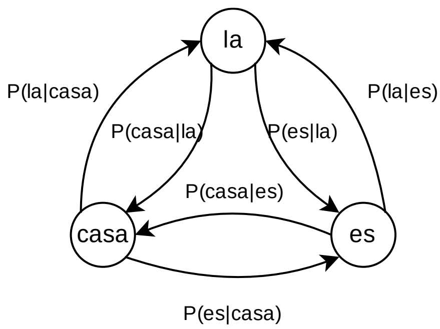
- Por cada frase de entrenamiento:

  "la casa es linda":



- No hace falta segmentar la frase en fonemas "a mano"

4) Modelo de lenguaje:



Para las transiciones entre palabras: bigramas:

$$P(w_4|w_3, w_2, w_1) \simeq P(w_4|w_3)$$

$$P(w_4|w_3) = \frac{N(w_3 w_4)}{N(w_3)}$$

Puede haber muchas probabilidades iguales a 0 $\Rightarrow$ suavizado: Backoff, Kneser-Ney

5) Reconocimiento: Viterbi en la red completa: **Palabras (modelo de lenguaje) -> HMM de Fonemas concatenados para cada palabra**