BaumWelch Implementation

Rodrigo Arriaza

1/18/2022

Introducción

El modelo oculto de Markov se introdujo como una técnica para segmentación de ADN por Gary Churchill en 1989. Desde entonces, este modelo se ha utilizado en distintios análisis de genómica como encontrar genes y predecir funciones de proteínas. La principal razón de su uso, es que este tipo de datos es altamente ruidoso, pues incluso en trozos con alto contenido de CG pueden haber largas cadenas de As y Ts. Por lo tanto, modelos simples multinomiales no pueden capturar esta propiedad de las secuencias de ADN.

Por otro lado, el modelo oculto de Markov (HMMs por sus siglas en inglés), es una forma de generar una secuencia que ha sido indirectamente creada por una cadena de Markov. En cada posición de la secuencia, la cadena de Markov se encuentra en un estado oculto (no observable), pero sí se puede observar la base generada (Cristianini and Hahn (2006)).

Este modelo tiene dos parámetros importantes: la matriz de transición y la matriz de emisión. La primera nos dice la probabilidad de pasar de un estado a otro, mientras la segunda nos dice la probabilidad de producir un símbolo (una base), desde cada estado. La matriz de transición tiene dimensiones (N, N), donde N es el número de estados ocultos; y la matriz de emisión (N, M), donde M es el tamaño del alfabeto, que en el caso de secuencias de ADN será $M = 4$: (A, C, G, T).

Por lo tanto, para crear un modelo oculto de Markov se deben conocer los valores de las matrices de transición y emisión. Este es un problema para el cual no existe una solución exacta, pero existen algoritmos de máxima verosimilitud con los cuales se puede obtener una solución aproximada. Tal es el caso del algoritmo de Baum-Welch.

Algoritmo Baum-Welch

El algoritmo es un caso especial de esperanza-maximización. El objetivo es predecir las matrices A y B (transición y emisión). Para ello, se utiliza el algoritmo forward-backward (Oudelha and Ainon (2010)).

Forward

En este paso se busca encontrar la probabilidad que el modelo oculto de Markov esté en el estado s en el momento t, dada una secuencia de estados visibles V_T:

$\alpha_j = p(v(1), p(v2), ..., v(t), s(t) = j)$

La ecuación anterior se puede separar en el caso base $t = 1$ y el caso recursivo $t > 1$.

Cuando t=1:

$\alpha_j(1) = p(v_k(1), s(1) = j) = p(v_k(1)|s(1) = j)p(s(1) = j)$

La probabilidad que el estado inicial sea j está dada por la distribución inicial π_j, por lo que podemos reemplazarlo en el paso anterior:

$= \pi_j p(v_k(1)|s(1) = j)$

$= \pi_j b_{jk}$, donde b_{jk} es la probabilidad de emitir el símbolo k desde el estado j.

Cuando t=2:

$\alpha_j(2) = p(v_k(1), v_k(2), s(2) = j)$

$= \sum_{i=1}^{M} p(v_k(1), v_k(2), s(1) = i, s(2) = j)$

Se agrega la sumatoria de 1 hasta M y el término $s(1) = i$ para considerar los M diferentes estados ocultos, luego se extiende a:

$= \sum_{i=1}^{M} p(v_k(2)|s(2) = j, v_k(1), s(1) = i) * p(v_k(1), s(2), s(1) = i)$

$= \sum_{i=1}^{M} p(v_k(2)|s(2) = j) * p(s(2) = j|s(1) = i) * p(v_k(1), s(1) = i)$

Del último paso se observa que se puede sacar $p(v_k(2)|s(2) = j$ de la sumatoria y reemplazar por $b_{jk}(2)$, que es la probabilidad de emitir el símbolo k desde el estado j en el momento t=2. De igual forma el término $p(v_k(1), s(1) = i$ se sustituye por $\alpha_i(1)$, que es la probabilidad forward en el tiempo t=1, que ya ha sido calculado en el paso anterior. Finalmente se obtiene:

$\alpha_j(2) = b_{jk}(2) * \sum_{i=1}^{M} a_{i2}\alpha_i(1)$.

Y de forma general para el paso t+1:

$\alpha_j(t+1) = p(v_k(1)...v_k(t+1), s(t+1) = j)$

$= \sum_{i=1}^{N} p(v_k(1)...v_k(t+1), s(t) = i, s(t+1) = j)$

$= \sum_{i=1}^{N} p(v_k(t+1)|s(t+1) = j, v_k(1)...v_k(t), s(t) = i) * p(v_k(1)...v_k(t), s(t+1) = j, s(t) = i)$

$= \sum_{i=1}^{N} p(v_k(t+1)|s(t+1) = j) * p(s(t+1) = j|s(t) = i) * p(v_k(t), s(t) = i)$

De forma similar al paso t=2, en esta última ecuación, se puede sustituit $p(v_k(t), s(t) = i)$ por $\alpha_i(t)$, siendo esto la probabilidad forward de producir en el momento t el símbolo k desde el estado i. Además el primer término puede salir de la sumatoria y sustituirse por $b_{jk(t+1)}$ y obtener así:

$\alpha_j(t+1) = b_{jk(t+1)} * \sum_{i=1}^{N} a_{ij}\alpha_i(t)$ (Jana (2019))

Backward

Este algoritmo es la contraparte del algoritmo forward, buscando ahora calcular $\beta_t(i)$, que es la probabilidad de la secuencia observada, dado el estado i en el tiempo t. Esto se calcula de la siguiente manera:

$\beta_i(t) = p(v_k(t+1)...v_k(T)|s(t) = i)$

$= \sum_{j=0}^{M} p(v_k(t+1)...v_k(T), s(t+1) = j|s(t) = i)$

$= \sum_{j=0}^{M} p(v_k(t+2)...v_k(T)|s(t+1) = j) * p(v_k(t+1)|s(t+1) = j) * p(s(t+1) = j|s(t) = i)$

$\beta_i(t) = \sum_{j=0}^{M} \beta_j(t+1)b_{jk(t+1)}a_{ij}$

donde a_{ij} es la probabilidad de pasar del estado i al estado j, $b_{jk(t+1)}$ es la probabilidad de emitir el simbolo k desde el estado j en el momento t+1 y $\beta_j(t+1)$ es la backward probability calculada en la iteración anterior, ya que este algoritmo va desde el final t=T hacia el principio, t=1.

Actualización Baum-Welch

Una vez definido el cálculo de $\alpha(t)$ y $\beta(t)$, la actualización de las matrices de transición y emisión procede de esta manera:

$a_{ij}^* = \frac{\sum \xi_{ij}(t)}{\sum \gamma_i(t)}$

$b_{ij}^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$

donde $\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{k=1}^{N} \sum \alpha_k(t) a_{kw} \beta_w(t+1) b_w(y_{t+1})}$ es la probabilidad de estar en el estado i e j en los tiempos t y t+1 respectivamente, dada la secuencia observada Y, y los parámetros $\theta = [A, B]$.

Mientras que $\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$ es la probabilidad de estar en el estado i en el momento t.

Metodología

La implementación del algoritmo BaumWelch está disponible como paquete de R en este repositorio. Esta se compara con la existente en R dentro de la librería HMM, para estimar los parámetros de un HMM asumiendo que la secuencia del archivo ./fastaFiles/vif.fasta ha sido creada de tal forma.

Luego se generan 3 secuencias desde un modelo HMM predefiniendo las matrices de emisión y transmisión para verificar si con el algoritmo de Baum-Welch se obtienen resultados similares a los valores verdaderos.

Finalmente, la generación de las secuencias se realiza de la siguiente manera, usando la función creada también dentro del paquete:

```
set.seed(33)
seq1 <- create_sequence(t.mat, e.mat, 333, start.prob)
seq2 <- create_sequence(t.mat, e.mat, 333, start.prob)
seq3 <- create_sequence(t.mat, e.mat, 333, start.prob)
seq4 <- create_sequence(t.mat, e.mat, 333, start.prob)
```

Resultados

Comparación vs librería HMM

Table 1: Estimación de la matriz de transición usando implementación manual (izq) y la existente (der)

	H	L		H	L
H	0.7881	0.2119	H	0.7845	0.2155
L	0.5026	0.4974	L	0.4979	0.5021

Table 2: Estimación de la matriz de emisión usando implementación manual (izq) y la existente (der)

	A	C	G	T		A	C	G	T
H	0.3871	0.0000	0.4366	0.1764	H	0.3810	0.0000	0.4406	0.1785
L	0.2369	0.5442	0.0002	0.2187	L	0.2541	0.5328	0.0001	0.2130

Al momento de estimar los parámetros del modelo para la secuencia ./fastaFiles/vif.fasta, numéricamente la diferencia entre ambas funciones es de 0.02 para la matriz de transición y de 0.05 para la matriz de emisión, por lo cual, al igual como se observa en las tablas 1 y 2, se puede afirmar que ambas llegan al mismo resultado.

Adicionalmente se puede observar la convergencia de ambas implementaciones en las figuras 1. El cambio en la implementación manual se calcula de la siguiente manera $\Delta_A = abs(A_{t-1} - A_t)$ y $\Delta_B = abs(B_{t-1} - B_t)$, donde A y B son las matrices de transición y emisión respectivamente. Aquí se puede observar que en ambas implementaciones, los valores de las matrices varían bastante durante las primeras iteraciones, pero que luego de la iteración 300 se estabilizan.

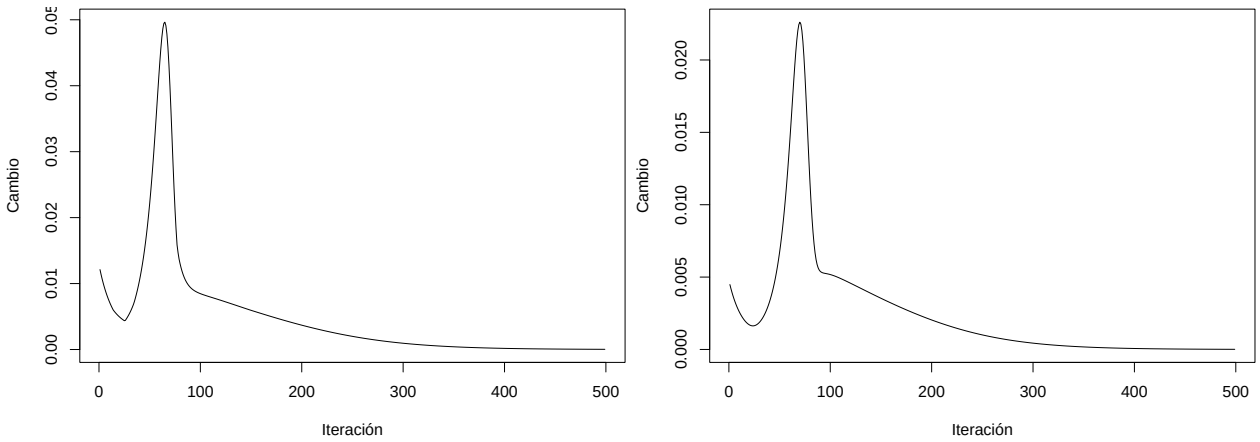

Figure 1: Cambio en los valores de los parámetros a lo largo del tiempo para implementación manual (izq) y existente (der)

Finalmente se puede comparar el tiempo de ejecución entre ambas implementaciones y se obtiene que la implementación manual requiere de 14.56 segundos mientras que la de libreria 20.19, por lo que en términos de tiempo de ejecución la implementación manual conlleva una mejora significativa.

Comparación BaumWelch vs valores verdaderos

Para hacer la estimación se utilizan las 4 secuencias descritas la sección de Metodología. La estimación para la primera secuencia se hace de forma aleatoria, pero para las siguientes se usa la salida de la estimación de la secuencia anterior:

```
set.seed(420)

# prediccion inicial con numeros aleatorios
A = matrix(runif(4), nrow=2)
A <- A/rowSums(A)
B = matrix(runif(8), nrow=2)
B <- B/rowSums(B)

row.names(A) <- colnames(A) <- c('H', 'L')
row.names(B) <- c('H', 'L')
colnames(B) <- c('A','C','G', 'T')

start.prob <- c(0.5, 0.5)
```

```
my.bw.preds.1 <- BaumWelch(seq1$sequence, A, B, start.prob,
                           row.names(A), colnames(B), n.iter = 300)
my.bw.preds.2 <- BaumWelch(seq2$sequence,
                           my.bw.preds.1$a, my.bw.preds.1$b, start.prob,
                           row.names(A), colnames(B), n.iter = 300)
my.bw.preds.3 <- BaumWelch(seq3$sequence,
                           my.bw.preds.2$a, my.bw.preds.2$b, start.prob,
                           row.names(A), colnames(B), n.iter = 300)
my.bw.preds.4 <- BaumWelch(seq4$sequence,
                           my.bw.preds.3$a, my.bw.preds.3$b, start.prob,
                           row.names(A), colnames(B), n.iter = 300)
```

Esto es equivalente a hacer la predicción para una secuencia de longitud 4 veces mayor, pero evita el problema de que los valores de α y β lleguen a cero (descrito más adelante). Además se puede observar como el error porcentual disminuye para cada cadena en la Figura 2, excepto para la última secuencia en la matriz de emisión, por lo que sería mejor parar en la tercera secuencia.

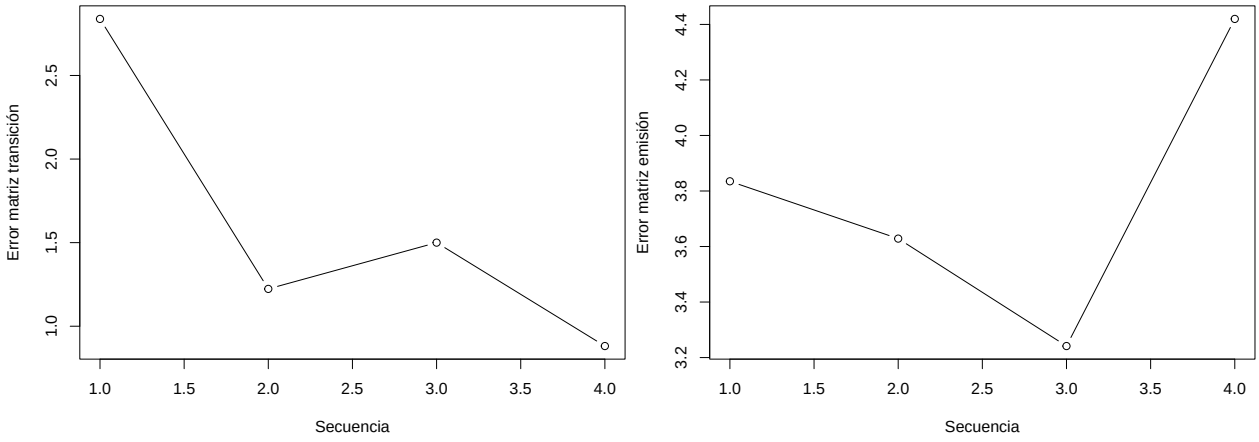

Figure 2: Suma error por cada secuencia

Sin embargo, el error es de apenas 0.88 % para la matriz de transición y de 4.42 para la de emisión, por lo que se puede afirmar que el resultado es aceptable.

Table 3: Estimación (izquierda) vs Real (derecha) matriz de transmisión

	H	L		H	L
H	0.6590	0.3410	H	0.7	0.3
L	0.4441	0.5559	L	0.3	0.7

Como se puede observar en 3, la estimación del parámetro A es bastante aceptable pues el algoritmo BaumWelch logra identificar que tanto para el estado H como para el L, la probabilidad de permanecer en el mismo estado es mayor que la de moverse. No obstante los valores de transición desde el estado L están más alejados de los valores verdaderos.

Table 4: Estimación (izquierda) vs Real (derecha) matriz de emisión

	A	C	G	T		A	C	G	T
H	0.1723	0.4149	0.3563	0.0565	H	0.1	0.4	0.3	0.2
L	0.3491	0.0000	0.0000	0.6509	L	0.4	0.1	0.1	0.4

Por otro lado, como se observa en 4, el algoritmo logra identificar que desde el estado H es más probable emitir una C o una G, e igualmente desde L producir As y Ts. Aunque desde este último se sobreestima la probabilidad de emitir T y se subestima la probabilidad de Cs y Gs. Sin embargo, en general se puede afirmar que las estimaciones de ambas matrices proveen una aproximación aceptable.

Problemas algoritmo BW y posibles alternativas

Uno de los principales problemas del algoritmo Baum-Welch es que tanto en el paso forward como backward, si la secuencia es muy larga, las probabilidades α_j y β_j tienden a cero. Esto es problemático, porque luego en el paso de actualización $\gamma = \frac{0}{0}$. Esto se puede resolver haciendo una reescalando los parámetros de la siguiente manera:

En vez de calcular

$\alpha_j(t+1) = p(x_n|s_{t+1}) \sum_i \alpha_j(t) * p(s_{t+1}|s_t = i)$

se puede usar la siguiente ecuación

$p(s_{t+1}|x_1, ..., x_n) = \frac{p(x_n|s_{t+1}) \sum_i p(s_{t+1}|s_t=i) * p(s_t=i|x_1,...,x_n)}{p(x_n|x_1,...,x_{n-1})}$

Esto implica, sin embargo, modificar el algoritmo, de tal forma de dejar de guardar la matriz α_j y guardar únicamente las distribuciones filtradas (Cappe and Moulines (2005)).

De manera similar, se debe modificar el cálculo de β de la siguiente forma:

$\hat{\beta}(t) = \beta(t) \frac{p(x_{1:n})}{p(x_{1:T})} = \frac{p(t|x_{1:T})}{p(t|x_{1:n})}$ (Cappe and Moulines (2005)).

Otro problema común es que, dependiendo de la estimación inicial, el algoritmo puede estar sesgado y quedarse en un máximo local. Esto se puede resolver usando distintias inicializaciones aleatorias o usando una metodología más elaborada como se propone en (Oudelha and Ainon (2010)), utilizando algoritmos genéticos. Estos pueden encontrar el óptimo global, ya que buscan en múltiples puntos del espacio de solución de forma simultánea. Sin embargo, este tipo de algoritmo puede tardarse mucho en converger, por lo que en el artículo mencionado anteriormente, se realizó un híbrido entre estos y Baum Welch, alcanzando resultados superiores a BW normal.

Conclusiones

En este trabajo se logró implementar el algoritmo de Baum Welch estándar y obtener resultados muy similares a la implementación existente en la librería HMM, para predecir los parámetros de un modelo de Markov. Además la implementación de este trabajo fue 27.88% más rápido.

Por otro lado, se observó que el resultado del algoritmo se asemeja a los valores verdaderos de un HMM simulado. Sin embargo, en este caso, los resultados no son tan precisos, pero si proveen una aproximación. Para mejorar la precisión se aconseja utilizar la reescala de los parámetros α y β descrita en la sección anterior.

Referencias

Cappe, O., and E. Moulines. 2005. *Inference in Hidden Markov Models*. New York: Springer.

Cristianini, N., and M. Hahn. 2006. *Introduction to Computational Genomics*. New York: Cambridge University Press.

Jana, Abhisek. 2019. "Forward and Backward Algorithm in Hidden Markov Model." http://www.adeveloperdiary.com/data-science/machine-learning/forward-and-backward-algorithm-in-hidden-markov-model/.

Oudelha, M., and R. Ainon. 2010. "HMM Parameters Estimation Using Hybrid Baum-Welch Genetic Algorithm." *IEEE*, no. 5561388: 542–45.