Modelos ocultos de Markov (HMM)

Georgina Flesia

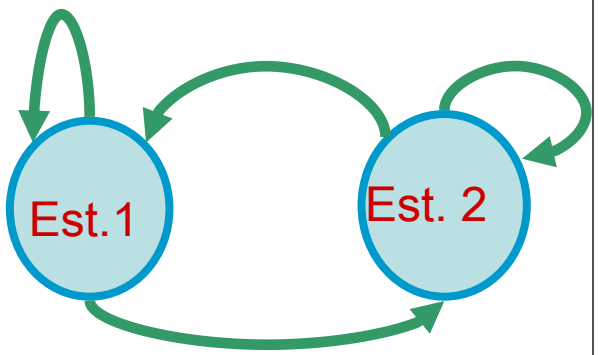
Hidden Markov models

- Belief nets son una forma muy poderosa de representar dependencias
- Si las dependencias tienen temporalidad, es decir se observa un proceso que se desarrolla en el tiempo, pueden observarse estados que estan muy influenciados por el estado anterior
- A pesar de que generan modelos mas complejos, las mismas ideas reaparecen
- HMM tienen una cantidad de parametros cuyos valores explican los patrones de cada categoria.
- Luego, un nuevo patron puede ser clasificado por el modelo por el estado que mejor explica el patron.

Definición de un modelo oculto de Markov

Un modelo oculto de Markov (HMM) es un conjunto finito de estados.

Las transiciones entre estados están dadas por un conjunto de probabilidades de transición.



En cualquier estado particular, la observación puede ser generada, de acuerdo a la distribución de probabilidades de emisión.

Sólo el resultado observable, no el estado, es visible a un observador externo por lo que los estados están “ocultos”.

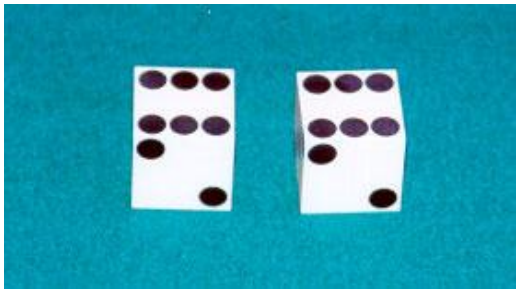
Ejemplos de modelos ocultos de Markov

- Texto escrito por Shakespeare que en alguna partes ha sido editado por un mono
- Un casino tiene dos dados, uno cargado y el otro no. Alterna entre ambos
- Una secuencia de DNA con segmentos que codifican y otros que no.

Ejemplos

Caso	Observables	Estado oculto
Texto	alfabeto	Shakespeare/mono
Dado	1-6	dado justo (F) /dado cargado (L)
DNA	A,C,G,T (bases)	codifica/no-codifica

Ejemplo: El Casino “ocasionalmente” deshonesto



Un casino tiene dos dados:

- Dado “balanceado”
  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- Dado cargado
  P(1) = P(2) = P(3) = P(5) = 1/10
  P(6) = 1/2

El casino alterna entre el dado balanceado y el cargado una vez cada 20 turnos

Juego:

1. Apostamos $1
2. Tiramos (siempre con un dado balanceado)
3. El Casino tira (tal vez con un dado balanceado, tal vez con uno cargado)
4. EL número mas alto gana $2

Pregunta # 1 – Evaluación

Dada

Una secuencia de tiradas del casino

12455264621461461361366616646616366163661636165156151151461235623 44

PREGUNTA

¿Cuan probable es esta secuencia, dado nuestro modelo de como opera el casino?

Este es el problema de **EVALUACIÓN** en HMMs

Pregunta # 2 – Decodificación

Dada

Una secuencia de tiradas del casino

12455264621461461361366616646616366163661636161515615115146123562344

PREGUNTA

¿Que partes de la secuencia fueron generadas por el dado cargado y cuales por el dado justo?

Este es el problema de **DECODIFICACIÓN** en HMMs

Pregunta # 3 – Aprendizaje

Dada

Una secuencia de tiradas del casino

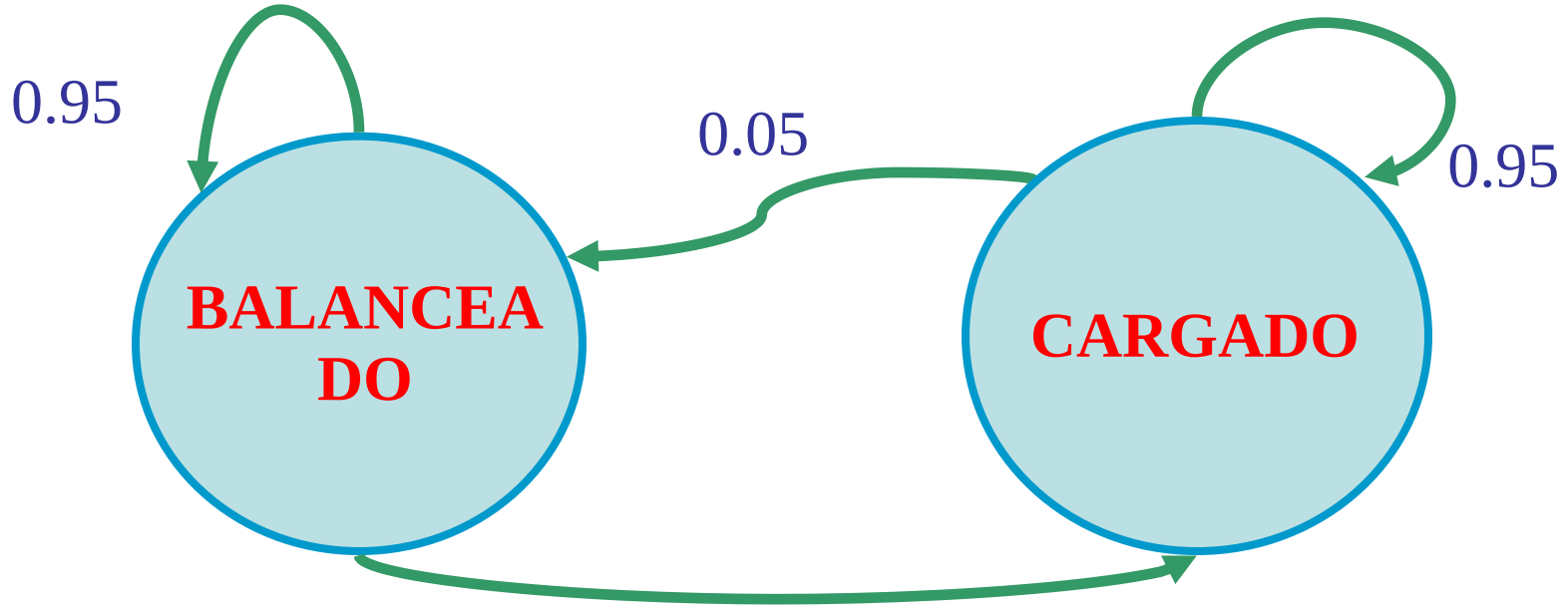
12455264621461461361366616646616366163661636161515615115146123562344

PREGUNTA

¿Cómo está cargado el dado cargado? ¿Cada cuanto cambia el casino entre el dado cargado y el justo?

Este es el problema de **APRENDIZAJE** en HMMs

El HMM del casino deshonesto



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

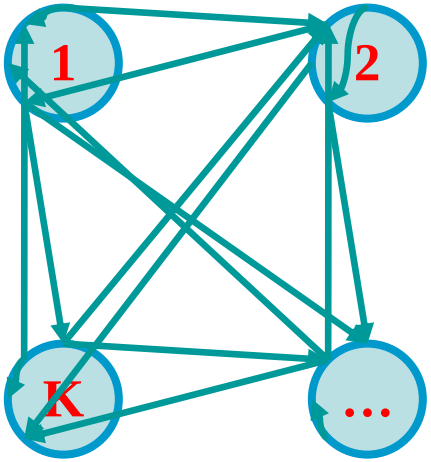
Definición de un modelo oculto de Markov

Definición: Un modelo oculto de Markov (HMM)

- Alfabeto $\Sigma = \{ b_1, b_2, \ldots, b_M \}$ M simbolos
- Conjunto de estados $S = \{ 1, ..., K \}$ K estados
- Probabilidades de transición entre dos estados cualesquiera

  a_{ij} = prob. de transición del estado i al estado j

  $a_{i1} + \ldots + a_{iK} = 1$, para todos los estados i = 1…K



- Probabilidades iniciales a_{0i}

  $a_{01} + \ldots + a_{0K} = 1$

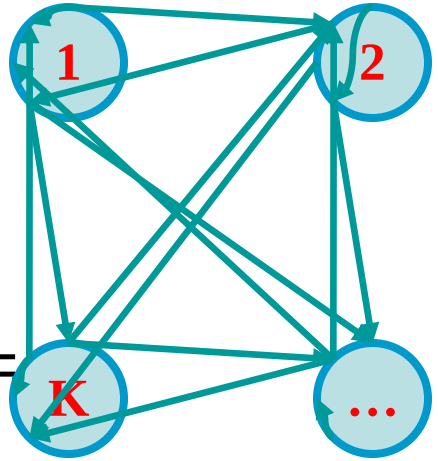
- Probabilidades de emisión dentro de cada estado

  $e_i(b) = P(x_i = b \mid s_i = k)$

  $e_i(b_1) + \ldots + e_i(b_k) = 1$, para todos los estados i = 1…K

Un HMM no tiene memoria

En cada paso de tiempo t,
Lo único que afecta los futuros
estados es el estado actual s_t



$P(s_{t+1} = k \mid \text{“cualquier cosa que pasó”}) =$

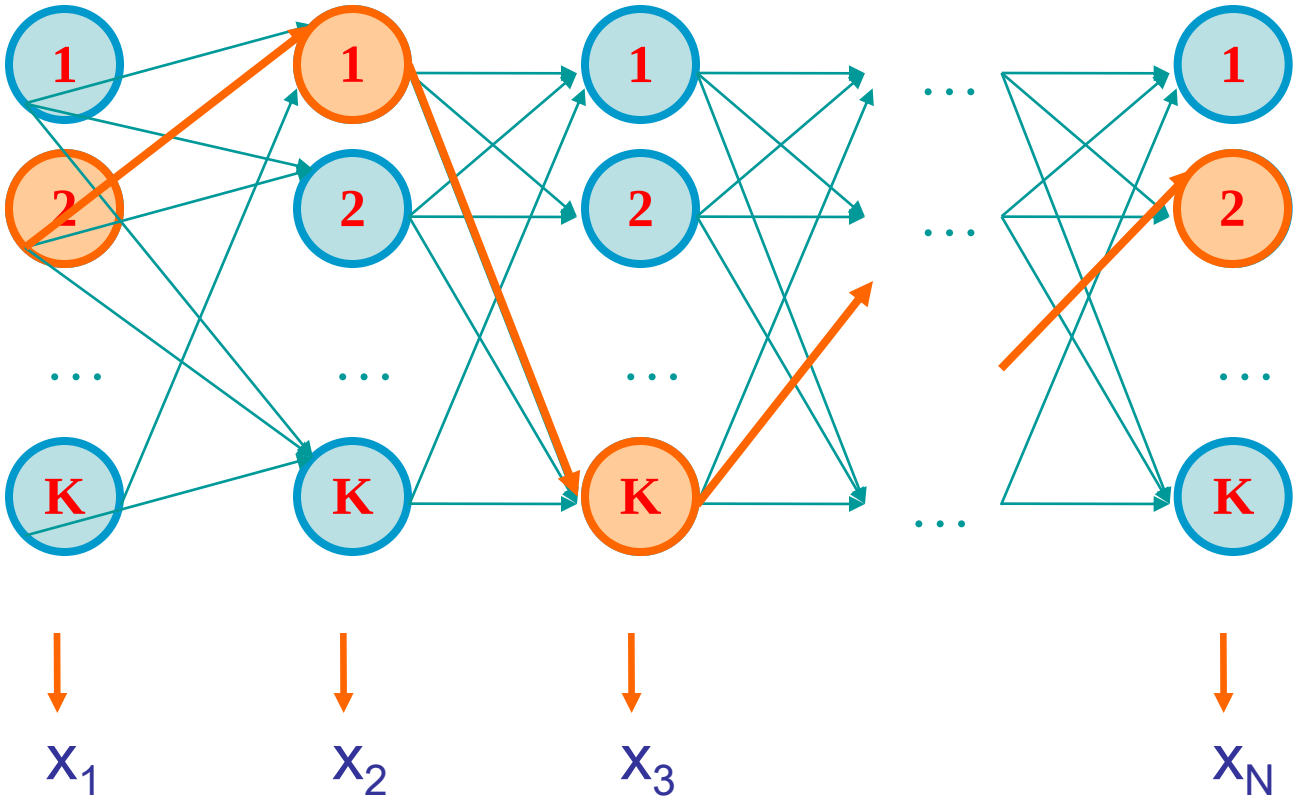
$P(s_{t+1} = k \mid s_1, s_2, \ldots, s_t, x_1, x_2, \ldots, x_t) =$

$P(s_{t+1} = k \mid s_t)$

Secuencias de observables y camino de estados

Dada una secuencia de observaciones x = $x_1 \ldots\ldots x_N$,

Y la secuencia de estados s = s_1, ……, s_N

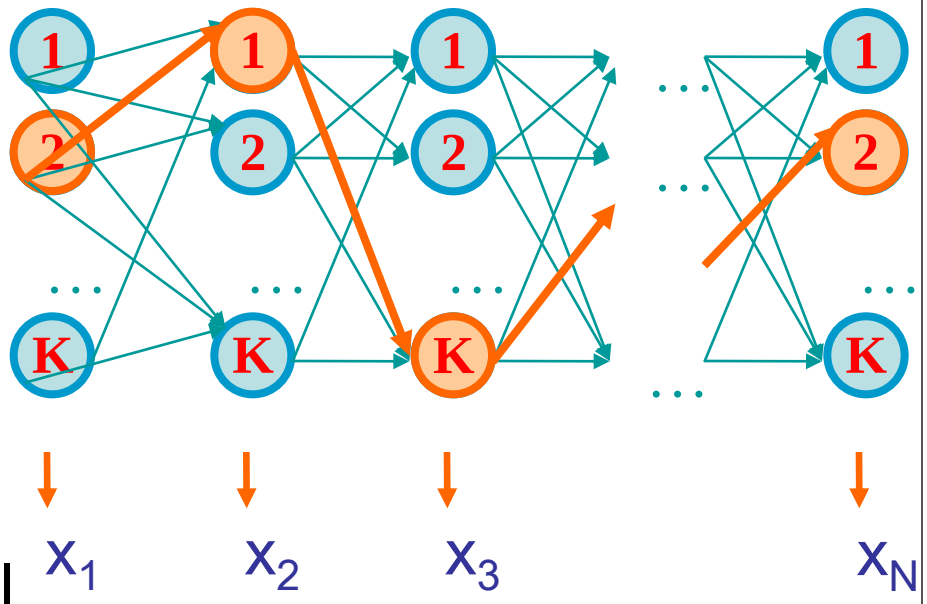


Verosimilitud de un camino de estados

Dados una secuencia $x = x_1 \ldots\ldots x_N$

Y un camino de

Estados $s = s_1, \ldots\ldots, s_N,$



Para averiguar cuan probable es el

Camino: (dado nuestro HMM)

$$P(x, s) = P(x_1, \ldots, x_N, s_1, \ldots\ldots, s_N)$$

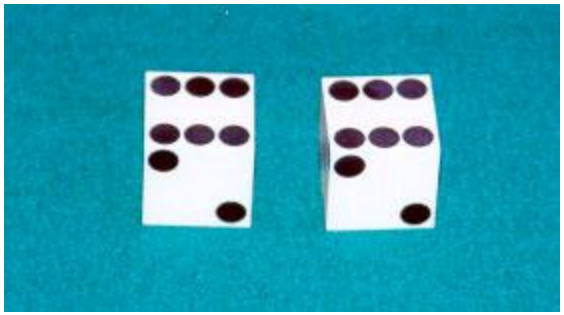
$$= P(x_N, s_N \mid s_{N-1})\, P(x_{N-1}, s_{N-1} \mid s_{N-2}) \ldots\ldots P(x_2, s_2 \mid s_1)\, P(x_1, s_1)$$

$$= P(x_N \mid s_N)\, P(s_N \mid s_{N-1}) \ldots\ldots P(x_2 \mid s_2)\, P(s_2 \mid s_1)\, P(x_1 \mid s_1)\, P(s_1)$$

$$= a_{0s1}\, a_{s1s2} \ldots\ldots a_{sN-1sN}\, e_{s1}(x_1) \ldots\ldots e_{sN}(x_N)$$

Ejemplo: El casino deshonesto

Imaginemos que la secuencia de dados es:



x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

Entonces, cuan probable es

S= Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

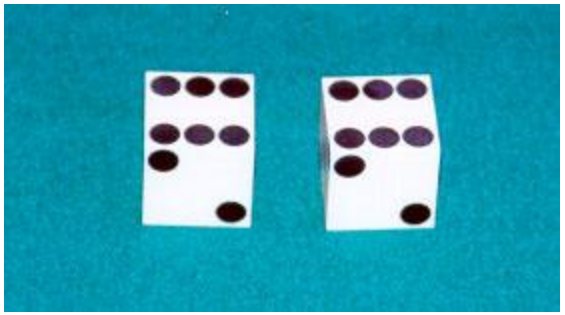
(imaginamos inicialmente $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

$½ \times$ P(1 | Fair) P(Fair | Fair) P(2 | Fair) P(Fair | Fair) … P(4 | Fair) =

$½ \times (1/6)^{10} \times (0.95)^9 = .000000005211586472 11 = 0.5 \times 10^{-9}$

Ejemplo: El casino deshonesto

Entonces, la probabilidad de que el dado sea Balanceado en toda la corrida es solamente 0.521×10^{-9}



Pero…¿Cual es la probabilidad de

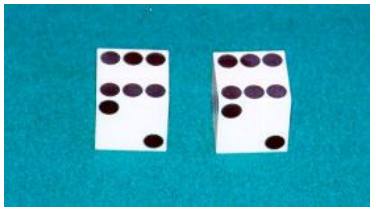
S= Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$\frac{1}{2} \times P(1 \mid Loaded)\ P(Loaded, Loaded) \ldots P(4 \mid Loaded) =$

$\frac{1}{2} \times (1/10)^8 \times (1/2)^2\ (0.95)^9 = .00000000078781176215 = 7.9 \times 10^{-10}$

Entonces, despues de todo es 6.59 veces mas probable que el dado haya sido siempre balanceado, a que haya sido siempre cargado.

Ejemplo: El casino deshonesto



Imaginemos que la secuencia de dados es:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

Entonces, cuan probable es

S= F, F, …, F?

$\frac{1}{2} \times (1/6)^{10} \times (0.95)^{9} = 0.5 \times 10^{-9}$,

igual que antes

¿Cuan probable es S= L, L, …, L?

$\frac{1}{2} \times (1/10)^{4} \times (1/2)^{6} (0.95)^{9} = .00000049238$

$= 0.5 \times 10^{-7}$

Entonces, es 100 veces mas probable que el dado esté cargado

Las tres grandes preguntas sobre HMMs

- Evaluación

  DADO un HMM M, y una secuencia x,

  ENCUENTRE Prob[x | M]

- Decodificación

  DADO un HMM M, y una secuencia x,

  ENCUENTRE la secuencia de estados s que maximiza P[x, s | M]

- Aprendizaje

  DADOS un HMM M, con probs transición/emisión desconocidas,
  y una secuencia x,

  ENCUENTRE los parámetros $\theta = (e_i(.), a_{ij})$ que maximizan P[x | θ]

Que no nos confunda la notación!

P[x | M]: La probabilidad de que la secuencia x haya sido generada por el modelo

El modelo es:

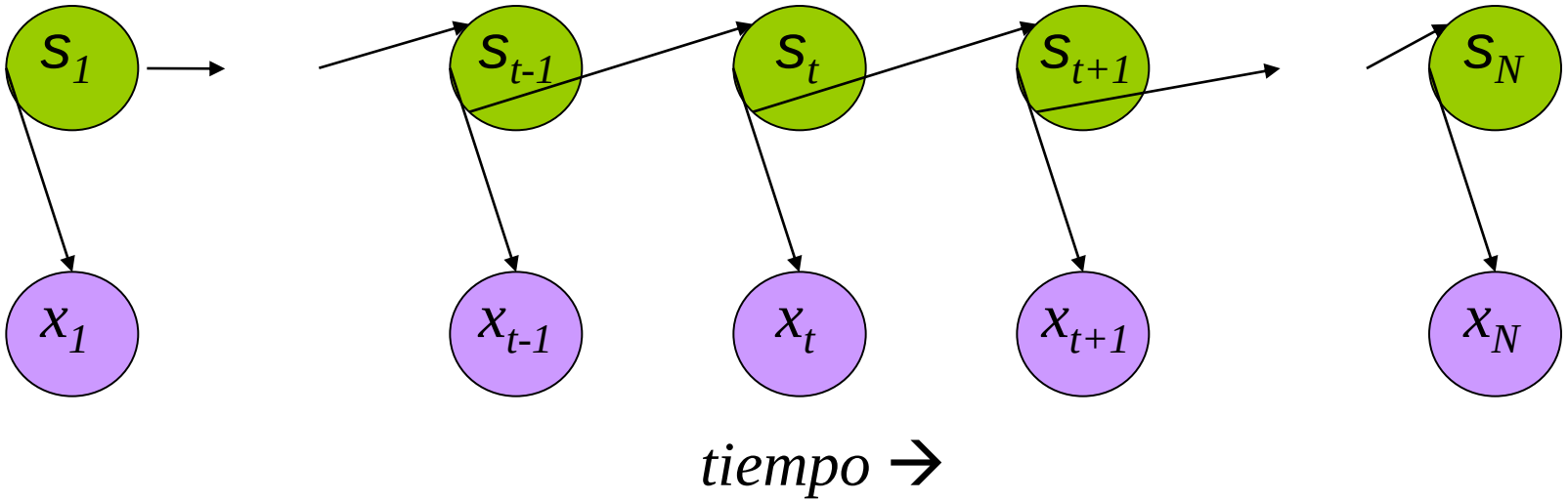
arquitectura (#estados, etc)+ parámetros $\theta = a_{ij}, e_i(.)$

Entonces, P[x | θ], y P[x] son lo mismo, en lo que se refiere a la arquitectura y al modelo entero

De igual forma, P[x, s | M] y P[x, s] son lo mismo

En el problema de APRENDIZAJE siempre escribimos P[x | θ] para enfatizar que buscamos el θ que maximiza P[x | θ]

HMM: Línea temporal

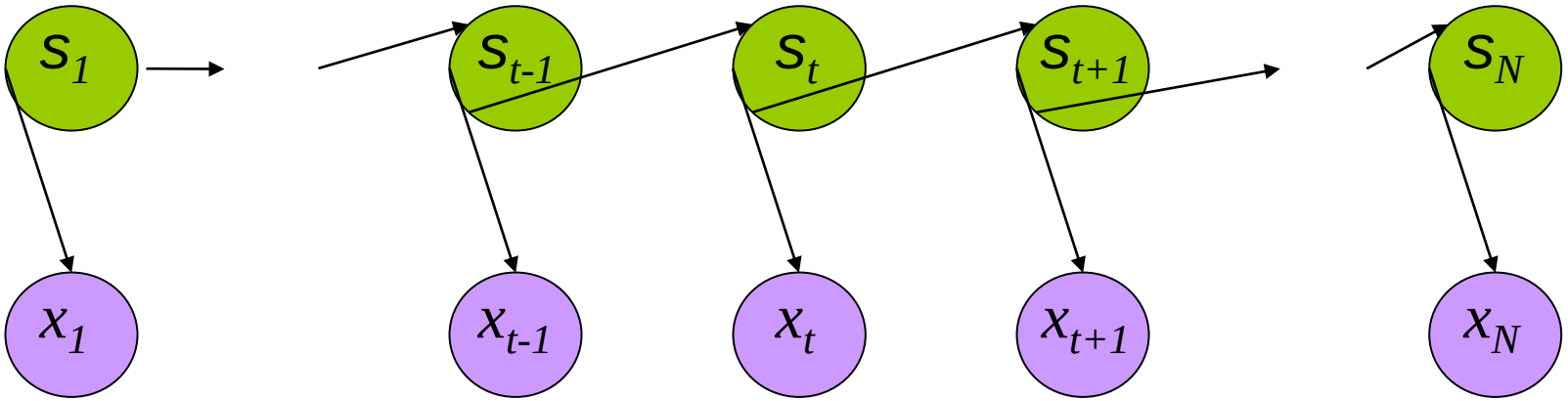


- Las flechas indican dependencias estocásticas.
- Los s's son ***estados ocultos,*** acada uno depende solamente del estado previo.
  - Proceso Markoviano.
- Los x's son ***observaciones,*** dependen solamente del estado oculto correspondiente.

Problema 1: Evaluación

Encontrando la verosimilitud de una secuencia generada por el modelo

Probabilidad de una observación



Dados una secuencia de observaciones y un modelo, vimos que era fácil calcular la probabilidad de obtener dicha secuencia...simplemente multiplicamos las probabilidades individuales...

Si no conocemos cual fue la secuencia de estados, entonces una estimacion sería calcular las probabilidades para los distintos caminos posibles y elegir la máxima

Evaluacion HMM

- Para una dada secuencia de longitud N tenemos alrededor de 2N cálculos.
- Si K es el número de estados en el gráfico.
- Hay K^N secuencias de estados posibles.
- Complejidad : $O(2NK^N)$.
- Pero esto es fuerza bruta...podemos intentar algo mejor.
- Es el algoritmo llamado HMM Forward

El algoritmo Forward

Queremos calcular

P(x) = probabilidad de la secuencia x, dado el HMM

Si sumamos sobre todos las posibles formas de generar x:

$$P(x) = \Sigma_s P(x, s) = \Sigma_s P(x \mid s) P(s)$$

Para evitar sumar sobre un número de caminos exponencial en s, definimos la probabilidad forward

$$\alpha_k(i) = P(x_1 \ldots x_i, s_i = k)$$

El algoritmo Forward

Probabilidad forward:

$\alpha_l(i) = P(x_1 \ldots x_i, s_i = l)$

$= \Sigma_{s1 \ldots si-1} P(x_1 \ldots x_{i-1}, s_1, \ldots, s_{i-1}, s_i = l)\, e_l(x_i)$

$= \Sigma_k \Sigma_{s1 \ldots si-2} P(x_1 \ldots x_{i-1}, s_1, \ldots, s_{i-2}, s_{i-1} = k)\, a_{kl}\, e_l(x_i)$

$= e_l(x_i) \Sigma_k \alpha_k(i-1)\, a_{kl}$

Relación de recurrencia entre $\alpha_l(i)$ y $\alpha_k(i-1)$

El algoritmo Forward

Podemos calcular $\alpha_k(i)$ para todo k, i, usando programación dinámica

Inicialización:

$\alpha_0(0) = 1$

$\alpha_k(0) = 0$, para todo $k > 0$

Iteración:

$\alpha_l(i) = e_l(x_i) \Sigma_k \alpha_k(i-1) a_{kl}$

Terminación:

$P(x) = \Sigma_k \alpha_k(N) a_{k0}$

Donde, a_{k0} es la probabilidad de que el estado que termina sea k

Problema 2: Decodificando

Encontrando la mejor secuencia de estados

Ejemplo de decodificacion

Dada una secuencia de observaciones ***X***, encuentre la secuencia de est. ***s*** .

(1) Texto Shakespeare (*s*) vs mono (*m*)

x = *..aefjkuhrgnandshefoundhappinesssdmcamoe…*
s = *..mmmmmmsssssssssssssssssssssssssssmmmmmm*…

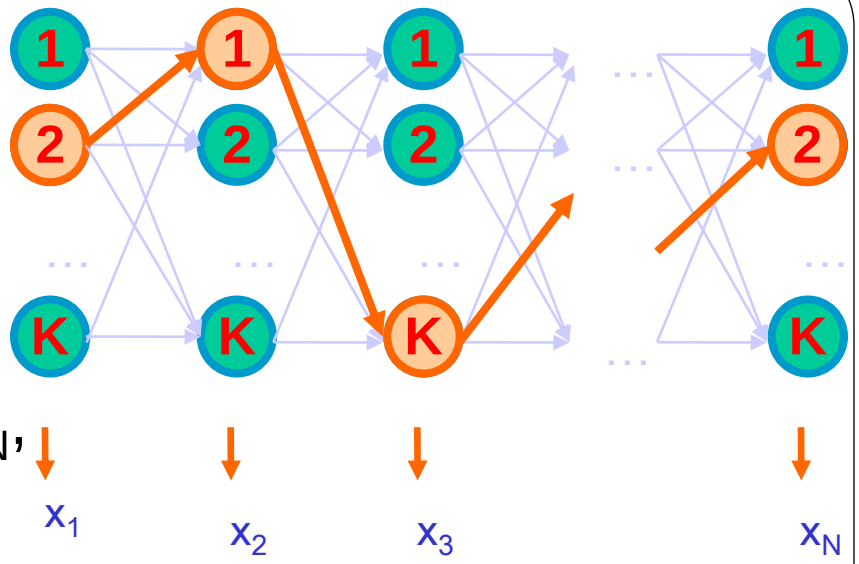
(2) Dado fair (*F*) vs loaded (*L*)

x = *…13245564436636345566116345621661124536…*
s = *…LLLLLLLLLLLLFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLL…*

(3) DNA coding (*C*) vs non-coding (*N*)

x = *…AACCTTCCGCGCAATATAGGTAACCCCGG*…
s = *…NNCCCCCCCCCCCCCCCCCCCNNNNNNNN*…

Decodificando



DADO $x = x_1 x_2 \ldots\ldots x_N$

Queremos encontrar $= s_1, \ldots\ldots, s_N$,
tal que P[x, s] esté maximizado

$s^* = \text{argmax}_s\ P[\ x, s\ ]$

Podemos usar programación dinámica

Sea $V_k(i) = \max_{\{s1,\ldots,si-1\}} P[x_1 \ldots x_{i-1}, s_1, \ldots, s_{i-1}, x_i, s_i = k]$

= Probabilidad de la secuencia de estados mas verosimil que termina en el estado $s_i = k$

El algoritmo de Viterbi

Input: $x = x_1 \ldots\ldots x_N$

Inicialización:

$V_0(0) = 1$ (0 es la posición inicial)

$V_k(0) = 0$, para todo $k > 0$

Iteración:

$V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$

$ptr_j(i) = \arg (\max_k a_{kj} V_k(i-1))$

Terminación:

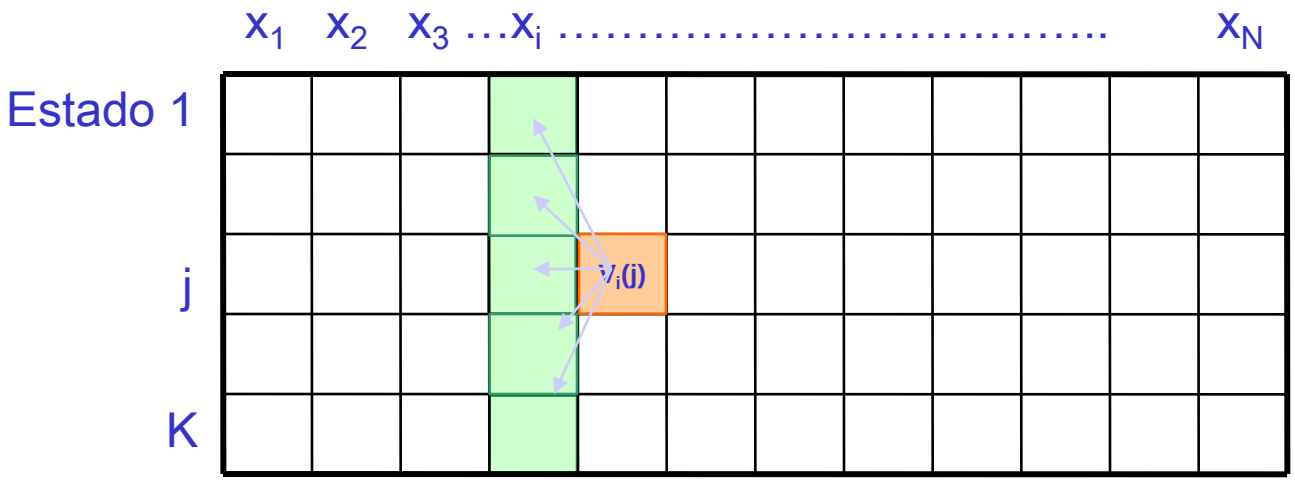
$P(x, s^*) = \max_k V_k(N)$

Rastreo:

$s_N^* = \text{argmax}_k V_k(N)$

$s_{i-1}^* = ptr_{si} (i)$

El algoritmo de Viterbi

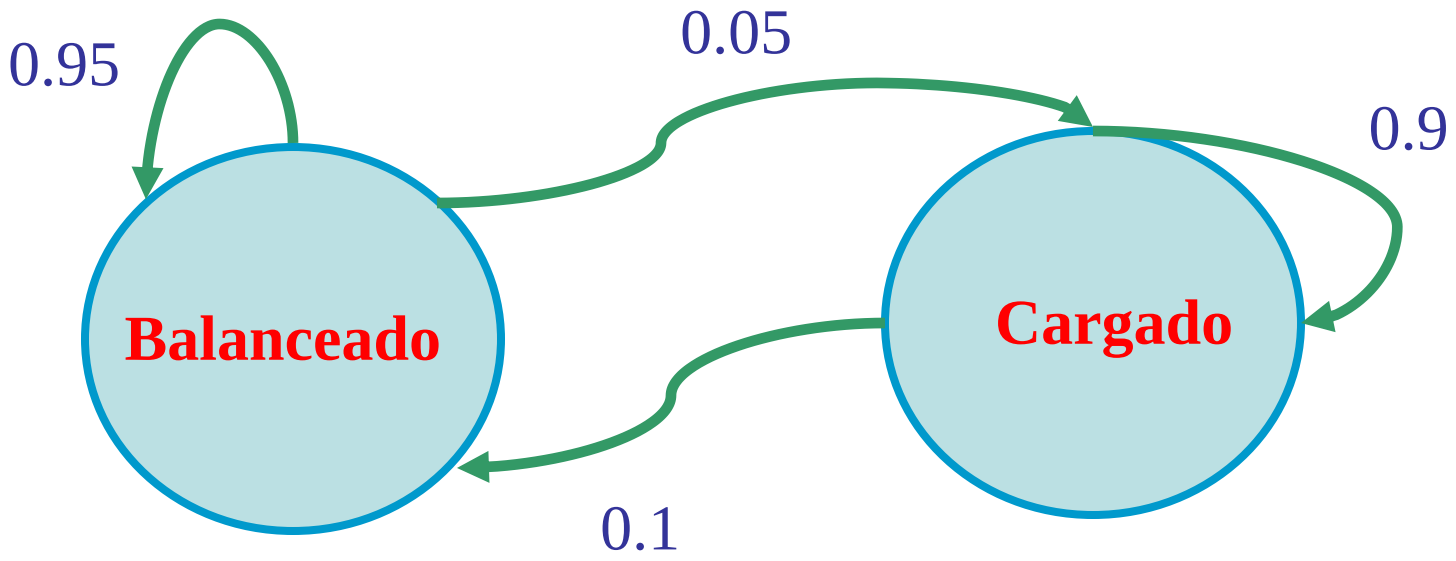


Es similar a "alinear" un conjunto de estados de una secuencia

Complejidad temporal: $O(K^2N)$

Complejidad espacial: $O(KN)$

Casino II (leve cambio en el uso del dado)



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

Ejemplo: El casino deshonesto II

- Dada la secuencia de dados:
  x = 3, 1, 5, 6, 1, 6, 4, 6, 4, 4, …

$V_0(0) = 1$ (estado Begin), $V_k(0) = 0$, para todo $k > 0$

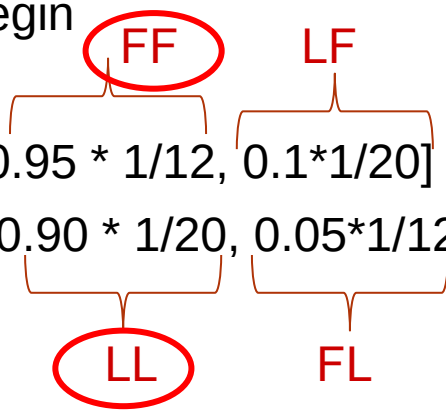
$V_j(1) = e_j(x_i) \times \max_k a_{kj} V_k(i\text{-}1)$

$V_F(1) = 1/6 * \max_k [½ * 1, 0.95*0, 0.1*0] = 1/12$

$V_L(1) = 1/10 * \max_k [½ * 1, 0.9*0, 0.05*0] = 1/20$

$ptr_F(1)$ = Estado Begin

$ptr_L(1)$ = Estado Begin



$V_F(2) = 1/6 * \max_k [0.95 * 1/12, 0.1*1/20] = 1/6*\max(0.079, 0.05) = 0.013$

$V_L(2) = 1/10 * \max_k [0.90 * 1/20, 0.05*1/12] = 1/10*\max(0.045, 0.041) = 0.0045$

$ptr_F(2)$ = Estado F

$ptr_L(2)$ = Estado L

Ejemplo: El casino deshonesto II

- Dada la secuencia de dados:

  x = 3, 1, 5, 6, 1, 6, 4, 6, 4, 4, …

$V_j(1) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$

FF LF

$V_F(3) = 1/6 * \max_k [0.95 * 0.013, 0.1*0.0045] = 1/6*\max(0.0123,0.00045) = 0.002$

$V_L(3) = 1/10 * \max_k [0.90 * 0.0045, 0.05*0.013] = 1/10*\max(0.004,0.00065) = 0.0004$

$ptr_F(3)$ = Estado F LL FL

$ptr_L(3)$ = Estado L

FF LF

$V_F(4) = 1/6 * \max_k [0.95 * 0.002, 0.1* 0.0004] = 1/6*\max(0.0019,0.00004) = 0.0003$

$V_L(4) = 1/2 * \max_k [0.90 * 0.0004, 0.05* 0.002] = 1/2*\max(0.00036,0.0001) = 0.00005$

$ptr_F(4)$ = Estado F LL FL

$ptr_L(4)$ = Estado L

Ejemplo: El casino deshonesto II

Dada la secuencia de dados:

x = 3, 1, 5, 6, 1, 6, 4, 6, 4, 4, …

Estado/obs.	3	1	5	…… x_N		
Begin	1					
F	0	.083	.013			
L	0	.05	.0045			

Testeo del algoritmo de Viterbi

Una secuencia de 300 tiradas de dados justos (F) o cargados (L)

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222554416665665635643243641315134651463534111264146262533 56
Die     FFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

El algoritmo de Viterbi – detalle práctico

Los errores de underflow pueden ser un problema

$P[x_1,\ldots., x_i, s_1, \ldots, s_i] = a_{0s1}\, a_{s1s2}\ldots\ldots a_{si}\, e_{s1}(x_1)\ldots\ldots e_{si}(x_i)$

Números muy chicos!

Solución: Tomar el logaritmo a todos los valores

$V_l(i) = \log e_k(x_i) + \max_k [V_k(i-1) + \log a_{kl}]$

Relación entre Forward y Viterbi

VITERBI

Inicialización:

$V_0(0) = 1$

$V_k(0) = 0$, para todo $k > 0$

Iteración:

$$V_j(i) = e_j(x_i) \ \mathbf{max_k} \ V_k(i\text{-}1)\, a_{kj}$$

Terminación:

$$P(x, \pi^*) = \mathbf{max_k} \ V_k(N)$$

FORWARD

Inicialización:

$\alpha_0(0) = 1$

$\alpha_k(0) = 0$, para todo $k > 0$

Iteración:

$$\alpha_l(i) = e_l(x_i) \ \Sigma_{\mathbf{k}} \ \alpha_k(i\text{-}1)\, a_{kl}$$

Terminación

$$P(x) = \Sigma_{\mathbf{k}} \ \alpha_k(N)\, a_{k0}$$

Motivación para el algoritmo Backward

- El algoritmo de Viterbi encuentra el camino más probable a través del modelo pero eso no es lo único que nos interesa.
- Estamos interesados en calcular, por ejemplo, el estado más probable para cualquier observación x_t, o en general queremos saber, dada la secuencia observada x, la probabilidad de que el sistema esté en el estado k en la posicion t

$$L_k(t)=P(s_t = k \mid x),$$

O la probabilidad de que el sistema esté en el estado k en la posicion t y en el estado l en la posicion t+1

$$H_{k,l}(t)=P(s_t = k, s_{t+1} = l \mid x),$$

Probabilidad forward

$$\alpha_l(i) = P(x_1 \ldots x_i, s_i = l)$$
$$= \Sigma_{s1 \ldots si-1} P(x_1 \ldots x_{i-1}, s_1, \ldots, s_{i-1}, s_i = l)\, e_l(x_i)$$
$$= \Sigma_k \Sigma_{s1 \ldots si-2} P(x_1 \ldots x_{i-1}, s_1, \ldots, s_{i-2}, s_{i-1} = k)\, a_{kl}\, e_l(x_i)$$
$$= e_l(x_i) \Sigma_k \alpha_k(i-1)\, a_{kl}$$

Relación de recurrencia entre $\alpha_l(i)$ y $\alpha_k(i-1)$

Probabilidad Backward

$$\beta_k(i) = P(x_{i+1} \ldots x_N \mid s_i = k)$$
$$= \Sigma_{si+1 \ldots sN} P(x_{i+1}, x_{i+2}, \ldots, x_N, s_{i+1}, \ldots, s_N \mid s_i = k)$$
$$= \Sigma_l \Sigma_{si+1 \ldots sN} P(x_{i+1}, x_{i+2}, \ldots, x_N, s_{i+1} = l, s_{i+2}, \ldots, \pi_N \mid s_i = k)$$
$$= \Sigma_l e_l(x_{i+1})\, a_{kl}\, \Sigma_{si+1 \ldots sN} P(x_{i+2}, \ldots, x_N, s_{i+2}, \ldots, s_N \mid s_{i+1} = l)$$
$$= \Sigma_l e_l(x_{i+1})\, a_{kl}\, \beta_l(i+1)$$

Relación de recurrencia entre $\beta_k(i)$ y $\beta_l(i+1)$

Las probabilidades $L_k(t)$ y $H_{k,l}(t)$ y $P(x)$ se calculan

$$L_k(t) = P(s_t = k \mid x) = P(s_t = k, x)/P(x)$$
$$= P(x_1 \ldots x_i, s_i = k, x_{i+1} \ldots x_N)/P(x)$$
$$= P(x_1 \ldots x_i, s_i = k)\, P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, s_i = k)/P(x)$$
$$= P(x_1 \ldots x_i, s_i = k)\, P(x_{i+1} \ldots x_N \mid s_i = k)/P(x)$$
$$= \alpha_k(i)\, \beta_k(i)\, /P(x)$$

$$H_{k,l}(t) = P(s_t = k, s_{t+1} = l \mid x) = P(s_t = k, s_{t+1} = l, x)/P(x)$$
$$= \alpha_k(i) a_{k,l}\, e_l(x_{i+1}) \beta_l(i+1)/P(x)$$

$$P(x) = \Sigma_k\, \alpha_k(i)\, \beta_k(i)$$

El algoritmo Backward

Podemos calcular $\beta_k(i)$ para todo k, i, otra vez...usando programación dinámica

Inicialización:

$\beta_k(N) = a_{k0}$, para todo k

Iteración:

$\beta_k(i) = \Sigma_l \, e_l(x_{i+1}) \, a_{kl} \, \beta_l(i+1)$

Terminación:

$P(x) = \Sigma_l \, a_{0l} \, e_l(x_1) \, \beta_l(1)$

Complejidad computacional

Complejidad de los algoritmos Forward y Backward

Tiempo: $O(K^2N)$

Espacio: $O(KN)$

Técnica práctica para evitar errores de underflow

Viterbi: suma de logaritmos

Forward/Backward: rescaleo en cada posición multiplicando por una constante

Problema 3: Aprendizaje

Reestimando los parámetros del modelo basado en los datos de entrenamiento

Estimación de parámetros – caso I

- Tenemos un conjunto de secuencias de ejemplo del tipo de las que queremos que el modelo ajuste (secuencias de entrenamiento), que suponemos independientes.
- Si conocieramos el camino de estados que recorrió el modelo, los estados no están ocultos (el modelo oculto de Markov se transforma en una cadena de Markov), en la cual los estimadores de máxima verosimilitud para las frecuencias de emisión y transición se obtienen a partir de las frecuencias de observaciones.
- Si tenemos información (biológica o física) que nos aporte información previa a la distribución de probabilidades podemos agregarsela al modelo como pseudocuentas.

Estimación de parámetros – caso II

- Objetivo: Dada una secuencia de observaciones, encuentre el modelo más probable que genere esa secuencia.
- Problema: No conocemos las frecuencias relativas de los estados ocultos visitados.
- No se conocen soluciones analíticas
- Nos acercamos a la solución por sucesivas aproximaciones.
- El problema es ahora de optimización, por lo que se pueden usar muchas heurísticas (simulated annealing, algoritmos genéticos, etc)

El algoritmo de Baum-Welch

- Este es el algoritmo de Expectation-Maximization (EM) para la estimación de parámetros.
- Aplicable a cualquier proceso estocástico, en teoría.
- Encuentra las frecuencias esperadas de los posibles valores de las variables ocultas.
- Calcula las distribuciones de maxima verosimilitud de las variables ocultas en base a las probabilidades forward y backward.
- Repite estos pasos hasta satisfacer algún criterio de “convergencia.”

Máxima probabilidad a posteriori

- Supongamos que hay una distribución de probabilidades $P(\Theta)$ de los parámetros. Entonces, por el teorema de Bayes, dada una observación x, la probabilidad a posteriori
- $$P(\Theta \mid x) = P(x|\Theta)\ P(\Theta)/P(x)$$
- Como P(x) es independiente de Θ, el mejor estimador está dado por la máxima probabilidad a posteriori

$$\Theta_{MAP} = \mathit{argmax}_{\Theta}\ P(x|\Theta)\ P(\Theta)$$

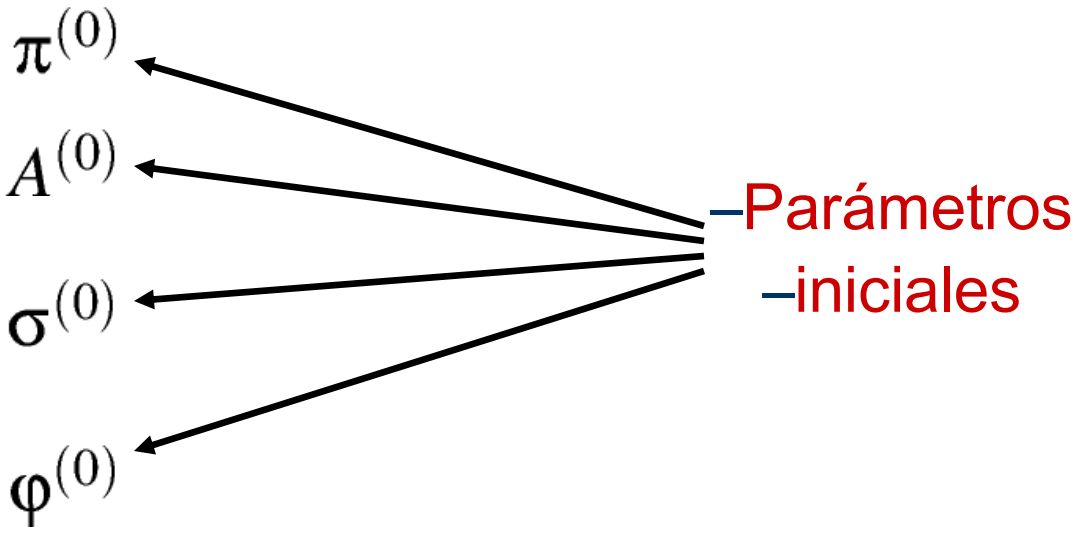
Máxima verosimilitud

- Denotemos con Θ los parámetros (probabilidades de transición y emisión) del HMM.
- Para una observación *x*, determinamos Θ usando el criterio de máxima verosimilitud
- $$\Theta_{ML} = \text{argmax}_{\Theta} P(x|\Theta)$$
- Si Θ_0 se usa para generar un conjunto de observables$\{x_i\}$, entonces

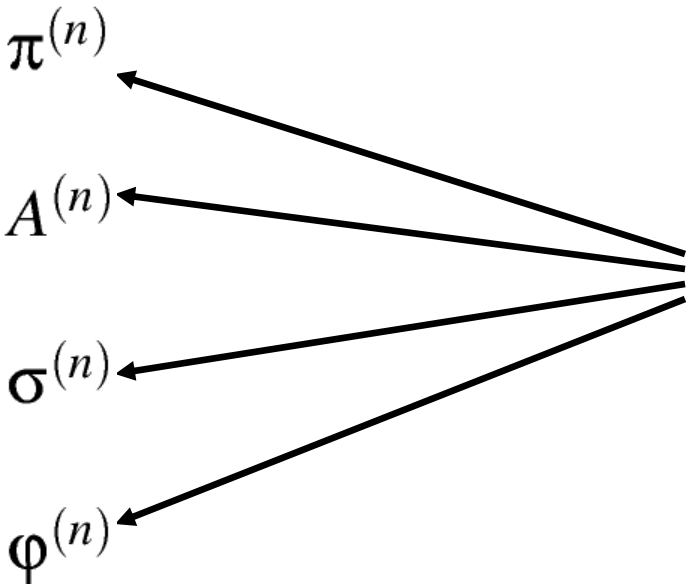
$$\Sigma_{xi} P(x_i|\Theta_0) \log P(x_i|\Theta)$$

- Se maximiza para $\Theta = \Theta_0$. de esta forma podemos encontrar Θ_{ML} por iteración (algoritmo de Baum-Welch, tambien llamado de backward-forward).

- Estimación de los parámetros de modelo:
  - Algoritmo de Baum-Welch (EM)

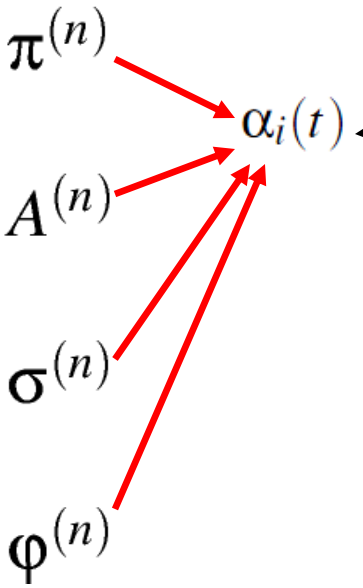


– Estimación de los parámetros de modelo:
  – Algoritmo de Baum-Welch (EM)



– Salida de la n-ésima iteración de EM

- Estimación de los parámetros de modelo:
  - Algoritmo de Baum-Welch (EM)



- Algoritmo Forward

$$\alpha_i(t) = P(O_1, O_2, \ldots, O_t, s_t = i | A^{(n)}, b^{(n)}, \pi^{(n)})$$

$$i = 1, 2, \ldots, M, \quad t = 1, 2, \ldots, T$$

regla recursiva:

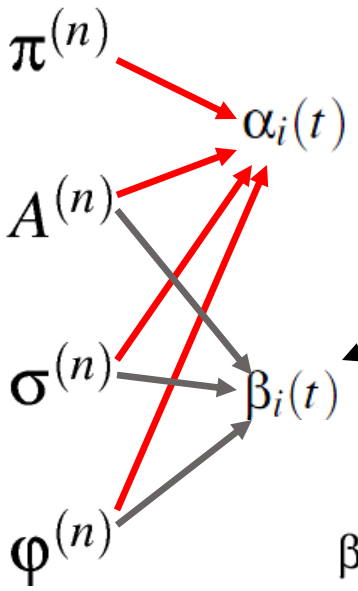
$$\alpha_i(1) = \pi_i^{(n)} b_i^{(n)}(O_1) \quad \forall i = 1, 2, \ldots, M$$

$$\alpha_j(t+1) = \left[\sum_{i=1}^{M} \alpha_i(t) a_{ij}^{(n)} \right] b_j^{(n)}(O_{t+1}) \quad \forall t = 1, 2, \ldots, T-1 \; \forall j = 1, 2, \ldots, M$$

- En cada pixel (x,y)

– Estimación de los parámetros de modelo:

– Algoritmo de Baum-Welch (EM)



– Algoritmo Backward

$$\beta_i(t) = P(O_{t+1}, \ldots, O_T | s_t = i, A^{(n)}, b^{(n)}, \pi^{(n)})$$
$$i = 1, 2, \ldots, M, \quad t = 1, 2, \ldots, T-1.$$

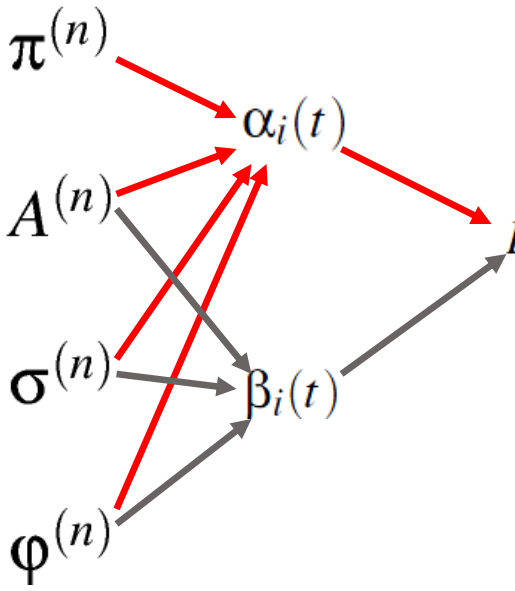
regla recursiva:

$$\beta_i(T) = 1 \quad \forall i = 1, 2, \ldots, M$$

$$\beta_i(t) = \sum_{j=1}^{M} a_{ij}^{(n)} b_j^{(n)}(O_{t+1}) \beta_j(t+1) \quad \forall t = T-1, T-2, \ldots, 1 \; \forall j = 1, 2, \ldots, M$$

– En cada pixel (x,y)

- Estimación de los parámetros de modelo:
  - Algoritmo de Baum-Welch (EM)

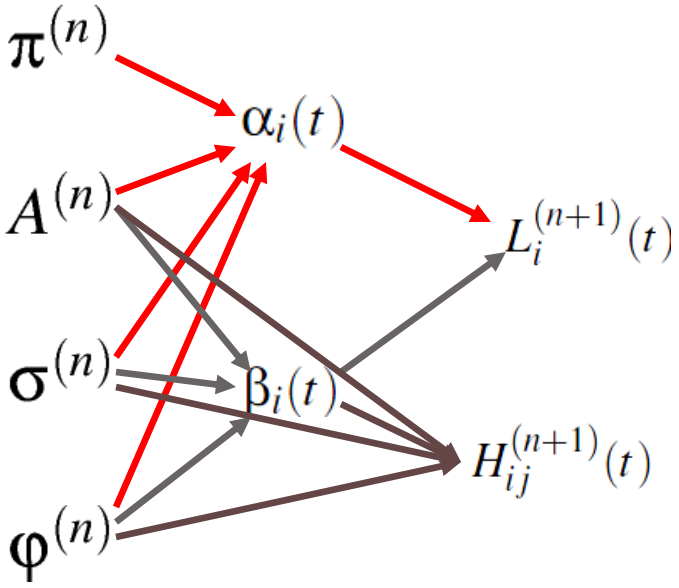


$$L_i^{(n+1)}(t) = P(s_t = i | O, A^{(n)}, b^{(n)}, \pi^{(n)})$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{M} \alpha_j(t)\beta_j(t)} \qquad i = 1, 2, \ldots, M, \;\; t = 1, 2, \ldots, T$$

- En cada pixel (x,y)

– Estimación de los parámetros de modelo:
  – Algoritmo de Baum-Welch (EM)



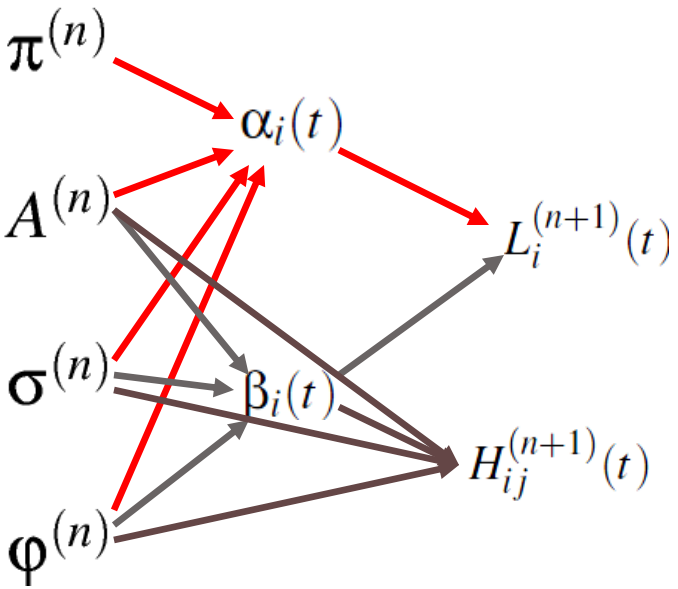
$$H_{ij}^{(n+1)}(t) = P(s_t = i, s_{t+1} = j | O, A^{(n)}, b^{(n)}, \pi^{(n)})$$

$$= \frac{L_i^{(n+1)}(t) a_{ij}^{(n)} b_j^{(n)}(o_{t+1}) \beta_j(t+1)}{\beta_i(t)}$$

– En cada pixel (x,y)

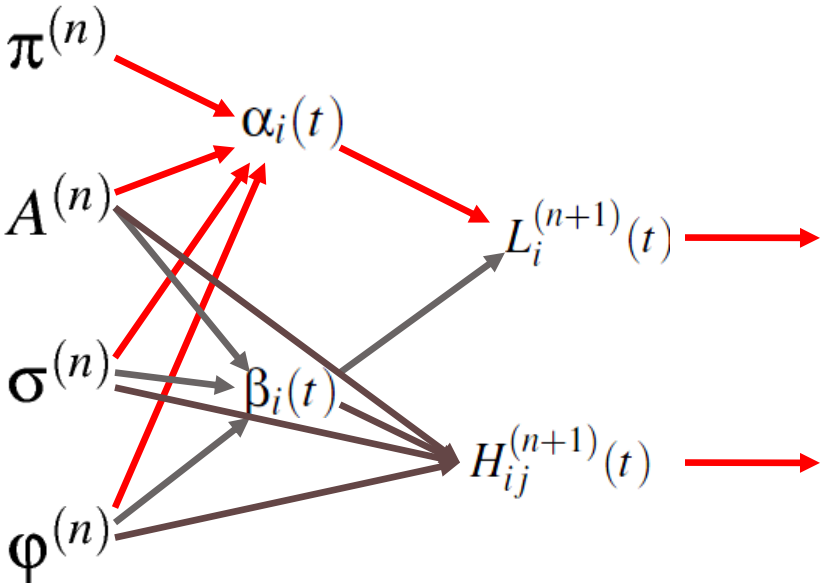
$$i, j = 1, 2, \ldots, M, \quad t = 1, 2, \ldots, T-1$$

– Estimación de los parámetros de modelo:
  – Algoritmo de Baum-Welch (EM)



– En cada pixel (x,y)

–Estimación de los parámetros de modelo:
– Algoritmo de Baum-Welch (EM)



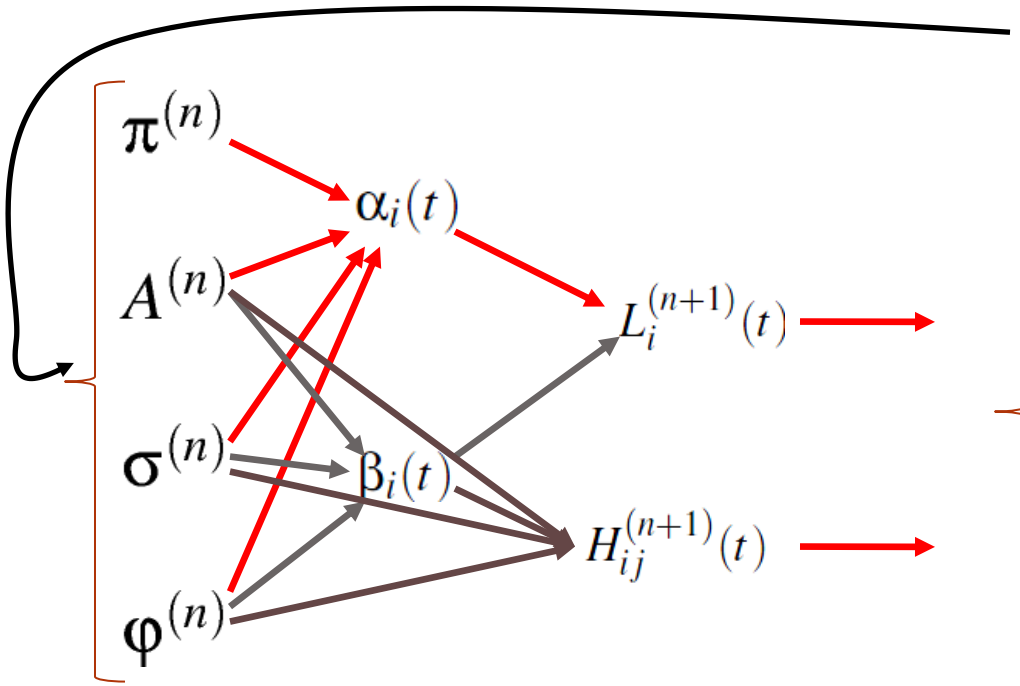
$$\hat{\pi}_i^{(n+1)} = \frac{1}{N \times M} \sum_{x,y} L_i^{(x,y)}(1)$$

$$\hat{a}_{ij}^{(n+1)} = \frac{\sum_{x,y} \sum_{t=1}^{T-1} H_{ij}^{(x,y)}(t)}{\sum_{x,y} \sum_{t=1}^{T-1} L_i^{(x,y)}(t)}$$

$$\hat{\varphi}^{(n+1)} = \sqrt{2} \frac{\sum_{x,y} \sum_{t=1}^{T} L_2^{(x,y)}(t) |O_t^{(x,y)}|}{\sum_{x,y} \sum_{t=1}^{T} L_2^{(x,y)}(t)}$$

$$\hat{\sigma}^{(n+1)} = \frac{\sum_{x,y} \sum_{t=1}^{T} L_1^{(x,y)}(t) (O_t^{(x,y)})^2}{\sum_{x,y} \sum_{t=1}^{T} L_1^{(x,y)}(t)}$$

–Estimación de los parámetros de modelo:
– Algoritmo de Baum-Welch (EM)



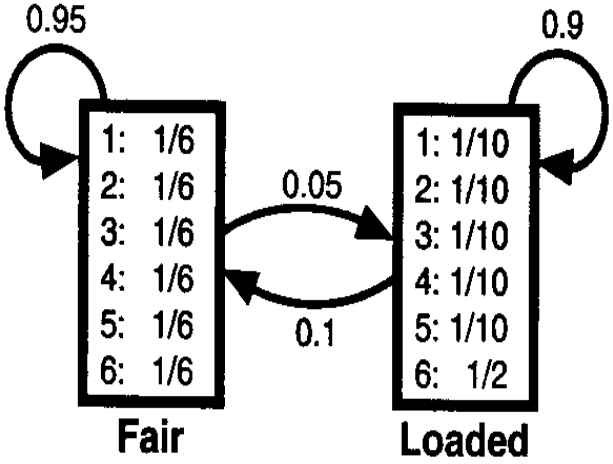
$$\hat{\pi}_i^{(n+1)} = \frac{1}{N \times M} \sum_{x,y} L_i^{(x,y)}(1)$$

$$\hat{a}_{ij}^{(n+1)} = \frac{\sum_{x,y} \sum_{t=1}^{T-1} H_{ij}^{(x,y)}(t)}{\sum_{x,y} \sum_{t=1}^{T-1} L_i^{(x,y)}(t)}$$

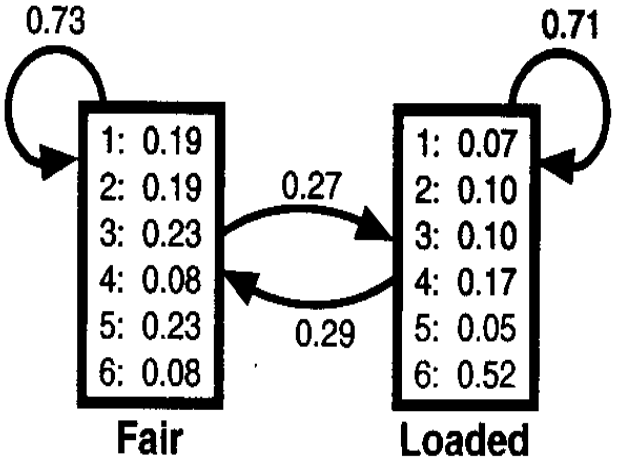
$$\hat{\varphi}^{(n+1)} = \sqrt{2} \frac{\sum_{x,y} \sum_{t=1}^{T} L_2^{(x,y)}(t) |O_t^{(x,y)}|}{\sum_{x,y} \sum_{t=1}^{T} L_2^{(x,y)}(t)}$$

$$\hat{\sigma}^{(n+1)} = \frac{\sum_{x,y} \sum_{t=1}^{T} L_1^{(x,y)}(t) (O_t^{(x,y)})^2}{\sum_{x,y} \sum_{t=1}^{T} L_1^{(x,y)}(t)}$$

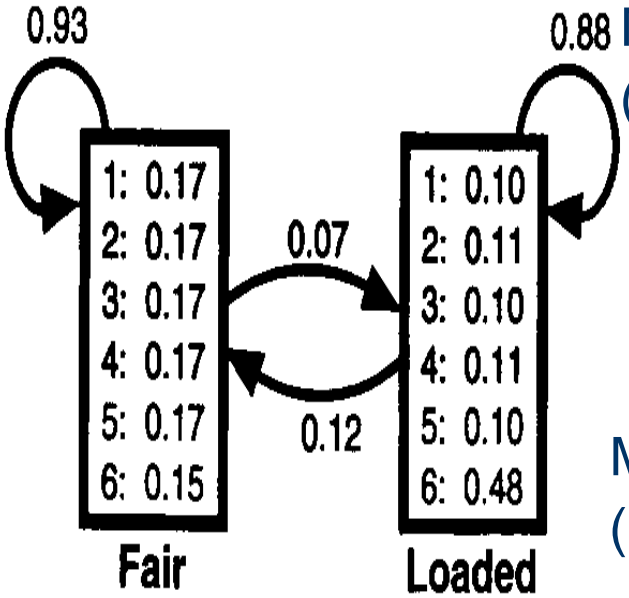
Ejemplo de Baum-Welch



Modelo real



Modelo estimado (300 tiradas)



Modelo estimado (30000 tiradas)

Algoritmo de Baum-Welch – Comentarios

Complejidad temporal:

iteraciones $\times$ $O(N^2T)$

- Garantiza incrementar la (log) verosimilitud del modelo
- No garantiza encontrar los mejores parámetros GLOBALES
- Converge a un óptimo LOCAL, dependiente de las condiciones iniciales
- Si hay demasiados parámetros/ el modelo es muy largo - Sobre-entrenamiento
- Algunas veces se reemplaza la reestimación de Baum-Welch por un entrenamiento basado sólo en el camino más probable (Viterbi training)