Reconocimiento Automático del Habla
2023-2024

Fundamentos en RAH:
Modelos Ocultos de Markov
(Fundamentos)

MIARFID-RAH mcastro@dsic.upv.es

Modelos Ocultos de Markov

Procesos estocásticos a dos niveles

Los Modelos Ocultos de Markov son procesos estocásticos a dos niveles:

- el de los **estados**, que *no* es observable,
- y el de **eventos físicos**, que *sí* es observable.

Decimos que los eventos físicos se **emiten** en los estados.

Ejemplos

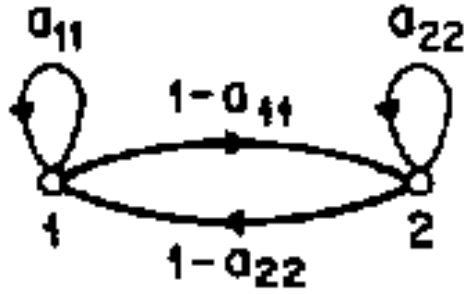
El modelo cara/cruz Estamos en una habitación con una cortina que nos impide ver a una persona que tiene dos monedas cargadas. Esa persona lanza repetidamente una u otra (aleatoriamente) al aire y nos informa de si sale cara (H, por *head*) o cruz (T, por *tail*).

Un ejemplo de secuencia de observaciones es éste:

HHTTHTHHTTHTHTHHTTTHTHHTHHT

No tenemos forma de saber cada H o T de qué moneda es. ¿Cómo explicamos/modelamos las secuencias que observamos?

Podríamos modelar este escenario con un HMM. Cada estado correspondería a una moneda. El hecho de que la persona cambia de moneda aleatoriamente correspondería al hecho de cambiar de estado. La asunción markoviana impone que la probabilidad de coger una u otra moneda dependa exclusivamente de la última moneda lanzada.



P(H) = P_1 P(H) = P_2

P(T) = $1-P_1$ P(T) = $1-P_2$

O = H H T T H T H H T T H...

S = 2 1 1 2 2 2 1 2 2 1 2...

En cada estado hay una probabilidad de "emitir" H o T.

Para completar el modelo, hemos de conocer el valor de 4 parámetros:

- a_{11},
- a_{22},
- $P_1(H)$,
- $P_2(H)$.

Sólo son 4 porque el resto se puede deducir de ellos:

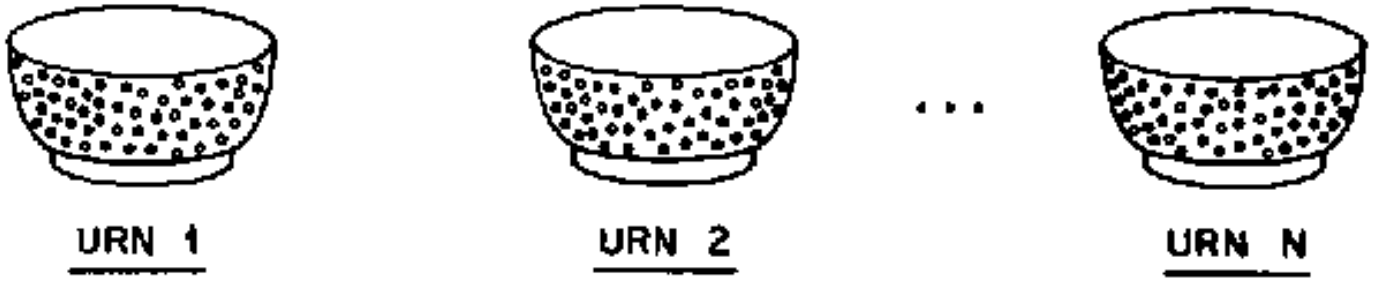
- $a_{12} = 1 - a_{11}$,

- $a_{21} = 1 - a_{22}$,
- $P_1(T) = 1 - P_1(H)$,
- $P_2(T) = 1 - P_2(H)$.

¿Podemos estimar esos cuatro valores observando únicamente el resultado de una o más secuencias de lanzamientos?

El modelo urna/bola Tratemos con una situación más complicada. Hay N urnas con bolas de M colores distintos. Un diablo actúa tras una cortina que oculta las urnas de acuerdo con el siguiente procedimiento:

1. escoge una urna inicial,
2. extrae una bola y nos la muestra,
3. restituye la bola a la urna de la que la extrajo,
4. selecciona una nueva urna de acuerdo con ciertas probabilidades asociadas a la urna de la que extrajo al última bola,
5. vuelve al paso 2 si aún no se ha extraído un número predeterminado de bolas..



URN 1

$P(\text{RED}) = b_1(1)$
$P(\text{BLUE}) = b_1(2)$
$P(\text{GREEN}) = b_1(3)$
$P(\text{YELLOW}) = b_1(4)$
$\vdots$
$P(\text{ORANGE}) = b_1(M)$

URN 2

$P(\text{RED}) = b_2(1)$
$P(\text{BLUE}) = b_2(2)$
$P(\text{GREEN}) = b_2(3)$
$P(\text{YELLOW}) = b_2(4)$
$\vdots$
$P(\text{ORANGE}) = b_2(M)$

URN N

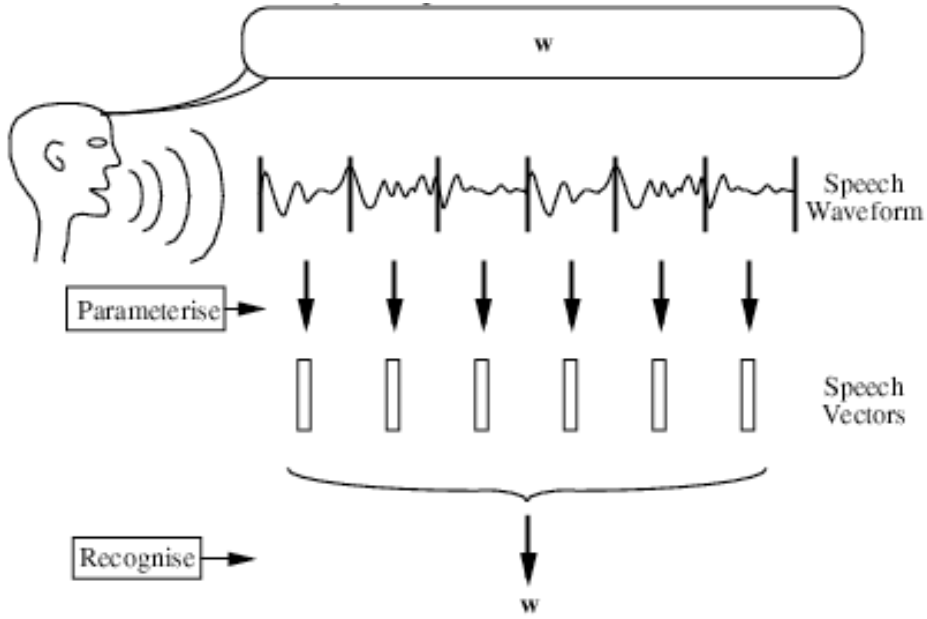
$P(\text{RED}) = b_N(1)$
$P(\text{BLUE}) = b_N(2)$
$P(\text{GREEN}) = b_N(3)$
$P(\text{YELLOW}) = b_N(4)$
$\vdots$
$P(\text{ORANGE}) = b_N(M)$

Si sólo podemos ver las bolas que extrae el diablo, ¿cómo modelamos el sistema?

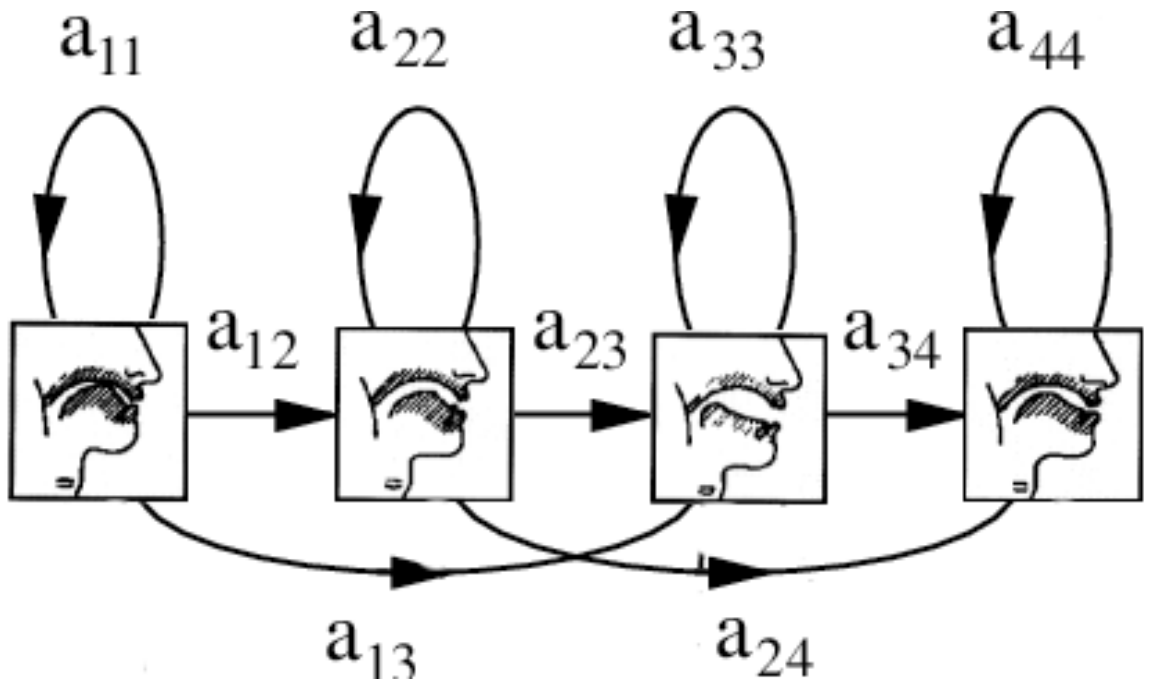
Un HMM modela este escenario con:

- un **estado** por urna,
- una **probabilidad de transición** entre estados que supedita la elección del siguiente estado a (únicamente) el estado actual,
- una probabilidad de "emisión de bolas de color" asociada a cada urna (dada por la proporción de bolas de cada color en la urna),
- una **probabilidad** de que cada urna sea la "urna **inicial**".

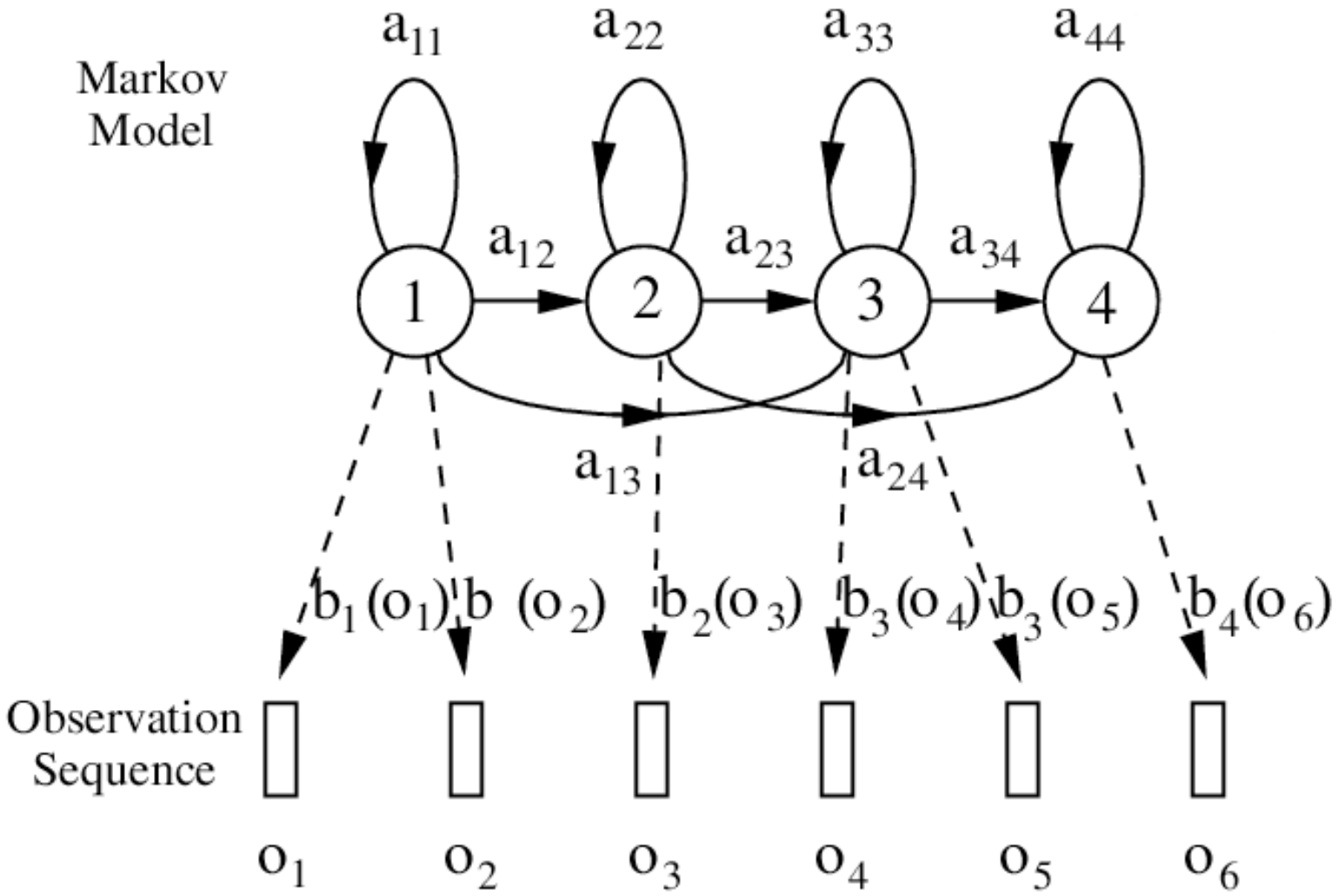
El habla


w
Speech
Waveform
Parameterise
Speech
Vectors
Recognise
w

Asimilamos estados a configuraciones del aparato fonador. De una configuración se puede pasar a otra de acuerdo con ciertas reglas probabilísticas.



En cada estado se puede emitir un sonido de entre los de un conjunto con cierta distribución de probabilidad.


Markov Model
a11
a22
a33
a44
a12
a23
a34
1
2
3
4
a13
a24
b1(o1)
b (o2)
b2(o3)
b3(o4)
b3(o5)
b4(o6)
Observation Sequence
o1
o2
o3
o4
o5
o6

Formalismo

Un modelo oculto de Markov se caracteriza por:

- Un conjunto de N **estados**.
- Un espacio de **características observables**. Asumiremos por el momento que es un conjunto de M símbolos (Modelos Ocultos de Markov **discretos**).
- Una matriz $A = \{a_{ij}\}$ de **probabilidad de transición** entre estados.

  El elemento a_{ij} indica la probabilidad de transitar al estado j si se está en el estado i.

  Se cumple:

$$a_{ij} \geq 0, \qquad 1 \leq i, j \leq N;$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad 1 \leq i \leq N.$$

- Una distribución de **probabilidad de emisión** de símbolos en cada estado: $B = \{b_j(k)\}$, para $1 \leq j \leq N$, $1 \leq k \leq M$.

  El elemento $b_j(k)$ indica la probabilidad de emitir una observación k en el estado j.

  Se cumple:

$$b_i(k) \geq 0, \qquad 1 \leq i \leq N, \quad 1 \leq k \leq M;$$

$$\sum_{k=1}^{M} b_i(k) = 1, \qquad 1 \leq i \leq N.$$

- Una distribución de **probabilidad de estado inicial**: $\Pi = \{\pi_i\}$, para $1 \leq i \leq N$.

  El elemento π_i indica la probabilidad de que el primer estado en el que estamos (y desde el que se emite el primer símbolo) sea el estado i.

  Se cumple:

$$\pi_i \geq 0, \qquad 1 \leq i \leq N;$$

$$\sum_{i=1}^{N} \pi_i = 1, \qquad 1 \leq i \leq N.$$

Un modelo queda descrito, pues, con $\lambda = (A, B, \Pi)$.

Los tres problemas básicos de los HMM

Problema 1: El problema de la evaluación Dada una secuencia de observaciones $O = o_1 o_2 \ldots$ y un modelo $\lambda = (A, B)$, ¿cómo calculamos eficientemente la probabilidad de que λ genera a O, es decir, $P(O \mid \lambda)$?

Problema 2: El problema de la descodificación Dada una secuencia de observaciones $O = o_1, o_2, \ldots, o_T$ y un modelo $\lambda = (A, B)$, ¿cómo calculamos la secuencia de estados $x_0, x_1, \ldots x_{T+1}$ que mejor "explica" las observaciones?

Problema 3: El problema del entrenamiento Dada una secuencia de observaciones O, ¿cómo estimamos los parámetros de $\lambda = (A, B)$ para que $P(O \mid \lambda)$ sea máxima?

Problema 1: El problema de la evaluación

Dada una secuencia de observaciones $O = o_1, o_2, \ldots, o_T$ y un modelo $\lambda = (A, B)$, ¿Cómo calculamos eficientemente $P(O \mid \lambda)$?

Podríamos enumerar todas las secuencias posibles de $T + 2$ estados (que emiten un total de T símbolos), calcular para cada una de ellas la probabilidad de generar O y sumar todas estas probabilidades ponderándolas por la probabilidad de cada secuencia.

Tomemos una secuencia cualquiera:

$$X = x_0 x_1 \ldots x_{T+1}.$$

1. La probabilidad de observar O dados X y un modelo λ es

$$P(O \mid X, \lambda) = \prod_{t=1}^{T} P(o_t \mid x_t, \lambda) = \prod_{t=1}^{T} b_{x_t}(o_t).$$

2. La probabilidad de la secuencia de estados X es:

$$P(X \mid \lambda) = a_{x_0 x_1} \cdot a_{x_1 x_2} \cdot \ldots \cdot a_{x_T x_{T+1}}.$$

3. La probabilidad conjunta es:

$$P(O, X \mid \lambda) = P(O \mid X, \lambda) \cdot P(X \mid \lambda)$$

Finalmente, la probabilidad de O dado λ es:

$$\begin{aligned} P(O \mid \lambda) &= \sum_X P(O \mid X, \lambda) \cdot P(X \mid \lambda) \\ &= \sum_{x_0 x_1 \cdots x_{T+1}} a_{x_0 x_1} \cdot b_{x_1}(o_1) \cdot a_{x_1 x_2} \cdot \ldots \cdot a_{x_{T-1} x_T} \cdot b_{x_T}(o_T) \cdot a_{x_T x_{T+1}} \\ &= \sum_{x_0 x_1 \cdots x_{T+1}} a_{x_0 x_1} \cdot \prod_{t=1}^{T} b_{x_t}(o_t) \cdot a_{x_t x_{t+1}}. \end{aligned}$$

Un cálculo directo de la fórmula es $O(TN^T)$

¿Se puede calcular más eficientemente?

El algoritmo forward Definimos la "**probabilidad forward**" como la probabilidad de observar la secuencia parcial $o_1 o_2 \cdots o_t$ y estar en el estado j de un modelo λ en el instante t:

$$\alpha_j(t) = P(o_1 o_2 \cdots o_t, x_t = j \mid \lambda).$$

El valor de $\alpha_j(t)$ puede calcularse recursivamente:

$$\alpha_j(t) = \begin{cases} 1, & \text{si } t = 0 \text{ y } j = 1; \\ \left(\sum_{i=1}^{N-1} \alpha_i(t-1) \cdot a_{ij} \right) b_j(o_t), & \text{si } 1 < t \leq T \text{ y } 1 < j < N; \\ \sum_{i=1}^{N-1} \alpha_i(T) \cdot a_{iN}, & \text{si } t = T+1 \text{ y } j = N; \\ 0, & \text{en otro caso.} \end{cases}$$

La probabilidad de observar una secuencia O dado el modelo λ es

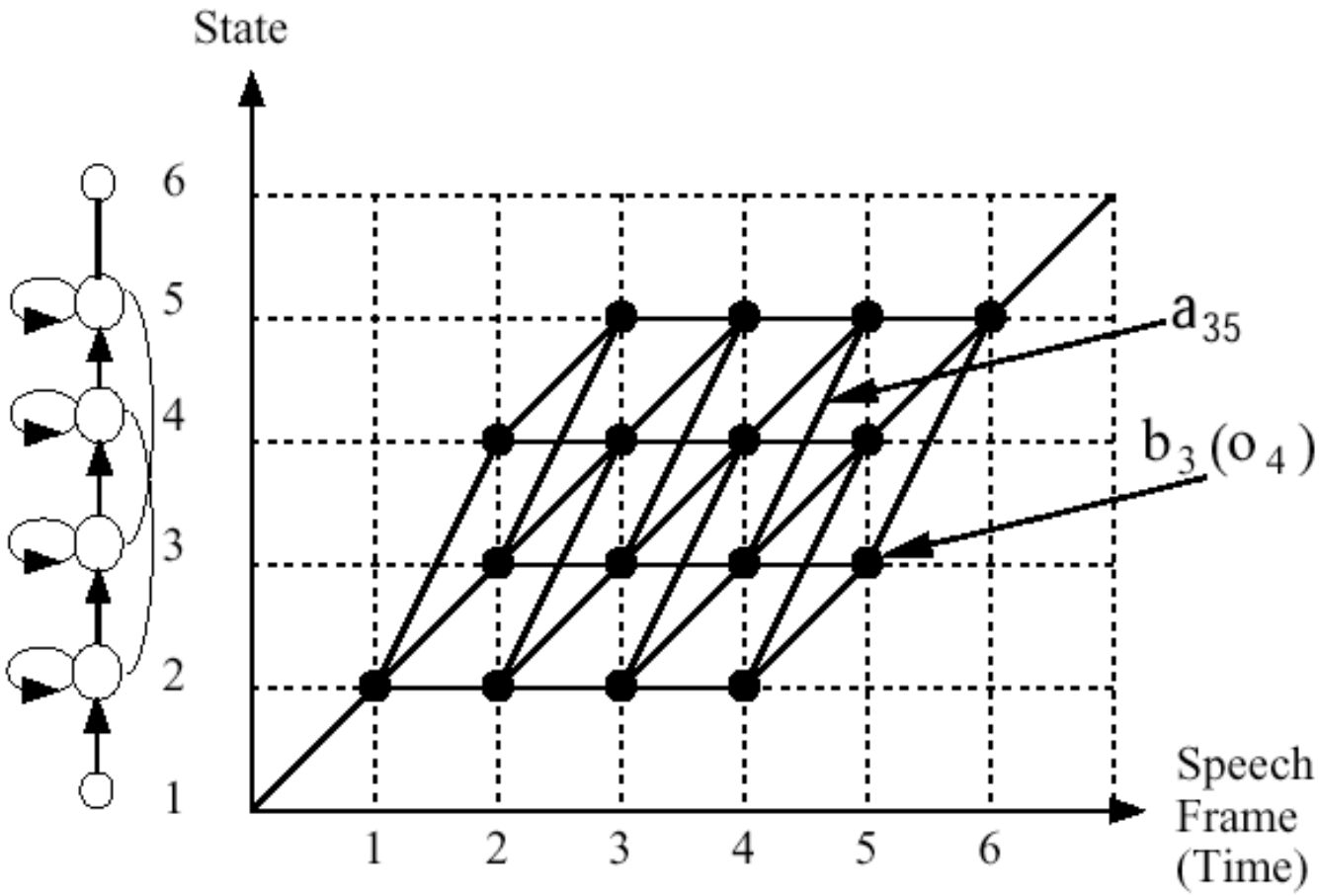
$$P(O \mid \lambda) = \alpha_N(T+1).$$

La recursión de α puede expresarse también así:

$$\alpha_j(t) = \begin{cases} 1, & \text{si } t = 0 \text{ y } j = 1; \\ \left(\sum_{i:a_{ij} \neq 0} \alpha_i(t-1) \cdot a_{ij} \right) b_j(o_t), & \text{si } 1 \leq t \leq T \text{ y } 1 < j < N; \\ \sum_{i:a_{iN} \neq 0} \alpha_i(T) \cdot a_{iN}, & \text{si } t = T+1 \text{ y } j = N; \\ 0, & \text{en otro caso.} \end{cases}$$

Esta versión hace explícito que sólo hemos de considerar los estados i "predecesores" de cada estado j.

El valor de alfa puede calcularse eficientemente mediante un algoritmo iterativo (programación dinámica) con un coste temporal $O(N^2T)$.



Trellis para HMM con topología de Bakis.

El procedimiento backward También podemos definir la "**probabilidad backward**", $\beta_i(t)$. Es la probabilidad de la observación parcial $o_{t+1}o_{t+2}\dots o_T$ partiendo del estado i del modelo λ en el instante t:

$$\beta_i(t) = P(o_{t+1}o_{t+2}\dots o_T \mid x_t = i, \lambda).$$

Tenemos que:

$$P(O \mid \lambda) = \beta_1(0).$$

Se puede calcular β recursivamente:

$$\beta_i(t) = \begin{cases} a_{iN}, & \text{si } t = T; \\ \displaystyle\sum_{j=2}^{N-1} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_j(t+1), & \text{si } 0 \leq t < T \text{ y } 1 \leq i < N; \\ 0, & \text{en otro caso.} \end{cases}$$

O, alternativamente,

$$\beta_i(t) = \begin{cases} a_{iN}, & \text{si } t = T; \\ \sum_{j:a_{ij}\neq 0} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_j(t+1), & \text{si } 0 \leq t < T \text{ y } 1 \leq i < N; \\ 0, & \text{en otro caso.} \end{cases}$$

Las ecuaciones se pueden resolver iterativamente en $O(TN^2)$.

Problema 2: El problema de la secuencia óptima de estados

¿Qué entendemos por secuencia óptima de estados? Es la secuencia de estados que con mayor probabilidad ha producido la secuencia de observaciones, es decir, la que maximiza

$$P(X \mid O, \lambda).$$

Puntuación de Viterbi Dada una secuencia de observaciones O y un modelo λ, definimos $\phi_t(j)$ como la probabilidad de la secuencia de estados que finaliza en el estado j, y más probablemente genera el prefijo $o_1 o_2 \cdots o_t$:

$$\phi_t(j) = \max_{x_0 x_1 \cdots x_t} P(x_0 x_1 \cdots x_t, x_t = j, o_1 \cdots o_t \mid \lambda).$$

$\phi_j(t)$ es la **puntuación de Viterbi**.

Podemos calcular recursivamente $\phi_t(j)$ (Viterbi):

$$\phi_j(t) = \begin{cases} 1, & \text{si } t = 0 \text{ y } j = 1; \\ \max\limits_{1 \le i < N} (\phi_i(t-1) \cdot a_{ij}) \cdot b_j(o_t), & \text{si } 1 \le t \le T \text{ y } 1 < j < N; \\ \max\limits_{1 \le i < N} \phi_i(T) \cdot a_{iN}, & t = T+1 \text{ y } j = N; \\ 0, & \text{en otro caso.} \end{cases}$$

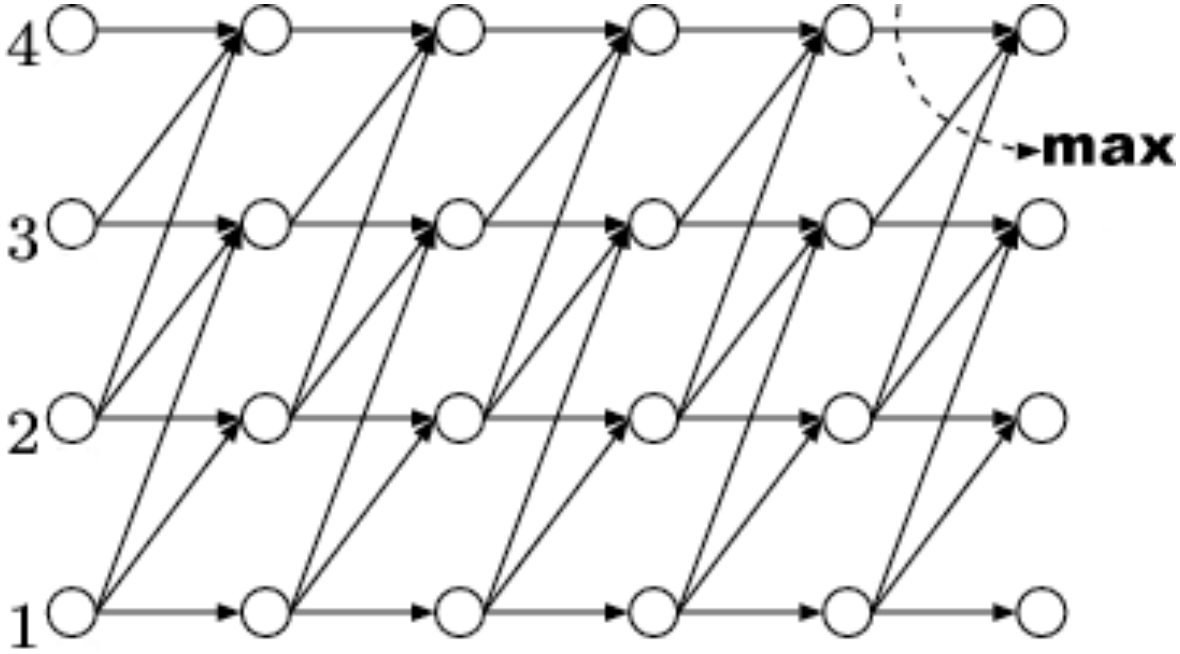
La secuencia más probable dada una secuencia de observaciones O tiene probabilidad

$$\max_X P(X \mid O, \lambda) = \phi_N(T).$$

Nuevamente, podemos reescribir la ecuación recursiva para hacer explícito que la recursión

sólo considera estados predecesores:

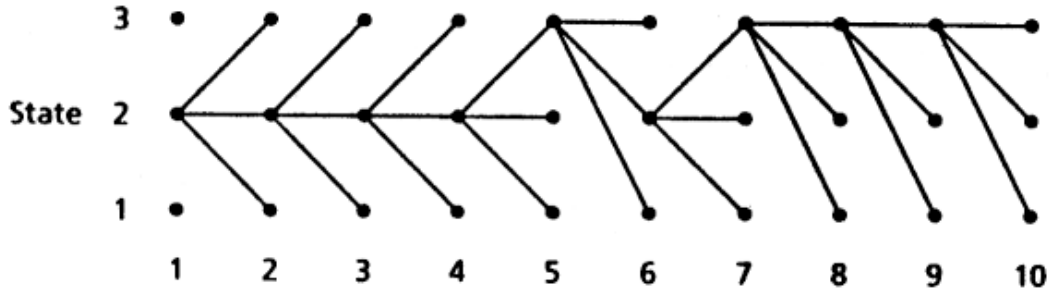
$$\phi_j(t) = \begin{cases} 1, & \text{si } t = 0 \text{ y } j = 1; \\ \max\limits_{i:a_{ij}\neq 0} (\phi_i(t-1) \cdot a_{ij}) \cdot b_j(o_t), & \text{si } 1 \leq t \leq T \text{ y } 1 < j < N; \\ \max\limits_{i:a_{iN}\neq 0} \phi_i(T) \cdot a_{iN}, & t = T+1 \text{ y } j = N; \\ 0, & \text{en otro caso.} \end{cases}$$



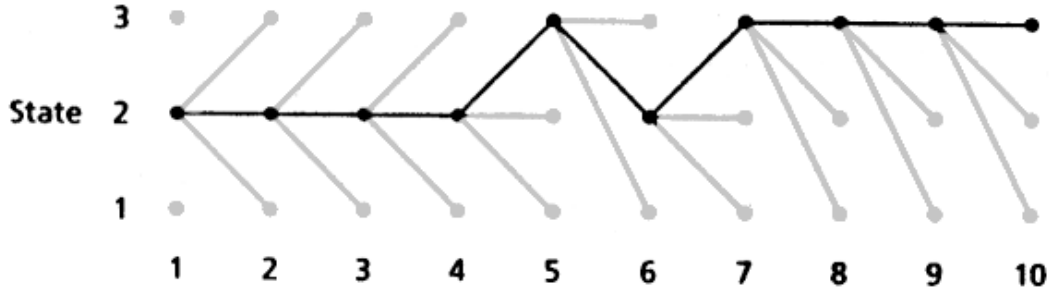
Trellis para la puntuación de Viterbi.

Recuperación de la secuencia de estados No nos interesa tanto $\phi_j(t)$ como la propia secuencia de estados que produjo el valor óptimo.

Resulta sencillo extraer la secuencia óptima de estados haciendo *backtracing*:



Punteros atrás en un Trellis.



Camino óptimo desde el nodo $(10, 3)$ *siguiendo los punteros hacia atrás en un Trellis.*

Una ventaja de la puntuación de Viterbi La secuencia de estados que **maximiza la probabilidad** de generar una secuencia determinada es la misma que **maximiza el logaritmo de dicha probabilidad**.

Trabajar con logaritmos de probabilidad (**logprob**) transforma los productos en sumas, operaciones menos costosas computacionalmente y que, además, evitan la comisión de *underflows*:

$$\phi'_j(t) = \begin{cases} 0, & \text{si } t = 0 \text{ y } j = 1; \\ \max\limits_{i:a_{ij}\neq 0} (\phi'_i(t-1) + \log a_{ij}) + \log b_j(o_t), & \text{si } 1 \leq t \leq T \text{ y } 1 < j < N; \\ \max\limits_{i:a_{iN}\neq 0} \phi'_i(T) + \log a_{iN}, & t = T+1 \text{ y } j = N; \\ \infty, & \text{en otro caso.} \end{cases}$$

Nota: es frecuente plantear la recursión como minimización y trabajar con el logaritmo cambiado de signo.

Problema 3: El problema del entrenamiento

Se trata del más difícil de los tres problemas.

No hay forma conocida de determinar analíticamente qué valores de $\lambda = (A, B)$ maximizan la probabilidad de $P(O \mid \lambda)$.

Reestimación de parámetros Queremos reestimar $A = \{a_{ij}\}$ y $B = \{b_j(k)\}$ de modo tal que se maximice

$$P(O \mid \lambda).$$

Si partimos de A y B arbitrarios es posible obtener una estimación $\bar{A} = \{\bar{a}_{ij}\}$ y $\bar{B} = \{\bar{b}_j(k)\}$ que aumente o deje igual el valor de $P(O \mid \lambda)$ mediante un procedimiento iterativo.

$$
\begin{aligned}
\bar{a}_{ij} &= \frac{\text{número esperado de veces que se usa la transición } i,j \text{ al generar } O}{\text{número esperado de veces que se visita } i \text{ al generar } O} \\
&= \frac{\sum_{t=0}^{T} P(x_t = i, x_{t+1} = j, O \mid \lambda)}{\sum_{t=0}^{T+1} P(x_t = i, O \mid \lambda)}, \\
\bar{b}_j(k) &= \frac{\text{número esperado de veces que se observa el símbolo } k \text{ en } j \text{ al generar } O}{\text{número esperado de veces que se visita } j \text{ al generar } O} \\
&= \frac{\sum_{t=0}^{T+1} P(x_t = j, o_t = k, O \mid \lambda)}{\sum_{t=0}^{T+1} P(x_t = j, O \mid \lambda)}.
\end{aligned}
$$

Observa que

$$
\begin{aligned}
P(x_t = i, O \mid \lambda) &= \alpha_i(t) \cdot \beta_i(t), \\
P(x_t = i, x_{t+1} = j, O \mid \lambda) &= \alpha_i(t) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_j(t+1), \\
P(x_t = j, o_t = k \mid \lambda) &= \alpha_j(t) \cdot \beta_j(t) \cdot \delta(o_t, k), \\
P(O \mid \lambda) &= \alpha_N(T),
\end{aligned}
$$

donde $\delta(o_t, k)$ es 1 si $o_t = k$ y 0 en caso contrario.

Tenemos que, para $1 < i, j < N$,

$$\bar{a}_{ij} = \frac{\sum_{t=0}^{T-1} \alpha_i(t) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_j(t+1)}{\sum_{t=0}^{T} \alpha_i(t) \cdot \beta_i(t)}$$

Y en los estados inicial $(i = 1)$ y final $(j = N)$:

$$\begin{aligned} \bar{a}_{1j} &= \frac{\alpha_j(1)\beta_j(1)}{\alpha_N(T+1)} \\ \bar{a}_{iN} &= \frac{\alpha_i(T)\beta_i(T)}{\sum_{t=0}^{T} \alpha_i(t)\beta_i(t)} \end{aligned}$$

El número esperado de veces que se observa el símbolo k en el estado j es

$$\sum_{t=0}^{T+1} P(x_t = j, o_t = k, O \mid \lambda) = \sum_{t=0}^{T+1} P(x_t = j, O \mid \lambda) \cdot \delta(o_t, k).$$

Así pues,

$$\bar{b}_j(k) = \frac{\sum_{t=0}^{T+1} \alpha_j(t) \cdot \beta_j(t) \cdot \delta(o_t, k)}{\sum_{t=0}^{T} \alpha_j(t) \cdot \beta_j(t)}$$

Si tenemos un modelo $\lambda = (a, b)$ y estimamos con él el modelo $\bar{\lambda} = (\bar{A}, \bar{B})$, $\bar{\lambda}$ satisface

$$P(O \mid \lambda) \leq P(O \mid \bar{\lambda}).$$

El procedimiento iterativo de entrenamiento: el algoritmo Baum-Welch

1. Se parte de λ inicial arbitrario.
2. Calcular $\bar{\lambda}$ a partir de λ.
3. Si $P(O \mid \lambda) = P(O \mid \bar{\lambda})$ (o está "suficientemente" próximo), terminar; si no, hacer que λ sea igual a $\bar{\lambda}$ y volver al paso 1.

Inicialización del entrenamiento La calidad de la solución hallada por el método de Baum-Welch depende del modelo inicial. ¿Cómo podemos encontrar un buen modelo inicial?

1. ¿Cómo estimamos A?

   La experiencia indica que una inicialización uniforme de A proporciona buenos resultados.

2. ¿Cómo estimamos B?

   Hemos de tener una buena estimación inicial de $b_j(k)$, para $1 \leq j \leq N$ y $1 \leq k \leq M$.

   Podemos segmentar la entrada en tantas partes como estados tiene el modelo y asignar inicialmente cada segmento a un estado. La segmentación puede ser:

   - En partes iguales.
   - Manualmente.
   - Con un procedimiento automático como el "segmental k-means" con agrupamiento.

Entrenamiento con conjuntos de secuencias Hemos desarrollado el método suponiendo que sólo hay una secuencia de observaciones O.

Una buena estimación requiere un conjunto grande de secuencias de observaciones $\mathcal{O} = \{O^1, O^2, \ldots, O^S\}$.

Queremos maximizar ahora

$$P(\mathcal{O} \mid \lambda) = \prod_{s=1}^{S} P(O^s \mid \lambda).$$

Basta con reestimar los parámetros considerando todas las secuencias de observaciones en cada iteración.

$$\begin{aligned}
\bar{a}_{ij} &= \frac{\sum_{s=1}^{S} P(O^s \mid \lambda) \sum_{t=1}^{T_s} \alpha_i^s(t) a_{ij} b_j(o_{t+1}^k) \beta_j^s(t+1)}{\sum_{s=1}^{S} P(O^s \mid \lambda) \sum_{t=1}^{T_s} \alpha_i^s(t) \beta_i^s(t)}, \\
\bar{b}_j(k) &= \frac{\sum_{s=1}^{S} P(O^s \mid \lambda) \sum_{t=1}^{T_s - 1} \alpha_i^s(t) \beta_i^s(t) \delta(o_t, k)}{\sum_{s=1}^{S} P(O^s \mid \lambda) \sum_{t=1}^{T_s} \alpha_i^s(t) \beta_i^s(t)}.
\end{aligned}$$

El problema de la falta de datos

Al estimar $b_j(k)$ es posible que no se observe nunca el símbolo k en el estado j y se infiera que $b_j(k) = 0$.

El modelo resultante asigna probabilidad 0 a cualquier secuencia que suponga la emisión del símbolo k en j.

Soluciones:

- Conseguir **más datos** para entrenamiento.

  Difícil y/o costoso.

- **Reducir el número de estados** del modelo.

  Puede haber razones físicas que desaconsejen eliminar estados.

- **Suavizar la estimación** fijando umbrales de probabilidad mínima y reescalando las probabilidades para satisfacer las restricciones estocásticas.

$$\hat{b}_j(k) = \begin{cases} s \cdot b_j(k), & \text{si } b_j(k) \geq \nu; \\ \nu, & \text{en otro caso.} \end{cases}$$

El valor de s se fija para que $\sum_{k=1}^{M} b_j(k) = 1$.

Un problema similar puede surgir con la estimación de los a_{ij}: si uno de ellos llega a 0, ya no puede tomar un valor distinto por el procedimiento de reestimación estudiado. Es preciso fijar algún umbral mínimo.