

Informe de Práctica N°3

Conociendo Glimmer

Rodrigo Assar C.
rassar@dim.uchile.cl

La dificultad del problema

Una secuencia de ADN consta de dos hebras cada una escrita en un lenguaje cuyo alfabeto es $\Sigma = \{a, c, g, t\}$ y complementarias entre si. A cada elemento del alfabeto lo llamaremos “base” y un “gen” será visualizado como una subsecuencia (subpalabra) de ADN escrita en el lenguaje $\Sigma^* = \{S : S \text{ string para el alfabeto } \Sigma\}$. El problema se divide en dos, se quiere encontrar subsecuencias específicas en la secuencia (genes) pero no tengo candidatos que buscar, por lo que se agrega el problema de determinar qué es lo que clasifico como genes. Es en este último punto en que interviene la incorporación de conceptos biológicos, debe haber “seales” en el ADN que me avisen de la presencia de genes, por ejemplo secuencias de “start” y “stop” que me indiquen con algún grado de certeza inicio y fin de genes. Por último para decidir entre posibles genes debo tener una herramienta de discriminación, los modelos “IMM” e “ICM” ayudan a realizar dicha tarea.

Fundamento matemático

Supongamos un string S de largo “ n ”, $S = s_1 s_2 \dots s_n$ ($S \in \Sigma^*$). Entonces, aplicando probabilidades condicionales

$$\mathbb{P}(S) = \mathbb{P}(s_n | s_1 \dots s_{n-1}) \mathbb{P}(s_{n-1} | s_1 \dots s_{n-2}) \dots \mathbb{P}(s_2 | s_1) \mathbb{P}(s_1) \quad (1)$$

Es decir

$$\mathbb{P}(S) = \mathbb{P}(s_1) \prod_{i=2}^n \mathbb{P}(s_i | s_1 \dots s_{i-1}) \quad (2)$$

Del mismo modo, para la probabilidad de S dado el contexto (letras que aparecen previamente a S en la secuencia de ADN), si llamamos “ $\text{context}(s_i)$ ” al contexto de s_i . Notando que $\text{context}(s_k) = \text{context}(s_{k-1}) \text{ concat } \{s_{k-1}\}$ y que $\text{context}(s_1) = \text{context}(S)$, se tiene que:

$$\mathbb{P}(S | \text{context}(S)) = \prod_{i=1}^n \mathbb{P}(s_i | \text{context}(s_i)) \quad (3)$$

Es decir, tenemos que la probabilidad de un string dado un contexto depende de la probabilidad de cada letra del string dado su contexto. Entonces, una

forma de estimar dicha probabilidad es estimar la probabilidad de cada letra del string dado su contexto y multiplicar dichas estimaciones.

Según como haga estas estimaciones se definen los modelos de orden k (Modelo de Markov uniforme), IMM (Interpolated Markov Model) e ICM (Interpolated Context Model).

Modelo uniforme de Markov de orden k

En este caso para estimar cada probabilidad tomamos un contexto de largo fijo k . Es decir:

$$\mathbb{P}(S|\text{context}(S)) \approx \prod_{i=1}^n \mathbb{P}(s_i | s_{i-k} \dots s_{i-1}) \quad (4)$$

y

$$\mathbb{P}(s_i | s_{i-k} \dots s_{i-1}) \approx \frac{f(s_{i-k} \dots s_i)}{\sum_{b \in \Sigma} f(s_{i-k} \dots s_{i-1} b)}, \quad (5)$$

donde f representa la frecuencia del evento en la secuencia de entrenamiento.

IMM

En este caso no se usa un orden fijo para aproximar, sino una combinación lineal convexa de diferentes órdenes (Glimmer1 usa hasta orden 8). Es decir, se tiene:

$$\mathbb{P}(s_i | \text{context}(s_i)) \approx \sum_{k=1}^8 c_k \mathbb{P}(s_i | s_{i-k} \dots s_{i-1}) + c_0 \mathbb{P}(s_i), \quad (6)$$

donde

$$\sum_{k=0}^8 c_k = 1 \quad (7)$$

Notemos que este modelo incluye al anterior, puesto que cuando $c_8 = 1$ ambos coinciden.

ICM

En el modelo ICM (interpolated context model) se estiman las probabilidades de aparición de cada letra del alfabeto según su contexto de otra manera. Sea S una palabra genérica de largo n (12), se ubica la posición en S más relacionada con la posición n la que llamaremos max_1 y se registra $\mathbb{P}(s_n = b)$, para cada $b \in \Sigma$. La segunda etapa es determinar, para cada $d \in \Sigma$ la posición más relacionada con n cuando $s_{max_1} = d$ a la que llamaremos max_2 , registrándose $\mathbb{P}(s_n = b | s_{max_1} = d)$, para cada $b \in \Sigma$. La tercera etapa consiste en seleccionar, para cada d y $e \in \Sigma$, la posición más relacionada con n cuando $s_{max_1} = d$ y $s_{max_2} = e$, la que llamaremos max_3 , y anotar las probabilidades $\mathbb{P}(s_n = b | s_{max_1} = d, s_{max_2} = e)$. El proceso continúa de igual modo hasta que se logra cierta profundidad en el árbol (8), de este modo se tiene que:

$$\mathbb{P}(s_i | context(s_i)) \approx \sum_{k=1}^8 c_k \mathbb{P}(s_i | s_{max_k} \dots s_{max_1}) + c_0 \mathbb{P}(s_i), \quad (8)$$

donde

$$\sum_{k=0}^8 c_k = 1 \quad (9)$$

La noción de posiciones más relacionadas, se estudia mediante su “información mutua”. Esto es, sean las variables aleatorias S_i y S_j representando a las bases en las posiciones i y j de S (s_i y s_j) respectivamente, se define

$$I(S_i; S_j) = \sum_{s_i \in \Sigma} \sum_{s_j \in \Sigma} \mathbb{P}(s_i, s_j) \log \frac{\mathbb{P}(s_i) \mathbb{P}(s_j)}{\mathbb{P}(s_i, s_j)}, \quad (10)$$

que mide la dependencia entre las variables. Notar que si son completamente independientes la información mutua se anula.

Notemos que el modelo IMM es un caso particular de este modelo, ICM se reduce a este cuando $max_1 = n - 1$, $max_2 = n - 2$, \dots $max_8 = n - 8$. Es decir IMM corresponde al caso en que cada posición es mayor relacionada con la posición anterior.

Escritura alternativa de ICM

Antes de continuar, notemos que definiendo inductivamente

$$IMM_k(s_i) = \lambda_k(s_{i-1})\mathbb{P}(s_i|s_{i-k} \dots s_{i-1}) + (1 - \lambda_k(s_{i-1}))IMM_{k-1}(s_i) \quad (11)$$

con IMM_0 = probabilidad simple y desarrollando $IMM_8(s_i)$ usando (12) se tiene que

$$IMM_8(s_i) = \sum_{k=1}^8 c_k \mathbb{P}(s_i|s_{i-k} \dots s_{i-1}) + c_0 \mathbb{P}(s_i) \quad (12)$$

con c_k en función de los λ y cumpliendo que $\sum_{k=1}^8 c_k = 1$. Entonces la aproximación que se efectúa para IMM corresponde a:

$$\mathbb{P}(S|context(S)) \approx \prod_{i=1}^n IMM_8(s_i) \quad (13)$$

para el modelo IMM. Análogamente para ICM, se tiene que podemos definir

$$ICM_k(s_i) = \lambda_k(s_{i-1})\mathbb{P}(s_i|s_{max_k} \dots s_{max_1}) + (1 - \lambda_k(s_{i-1}))ICM_{k-1}(s_i), \quad (14)$$

donde ICM_0 es la probabilidad simple e $ICM_k(s_i)$ corresponde a icm con profundidad k. Cumpliéndose que la aproximación antes vista para ICM corresponde a:

$$\mathbb{P}(S|context(S)) \approx \prod_{i=1}^n ICM_8(s_i) \quad (15)$$

Funcionamiento de Glimmer: Modo de uso

Para el uso de esta herramienta se dispone de varios programas, los principales son:

long-orfs
 get-putative
 extract
 build-icm
 glimmer2

El procedimiento consta de tres partes:

- Generar subsecuencias de posibles genes, a partir de la secuencia completa de ADN.
- A partir de estas subsecuencias crear el modelo (en realidad 3 modelos) IMM o ICM, según se trate de Glimmer1 o Glimmer2.
- Aplicando el modelo creado, entregar los genes encontrados en la secuencia de ADN.

La siguiente es la forma en que Glimmer2 aplica dicho procedimiento:

- A partir de la secuencia completa, usando long-orfs se determinan las secuencias START-STOP más largas, el resultado de este programa es modificado en su formato por get-putative y por extract de modo que se entregue el código entre el START y STOP para cada secuencia. Básicamente funciona encontrando los codones de STOP y ubicándose en el START más lejano seleccionando el gen si es lo suficientemente largo.

long-orfs admite varias opciones:

- Para modificar largo mínimo de gen (que por defecto es 500), de modo de entregar subsecuencias más largas que dicho número.
- Considerar el genoma como no circular.
- Para modificar el largo máximo de overlaps permitidos (que por defecto es 30).
- Se puede también modificar el porcentaje máximo de overlap (por defecto es 10) que dicho número sean ignorados.

Los métodos de búsqueda de subsecuencias de START y STOP, se implementan en el programa “gene.cc”, que sirve de librería. Más detalles de este programa puede encontrar en la sección “Incorporación de elementos biológicos”.

```
long-orfs ecoliseq > forma.coord \\
get-putative < forma.coord > lista.coord \\
extract ecoliseq lista.coord > entrena.train
```

- Terminado este proceso, estas secuencias se usan como entrenamiento para crear 3 modelos ICM, uno por marco de lectura adireccional, a través del programa build-icm que registra las probabilidades contextuales para cada letra en el alfabeto.

Se dice que las secuencias se usan de entrenamiento para build-icm pues a

partir de ellas se crea el modelo ICM usando la siguiente aproximación clásica. Sea S string, A conjunto de strings. con f representando la frecuencia del hecho en la secuencia de entrenamiento.

$$\mathbb{P}(S \in A) \approx \frac{f(S \in A)}{f(S \in \Sigma^*)} \quad (16)$$

aplicada a probabilidades de contexto (condicionales) como en (5).

El proceso entonces continúa como sigue

```
build-icm < entrena.train > train.model
```

-Para finalizar, se discrimina entre los posibles genes, esto mediante un sistema de puntajes que efectúa GLIMMER2 a partir del modelo ICM y considerando START, STOP y RIBOSOME-PATTERN. Este consiste básicamente en estimar como se explicó previamente $\mathbb{P}(S|\text{contexto}(S))$ para cada posible gen en cada uno de los tres modelos (se trata solo de tres modelos pues el sentido del gen está determinado por el codón de STOP) , clasificándolo como gen si es que cierto estimador que considera estas probabilidades sobrepasa un valor de barrera y la probabilidad independiente (que el string sea independiente del contexto) es lo suficientemente baja.

Para cumplir su cometido Glimmer2 utiliza métodos y funciones definido en la librería “icm.h” que cumplen la labor de capturar la información obtenida a través de build-icm para que glimmer2 la utilice. Además esta librería incluye la función “Fast-Evaluate” a través de la cual se calcula el logaritmo de la probabilidad de contexto de un string dado, para cada modelo ICM (los tres).

```
glimmer2 ecoli.seq train.model > result.gli
```

Suposición fundamental de Glimmer

Notemos que lo efectuado por Glimmer busca registrar el comportamiento probabilístico de los genes y ,luego buscar en la secuencia de ADN qué sub-secuencias se amoldan mejor a este comportamiento. Lo primero lo hace seleccionando un primer grupo de candidatos a genes a través de long-orfs y registrando su comportamiento probabilístico respecto a distribuciones condicionales. Lo segundo se mencionó en términos generales en la sección recién pasada y se detalla en la sección “Buscando genes”. Entonces se ha hecho la siguiente asunción:

- Las bases que forman genes siguen una distribución condicional y esta los distingue del resto del ADN.

El resultado satisfactorio de Glimmer nos llevaría a pensar que esto es verdad. Para mejorar el funcionamiento de Glimmer, se puede mejorar el registro del comportamiento probabil

Incorporación de elementos biológicos

Los elementos biológicos se incorporan en las funciones de búsqueda de START y STOP definidas en “gene.h”. Estas son:

- Find-Stop-Codons
- Is-Forward-Stop
- Is-Reverse-Stop
- Is-Forward-Start
- Is-Forward-Stop

y buscan de manera exacta los codones de START y STOP respectivamente. Resulta fácil modificar dichas funciones si se quiere utilizar otras secuencias de inicio y fin.

Estas funciones son utilizadas por Choose-Start para seleccionar los inicios de los candidatos a genes. Si se le entrega a glimmer2 una entrada para que considere RIBOSOME-PATTERN característico de la especie, la búsqueda de los START es modificada, interviniendo ahora el concepto de energía de enlace entre las bases, en este caso no se hace una búsqueda de codones de inicio exactos (en este último caso interviene la función Edit-Distance definida en glimmer2). Si esto no se hace se buscan de manera exacta.

Construcción modelo ICM, estimando $\lambda_k(s_{i-1})$

- Para estimar la probabilidad del string S en el contexto dado, se utiliza una estimación para $\lambda_k(s_{i-1})$ en vez de estimar c_k .

La determinación de los multiplicadores λ se hace mediante la función “chi2-test”, se basa en analizar la significancia de agregar un nuevo orden en la estimación de $\mathbb{P}(s_x | context(s_x))$.

Sean:

$$\begin{aligned} S_{x,k} &= s_{x-k} s_{x-k+1} \dots s_{x-1} \\ f_1 &= f(s_{x,k}, a) \\ f_2 &= f(s_{x,k}, c) \\ f_3 &= f(s_{x,k}, g) \\ f_4 &= f(s_{x,k}, t) \\ f &= f_1 + f_2 + f_3 + f_4 \end{aligned}$$

y sean las probabilidades ICM usando un contexto más corto (escaladas de modo de poder comparar uno a uno)

$$\begin{aligned} I_1 &= ICM_{k-1}(s_{x,k-1}, a) f \\ I_2 &= ICM_{k-1}(s_{x,k-1}, c) f \\ I_3 &= ICM_{k-1}(s_{x,k-1}, g) f \\ I_4 &= ICM_{k-1}(s_{x,k-1}, t) f \end{aligned}$$

se determina que tan “parecidos” son estos pares, de modo que si la diferencia es significativa $\lambda_k(s_{x-1})$ se acerca a 1 y de lo contrario a 0 (ver (12)).

Para ello se define el siguiente estadístico

$$Q = \sum_{i=1}^4 \frac{(f_i - I_i)^2}{I_i} \quad (17)$$

el cual, suponiendo $F=(f_1, f_2, f_3, f_4)$ siguiendo una distribución multinomial, se rige por una distribución χ_3^2 . Se utilizan los siguientes arreglos:

$$\begin{aligned} \text{Chi2Val [7]} &= \{2.37, 4.11, 6.25, 7.81, 9.35, 11.3, 12.8\} \\ \text{Significance [7]} &= \{0.50, 0.75, 0.90, 0.975, 0.99, 0.99\} \end{aligned}$$

, de modo que para cada $i \in I = \{0, 1, 2, 3, 4, 5, 6\}$ se tiene que

$$P(\chi_3^2 > Chi2Val[i]) = 1 - Significance[i]$$

Así, notando que si $Q \leq Chi2Val[i]$ entonces $\mathbb{P}(\chi_3^2 > Q) \geq 1 - Significance[i]$, se tiene que cuando $Q \leq Chi2Val[0]$ la diferencia es no significativa ($\lambda_k(s_{x-1}) = 0,0$).

Por otra parte, si $Q > Chi2Val[i]$ para todo $i \in I$ se tiene que $\mathbb{P}(\chi_3^2 > Q) \leq 1 - Significance[i]$ para toda significancia i considerada. En este caso se acepta que $\lambda_k(s_{x-1}) = 1,0$.

Para los casos intermedios se define λ como sigue:

- Sea i el menor natural $i < 7$ tal que $Q \leq Chi2Val[i]$.

Es decir, se selecciona i de modo tal que

$$\begin{aligned} \mathbb{P}(\chi_3^2 > Q) &\leq 1 - Significance[i - 1] \\ \mathbb{P}(\chi_3^2 > Q) &\geq 1 - Significance[i] \end{aligned}$$

- Se define c tal que

$$c = Sign[i - 1] + (Sign[i] - Sign[i - 1]) \frac{(Q - Chi2Val[i - 1])}{Chi2Val[i] - Chi2Val[i - 1]} \quad (18)$$

- Por último, se calcula λ como:

Si $c \geq 0,5$

$$\lambda_k(s_{x-1}) = c \frac{f}{SAMPLE - WEIGHT},$$

donde $SAMPLE - WEIGHT = 400$. Este valor habría sido determinado de modo tal que se tiene un 95 % de confianza en que la probabilidad obtenida esté en el margen más menos 0,5 de la probabilidad real a partir de la muestra.

Si no

$$\lambda_k(s_{x-1}) = 0$$

El resto de las funciones de build-icm cumplen la labor de calcular las informaciones mutuas y con ello realizar el proceso de construcción de los tres modelos ICM.

Glimmer2: Buscando genes

El modelo ICM estima probabilidades de aparición de cada letra del alfabeto según el contexto, para llevar dicha probabilidad a un string debemos considerar dicha estimación para cada letra del string en el contexto dado y multiplicarlas. Luego de ello, se confecciona un estimador (en realidad 2) para elegir entre los scores obtenidos para cada forma de lectura.

Los estadísticos usados por Glimmer se construyen del siguiente modo:

Sea un string S , llamemos $S[i]$ al valor entregado por

`Fast_Evaluate`

para dicho string con el frame i ($i \in \{0,1,2,3,4,5,6\}$).

Se denomina Min al menor de los seis valores y Max al mayor, si se tiene que

$$Min < Max + MAXLOGDIFF$$

con $MAXLOGDIFF = -46.0$.

Entonces se define

$$Min = Max - 46,0$$

Luego se definen:

$$W[i] = \exp(S[i] - Min)$$

y la suma Sum de los $W[i]$, con $i \in \{0,1,2,3,4,5,6\}$.

para llegar finalmente a:

$$Score[i] = 100 \frac{W[i]}{Sum}$$

Si se usa el criterio con independencia, se define de forma análoga, con $i \in \{0,1,2,3,4,5\}$ WeakScore (esto se usa si hay problemas creados por un puntaje alto de independencia que reduce a $Score[i]$ por debajo de lo aceptado). Ambos tienen como tope superior el valor 99 e inferior el 0.

En $Score[6]$ se calcula este puntaje para cuando hay independencia con el contexto (la probabilidad, en realidad el logaritmo de esta, es calculada por

la funcion `IndepEval` definida en `glimmer2`).

Usando estas dos mediciones, el candidato se clasifica, en cierto frame i , como:

-Tentative Gene, si se tiene alguna de las siguientes condiciones:

- 1) $\text{Score}[i] \geq \text{ThresholdScore}$ (que por defecto es 90).
- 2) $\text{Votes}[i] + \text{Score}[i] \geq \text{VoteThreshold}$ (que por defecto es 150).
- 3) $\text{WeakScore} \geq \text{ThresholdScore}$ y $\text{GeneLen} \geq \text{MinWeakLen}$.

-Si además se tiene 1) entonces se clasifica como `REGULAR`.

-Sino, si se tiene 2) se clasifica como `VOTED`.

-Si tampoco esto se tiene pero se tiene 3), se clasifica `WEAK`.

Si no se tiene ninguno se clasifica como `NONE`.

Tratamiento de Overlaps

Diremos que dos subsecuencias distintas estan en overlap si existen posiciones que pertenecen a ambas. Esta situación requiere un tratamiento especial por parte de Glimmer, ya que se asume por una cuestion biológica que en una secuencia de ADN los traslapes producidos son escasos y de pequeno tamaño.

Para analizar el modo de actuar de Glimmer llamaremos a las subsecuencias A y B y supondremos que el puntaje asociado a A es mayor al asociado a B .

-Caso1

$$A \ 5' \text{ —————} > \ 3'$$

$$3' < \text{ —————} \ 5' \ B$$

Se trata de eliminar el overlap moviendo el comienzo de A y de B a otros codones de `START`. Si esto no se logra y A es significativamente más largo que B entonces B es rechazado. Si no A y B son llamados genes con la anotación `doubtful overlap`.

Para mover el comienzo de los genes predichos el programa acorta el gen shifteando la localizacion del `START` a la siguiente (en el sentido del gen) posición de `START` (que no este más allá del codón de `STOP`), si esto no resuelve el problema se mueve nuevamente dicha posición hasta solucionar el

problema o hasta que el gen sea más corto que “minimum gene length”.

-Caso2

A 5' —————> 3'

B 5' —————> 3'

Se trata de resolver el overlap moviendo el comienzo de B . Si esto no se logra entonces B es rechazado, sino ambos son listados como genes con la anotación overlap.

-Caso3

A 5' —————> 3'

B 5' —————> 3'

Se trata de resolver el overlap moviendo el comienzo de A , lo cual se hace sólo si el overlap es una fracción relativa pequeña del largo de A . Si el ajuste de A no es exitoso B es rechazado.

-Caso4

3' <————— 5' A

B 5' —————> 3'

Se trata de mover ambos comienzos. Primero movemos el inicio de B hasta la región de overlap de puntaje más alto para B , luego movemos el de A hasta su puntaje más alto. Nuevamente lo mismo para B y continuamos el proceso hasta que el overlap es eliminado o no podemos mover más lejos los comienzos.