BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

POSGRADO EN MATEMÁTICAS

PROCESOS DE MARKOV APLICADOS AL ESTUDIO DE SECUENCIAS DE ADN

T E S I S

Presentada al

Posgrado de Matemáticas

como requisito parcial para la obtención del grado de

Maestro en Ciencias (Matemáticas)

Por:

Aranzazú Ortega Ponce

Director de tesis:

Dr. Hugo Adán Cruz Suárez

Puebla, Puebla. 2015.

DR. JOSÉ ENRIQUE BARRADAS GUEVARA
SECRETARIO DE INVESTIGACIÓN Y
ESTUDIOS DE POSTGRADO, FCFM-BUAP
P R E S E N T E:

Por este medio le informo que el(la) C:

ARANZAZÚ ORTEGA PONCE

estudiante de la Maestría en Ciencias (Matemáticas), ha cumplido con las indicaciones que el Jurado le señaló en el Coloquio que se realizó el día 14 de agosto de 2015, con la tesis titulada:

"Procesos de Markov aplicados al estudio de secuencias de ADN"

Por lo que se le autoriza a proceder con los trámites y realizar el examen de grado en la fecha que se le asigne.

A T E N T A M E N T E.
H. Puebla de Z. a 19 de enero de 2016.

DR. FERNANDO MACÍAS ROMERO
COORDINADOR DEL POSTGRADO
EN MATEMÁTICAS.

Ccp.Archivo
DR. JEBG/mtv

Facultad
de Ciencias
Físico Matemáticas

Av. San Claudio y 18 sur, edif. 111 A,
Ciudad Universitaria, Col. San
Manuel, Puebla, Pue. C.P. 72570
01 (222) 229 55 00 Ext. 7550 y 7552

Al gran amor de mi vida y principal motivo de alegría: mi hija.

Agradecimientos

Quiero agradecer en primer lugar a Dios por la vida y todas las bendiciones que he recibido de Él, en especial por permitirme alcanzar esta meta.

A mis padres por su amor incondicional, porque sin importar las condiciones he contado siempre con su apoyo y motivación para alcanzar mis metas. Gracias por ser un ejemplo de esfuerzo y superación.

A mis hermanos Judith, Roberto y Oscar porque sé que siempre puedo contar con ellos y de cada uno de ellos he recibido valiosas enseñanzas.

A mi asesor, el Dr. Hugo Adán Cruz Suárez, por haberme guiado en este periodo con dedicación y paciencia, por estar siempre dispuesto a compartir sus conocimientos con el fin de enriquecer este trabajo. Y agradezco sobre todo su comprensión y apoyo en el momento que fue necesario.

A la Dra. Hortensia Reyes Cervantes, al Dr. Víctor Hugo Vázquez Guevara, al Dr. Fernando Luna Velasco y a la Dra. Gladys Linares Fleites por tomarse el tiempo de revisar la tesis, por sus comentarios y observaciones con el fin de mejorar este trabajo.

A mis compañeros y amigos por su apoyo tanto en el ámbito académico como personal.

Finalmente al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haber financiado mis estudios y hacer posible el culminar este trabajo.

Aranzazú Ortega

Índice general

Introducción

En el presente trabajo se estudian los Modelos Ocultos de Markov (MOM) aplicados al estudio de secuencias de ADN, los tres problemas asociados a éstos: 1) Calcular la probabilidad de una secuencia de observaciones dado un modelo, 2) Encontrar la sucesión de estados ocultos más factible que genera cierta secuencia de observaciones y 3) Determinar los parámetros de un modelo dada una secuencia de observaciones; así como la forma de resolverlos. Para ello se hace uso de algoritmos de programación dinámica y de algunas otras herramientas diseñadas para el manejo de datos correspondientes a Genómica. Como aplicación práctica haciendo uso de los MOM y del algoritmo de Viterbi [19], usado para resolver el segundo problema, se logró identificar la secuencia de estados ocultos correspondientes a una región del ADN conocida como islas CpG y que constituyen aproximadamente el 40 % de los promotores, es decir los que marcan el inicio del proceso de transcripción mediante el cual el ADN codifica en proteínas. Así mismo, se describen los dos algoritmos de programación dinámica utilizados para alinear un par de secuencias y se aplica a dos secuencias de proteínas de diferentes organismos [10]. Finalmente, se muestra una variante de los MOM llamada Perfiles de Modelos Ocultos de Markov utilizada para modelar familias de proteínas.

El ADN se aisló por primera vez durante el invierno de 1869, quien lo hizo fue el médico suizo Friedrich Miescher mientras trabajaba en la Universidad de Tubinga. Miescher realizaba experimentos acerca de la composición química del pus de vendas quirúrgicas desechadas, cuando notó una sustancia desconocida que caracterizó químicamente más tarde [7]. La llamó *nucleína*, debido a que la había extraído a partir de núcleos celulares [8]. Se necesitaron casi 70 años de investigación para poder identificar los componentes y la estructura de los ácidos nucleicos. En 1919 Phoebus Levene identificó que un nucleótido está formado por una base nitrogenada, un azúcar y un fosfato [15]. Levene sugirió que el ADN generaba una estructura con forma de solenoide (muelle) con unidades de nucleótidos unidos a través de los grupos fosfato. En 1930 Levene y su maestro Albrecht Kossel probaron que la nucleína de Miescher es un ácido desoxirribonucleico (ADN) formado por cuatro bases nitrogenadas (citosina (C), timina (T), adenina (A) y guanina (G)), el azúcar desoxirribosa y un grupo fosfato, y que, en su estructura básica, el nucleótido está compuesto por un azúcar unido a la base y al fosfato [6]. Sin embargo, Levene pensaba que la cadena era corta y que las bases se repetían en un orden fijo. En 1937 William Astbury produjo el primer patrón de difracción de rayos X que mostraba que el ADN tenía una

estructura regular [1]. Finalmente James Watson y Francis Crick propusieron en 1953 el modelo de la doble hélice de ADN [24]. En la actualidad la forma de representar el ADN es nombrando sus bases, es decir como secuencia de las letras A, C, G y T.

Con la aparición de secuenciadores automáticos la cantidad disponible de secuencias de ADN se incrementó, así que fue necesario encontrar modelos y herramientas que facilitaran su estudio. La bioinformática se encarga, entre otras cosas, del diseño de herramientas computacionales para el discernimiento de la estructura del ADN e identificación de las diferentes partes del mismo. Entre los métodos usados destacan los métodos probabilísticos como son los MOM, los cuales en un principio se aplicaron al reconocimiento del habla pero con el tiempo se han aplicado a diversas áreas, entre ellas la bioinformática [3]. Un Modelo Oculto de Markov es un proceso estocástico bivariado ya que consta de un proceso estocástico de estados ocultos, que forman una cadena de Markov, y un proceso estocástico de eventos observados; que en nuestro caso serán las secuencias de ADN.

La secuenciación autómatica del ADN y el estudio de estos datos ha permitido el desarrollo de grandes proyectos como lo es el del genoma humano. En la actualidad también ha permitido la detección de enfermedades, pues una vez que se conoce la relación de cierta mutación puntual con dicha enfermedad se puede identificar mediante este tipo de estudios, un ejemplo de ello es el estudio del cáncer [16, 22]. O bien, para la identificación de genes o pseudogenes [5].

El trabajo está organizado de la siguiente forma:

- En el primer Capítulo se habla de las nociones de Biología necesarias para llevar a cabo y entender el trabajo, es decir, de la composición, características y funciones del ADN.
- En el segundo Capítulo se mencionan algunas técnicas estadísticas utilizadas para el estudio de las secuencias del ADN y de cómo, mediante propiedades estadísticas, podemos identificar diferentes regiones dentro el ADN.
- En el tercer Capítulo se da la definición de un Modelo Oculto de Markov de manera general, se dan ejemplos sencillos que permiten familiarizarse con los conceptos. Se describen los tres problemas asociados a los MOM, así como la forma de resolverlos.
- Finalmente, en el cuarto Capítulo se aplican los Modelos Ocultos de Markov al estudio de secuencias de ADN, se habla de proteínas, su relación con el ADN y se trabaja con datos reales tanto en secuencias de ADN como con secuencias de proteínas.

Capítulo 1

ADN (Ácido Desoxirribonucleico)

El ADN, es un ácido nucleico que contiene la información genética de los seres vivos, por la cual se sintetizan y se desarrollan las proteínas. Visto desde el punto de vista químico es un polímero de nucleótidos, es decir, una macromolécula formada por la unión de nucleótidos, donde a su vez, cada nucleótido es una molécula formada por una pentosa, una base nitrogenada y un grupo fostato, como se muestra en la Figura 1.1.

En el caso del ADN la pentosa es la desoxirribosa y la base nitrogenada puede ser Adenina, Citosina, Guanina o Timina (A,C, T, G); la Adenina sólo puede unirse a la Timina y la Guanina a la Citosina, por eso decimos que la Adenina es complementaria de la Timina y la Citosina es complementaria de la Guanina. Así, lo que distingue a un nucleótido de otro es la base nitrogenada; y por ello la secuencia del ADN se especifica nombrando sólo el orden de sus bases. Un ejemplo de secuencia de ADN puede ser: ATCCGATTAAGCTCATGGCT.

En la actualidad se dispone de la secuencia completa del ADN de muchos organismos, ya que la tecnología nos permite leer el ADN a gran velocidad. El *Método Sanger* es el más usado para secuenciar el ADN. La base de este método es el proceso biológico de la replicación del ADN. El método de secuenciación ideado por Sanger hace uso de dideoxinucleótidos (dd) que carecen del grupo hidroxilo del carbono $3'$, de manera que cuando uno de estos nucleótidos se incorpora a una cadena de ADN en crecimiento, esta

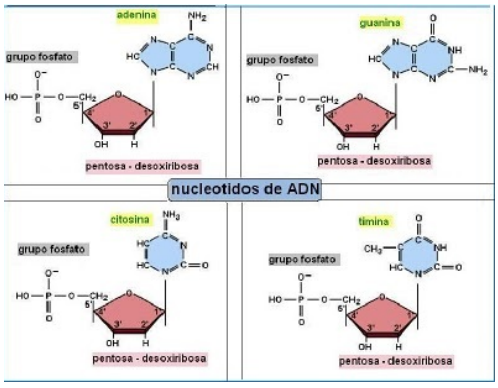


Figura 1.1: Nucleótidos del ADN

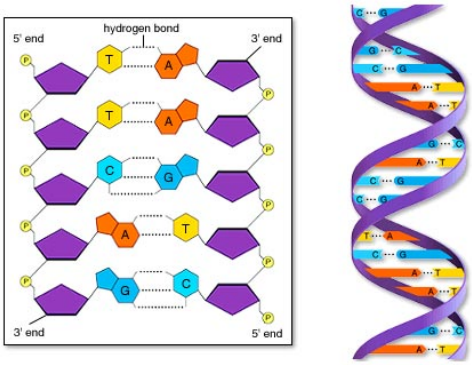


Figura 1.2: Estructura del ADN

cadena no puede continuar elongándose. Esto es así ya que la ADN polimerasa necesita un grupo terminal 3′ OH para añadir el siguiente nucleótido.

El método comienza una vez que se aísla y se clona el ADN que se desea secuenciar. Este ADN se desnaturaliza y se emplea una sola hebra para la secuenciación. En la secuenciación se utiliza un "primer" (una pieza pequeña -de 20 a 30 bases- de ADN de una sola hebra) que se encarga de suministrar el terminal 3′ OH que necesita la ADN polimerasa para comenzar a elongar. Se preparan cuatro tubos de reacción, cada uno con el ADN molde de hebra sencilla que se desea secuenciar, con ADN polimerasa, con el "primer" y con los cuatro nucleótidos trifosfatados.

A cada tubo se le añade una pequeña proporción de un dideoxinucleótido trifosfato; un tubo con ddATP (dideoxiadenina trifosfato), otro con ddTTP (dideoxitimina trifosfato), el tercero con ddGTP (dideoxiguanina trifosfato) y el cuarto con ddCTP (dideoxicitosina trifosfato). En cada uno de estos tubos se producen cadenas de ADN de distintas longitudes, terminando todas en el lugar en el que se incorporó el dideoxinucleótido correspondiente añadido al tubo. Los productos de las 4 reacciones, cada una conteniendo una pequeña cantidad de uno de los cuatro dideoxinucleótidos, son cargados en un gel de agarosa y sometidos a electroforesis. Así, obtendremos un patrón de bandas en orden, del cual es posible deducir la secuencia del ADN introducido [17].

En 1953, el bioquímico estadounidense James Watson y el biofísico británico Francis Crick publicaron la primera descripción de la estructura del ADN. Su modelo adquirió tal importancia, que los científicos obtuvieron en 1962 el Premio Nobel de Medicina por su trabajo. La estructura del ADN es una especie de escalera retorcida denominada *doble hélice*, llamada así pues está formada por dos hebras cada una constituida por una secuencia de nucleótidos encadenados, donde cada hebra contiene toda la información presente en la otra y están unidas entre sí por unas conexiones denominadas puentes de hidrógeno.

En concreto, la molécula de desoxirribosa ocupa el centro del nucleótido, de un lado tiene un grupo fosfato y del otro una base. El grupo fosfato está a su vez unido a la desoxirribosa del nucleótido adyacente de la cadena. Estas subunidades enlazadas desoxirribosa-fosfato forman los lados de la escalera; las bases están enfrentadas por parejas, mirando hacia el interior, y forman los escalones como se muestra en la Figura 1.2.

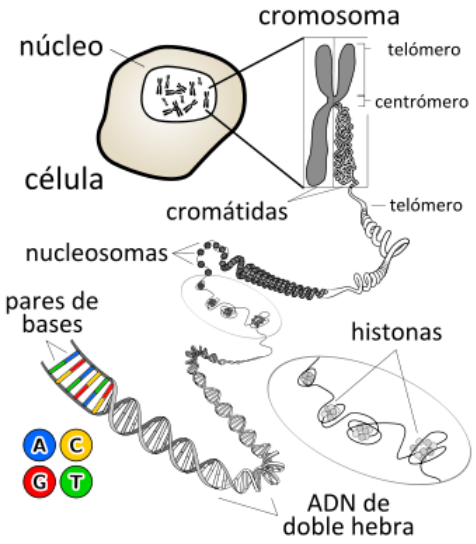


Figura 1.3: ADN dentro de la célula

En la célula, el ADN está organizado en estructuras llamadas cromosomas. Los cromosomas varían ampliamente entre los diferentes organismos. Las células humanas contienen 23 pares de cromosomas, un miembro de cada par heredado de la madre y el otro del padre. Los dos cromosomas en un par son prácticamente idénticos, con la excepción del cromosoma sexual, para el cual hay dos tipos, X e Y (XX para la mujer y XY para el hombre, siendo la pareja la que determina el sexo). Cada célula del cuerpo contiene copias idénticas del conjunto entero de 23 pares de cromosomas. En los organismos eucariotas (por ejemplo, animales, plantas y hongos) la mayor parte del ADN se almacena dentro del núcleo celular y una mínima parte en elementos celulares llamados mitocondrias, y en los plastos y los centros organizadores de microtúbulos o centríolos, en caso de tenerlos. Los organismos procariotas (bacterias y arqueas) lo almacenan en el citoplasma de la célula. Podemos ver la situación del ADN dentro de la célula en la Figura 1.3.

Podríamos hacer la comparación del ADN con una receta, puesto que contiene la información genética necesaria para el desarrollo del organismo. Sin embargo para poder hacer uso de esa información debemos copiar la información del ADN en moléculas de ARN mediante un proceso llamado *transcripción*, éstas una vez procesadas en el núcleo celular, pueden salir al citoplasma para su utilización posterior [2].

1.1. Funciones Biológicas del ADN

Como hemos mencionado, el ADN contiene la información genética necesaria para el desarrollo del organismo, por lo que entre sus funciones biológicas se encuentran:

a) El almacenamiento de información, la cual se encuentra presente en los genes y cuyo conjunto se denomina genoma.

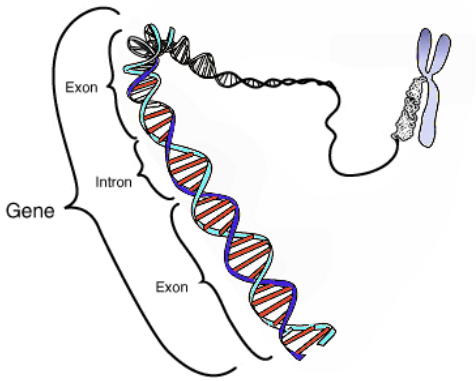


Figura 1.4: Estructura del gen

b) La codificación de proteínas, es decir, utilizar la información presente en el ADN y convertirla en proteínas mediante los procesos de transcripción y traducción.

c) Finalmente, la autoduplicación para asegurar la transmisión de la información a las células hijas durante la división celular, este proceso recibe el nombre de replicación del ADN.

Abajo se describen con mayor detalle estas funciones.

1.1.1. Genes y Genoma

El ADN se puede considerar como un almacén cuyo contenido es la información (mensaje) necesaria para construir y sostener el organismo en el que está presente, ésta se transmite de generación en generación. Al conjunto de información que cumple esta función se le denomina genoma, y el ADN que lo constituye, ADN genómico.

La información genética de un genoma está contenida en los genes, y al conjunto de toda la información que corresponde a un organismo se le denomina su genotipo. Un gen es una unidad de herencia que influye en una característica particular de un organismo (por ejemplo en el color de los ojos, cabello), ya que son éstos las secuencias de ADN que constituyen la unidad fundamental, física y funcional de la herencia. Contienen un "marco de lectura abierto" (open reading frame) que puede transcribirse, además de secuencias reguladoras, tales como promotores y enhancers, que controlan la transcripción del marco de lectura abierto. La estructura general de un gen puede verse en la Figura 1.4.

Desde este punto de vista, las obreras de este mecanismo son las proteínas. Estas pueden ser estructurales, como las proteínas de los músculos, cartílagos, pelo, etc., o funcionales, como la hemoglobina o las innumerables enzimas del organismo. La función principal de la herencia es la especificación de las proteínas, siendo el ADN una especie de plano o receta para producirlas. La mayor parte de las veces la modificación del ADN provocará una disfunción proteica que dará lugar a la aparición de alguna enfermedad. Pero en determinadas ocasiones, las modificaciones podrán provocar cambios beneficiosos que darán lugar

a individuos mejor adaptados a su entorno.

Las aproximadamente treinta mil proteínas diferentes en el cuerpo humano están constituidas por veinte aminoácidos diferentes, y una molécula de ADN debe especificar la secuencia en que se unen dichos aminoácidos.

En el proceso de elaborar una proteína, el ADN de un gen se lee y se transcribe como ARN. Este ARN sirve como mensajero entre el ADN y la maquinaria que elaborará las proteínas y por eso recibe el nombre de ARN mensajero o ARNm. El ARN mensajero sirve de molde a la maquinaria que elabora las proteínas, para que ensamble los aminoácidos en el orden preciso para armar la proteína.

El dogma central de la biología molecular establecía que el flujo de actividad y de información era: ADN $\rightarrow$ ARN $\rightarrow$ proteína. No obstante, en la actualidad ha quedado demostrado que este "dogma" debe ser ampliado, pues se han encontrado otros flujos de información: en algunos organismos (virus de ARN) la información fluye de ARN a ADN; este proceso se conoce como "transcripción inversa o reversa", también llamada "retrotranscripción". Además, se sabe que existen secuencias de ADN que se transcriben a ARN y son funcionales como tales, sin llegar a traducirse nunca a proteína: son los ARN no codificantes [9].

1.1.2. Transcripción y Traducción

Uno de los paradigmas principales de la biología establece que a partir de la información genética codificada en el ADN podemos obtener proteínas, el paradigma también establece que hay una molécula intermedia entre el ADN que está en el núcleo y las proteínas, las cuales están presentes principalmente en el citosol de la célula, esa molécula intermedia es conocida como ARN y es la intermediaria, pues es capaz de salir del núcleo, llegar al citosol y generar las proteínas. El proceso por el cual transferimos el ADN hacia el ARN se le conoce como transcripción del ADN y el proceso por el cual transferimos el ARN y lo codificamos en proteínas se conoce como la traducción.

La secuencia conocida como promotor se une a los llamados factores de transcripción y marcan el inicio del proceso, una vez que están unidas, la enzima llamada ARN polimerasa separa los segmentos del ADN para preparar la llegada de los nucleótidos complementarios del ARN y así se forma la nueva molécula ARN mensajero (ARNm), cabe mencionar que la diferencia entre los nucleótidos de ADN y ARN, radica en el azúcar y una de las bases que es reemplazada por Uracilo. Por otra parte la célula produce el ARN de transferencia (ARNt) y ARN ribosomal (ARNr) las cuales en su forma final juegan un rol vital en el proceso de traducción y por lo tanto, en la síntesis de proteínas.

1.1.3. El ADN no Codificante

El ADN del genoma de un organismo puede dividirse conceptualmente en dos: el que codifica las proteínas (los genes) y el que no codifica. Sólo alrededor del 1.5 % del genoma humano consiste en exones que codifican proteínas (20,000 a 25,000 genes), mientras que más del 90 % consiste en ADN no codificante.

El ADN no codificante (también denominado ADN basura o junk DNA) corresponde a secuencias del genoma que no generan una proteína (procedentes de transposiciones, duplicaciones, translocaciones y recombinaciones de virus, etc.), incluyendo los intrones. Hasta hace poco tiempo se pensaba que el ADN no codificante no tenía utilidad alguna, pero estudios recientes indican que eso es inexacto. Entre otras funciones, se postula que el llamado "ADN basura" regula la expresión diferencial de los genes [23]. Por ejemplo, algunas secuencias tienen afinidad hacia proteínas especiales que tienen la capacidad de unirse al ADN (como los homeodominios, los complejos receptores de hormonas esteroides, etc.), con un papel importante en el control de los mecanismos de trascripción y replicación. Estas secuencias se llaman frecuentemente "secuencias reguladoras", y los investigadores suponen que sólo se ha identificado una pequeña fracción de las que realmente existen. Un grupo de investigadores de la Universidad de Yale ha descubierto una secuencia de ADN no codificante que sería la responsable de que los seres humanos hayan desarrollado la capacidad de manipular objetos o herramientas.

Capítulo 2

Análisis de las Secuencias de ADN

Como hemos dicho, el ADN puede representarse como secuencias de nucleótidos. Dentro de éstas hay secuencias codificantes (genes), separadas por largas regiones intergénicas de secuencias no codificantes. La mayoría de los genes eucariotas tienen un nivel adicional de organización: dentro de cada gen, las secuencias codificantes, conocidas como exones, son interrumpidas por tramos de secuencias no codificantes, conocidas como intrones. Durante la transcripción del ADN en ARNm, estos últimos se eliminan mediante el proceso de maduración en la etapa conocida como splicing. Las regiones intergénicas y los intrones tienen diferentes propiedades que los exones, para identificarlas podemos utilizar procedimientos estadísticos y comprobar si una parte no caracterizada de ADN es parte de la región codificante del gen. El modelo está basado en un conjunto de datos de "entrenamiento" y en el más sencillo se asume que los nucleótidos en diferentes posiciones son independientes e idénticamente distribuidos (iid). Si éste es el caso, las diferencias entre ADN codificante y no codificante podrían detectarse por las diferencias entre las frecuencias de los cuatro nucleótidos en los casos distintos [11]. Si los nucleótidos no son independientes entonces lo modelaremos como una Cadena de Markov, con las características que se mencionan a continuación.

2.1. Contrastes de Independencia

La precisión de los procedimientos que se llevan a cabo depende de la precisión de las suposiciones hechas. Podemos creer que suponer que hay independencia estadística entre los nucleótidos suele ser una simplificación excesiva, puesto que puede haber correlaciones, por ejemplo, entre los nucleótidos debido a su pertenencia a uno u otro codón (triplete de nucleótidos). Por ello, es importante, entre otras cosas, desarrollar un contraste de independencia sobre la secuencia de nucleótidos. Este contraste está basado en el análisis de una cadena de Markov. Al utilizar cadenas de Markov en el análisis de secuencias, el concepto del "tiempo t" se reemplaza por la "posición a en la secuencia".

Sea $E = \{A, C, G, T\}$, $\Omega = \{\omega = (\omega_1, \omega_2, \omega_3, ...) \mid \omega_i \in E\}$, $\Im = \wp(\Omega)$ y $P(F) = \sum_{w_i \in F} p_i$, donde $p_i = P(\{w_i\})$ con $w_i \in E$ indica la probabilidad de que aparezca el nu-

cleótido i en la secuencia, con $i \in E$. La terna $(\Omega, \Im, P)$ es nuestro espacio de probabilidad y definimos el proceso estocástico $X_a(\omega) = \omega_a$ como "el nucleótido en la posición a de la secuencia de ADN" y cuyo espacio de estados será E, de tamaño $k = 4$.

Proposición 1. El estimador $\widehat{p_{ij}} = \frac{n_{ij}}{\sum_{k=1}^{m} n_{ik}}$ es el estimador de máxima verosimilitud de p_{ij}.
Donde p_{ij} denota la probabilidad de transición del estado i al estado j y n_{ij} el número de transiciones observadas desde el estado i al estado j, con $1 \leq i, j \leq m$.

Demostración. Supongamos que queremos estimar los $m^2 - m$ parámetros p_{ij} $(i \neq j)$ de una Cadena de Markov de m estados a partir de una realización $x_1, x_2, ..., x_T$. La verosimilitud condicionada a la primera observación es

$$L = \prod_{i=1}^{m} \prod_{j=1}^{m} p_{ij}^{n_{ij}}.$$

Entonces la log-verosimilitud es

$$l = \sum_{i=1}^{m} (\sum_{j=1}^{m} n_{ij} log p_{ij}) = \sum_{i=1}^{m} l_i,$$

donde $l_i = \sum_{j=1}^{m} n_{ij} log p_{ij}$ y podemos maximizar l maximizando cada l_i separadamente. Sustituyendo $1 - \sum_{k \neq i} p_{ik}$ para p_{ii}, diferenciando l_i con respecto a la probabilidad de transición p_{ij} fuera de la diagonal, e igualando la derivada a cero obtenemos

$$0 = \frac{-n_{ii}}{1 - \sum_{k \neq i} p_{ik}} + \frac{n_{ij}}{p_{ij}} = -\frac{n_{ii}}{p_{ii}} + \frac{n_{ij}}{p_{ij}}.$$

Por lo tanto, a menos que el denominador sea cero en la ecuación anterior, $n_{ij} p_{ii} = n_{ii} p_{ij}$, y así $p_{ii} \sum_{j=1}^{m} n_{ij} = n_{ii}$. Esto implica que en un máximo (local) de la verosimilitud,

$$p_{ii} = \frac{n_{ii}}{\sum_{j=1}^{m} n_{ij}} \qquad y \qquad p_{ij} = \frac{n_{ij} p_{ii}}{n_{ii}} = \frac{n_{ij}}{\sum_{j=1}^{m} n_{ij}}.$$

El estimador $\widehat{p_{ij}} = \frac{n_{ij}}{\sum_{k=1}^{m} n_{ik}}$ es por lo tanto el estimador de máxima verosimilitud de p_{ij}. □

Desarrollaremos un test de independencia chi-cuadrado (χ^2) sobre la secuencia de nucleótidos que queremos analizar, que contrasta la hipótesis nula de que los nucleótidos en posiciones distintas sean independientes, frente a la hipótesis alternativa de que un nucleótido en una posición dada dependa del nucleótido en la posición anterior. Por lo tanto, la hipótesis alternativa la pensamos como una cadena de Markov donde la probabilidad de un nucleótido en una posición dada, dependa del nucleótido en la posición anterior.

Este contraste es de interés para evaluar si el modelo de Markov bajo la hipótesis alternativa describe la realidad significativamente mejor que el modelo de independencia y, por tanto, podría aumentar la precisión de nuestros procedimientos de predicción. Supongamos que la longitud de la secuencia de ADN que queremos analizar es n. El contraste estadístico de independencia es un contraste de asociación en una tabla 4×4 de doble entrada como la mostrada en el Cuadro 2.1, denominada *tabla de contingencia*.

	A	C	G	T	Total
A	n_{11}	n_{12}	n_{13}	n_{14}	$n_{1,}$
C	n_{21}	n_{22}	n_{23}	n_{24}	$n_{2,}$
G	n_{31}	n_{32}	n_{33}	n_{34}	$n_{3,}$
T	n_{41}	n_{42}	n_{43}	n_{44}	$n_{4,}$
Total	$n_{,1}$	$n_{,2}$	$n_{,3}$	$n_{,4}$	n

Cuadro 2.1: Tabla de contingencia

Donde n_{ij} denota el número de transiciones observadas del estado i al estado j; $n_{i,}$ corresponde a la suma de la fila i, que será el número de transiciones desde el estado i, y $n_{,j}$ representan la suma de la columna j, que será el número de transiciones que llegan al estado j, con $i, j \in E$. Luego, podremos estimar las probabilidades de transición p_{ij} utilizando el estadístico $\widehat{p_{ij}} = \dfrac{n_{ij}}{n_{i,}}$.

Así, bajo la hipótesis nula de independencia entre nucleótidos, y dada la secuencia de ADN $\omega = ACGATTA$, por ejemplo, la probabilidad de tal secuencia bajo el modelo de independencia será $P(\{\omega\}) = p_A p_C p_G p_A p_T p_T p_A = p_A^3 p_C^1 p_G^1 p_T^2$.

Tenemos entonces un contraste de independencia con la hipótesis nula

$$H_0 : p_{ij} = p_i \times p_i, i, j \in E.$$

y podemos representar a la hipótesis alternativa como

$$H_a : p_{ij} \neq p_i \times p_i, i, j \in E.$$

Los datos deben aparecer como se muestra en el Cuadro 2.1, donde las filas representan el nucleótido en la posición a y las columnas el nucleótido en la posición $a+1$. Tomemos a $i, j \in E$, $n_{i,}$, $n_{,j}$, n_{ij} como antes y $\Sigma_{i,j \in E} n_{ij} = n$. La idea es realizar el contraste anterior comparando las frecuencias esperadas bajo la hipótesis nula denotadas por $T_{ij} = np_i p_j$, con las frecuencias observadas denotadas por $O_{ij} = n_{ij}$. Si las cantidades p_i y p_j no son conocidas, han de ser estimadas a partir de las frecuencias observadas de la siguiente forma

$$\widehat{p_i} = \frac{n_{i,}}{n}, \widehat{p_j} = \frac{n_{,j}}{n}, \tag{2.1}$$

y por lo tanto

$$T_{ij} = n\widehat{p_i}\widehat{p_j} = \frac{n_{i,} n_{,j}}{n}, \tag{2.2}$$

lo que hace perder $(k-1)(k-1)$ grados de libertad adicionales al estadístico de contraste

$$\chi^2 = \sum_{i=1}^{k}\sum_{j=1}^{k}\frac{(O_{ij}-T_{ij})^2}{T_{ij}} \simeq \chi^2_{(k-1)(k-1)} \tag{2.3}$$

que sigue, bajo la hipótesis nula, una distribución aproximada chi-cuadrado con $(k-1)(k-1)$ grados de libertad, con $k=4$. De esta forma, rechazaremos H_0 si

$$\chi^2 > \chi^2_{(k-1)(k-1),\alpha},$$

para α =0.05, es decir un nivel de significancia del 0.05. Por lo tanto, la hipótesis nula de independencia resulta ser la hipótesis nula de no asociación en la tabla. Pruebas de este tipo muestran que nucleótidos contiguos en posiciones de secuencias de ADN son a veces dependientes, y el modelo de una cadena de Markov de primer orden se ajusta a los datos reales significativamente mejor que el modelo de independencia, tanto en regiones de intrones como de exones.

2.2. Modelado de "Señales" en el ADN

En el contexto de la genómica, la anotación o puntuación es el proceso de marcado de los genes y otras características biológicas de la secuencia de ADN. El primer sistema software de anotación de genomas fue diseñado en 1995 por Owen White, éste construyó un software para localizar los genes, el ARN de transferencia, y otras características; así como para realizar las primeras atribuciones de función a esos genes.

Se conoce que los genes contienen "señales" en el ADN, que son secuencias de ADN con un propósito específico: indicar, por ejemplo, el comienzo y final de la región transcrita o los límites de los exones e intrones. Estas señales se utilizan para reconocer el gen, editarlo correctamente, y para traducirlo apropiadamente en proteína. Si la naturaleza fuese bondadosa, cada señal consistirá en una única secuencia de ADN que no aparecería en ningún otro lugar del ADN excepto donde sirviese para su propósito específico. En la realidad, hay muchas secuencias de ADN que realizan la misma función de señal; llamamos a éstas "miembros" de una señal. Además, los miembros de las señales también aparecen aleatoriamente en el ADN no funcional, haciendo difícil clasificar las señales funcionales de las no funcionales. En la práctica, no todos los miembros de una señal son conocidos. Nuestro objetivo es utilizar los miembros conocidos para evaluar la probabilidad de que una nueva secuencia no caracterizada de ADN sea también un miembro de la señal. Supondremos que los diferentes miembros surgen de antepasados comunes mediante procesos estocásticos, por lo tanto, es razonable construir un modelo estadístico de los datos. Algunas señales requieren solamente modelos simples, mientras que otras necesitan modelos más complejos. Cuando el modelo requerido es complejo y los datos son limitados, debemos tener cuidado eligiendo suposiciones simplificadoras en el modelo para utilizar los datos de la manera más eficiente posible [11].

Capítulo 3

Modelos Ocultos de Markov

Un modelo oculto de Markov (MOM) es un modelo matemático que puede ser usado para describir la evolución de eventos observados que dependen de factores internos, los cuales no son observados. Llamaremos al evento observado "símbolo" y al factor oculto subyacente a la observación "estado". Un MOM consiste de dos procesos estocásticos: un proceso estocástico de estados ocultos y un proceso estocástico de eventos observados. Los estados ocultos forman una cadena de Markov y la distribución de probabilidad de los símbolos observados depende de los estados ocultos subyacentes. Por esta razón, se dice que un modelo oculto de Markov es un proceso estocástico bivariado.

A continuación se define formalmente un MOM, se dan ejemplos sencillos y se describen los problemas asociados a los MOM, así como los algoritmos de programación dinámica utilizados para dar solución a estos últimos.

3.1. Definición de un Modelo Oculto de Markov (MOM)

Un Modelo Oculto de Markov discreto se define como una quíntupla (S, V, Π, A, B), donde

- S es el conjunto finito de estados ocultos, $S = \{s_1, s_2, ..., s_N\}$, N es la cantidad de estados en el modelo. El estado actual en el instante de tiempo t se denota como X_t.
- V es el conjunto de valores o símbolos diferentes que se pueden observar en cada uno de los estados, puede considerarse también un alfabeto finito. Cada uno de los símbolos que un estado puede emitir se denota como $\{v_1, v_2, ..., v_M\}$, donde M es el número de símbolos del alfabeto y cada v_k se refiere a un símbolo diferente. Una secuencia de observaciones se denota como un vector $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ donde cada observación Y_i, es un elemento de V, y T es la cantidad de observaciones en la secuencia.
- $\Pi = \{\pi_1, \pi_2, ..., \pi_N\}$ es la distribución de *probabilidad inicial* de los estados ocultos. De esta forma, $\pi_i = P(X_1 = s_i)$ constituye la probabilidad de que el sistema inicie en el estado s_i.

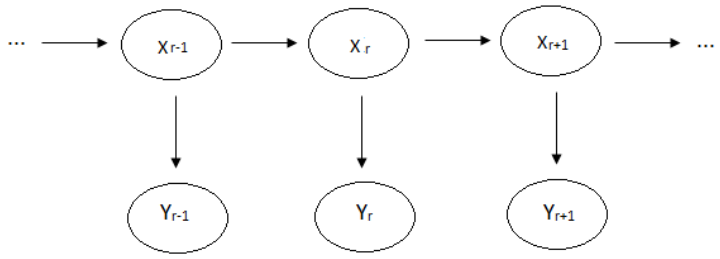


Figura 3.1: Arquitectura general de un modelo oculto de Markov

- $A = \{p_{ij}\}$ es la *matriz de las probabilidades de transición* entre estados, donde $p_{ij} = P(X_t = s_j \mid X_{t-1} = s_i)$ constituye la probabilidad de que el sistema se encuentre en el estado s_i en el instante $t-1$ y pase al estado s_j en el instante t.
- $B = \{e_i(k)\}$ es la *matriz de las probabilidades de emisión*, es decir, cada $e_i(k) = P(Y_t = v_k \mid X_t = s_i)$ constituye la probabilidad de que el sistema, estando en el estado s_i, genere la observación v_k.

La notación más compacta para un MOM es usando sus tres medidas de probabilidad $\lambda = (A, B, \Pi)$.

Veamos ahora dos ejemplos de cómo aplicar un MOM.

Ejemplo 1. Un MOM con dos estados: Imaginemos que en un casino utilizan un dado justo la mayoría de las veces, pero otras utilizan un dado cargado (las caras no tienen la misma probabilidad de aparecer). De esta forma, nuestros estados serán: {"dado libre", "dado cargado"}={"L", "C"} y los símbolos observados 1,2,3,4,5,6. Un modelo de Markov decide cuál de los dos dados juega, y dependiendo del estado del modelo, se aplicarán las probabilidades de emisión para el dado cargado o para el normal. Así, generamos una secuencia de símbolos que es el resultado de estar en dos estados diferentes. Los parámetros de nuestro modelo serán:

- La matriz de transición:

$$A = \begin{array}{c} \\ L \\ C \end{array} \begin{array}{c} \begin{array}{cc} L & C \end{array} \\ \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \end{array}$$

- La matriz de emisión:

$$B = \begin{array}{c} \\ L \\ C \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.5 \end{pmatrix} \end{array}$$

- Con probabilidad inicial Π =(0.5, 0.5).

Y está representado en la Figura 3.2.

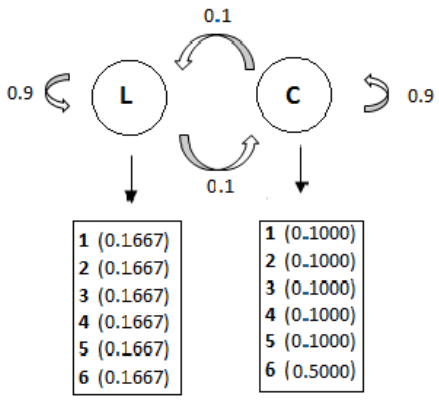


Figura 3.2: Modelo de Markov oculto asociado al ejemplo 1. Las transiciones entre el dado cargado y el dado libre se modelan como una cadena de Markov. Las flechas corresponden a las probabilidades de transición y las columnas a las probabilidades de emisión.

La secuencia observada por tal modelo de Markov oculto podría ser como la siguiente

$$\mathbf{Y}=45536531633635551333626651321416366516 66.$$

Si conocemos las propiedades de los dados y de la cadena de Markov subyacente u oculta, ¿podemos encontrar la secuencia de estados ocultos con mayor probabilidad de haber generado la secuencia de símbolos observados Y?, en otras palabras, ¿podemos averiguar qué dado ha sido utilizado por el casino en cada instante o posición de la secuencia de resultados? Más adelante, con el algoritmo de Viterbi, veremos cómo responder a estas preguntas, aquí simplemente mostramos la secuencia oculta que genera nuestra secuencia visible

$$\textit{Oculto:}\mathbf{X}=11111111111111111111222211111112222222 22$$
$$\textit{Visible:}\mathbf{Y}=45536531633635551333626651321416366516 66.$$

Observemos que el símbolo 6 ocurre con una frecuencia mayor cuando el estado en la secuencia oculta es 2, correspondiente al dado cargado, pero esta dependencia es simplemente probabilística. En las aplicaciones biológicas, aplicaremos nuestro modelo de Markov oculto en un conjunto de datos donde los estados ocultos son conocidos, incluso cuando no sepamos exactamente cuáles son las probabilidades de transición y emisión. Esto permite calcular las matrices de transición y emisión de nuestro modelo basadas en los datos y, por lo tanto, una mejor inferencia en los estados ocultos de nuevos datos.

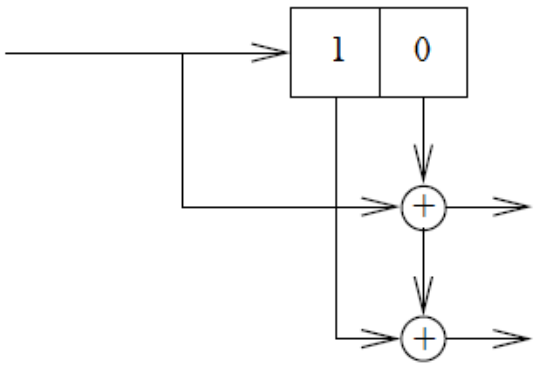


Figura 3.3: Código convolucional de razón $\frac{1}{2}$ con longitud de memoria 2.

Ejemplo 2. Queremos transmitir un mensaje codificado como una secuencia de bits $b_0, ..., b_m$, donde $b_i \in \{0, 1\}$ son los bits y $m \geq 1$ es la longitud del mensaje. Deseamos transmitir el mensaje mediante un canal que sabemos que afectará al mensaje puesto que introduce errores aleatoriamente. Consideraremos canales *discretos*, esto es, se supone que las entradas y las salidas del canal pertenecen a alfabetos finitos: $i_1, ..., i_q$ para las entradas y $o_1, ..., o_l$ para las salidas. Ya que consideraremos canales binarios, entonces las entradas y las salidas del canal de transmisión son bits, así $q = l = 2$ y $i_1, i_2 = o_1, o_2 = \{0, 1\}$.

Consideremos un codificador convolucional, el cual forma parte de las técnicas usadas en canales en los que la señal transmitida se ve corrompida. La característica principal de este código es que cada símbolo de m bits al ser codificado se transforma en un símbolo de n bits, donde m/n es la razón del código ($n \geq m$) [25]. Consideremos un codificador como el antes mencionado de razón $1/2$ con longitud de memoria 2, es decir que el mensaje de longitud m será transformado en un mensaje de longitud $2m$, esto es, enviamos $2m$ bits a través del canal con el fin de introducir algún tipo de redundancia para incrementar nuestra oportunidad de obtener un mensaje libre de errores. El principio de este codificador convolucional es mostrado en la Figura 3.3.

Debido a que la longitud de memoria es 2, hay 4 estados diferentes y el comportamiento de este codificador convolucional puede ser modelado como una máquina de 4 estados, donde el alfabeto de estados es $E = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. Denotamos por X_k el valor del estado en el tiempo k, $X_k = (X_{k,1}, X_{k,2}) \in E$. A la llegada del bit B_{k+1}, el estado se transforma a

$$X_{k+1} = (X_{k+1,1}, X_{k+1,2}) = (B_{k+1}, X_{k,1}).$$

Si la secuencia $\{B_k\}_{k\geqslant 0}$ de los bits de entrada es i.i.d. con probabilidad $P(B_k = 1) = p$, entonces $\{X_k\}_{k\geqslant 0}$ es una cadena de Markov con matriz de transición:

$$A = \begin{array}{c} \\ (0,0) \\ (0,1) \\ (1,0) \\ (1,1) \end{array} \begin{array}{c} \begin{array}{cccc} (0,0) & (0,1) & (1,0) & (1,1) \end{array} \\ \begin{pmatrix} 1-p & 0 & p & 0 \\ 1-p & 0 & p & 0 \\ 0 & 1-p & 0 & p \\ 0 & 1-p & 0 & p \end{pmatrix} \end{array}.$$

Y matriz de emisión:

$$B = \begin{array}{c} \\ (0,0) \\ (0,1) \\ (1,0) \\ (1,1) \end{array} \begin{array}{c} \begin{array}{cccc} (0,0) & (0,1) & (1,0) & (1,1) \end{array} \\ \begin{pmatrix} q & 0 & 0 & 1-q \\ q & 0 & 1-q & 0 \\ 0 & q & 0 & 1-q \\ 0 & q & 1-q & 0 \end{pmatrix} \end{array}.$$

Para cada bit de entrada, el codificador convolucional genera dos salidas de acuerdo a

$$S_k = (S_{k,1}, S_{k,2}) = (B_k \oplus X_{k,2}, B_k \oplus X_{k,2} \oplus X_{k,1})$$

Un codificador convolucional puede ser representado como un diagrama de transición de estado del tipo de la Figura 3.4. Los nodos son los estados y las ramas representan las transiciones que no tienen probabilidad cero. Si indexamos los estados con el índice de tiempo k y el índice de estado m, obtenemos el diagrama de Trellis de la Figura 3.5. El diagrama de Trellis muestra la progresión del tiempo de la secuencia de estados. Para toda secuencia de estados, existe un único camino a través del diagrama de Trellis y viceversa.

3.2. Los Tres Problemas Básicos en los Modelos Ocultos de Markov

Dada la forma del modelo oculto de Markov en la sección anterior, existen tres problemas básicos que deben resolverse para que el modelo pueda ser aplicado a sistemas reales. Dada una secuencia de observaciones $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ y un modelo $\lambda = (A, B, \Pi)$, los problemas a resolver son los siguientes:

- Problema 1 (P1). ¿Cómo podemos calcular de forma eficiente $P_\lambda(Y_1 = y_1, ..., Y_T = y_T)$? o dicho de otro modo ¿cómo calcular la probabilidad de obtener dicha secuencia de observaciones dado un modelo fijo?

- Problema 2 (P2). ¿Cómo encontrar la sucesión de estados ocultos más factible que pueda haber generado dicha secuencia de observaciones? (La que mejor explica la secuencia de observaciones dada).

- Problema 3 (P3). Dada $\mathbf{Y}$ ¿Cómo estimar los parámetros de λ para maximizar $P_\lambda(Y_1 = y_1, ..., Y_T = y_T)$? (El modelo que mejor explica la secuencia de observaciones

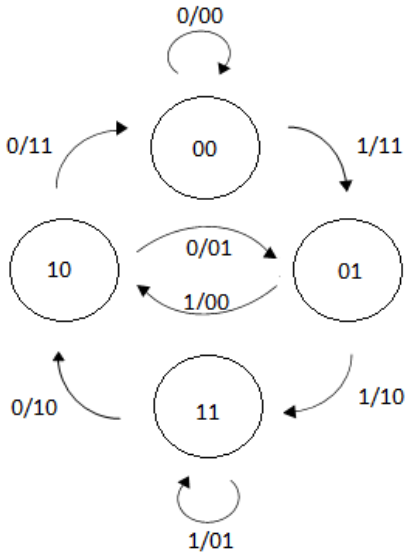


Figura 3.4: Diagrama de estados código convolucional de razón $\frac{1}{2}$ con longitud de memoria 2.

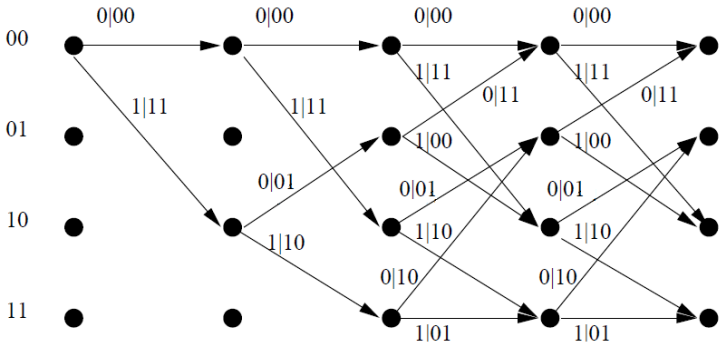


Figura 3.5: Diagrama de Trellis del código convolucional de razón $\frac{1}{2}$ con longitud de memoria 2.

dada), o en otras palabras, *entrenar* los parámetros del MOM dada una secuencia de observaciones. Por "*entrenar*" entenderemos el proponer un modelo a priori e ir mejorándolo al maximizar $P_\lambda(Y_1 = y_1, ..., Y_T = y_T)$.

Observaciones.

a) El primer problema consiste en la evaluación de una secuencia de observaciones dado el modelo. La resolución de este problema provee la probabilidad de que la secuencia fuera generada por ese modelo. Un ejemplo práctico puede ser presuponer la presencia de varios MOM's compitiendo entre sí. La evaluación de la secuencia en cada uno de ellos dará la oportunidad de elegir el modelo que mejor se ajusta con las observaciones.

b) El segundo problema es encontrar la cadena oculta del modelo, es decir, descubrir el camino que siguió la cadena de Markov. Este camino oculto es de utilidad para la clasificación de las observaciones.

c) El tercero de los problemas, es el problema crítico de los MOM puesto que permite adecuar en forma óptima los parámetros de un modelo a una secuencia de observaciones. También resulta el más complejo de resolver. Notemos que la solución a este problema permite la efectiva aplicación de los MOM a una cantidad de casos reales, a los que sería inviable su aplicación, pues determinar a priori las distribuciones correspondientes no es una opción en la mayoría de los casos. En lugar de eso, se dispone de una *secuencia de entrenamiento* con el fin de *enseñar* al MOM cuáles son las distribuciones que mejor se adaptan.

3.3. Solución a los Tres Problemas Básicos

3.3.1. Solución a P1: Probabilidad de una Observación

Se quiere calcular de manera eficiente la probabilidad de una secuencia de observaciones $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$, dado un modelo $\lambda = (A, B, \Pi)$, es decir, $P_\lambda(\mathbf{Y})$.
Recordando que cada Y_t depende solamente de X_t, (la observación en el tiempo t depende únicamente del estado en el tiempo t), entonces para producir la secuencia $\mathbf{Y}$ se debe tener una secuencia de estados $\mathbf{X} = (X_1, X_2, ..., X_T)$. Luego, esto permite inferir un sencillo método de resolución al problema en cuestión, simplemente se podría *calcular la probabilidad de obtener* $\boldsymbol{Y}$ *con cada una de las posibles secuencias de estados de longitud* T *que son posibles con el modelo, y finalmente sumar dichas probabilidades.* Notar que si N es el número de estados del modelo, existen N^T posibles secuencias de estados de longitud T.

Como la probabilidad de que ocurra la secuencia de estados $\mathbf{X} = (X_1, X_2, ..., X_T)$ en el modelo $\lambda = (A, B, \Pi)$ está dada por:

$$P_\lambda(\mathbf{X}) = \pi_{x_1} \prod_{t=1}^{T-1} p_{x_t x_{t+1}}. \tag{3.1}$$

La probabilidad de $\mathbf{Y}$, dada la secuencia $\mathbf{X}$ y el modelo $\lambda = (A, B, \Pi)$ es:

$$P_\lambda(\mathbf{Y} \mid \mathbf{X}) = \prod_{t=1}^{T} P_\lambda(Y_t = y_t \mid X_t = x_t) = e_{x_1}(y_1) \cdot e_{x_2}(y_2) \cdots e_{x_T}(y_T), \tag{3.2}$$

en donde se asume independencia entre observaciones. La probabilidad conjunta de $\mathbf{Y}$ y $\mathbf{X}$ (la probabilidad de obtener la secuencia de observaciones $\mathbf{Y}$ simultáneamente con la secuencia particular de estados $\mathbf{X}$) dado el modelo, como consecuencia de la regla del producto de (3.1) y (3.2) se tiene:

$$P_\lambda(\mathbf{Y}, \mathbf{X}) = P_\lambda(\mathbf{Y} \mid \mathbf{X}) P_\lambda(\mathbf{X}). \tag{3.3}$$

Como se comentó al principio, para obtener $P_\lambda(\mathbf{Y})$ se debe calcular (3.3) para cada secuencia posible de estados de tamaño T del modelo y luego calcular la suma de las mismas, por lo tanto:

$$P_\lambda(\mathbf{Y}) = \sum P_\lambda(\mathbf{Y} \mid \mathbf{X}) P_\lambda(\mathbf{X}). \tag{3.4}$$

Equivalentemente:

$$P_\lambda(\mathbf{Y}) = \sum \left(\pi_{x_1} e_{x_1}(y_1) \prod_{t=1}^{T-1} p_{x_t x_{t+1}} e_{x_{t+1}}(y_{t+1}) \right). \tag{3.5}$$

Se puede interpretar la ecuación (3.5) de la siguiente forma. Al inicio (tiempo t=1), el proceso se encuentra en el estado x_1 con probabilidad π_{x_1} y es generado el símbolo y_1 con probabilidad $e_{x_1}(y_1)$, en t=2, se produce una transición del estado x_1 al estado x_2 con probabilidad $p_{x_1 x_2}$, y se genera el símbolo y_2 con probabilidad $e_{x_2}(y_2)$. El proceso continua hasta que se llega al tiempo $t = T$, con una transición del estado x_{T-1} al estado x_T con probabilidad $e_{x_{T-1} x_T}$ y generando el símbolo y_T con probabilidad $e_{x_T}(y_T)$. En la Figura 3.6 se puede ver una representación gráfica de lo anterior.

Como ya se mencionó antes, si la cantidad de estados del modelo es N, la cantidad de secuencias de estados posibles de longitud T es N^T. Analizando la ecuación (3.5), se puede ver que para cada una de las N^T secuencias de la sumatoria se deben realizar $(2T - 1)$ multiplicaciones y $N^T - 1$ sumas. Por lo tanto, esta forma de calcular $P_\lambda(\mathbf{Y})$ es del orden de $(2TN^T)$ lo que la convierte en impracticable, así que para solucionar este problema debe hacerse uso de técnicas de programación dinámica (ver apéndice A), con el objetivo de recordar soluciones parciales en vez de recalcularlas. A continuación se mostrarán un par de técnicas para ese fin, denominados *proceso de avance* (forward) y *proceso de retroceso*

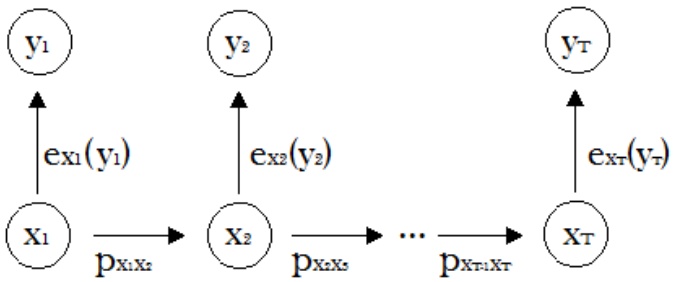


Figura 3.6: Proceso representativo de la ecuación (3.5)

(backward) [19].

Proceso de avance

Considérese la variable $\alpha_t(i)$, definida de la siguiente manera:

$$\alpha_t(i) := P_\lambda(Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t, X_t = i), \tag{3.6}$$

la cual representa la probabilidad conjunta de obtener la secuencia de observaciones parcial dada $(y_1, y_2, ..., y_t)$ hasta el tiempo t, y que el proceso se encuentre en el estado i en el tiempo t, dado el modelo $\lambda = (A, B, \Pi)$. Los resultados de $\alpha_t(i)$, para los estados del modelo en los diferentes momentos de tiempo, se pueden calcular de forma iterativa y luego pueden ser usados para determinar $P_\lambda(\mathbf{Y})$. El proceso es descrito en el siguiente algoritmo

1. Inicialización:
$$\alpha_1(i) = \pi_i e_i(y_1), \quad 1 \leq i \leq N. \tag{3.7}$$

2. Iteración:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) p_{ij}\right] e_j(y_{t+1}), \quad t = 1, 2, ..., T-1; \;\; 1 \leq j \leq N. \tag{3.8}$$

3. Terminación:
$$P_\lambda(Y_1 = y_1, ..., Y_T = y_T) = \sum_{i=1}^{N} \alpha_T(i). \tag{3.9}$$

El primer paso (3.7), inicializa todas las probabilidades de inicio potenciales, es decir, una para cada estado (N en total), como las probabilidades conjuntas de que inicie en el estado i y dicho estado genere la observación y_1.
El paso de iteración (3.8) es el más complejo y es el núcleo del algoritmo de avance. Para

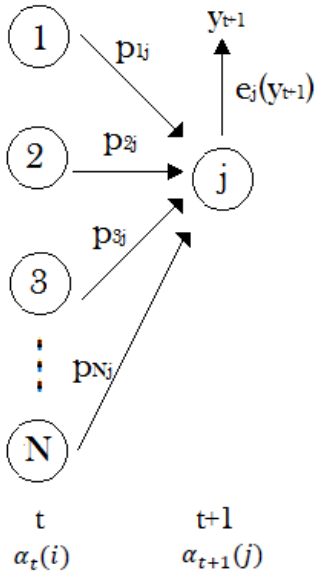


Figura 3.7: Representación gráfica del cálculo de $\alpha_{t+1}(j)$.

entenderlo, primero nótese que $\alpha_t(i)$ representa la probabilidad conjunta de que el proceso genere la sucesión de observaciones $(y_1, y_2, ..., y_t)$, estando en el estado i en el tiempo t, luego $\alpha_t(i)p_{ij}$ representa la probabilidad de que en el tiempo $t+1$ exista una transición desde el estado i hasta el estado j, simultáneamente a la probabilidad conjunta anterior para el tiempo t. Si se suman los productos para cada uno de los N estados posibles en el instante t, y se multiplica esa suma por $e_j(y_{t+1})$, (que es la probabilidad de que el símbolo y_{t+1} sea emitido por el j), se obtiene $\alpha_{t+1}(j)$, que será la probabilidad conjunta de obtener la secuencia parcial de observaciones $(y_1, y_2, ..., y_{t+1})$ y de encontrarse en el estado j en el instante de tiempo $t+1$. Se realiza el cálculo anterior para cada uno de los N estados y para los instantes de tiempo $t = 1, 2, ..., T-1$. En la Figura 3.7 se puede apreciar una representación del paso de iteración.

El paso de terminación (3.9) obtiene $P_\lambda(\mathbf{Y})$ sumando el valor de las N variables $\alpha_T(i)$. Nótese que por definición,

$$\alpha_T(i) = P_\lambda(Y_1 = y_1, Y_2 = y_2, ..., Y_{T-1} = y_{T-1}, Y_T = y_T, X_T = i). \tag{3.10}$$

Es decir, la probabilidad conjunta de obtener la secuencia de observaciones $(y_1, y_2, ..., y_T)$ y que en el tiempo T el proceso se encuentre en el estado i, si se hace extensivo eso a todos los N estados, y se calcula la suma, se tendrá precisamente $P_\lambda(\mathbf{Y})$, obteniendo la ecuación (3.9) el mismo resultado que la ecuación (3.5).

La Figura 3.8 muestra los estados y probabilidades necesarias para el cálculo de $\alpha_4(3)$, donde $\alpha_4(3) = \left(\sum_{i=1}^{5} \alpha_3(i)p_{i3}\right) e_3(y_4)$.

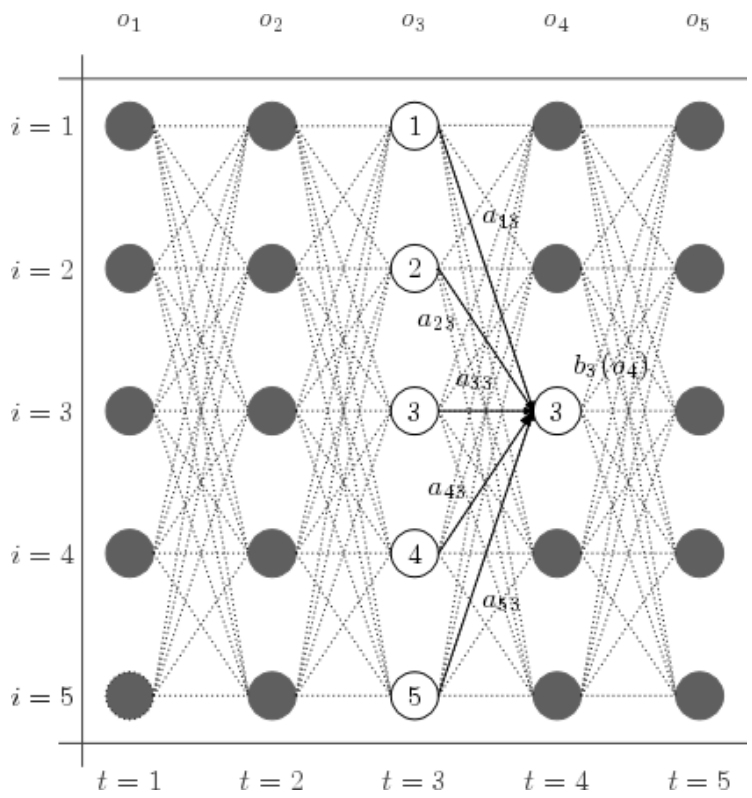


Figura 3.8: Estados y probabilidades necesarias para aplicar el algoritmo de avance.

Proceso de retroceso

Consideramos la variable

$$\beta_t(i) = P_\lambda(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T \mid X_t = i) \tag{3.11}$$

Es decir, la probabilidad de obtener la secuencia de observación parcial $(y_{t+1}, y_{t+2}, ..., y_T)$ desde el instante de tiempo $t+1$ hasta el instante de tiempo T (final), dado que el estado es i en el instante de tiempo t y dado el modelo $\lambda = (A, B, \Pi)$. Luego se pueden resolver las variables $\beta_t(i)$ nuevamente en forma inductiva y calcular el valor de $P_\lambda(\mathbf{Y})$ a través del siguiente algoritmo:

1. Inicialización:
$$\beta_T(i) = 1, \quad 1 \leq i \leq N. \tag{3.12}$$

2. Iteración:
$$\beta_t(i) = \sum_{j=1}^{N} p_{ij}\beta_{t+1}(j)e_j(y_{t+1}), \quad t = T-1, T-2, ..., 1; \;\; 1 \leq i \leq N. \tag{3.13}$$

3. Terminación:
$$P_\lambda(Y_1 = y_1, ..., Y_T = y_T) = \sum_{i=1}^{N} \beta_1(i)\pi_i e_i(y_1). \tag{3.14}$$

El primer paso (3.12), otorga el valor 1 a las N variables $\beta_T(i)$. El segundo paso (3.13), es el paso iterativo, nótese que $\beta_{t+1}(j)$ es la probabilidad de obtener la secuencia de observación parcial $(y_{t+2}, y_{t+3}, ..., y_T)$ dado que el estado en el momento $t+1$ es j y dado el modelo, la misma se multiplica por la probabilidad de transición entre el estado i y el j y por la probabilidad de que el estado j, en el tiempo $t+1$, emita el símbolo y_{t+1}, es decir, $e_j(y_{t+1})$. Obteniendo dicho producto para cada uno de los N estados posibles y sumándolos, se obtiene $\beta_t(i)$, que es precisamente la probabilidad de obtener la secuencia de observaciones parcial $(y_{t+1}, y_{t+2}, ..., y_T)$, dado que el estado en el momento t es i, y dado el modelo. En la Figura 3.9 se puede ver una representación de este paso.
El paso final (3.14) obtiene lo que se quería en la dirección inversa al algoritmo anterior, nótese que en este caso a $\beta_t(i)$ se le debe multiplicar la probabilidad de comenzar en el estado i y la probabilidad de que el estado i emita el símbolo y_1, esto último se consideraba en el algoritmo de avance (siguiendo la simetría) en el paso 1.

Por ejemplo, si quisiéramos calcular la probabilidad de obtener la secuencia de observación parcial $(y_3, ..., y_5)$ desde el momento de tiempo 3 hasta el momento de tiempo final, que en este caso es 5, es decir, $\beta_2(3)$ los estados y probabilidades necesarias serían las mostradas en la Figura 3.10. Y tendríamos que $\beta_2(3) = \sum_{j=1}^{5} p_{3j}\beta_3(j)e_j(y_4)$.

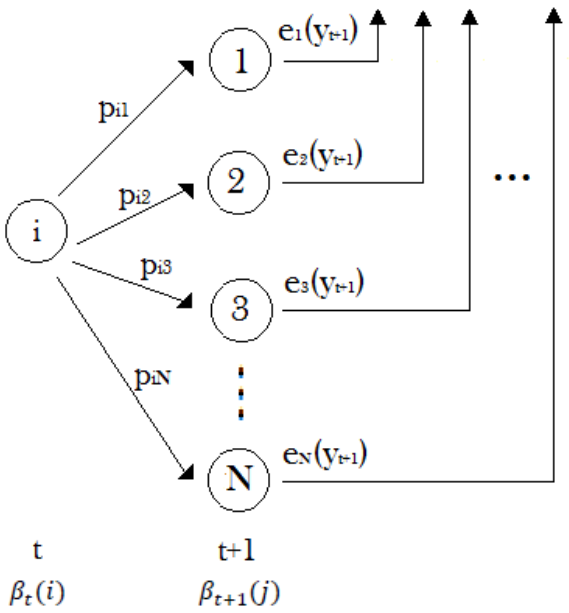


Figura 3.9: Representación gráfica del cálculo de $\beta_t(i)$.

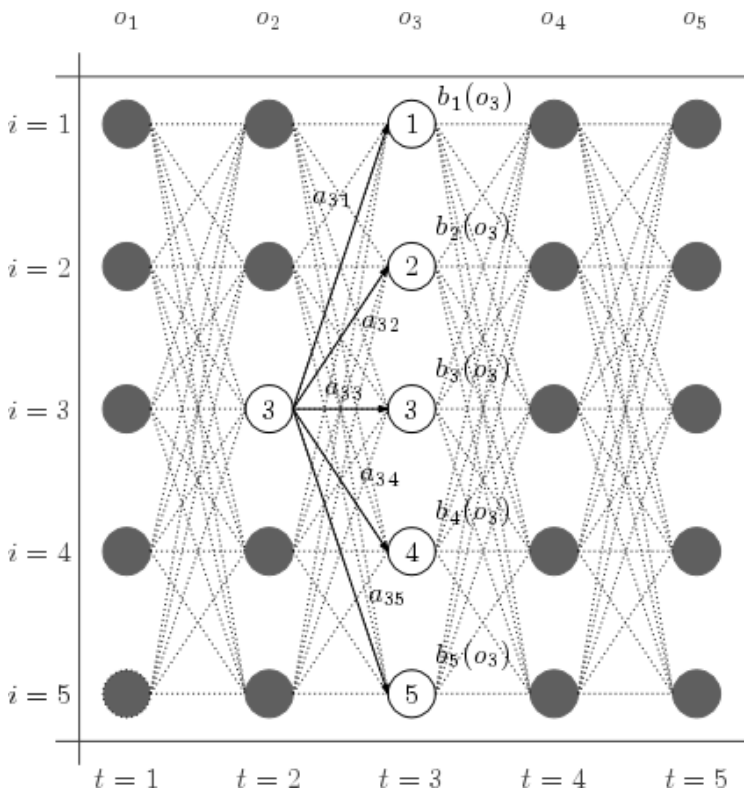


Figura 3.10: Estados y probabilidades necesarias para aplicar el algoritmo de retroceso.

3.3.2. Solución a P2: Secuencia de Estados más Probable

Como mencionamos, el objetivo es encontrar la secuencia de estados oculta óptima, dada una secuencia de observaciones y un modelo. Una forma de solucionarlo es la selección de los estados que son más probables individualmente en cada instante de tiempo. Es decir, encontrar el estado en donde la generación de una observación concreta en el momento t tenga la máxima probabilidad. Consideremos $\gamma_t(i)$ como

$$\gamma_t(i) = P_\lambda(X_t = i \mid Y_1 = y_1, ..., Y_T = y_T).$$

Esto es, la probabilidad de encontrarse en el estado i en el momento t, dada la secuencia de observaciones $\mathbf{Y}$ y el modelo $\lambda = (A, B, \Pi)$.

Usando $\gamma_t(i)$ podemos resolver el problema antes mencionado, es decir encontrar el estado con máxima probabilidad de haber generado una observación concreta en el momento t, se denotará a dicho estado como X_t^*, y será:

$$X_t^* = argmax_{1 \leq i \leq N} \left[\gamma_t(i)\right], \qquad 1 \leq t \leq T. \tag{3.15}$$

La Ecuación (3.15) hace que la cantidad de estados correctos obtenida sea máxima, sin embargo, podría mostrar contradicciones con la matriz de transición de probabilidades, y exhibir estados contiguos no aceptados por dicha matriz. Esto se debe a que la solución obtenida a base de esta ecuación obtiene el estado con mayor probabilidad de generar una observación concreta en cada momento, pero nunca considera la probabilidad de la secuencia de estados que está generando. La solución a este problema radica en descubrir la mejor secuencia considerando todos los instantes de tiempo, es decir, la secuencia de estados $X = (x_1, x_2, ..., x_T)$ que maximiza $P_\lambda(X_1 = x_1, ..., X_T = x_T \mid Y_1 = y_1, ..., Y_T = y_T)$. Para esto, existe un procedimiento que también se basa en técnicas de programación dinámica y es llamado *algoritmo de Viterbi.*

Se quiere descubrir la secuencia de estados más probable $\mathbf{X} = (X_1, X_2, ..., X_T)$, dada la secuencia de observaciones $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ y el modelo $\lambda = (A, B, \Pi)$, para ello se considera $\delta_t(i)$, que se define como:

$$\delta_t(i) = max_{x_1, x_2, ..., x_{t-1}} P(X_1 = x_1, X_2 = x_2, ..., X_t = i, Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t). \tag{3.16}$$

Es decir, $\delta_t(i)$ es la probabilidad de la mejor secuencia de estados que finaliza en el estado i teniendo en cuenta las primeras t observaciones.

Proposición 2. Sean i, j en el espacio de estados y t el tiempo. Se tiene que

$$\delta_{t+1}(i) = \left[max_{1 \leq i \leq N} \delta_t(i) p_{ij}\right] e_j(y_{t+1}). \tag{3.17}$$

Demostración. Para simplificar notación, sólo se escriben las realizaciones.

Se analiza el cálculo de $\delta_t(i)p_{ij}$, el cual se muestra a continuación:

$$\delta_t(i)p_{ij} = \left[max_{x_1,...,x_{t-1}} P_\lambda(x_1, ..., X_t = i, y_1, ..., y_t)\right] P_\lambda(X_{t+1} = j \mid X_t = i)$$

Pero como la maximización no incluye los estados de los momentos t y $t+1$, se puede tomar:

$$\delta_t(i)p_{ij} = max_{x_1,...,x_{t-1}} \left[P_\lambda(x_1, ..., X_t = i, y_1, ..., y_t) P_\lambda(X_{t+1} = j \mid X_t = i).\right]$$

Además se puede ver que:

$$P_\lambda(x_1, x_2, ..., X_t = i, y_1, y_2, ..., y_t) =$$

$$= P_\lambda(x_1, x_2, ..., x_{t-1}, y_1, y_2, ..., y_t \mid X_t = i) P_\lambda(X_t = i).$$

Entonces:

$$P_\lambda(x_1, ..., X_t = i, y_1, ..., y_t) P_\lambda(X_{t+1} = j \mid X_t = i) =$$

$$\begin{aligned}
&= P_\lambda(x_1, ..., x_{t-1}, y_1, ..., y_t \mid X_t = i) P_\lambda(X_t = i) P_\lambda(X_{t+1} = j \mid X_t = i) = \\
&\overset{1}{=} P_\lambda(x_1, ..., x_{t-1}, X_{t+1} = j, y_1, ..., y_t \mid X_t = i) P_\lambda(X_t = i) = \\
&\overset{2}{=} \frac{P_\lambda(x_1, ..., x_{t-1}, X_{t+1} = j, y_1, ..., y_t, X_t = i) P_\lambda(X_t = i)}{P_\lambda(X_t = i)} = \\
&\overset{3}{=} \frac{P_\lambda(x_1, ..., x_{t-1}, X_{t+1} = j, y_1, ..., y_t, X_t = i) P_\lambda P_\lambda(X_t = i)}{P_\lambda(X_t = i)} = \\
&\overset{4}{=} P_\lambda(x_1, ..., x_{t+1}, X_t = i, X_{t+1} = j, y_1, ..., y_t).
\end{aligned}$$

Donde en (1) se utiliza el hecho de que x_{t+1} depende solamente de x_t y del modelo, en (2) y (4) se aplicó la definición de probabilidad condicional y en (3) la regla de la multiplicación. Por lo tanto:

$$\delta_t(i)p_{ij} = max_{x_1,...,x_{t-1}} P_\lambda(x_1, ..., x_{t-1}, X_t = i, X_{t+1} = j, y_1, ..., y_t).$$

Luego, como se debe tomar $max_{1 \le i \le N} \delta_t(i)p_{ij}$ esto indica que el estado en el momento t, también debe maximizarse, es decir, forma parte de la mejor secuencia de estados que termina en este caso en el estado j, y por lo tanto:

$$max_{1 \le i \le N} \delta_t(i)p_{ij} = max_{x_1,...,x_{t-1}} P_\lambda(x_1, ..., X_{t+1} = j, y_1, ..., y_t).$$

Finalmente como:

$$e_j(y_{t+1}) = P_\lambda(y_{t+1} \mid X_{t+1} = j).$$

Y utilizando que la maximización no incluye el estado del momento $t+1$ se tiene que:

$$\left[max_{1 \le i \le N} \delta_t(i)p_{ij}\right] e_j(y_{t+1}) =$$

$$= max_{x_1,...,x_{t-1}} \left[P_\lambda(x_1, ..., X_{t+1} = j, y_1, ..., y_t) P_\lambda(y_{t+1} \mid X_{t+1} = j)\right].$$

Y como:

$$P_\lambda(x_1, ..., X_{t+1} = j, y_1, ..., y_t)P_\lambda(y_{t+1} \mid X_{t+1} = j) =$$

$$\begin{aligned}
&= P_\lambda(x_1, ..., x_t, y_1, ..., y_t \mid X_{t+1} = j)P_\lambda(X_{t+1} = j)P_\lambda(y_{t+1} \mid X_{t+1} = j) = \\
^{1}\ &= P_\lambda(x_1, ..., x_t, X_{t+1} = j, y_1, ..., y_{t+1} \mid X_{t+1} = j)P_\lambda(X_{t+1} = j) = \\
^{2}\ &= \frac{P_\lambda x_1, ..., x_t, y_1, ..., y_{t+1}, X_{t+1} = j)}{P_\lambda(X_{t+1} = j)} P(X_{t+1} = j) = \\
^{3}\ &= \frac{P_\lambda(x_1, ..., x_t, X_{t+1} = j, y_1, ..., y_{t+1}, X_{t+1} = j)P_\lambda}{P_\lambda(X_{t+1} = j)} P_\lambda(X_{t+1} = j) = \\
^{4}\ &= P_\lambda(x_1, ..., X_{t+1} = j, y_1, ..., y_{t+1}).
\end{aligned}$$

En (1) se utiliza el hecho de que y_{t+1} depende sólo de x_{t+1} y del modelo, en (2) y (4) se usó la definición de probabilidad condicional y en (3) la regla de la multiplicación.
Lo que implica que:

$$\begin{aligned}
&[max_{1 \le i \le N} \delta_t(i) p_{ij}]\, e_j(y_{t+1}) = \\
&= max_{x_1, ..., x_t} P_\lambda(x_1, ..., X_{t+1} = j, y_1, ..., y_{t+1}).
\end{aligned}$$

□

Algoritmo de Viterbi.

Se debe calcular $\delta_t(j)$ para cada instante de tiempo t y para cada estado j, para luego construir la secuencia de estados con un procedimiento en retroceso, para ello se necesita recordar el argumento que maximizó la Ecuación (3.17) para cada momento t y para cada estado j. Esto último se almacena en las variables $\varphi_t(j)$. A continuación se describen los pasos del algoritmo:

1. Inicialización:

$$\begin{aligned}
\delta_1(i) &= \pi_i e_i(y_1), \quad 1 \le i \le N. \\
\varphi_1(i) &= 0, \quad 1 \le i \le N.
\end{aligned}$$

2. Iteración:

$$\begin{aligned}
\delta_{t+1}(j) &= max_{1 \le i \le N} \left[\delta_t(i) p_{ij}\right] e_j(y_{t+1}) \quad 1 \le t \le T-1, \quad 1 \le j \le N. \\
\varphi_{t+1}(j) &= argmax_{1 \le i \le N} \left[\delta_t(i) p_{ij}\right] \quad 1 \le t \le T-1, \quad 1 \le i \le N.
\end{aligned}$$

3. Terminación:

$$x_T^* = argmax_{1 \le i \le N} \left[\delta_T(i)\right]. \tag{3.18}$$

3. Reconstrucción de la secuencia de estados:

$$x_t^* = \varphi_{t+1}(x_{t+1}^*) \quad t = T-1, T-2, ..., 1. \tag{3.19}$$

El algoritmo de Viterbi encuentra la secuencia de estados óptima con una eficiencia del orden de $O(N^2T)$.

Ejemplo: el casino deshonesto. Recordamos que la probabilidad de quedarse en cualquiera de los dos estados es 0.9 y la probabilidad de cambiar de estado es 0.1. Además, la probabilidad de obtener cualquier número en el dado libre es siempre la misma $(\frac{1}{6})$, mientras que en el dado cargado la probabilidad de obtener un 6 es 5 veces mayor que de obtener otro número. Veamos pues, un ejemplo sencillo de cómo aplicar el algoritmo de Viterbi para hallar la secuencia oculta dado que tenemos la secuencia de símbolos observada $\mathbf{Y} = (6, 2, 6)$. Así para el símbolo 6 tenemos

$$\frac{1}{2} \times \frac{1}{6} = \frac{1}{12},$$

para el dado libre, mientras que para el dado cargado obtenemos

$$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4} = \mathbf{0.25},$$

ahora, con el siguiente número observado que es el 2, obtenemos

$$\frac{1}{6} \times max\{(\frac{1}{12} \times 0.9, \frac{1}{4} \times 0.1)\} = 0.0125$$

$$\frac{1}{6} \times max\{(\frac{1}{12} \times 0.1, \frac{1}{4} \times 0.9)\} = \mathbf{0.0225}.$$

Finalmente lo aplicamos para el último número observado, es decir, el 6

$$\frac{1}{6} \times max\{(0.0125 \times 0.9, 0.0225 \times 0.1)\} = 0.000375$$

$$\frac{1}{2} \times max\{(0.0125 \times 0.1, 0.0225 \times 0.9)\} = \mathbf{0.005625}.$$

Por lo tanto, tomando los mayores valores de γ, que son los que están en negrita, obtenemos que la secuencia de estados ocultos óptima es $\mathbf{X} = (2, 2, 2)$ para la secuencia de símbolos observada $\mathbf{Y} = (6, 2, 6)$.

Ejemplo 2. Ahora veamos cómo aplicar el algoritmo de Viterbi a un caso particular del Ejemplo 2 visto en la sección anterior, el cual se representa en la Figura 3.11, donde se muestran los datos de entrada y cuáles deberían ser los datos de salida o codificados, además también podríamos obtener los estados correspondientes.

Para ilustrar la aplicación del algoritmo de Viterbi, supondremos que recibimos los datos con algún error en algún bit, todo lo anterior lo vemos representado en el Cuadro 3.1.

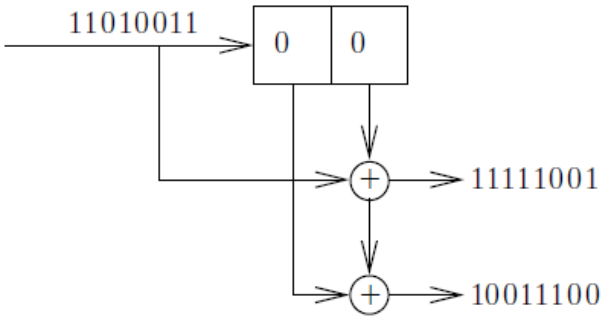


Figura 3.11: Código convolucional de razón 1/2

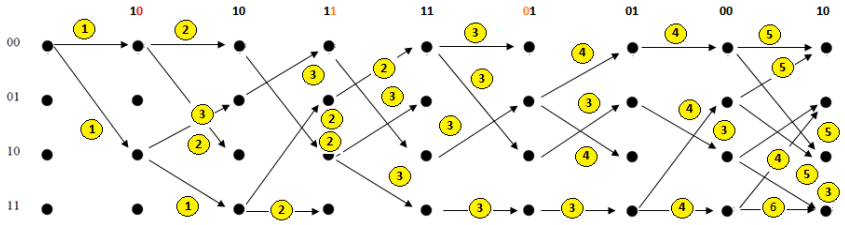


Figura 3.12: Diagrama de Trellis con la distancia de Hamming.

Datos	1	1	0	1	0	0	1	1
Estado presente	11	10	01	10	00	01	11	10
Codificado	10	00	01	11	11	10	10	11
Recibido	10	00	01	01	11	11	10	10

Cuadro 3.1: Datos de entrada y salida correspondientes al ejemplo 2.

Ahora bien, para aplicar el algoritmo de Viterbi hacemos uso de otro concepto: *distancia de Hamming*, la cual se define como la cantidad de bits que cambian de un bloque a otro y se denota como d^H. Básicamente lo que haremos será recorrer el diagrama de Trellis, hallar la distancia de Hamming entre los datos codificados y los datos recibidos para finalmente encontrar la ruta óptima que será aquella cuya distancia recorrida sea menor. Veamos la Figura 3.12, notemos que algunas ramas se han borrado puesto que la distancia era muy grande y por ende quedan descartadas, las etiquetas en amarillo representan la distancia de Hamming.

Finalmente, la ruta óptima es la mostrada en la Figura 3.13.

3.3.3. Solución a P3: Estimación de los Parámetros del Modelo

Conocido como el problema crucial de los MOM, debido a que determinar a priori las probabilidades correctas para un modelo en particular a priori es una tarea inviable en la mayoría de los problemas reales, éste es entre los tres problemas el más complicado

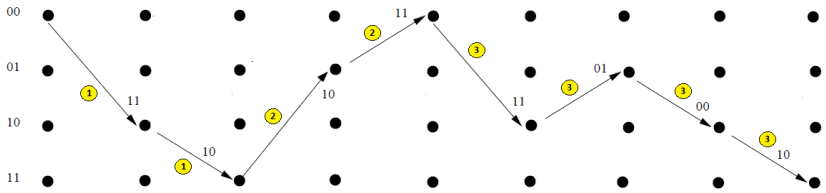


Figura 3.13: Ruta óptima encontrada con el algoritmo de Viterbi.

de resolver y no se conoce una manera analítica de resolverlo; es decir, no se conoce la forma analítica de estimar un conjunto de parámetros para el modelo, que maximice la probabilidad de una secuencia de observaciones en particular de una forma cerrada [20].

Por lo tanto es necesario determinar a través de una *secuencia de entrenamiento*, las probabilidades que *mejor se adaptan* al modelo, como se verá en esta sección, para dicha tarea se analizará un algoritmo, denominado algoritmo de Baum-Welch. La idea del mismo es maximizar *localmente* la probabilidad de una secuencia de observaciones, lo que se consigue a través de un *procedimiento iterativo*.

Como ilustración, el algoritmo de Baum-Welch, intuitivamente, se podría describir de la siguiente manera. Inicialmente no se tiene conocimiento de los parámetros que mejor se ajustan a un modelo $\lambda = (A, B, \Pi)$, pero se dispone de una secuencia de observaciones (denominada secuencia de entrenamiento), que puede utilizarse para maximizar $P_\lambda(\mathbf{Y})$. Como primer paso, se utilizan distribuciones de probabilidad preseleccionadas o se eligen aleatoriamente, entonces se identificarán las transiciones y símbolos de observación más probables. Se maximizarán las probabilidades de dichas transiciones y símbolos, y de esta forma se tendrá un nuevo modelo revisado y *mejor* que el anterior, es decir, con una mayor probabilidad de haber generado la secuencia de entrenamiento dada. Este proceso, denominado *proceso de entrenamiento*, se repite hasta que el modelo generado por el mismo, no difiera del generado en el paso anterior, o sea, hasta que no pueda *mejorarse* el modelo.

Pasando a una descripción formal, primero se define $\xi_t(i, j)$, que será la probabilidad conjunta de que el proceso se encuentre en el estado i, en el momento t, y que se produzca una transición hacia el estado j en el momento $t + 1$, dada la secuencia de observaciones $\mathbf{Y}$ y dado el modelo $\lambda = (A, B, \Pi)$, es decir, $\xi_t(i, j) = P_\lambda(X_t = i, X_{t+1} = j \mid \mathbf{Y})$. Se puede demostrar usando las definiciones de $\alpha_t(i)$ y $\beta_t(i)$ de la sección 3.3.1 que

$$\xi_t(i, j) = \frac{\alpha_t(i) p_{ij} e_j(y_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_t(k) p_{kl} e_l(y_{t+1}) \beta_{t+1}(l)}. \tag{3.20}$$

Sea $\gamma_t(i)$ la probabilidad de que el proceso se encuentre en el estado i en el momento t, dada una secuencia de observaciones y dado el modelo. Razón por la cual, es posible relacionar $\gamma_t(i)$ con $\xi_t(i, j)$, notar que la diferencia, es que la última requiere que el proceso se encuentre en el estado j en el momento $t + 1$, es decir,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j). \tag{3.21}$$

Si se suma $\gamma_t(i)$ a través del tiempo (desde $t = 1$ hasta $t = T-1$) se obtendrá el *número esperado de veces que el proceso efectúa transiciones desde el estado i*, y si se efectúa el mismo procedimiento con $\xi_t(i,j)$, se obtiene el *número esperado de veces que el proceso produce transiciones desde el estado i al estado j*:

$$\sum_{t=1}^{T} \gamma_t(i), \tag{3.22}$$

el cual es el número esperado de transiciones desde el estado i para una secuencia de observaciones $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ y

$$\sum_{t=1}^{T} \xi_t(i,j), \tag{3.23}$$

representa el número esperado de transiciones desde el estado i hacia el estado j para $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$.

Así, utilizando las expresiones (3.22), (3.23) y la definición de $\gamma_t(i)$, se puede obtener un método para lograr la reestimación de los parámetros de un MOM. Sea $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\Pi})$ el modelo resultante de la reestimación de los parámetros, con $\overline{A} = \{\overline{p_{ij}}\}$ $1 \leq i, j \leq N$, $\overline{B} = \{\overline{e_j(k)}\}$ $1 \leq j \leq N$, $1 \leq k \leq M$ y $\overline{\Pi} = \{\overline{\pi_i}\}$ $1 \leq i \leq N$, entonces se pueden utilizar las siguientes fórmulas de reestimación:

$$\overline{\pi_i} = \gamma_1(i), \tag{3.24}$$

$$\overline{p_{ij}} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}, \tag{3.25}$$

$$\overline{e_j(k)} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \tag{3.26}$$

donde, la primera ecuación será la cantidad de veces que el proceso se encuentra en el estado i al momento $(t = 1)$, la segunda el número esperado de transiciones desde el estado i al j, sobre el número esperado de transiciones desde el estado i y la última expresión el número esperado de veces en las que el proceso se encuentra en el estado j, observando el símbolo v_k sobre el número de veces en las que el proceso se encuentra en el estado j. Si el modelo inicial es $\lambda = (A, B, \Pi)$, entonces Baum demostró [20, 19] que se verifica que $\lambda = \overline{\lambda}$, o bien, el modelo $\overline{\lambda}$ tiene una probabilidad mayor que λ, donde la mayor probabilidad se refiere a que se verifica la desigualdad $P_{\overline{\lambda}}(\mathbf{Y}) > P_{\lambda}(\mathbf{Y})$, lo que significa que se ha encontrado un modelo $\overline{\lambda}$, que posee mayor probabilidad de generar la secuencia de observaciones $\mathbf{Y}$. Esto asegura que si se repite el proceso tomando en cada paso el nuevo modelo como la base para el posterior cálculo, se producirá un acercamiento a los parámetros reales. Lo anterior es la plataforma para el siguiente algoritmo.

1. Elegir distribuciones de probabilidad (A, B, Π), tal que $\lambda = (A, B, \Pi)$.

2. Calcular

$$\overline{\pi_i} = \gamma_1(i),\ \overline{p_{ij}} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)},\ \overline{e_j(k)} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}\ 1 \leq i, j \leq N, 1 \leq k \leq M. \quad (3.27)$$

3. Hacer $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\Pi})$.

4. Si $\overline{\lambda} = \lambda$ terminar y devolver $\overline{\lambda}$, si no es así, tomar a λ como $\overline{\lambda}$ y volver al paso 2.

Capítulo 4

Aplicación de los MOM a Secuencias de ADN

La mayoría de los escritos sobre los modelos de Markov ocultos pertenecen a la literatura del reconocimiento ortográfico, donde fueron aplicados por primera vez a principios de 1970 [19].

Con la aparición de los proyectos genómicos, se dispone de una gran cantidad de secuencias y con ellas el problema de cómo extraer de esos datos, experimentalmente obtenidos, la información subyacente, es decir, cómo descubrir las propiedades estadísticas o determinísticas que permitan hacer análisis, modelos y juntamente con la generación de estos últimos, obtener hipótesis que se confirmen a través de experimentación.

Así que nuevamente aparecen los MOM como una herramienta para el análisis de datos genómicos, donde la secuencia de símbolos observada será la secuencia de ADN o de proteínas y la secuencia de estados ocultos es la información subyacente que hay que descubrir.

4.1. P1: Probabilidad de una observación.

Aplicaremos el algoritmo de avance, descrito en el capítulo anterior, con el fin de evaluar la probabilidad de una observación dado un MOM.

Consideremos el siguiente MOM simple. El modelo está compuesto por dos estados: H (alto contenido del dinucleótido GC) y L (bajo contenido del dinucleótido GC). Podríamos suponer que el estado H caracteriza el ADN codificante, mientras que el estado L caracteriza el ADN no codificante. Representamos gráficamente el modelo como en la Figura 4.1.

Ahora bien, considere la secuencia de observaciones $\mathbf{Y} = GGCA$. ¿Cuál es la probabilidad, $P(\mathbf{Y})$, de que la secuencia haya sido generada por el MOM descrito? Esta probabilidad puede ser calculada usando el algoritmo de avance-retroceso descrito anteriormente, vemos que:

La probabilidad de que la secuencia $\mathbf{Y} = GGCA$ haya sido generada por el MOM es

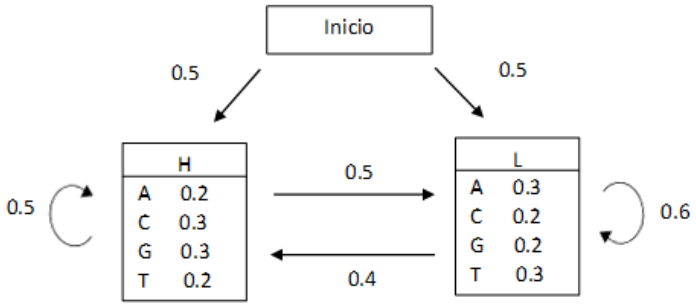


Figura 4.1: Estados y probabilidades necesarias para aplicar el algoritmo de avance.

	Inicio	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3+0.1*0.4*0.3=0.0345	... + ...	0.0013767
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027	... + ...	0.0024665

$P(\mathbf{Y}) = 0{,}0038432$.

Para evaluar la significancia de este valor, debemos compararlo con la probabilidad de que la secuencia $\mathbf{Y}$ no haya sido generada por el modelo, a esta probabilidad la denotaremos P_{bg}. Si todos los nucléotidos tienen la misma probabilidad, $p_{bg} = 0{,}25$; la probabilidad de observar $\mathbf{Y}$ es $P_{bg}(\mathbf{Y}) = p_{bg}^4 = 0{,}25^4 = 0{,}00396$.

Así que para este ejemplo en particular, es más probable que la secuencia no haya sido generada por el MOM ($P_{bg} > P$).

4.2. P2: Secuencia de estados ocultos más probable.

En genética, islas CpG o islas CG son regiones genómicas que contienen una alta frecuencia de los sitios CpG. La "p" en CpG se refiere al enlace fosfodiéster entre la citosina y la guanina, lo que indica que el C y el G están uno junto al otro en secuencia, independientemente de ser de una sola o de doble hebra. La definición formal de una isla CpG es una región con al menos 200 pb (pares de bases), y un porcentaje de GC que es mayor que 50 %, y con una razón del promedio "observado/esperado" mayor que 60 %. Este cociente se calcula dividiendo la proporción de dinucleótidos CpG en la región entre lo esperado cuando se asume independencia en una distribución multinomial, es decir, mediante la siguiente fórmula:

$$\frac{O}{E} = \frac{\frac{CpG}{L}}{\frac{C}{L} * \frac{G}{L}},$$

donde L es el número de pares de bases en la secuencia.

Las islas CpG constituyen aproximadamente el 40 % de los promotores de genes de mamíferos, es decir, los que marcan el inicio del proceso de transcripción del ADN, al resto del genoma que no constituye parte de las islas CpG lo llamaremos océano. A los estados correspondientes a las islas los denotaremos por la letra correspondiente y un +, es decir, A+, C+, G+ y T+. Mientras que a los estados de los océanos le corresponderá un signo menos, es decir, A-, C-, G- y T-. Algunas de las características de las islas CpG son las siguientes:

- Hay más C y G en las islas (más A y T en los océanos).
- La probabilidad de hallar una G después de un nucleótido será mayor en una isla si en la posición actual hay una C que si no la hay.

Se han realizado diversos estudios con el fin de identificar las regiones correspondientes a las islas CpG, entre ellos podemos encontrar el artículo "Redefining CpG islands using hidden Markov Models" (ver [14]) donde se usa una variante de los Modelos Ocultos de Markov llamada Modelos Ocultos de Markov jerárquicos.

Veamos un ejemplo de aplicación, utilizaremos la teoría estudiada anteriormente para buscar regiones de islas CpG en un organismo en particular, el del ratón. Analizaremos el cromosoma 10 de este organismo, utilizaremos el software R. Primero descargaremos el software *Bioconductor*, el cual es un proyecto basado en R y que nos permite acceder, analizar y entender datos en genómica, en nuestro caso será usado para acceder y analizar la información correspondiente al genoma del ratón, así mismo, utilizaremos las islas CpG estudiadas en el artículo "Redefining CpG islands using hidden Markov models" como datos de entrenamiento, pues calcularemos las tablas de contingencia de todos los posibles dinucleótidos en las regiones de las islas y en las de los océanos, con el fin de calcular uno de los paramétros de los Modelos Ocultos de Markov: la matriz de transición. Finalmente programaremos el algoritmo de Viterbi con el fin de determinar la secuencia de estados ocultos que genera la secuencia observada (Ver apéndice B, 1. Algoritmo de Viterbi).

Empezamos descargando el software Bioconductor, y el genoma correspondiente al ratón

```
[26] "BSgenome.Mmusculus.UCSC.mm10"
[27] "BSgenome.Mmusculus.UCSC.mm8"
[28] "BSgenome.Mmusculus.UCSC.mm9"
```

Mouse genome

```
|
|  organism: Mus musculus (Mouse)
|  provider: UCSC
|  provider version: mm9
|  release date: Jul. 2007
|  release name: NCBI Build 37
|
```

chr1	chr2	chr3	chr4	chr5
chr6	chr7	chr8	chr9	chr10
chr11	chr12	chr13	chr14	chr15
chr16	chr17	chr18	chr19	chrX
chrY	chrM	chr1_random	chr3_random	chr4_random
chr5_random	chr7_random	chr8_random	chr9_random	chr13_random
chr16_random	chr17_random	chrX_ramdom	chrY_random	chrUn_random

Accedemos a la información correspondiente al cromosoma 10:

129993255-letter "MaskedDNAString" instance (# for masking)
seq: ##...TTTTTTATTT-TAGTTGCATTCATTTGCCTATGAATT
masks:

	maskedwidth	maskedratio	active	names	desc
1	3145403	2.419666e-02	TRUE	AGAPS	assembly gaps
2	3	2.307812e-08	TRUE	AMB	intra-contig ambiguities
3	54256425	4.173788e-01	FALSE	RM	RepeatMasker
4	2861531	2.201292e-02	FALSE	TRF	Tandem Repeats Finder [period¡=12]

all masks together:

maskedwidth	maskedratio
57487075	0.4422312

all active masks together:

maskedwidth	maskedratio
3145406	0.02419669

> length(crom10)

[1] 129993255

y una vez obtenida ésta, tomamos nuestros datos de referencia y que llamaremos de entrenamiento, los cuales serán las posiciones de las islas de acuerdo al artículo anteriormente mencionado.

chr	start	end	length	CpGcount	GCcontent	pctGC	obsExp
chr10	3001252	3001468	217	7	104	0.479	0.562
chr10	3001990	3002096	107	6	54	0.505	0.900
chr10	3102618	3102897	280	12	147	0.525	0.627
chr10	3103777	3103948	172	7	87	0.506	0.640
chr10	3104651	3105129	479	30	245	0.511	0.965
chr10	3133948	3135475	1528	117	969	0.634	0.764

Donde las columnas del objeto anterior indican, en la primera columna el cromosoma, la segunda y tercera columna la posición inicial y final de la isla respectivamente, la cuarta columna la longitud de las regiones de islas, la quinta el radio de las islas, la sexta el número de dinucleótidos CG en cada isla, la séptima el porcentaje de dinucleótidos CG y la última el promedio de la razón "observado/esperado".

Por otra parte, cuando nos fijamos en la clase de nuestro objeto "crom10" pudimos ver que existen posiciones dentro del cromosoma 10 que no se muestran, accediendo a los datos obtenemos que hay existen 3145406 nucleótidos desconocidos. Ahora bien, nosotros buscamos determinar los parámetros del Modelo Oculto de Markov, así que para ello es necesario eliminar estos nucleótidos desconocidos y trabajar solamente con aquellos que han sido determinados.

Después de eliminar los nucleótidos desconocidos se calcula la matriz de transición de acuerdo a los 8 estados que ya mencionamos con anterioridad. Se calcula de la siguiente forma: se calculan las tablas de contingencia de todos los posibles dinucleótidos en las regiones de islas, así como en las regiones de océanos obteniendo dos matrices de 4 x 4. Estas matrices formarán los bloques diagonales de la que finalmente será la matriz de transición, los bloques restantes medirán las transiciones de islas a océanos y viceversa, para calcular éstas, se toma la primera letra antes de una isla y la primera después de ésta y se calculan las transiciones con la primera de la isla y la última, respectivamente.

Y se calcula la matriz de transición, la cual se muestra en la siguiente Tabla.

	A+	*C+*	*G+*	*T+*	*A-*	*C-*	*G-*	*T-*
A+	0,0009708738	0,0009708738	0,0009708738	0,0009708738	0,2009708738	0,1776699029	0,3038834951	0,3135922330
C+	0,0009165903	0,0009165903	0,0009165903	0,0009165903	0,2951420715	0,0164986251	0,3330522456	0,3116406966
G+	0,0007824726	0,0007824726	0,0007824726	0,0007824726	0,2832550861	0,3513302034	0,2206572770	0,1416275430
T+	0,0009624639	0,0009624639	0,0009624639	0,0009624639	0,2329162656	0,3493743985	0,2146294514	0,1992300289
A-	0,0000000000	0,9957537155	0,0000000000	0,0000000000	0,0010615711	0,0010615711	0,0010615711	0,0010615711
C-	0,0000000000	0,9971448965	0,0000000000	0,0000000000	0,0007137759	0,0007137759	0,0007137759	0,0007137759
G-	0,0000000000	0,9965811966	0,0000000000	0,0000000000	0,0008547009	0,0008547009	0,0008547009	0,0008547009
T-	0,0000000000	0,9953970081	0,0000000000	0,0000000000	0,0011507480	0,0011507480	0,0011507480	0,0011507480

Tabla. Matriz de transición.

Y se calcula la matriz de emisión:

$$B = \begin{array}{c} \\ A+ \\ C+ \\ G+ \\ T+ \\ A- \\ C- \\ G- \\ T- \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Se define de la siguiente manera la probabilidad inicial:

$$(0{,}2925967, 0{,}2067037, 0{,}2073051, 0{,}2933946, 0, 0, 0, 0)$$

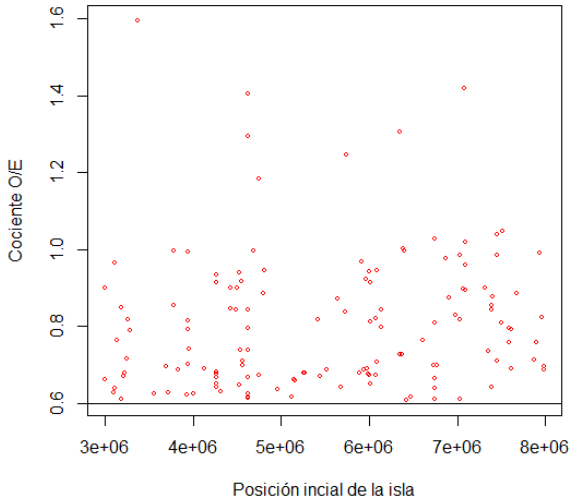
Luego, programando el algoritmo de Viterbi y aplicándolo a la secuencia de menor longitud (48) obtenemos:

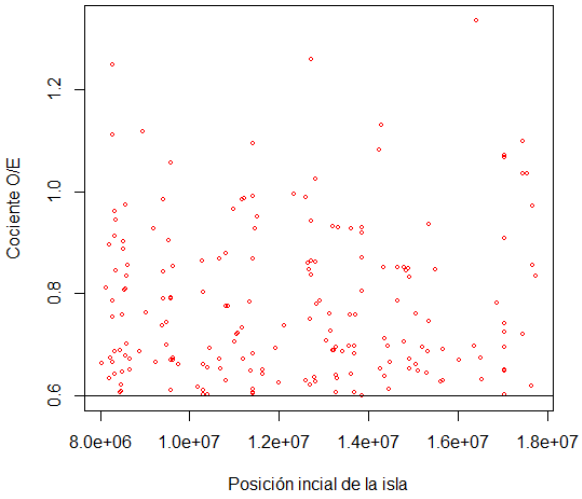
[1]"A_+""A_-""C_+""A_-""A_-""T_+""T_+""A_-""C_-""A_-""C_+""A_-"
[13]"G_-""A_-""A_-""A_-""T_-""A_-""A_-""A_+""A_-""G_+""T_+""A_-"
[25]"A_-""G_+""A_-""T_+""A_-""A_-""G_-""A_-""A_-""A_-""C_+""T_+"
[37]"G_+""C_+""C_-""A_-""C_+""A_-""T_+""A_-""A_-""A_+""C_+""A_-"

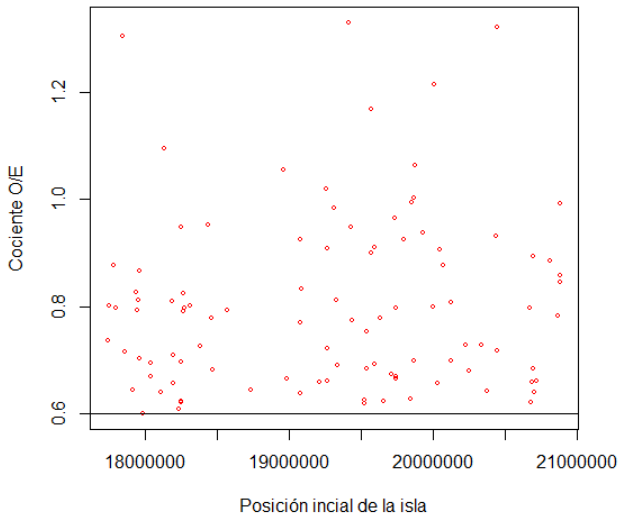
Así que finalmente se encuentra la secuencia oculta que generó la secuencia observada mediante la aplicación del algoritmo de Viterbi.

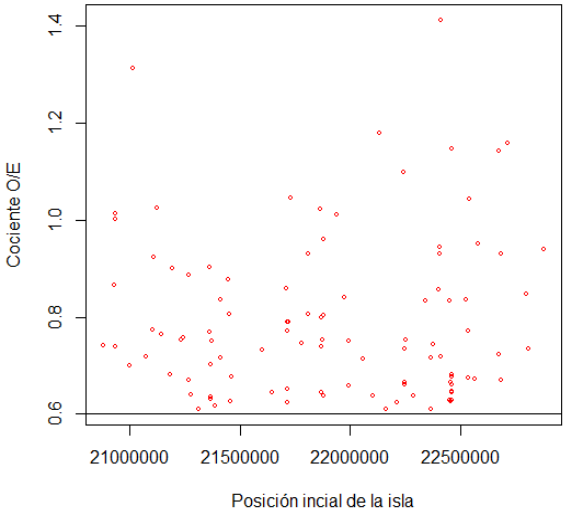
En la Figura 4.1.1 se incluyen las gráficas correspondientes a las islas CpG.

Figura 4.1.1. Regiones de islas CpG









4.3. P3: Estimación de los parámetros del modelo.

Este problema, puede aplicarse a una proteína y encontrar los parámetros para evaluar si pertenece a determinada familia.

4.3.1. Análisis de Proteínas

Las proteínas son moléculas formadas por cadenas de aminoácidos, que están representados por 20 letras del alfabeto, las cuales son: A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y. Así que, de forma análoga al caso del ADN, una proteína es representada por una secuencia de letras tomadas del alfabeto anterior.

Las proteínas se pueden agrupar en familias, es decir, en grupos que están relacionadas evolutivamente. Las proteínas que pertenecen una familia descienden de un antepasado común y poseen estructura, funciones y secuencias similares, así pues, para determinar qué proteínas pertenecen a una familia se busca que esa similitud sea significativa, lo cual se evalúa haciendo uso de métodos de alineamiento de secuencias.

Luego, al determinar si una secuencia desconocida es similar, en algún sentido, a secuencias conocidas podremos identificar y predecir su estructura y función.

Métodos de alineamiento

Las tarea más básica del análisis de secuencias es determinar si dos secuencias están relacionadas. Esto se hace primero alineando las secuencias (o parte de ellas) y luego decidir si es más probable que la alineación resulte porque las secuencias están relacionadas o sólo por azar.

Los puntos claves son:

- El tipo de alineación a considerar.
- El sistema de puntuación utilizado para clasificar las alineaciones.
- El algoritmo utilizado para encontrar la puntuación óptima de la alineación.
- Los métodos estadísticos utilizados para evaluar la significancia de la puntuación del alineamiento.

Cuando comparamos secuencias, buscamos evidencia de que provienen de un ancestro en común, aún cuando haya habido un proceso de mutación. El proceso mutacional básico que se considera son *sustituciones*, el cual cambia los residuos en una secuencia, e *inserciones* y *deleciones*, el cual aumenta o quita residuos. Nos referiremos tanto a las inserciones como a las deleciones, como *huecos*.

La puntuación total que asignaremos a una alineación será la suma de los términos para cada par de residuos alineados, más los términos para cada hueco.

Dado un sistema de puntuación, necesitamos un algoritmo para encontrar el alineamiento óptimo de un par de secuencias. Cuando ambas tienen la mismma longitud n, sólo

hay un posible alineamiento global de las secuencias completas, pero se complica cuando se permiten huecos (o cuando buscamos alineamientos locales entre subsecuencias de dos secuencias). Existen $\binom{2n}{n}$ posibles alineamientos globales entre 2 secuencias de longitud n. Sin embargo, no es computacionalmente viable enumerarlas todas, incluso para valores moderados de n, así que nuevamente haremos uso de un algoritmo de programación dinámica. Más adelante describimos explícitamente los algoritmos usados para el alineamiento global y local de secuencias de proteínas.

Alineamiento global

El primer problema que consideramos es el obtener el alineamiento global óptimo entre dos secuencias permitiendo huecos. El algoritmo de programación dinámica utilizado para resolver este problema es conocido como algoritmo de Needleman-Wunsch propuesto en 1970, pero la versión más eficiente fue dada por Gotoh en 1982 [13].

La idea es construir una alineación óptima usando las soluciones previas para alineamientos óptimos de subsecuencias más pequeñas.

Para implementar el algoritmo de Neddleman-Wunsch, es necesario definir:

- Una función de puntuación (σ). Definimos la puntuación que daremos a una mutación por sustitución, ésta toma valores en un subconjunto finito de $\mathbb{Z}$. Por ejemplo: 1 para coincidencia y -1 para no coincidencia.

- Una penalización por huecos (ς). Define la puntuación que daremos a una mutación por inserción o deleción, toma un valor de un subconjunto finito de $\mathbb{Z}^-$. Por ejemplo: -2.

- Una relación de recurrencia T. Define las acciones que repetiremos en cada iteración del algoritmo, se define como

$$T(i,j) = max \left\{ \begin{array}{l} T(i-1,j-1) + \sigma\left(S_1(i), S_2(j)\right), \\ T(i-1,j) + \varsigma, \\ T(i,j-1) + \varsigma \end{array} \right\}. \tag{4.1}$$

  Donde S_k es la k-ésima secuencia a alinear de tamaño m o n y con $i \in \{0,1,...,m+1\}$ y $j \in \{0,1,...,n+1\}$.

Ahora bien, el algoritmo consta de dos partes:

1. Llenar una matriz o tabla usando la relación de recurrencia previamente definida. Para construir esta matriz hay algunos aspectos a considerar como lo son:
   - Siendo S_1 de longitud m y S_2 de longitud n, la matriz será de tamaño $(m+1) \times (n+1)$.
   - T(0,0)=0.
   - Se llena de izquierda a derecha y de arriba a abajo.

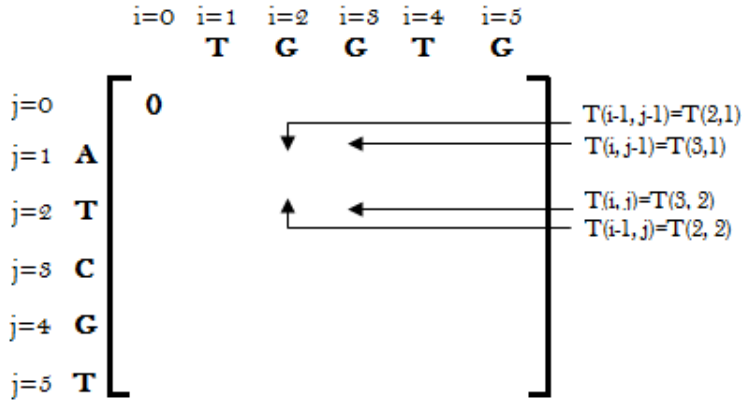


Figura 4.2: Primer paso para llenar la matriz

- Calculamos los valores de $T(i, j)$ para la fila 0 de izquierda a derecha.
- Calculamos los valores de $T(i, j)$ para la fila 1, luego la fila 2, etcétera.

2. El camino hacia atrás. Usamos la matriz del paso 1 para encontrar el mejor camino hacia atrás y por lo tanto el mejor alineamiento.

Ejemplo. Sean S_1 = TGGTG y S_2 = ATCGT secuencias a alinear, haremos uso del algoritmo de Needleman-Wunsch con el fin de encontrar el mejor alineamiento. Así, definimos σ como

$$\sigma\left(S_1(i), S_2(j)\right) = \begin{cases} 1 & coincidencia \\ -1 & no \quad coincidencia \end{cases}$$

y $\varsigma = -2$, finalmente a T como en la Ecuación (4.1).

Para construir la matriz, como ya hemos mencionado, hacemos uso de la relación de recurrencia y los aspectos antes mencionados. En nuestro caso la matriz será de tamaño 6×6.

El proceso anteriormente descrito se ilustra en las Figuras 4.3, 4.4, y así sucesivamente, hasta obtener la matriz representada en la Figura 4.5.

Luego, para llevar a cabo la segunda parte, empezamos en la esquina inferior derecha y seguimos las flechas para calcular el mejor valor para esa celda, y así sucesivamente hasta encontrar el camino hacia atrás que será el mejor alineamiento, lo cual se ilustra en la Figura 4.6.

Por lo tanto, obtenemos que el mejor alineamiento es el mostrado en el Cuadro 4.1.

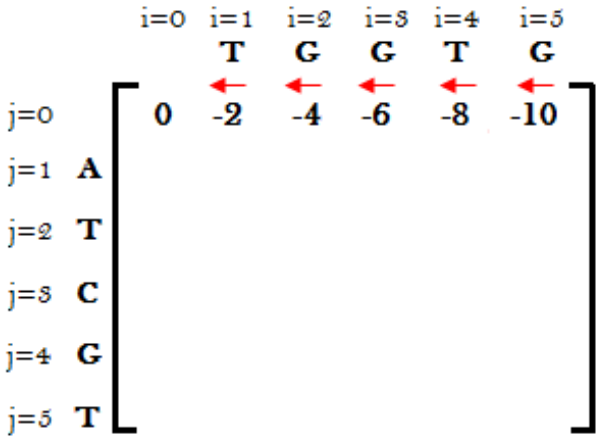


Figura 4.3: Segundo paso para llenar la matriz

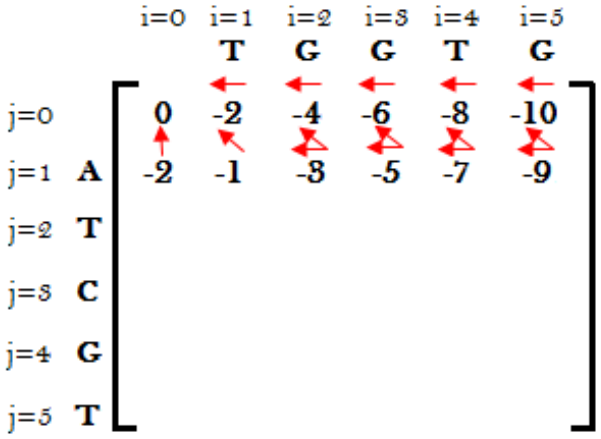


Figura 4.4: Tercer paso para llenar la matriz

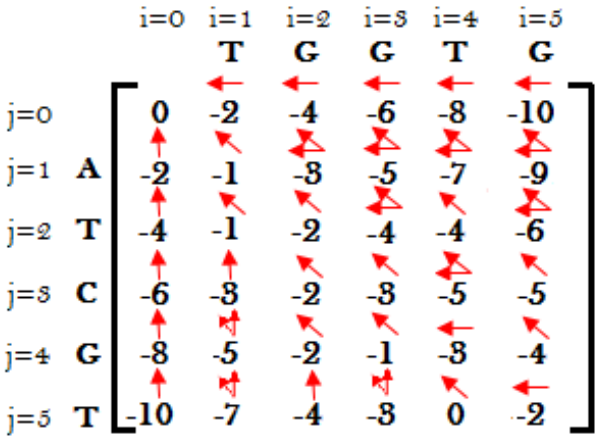


Figura 4.5: Matriz construida usando la relación de recurrencia.

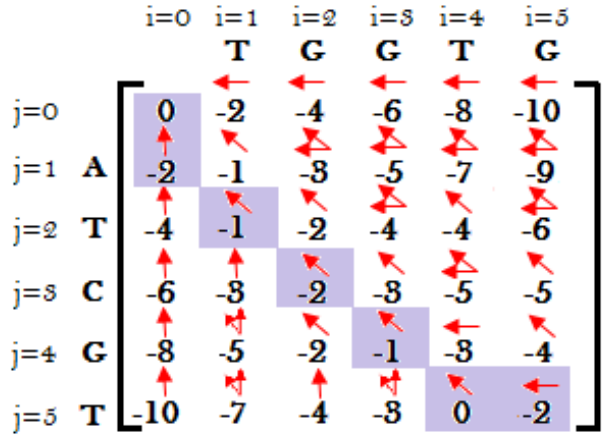


Figura 4.6: Alineamiento usando el algoritmo Needleman-Wunsch.

-	T	G	G	T	G
	|		|	|	
A	T	C	G	T	-

Cuadro 4.1: Alineamiento óptimo encontrado con el Algoritmo de Needleman-Wunsch.

Como aplicación a datos reales, descargamos la secuencia correspondiente a la proteína denominada Neurofascin de dos organismos diferentes: Gallus Gallus y Mus Musculus, que corresponden al pollo y al ratón, las cuales se encuentran disponibles en la base de datos del laboratorio NCBI (National Center for Biotechnology Information).

Se programó en R el algoritmo de Needleman Wunsch como se muestra en el apéndice C (1. Algoritmo de Needleman Wunsch) y lo aplicamos a las dos secuencias antes mencionadas. Obteniendo:

Se muestra un fragmento de la matriz de alineamiento:

.... A L T W T R V H L D T I Q

-1716 -1718 -1720 -1722 -1724 -1726 -1728 -1730 -1732 -1734 -1736 -1738 -1740

M -1713 -1715 -1717 -1719 -1721 -1723 -1725 -1727 -1729 -1731 -1733 -1735 -1737

A -1710 -1712 -1714 -1716 -1718 -1720 -1722 -1724 -1726 -1728 -1730 -1732 -1734

V -1707 -1709 -1711 -1713 -1715 -1717 -1719 -1721 -1723 -1725 -1727 -1729 -1731

S -1704 -1706 -1708 -1710 -1712 -1714 -1716 -1718 -1720 -1722 -1724 -1726 -1728

V -1701 -1703 -1705 -1707 -1709 -1711 -1713 -1715 -1717 -1719 -1721 -1723 -1725

V -1698 -1700 -1702 -1704 -1706 -1708 -1710 -1712 -1714 -1716 -1718 -1720 -1722

Q -1695 -1697 -1699 -1701 -1703 -1705 -1707 -1709 -1711 -1713 -1715 -1717 -1719

L -1692 -1694 -1696 -1698 -1700 -1702 -1704 -1706 -1708 -1710 -1712 -1714 -1716

M -1689 -1691 -1693 -1695 -1697 -1699 -1701 -1703 -1705 -1707 -1709 -1711 -1713

Y -1686 -1688 -1690 -1692 -1694 -1696 -1698 -1700 -1702 -1704 -1706 -1708 -1710

A -1683 -1685 -1687 -1689 -1691 -1693 -1695 -1697 -1699 -1701 -1703 -1705 -1707

G -1680 -1682 -1684 -1686 -1688 -1690 -1692 -1694 -1696 -1698 -1700 -1702 -1704

I -1677 -1679 -1681 -1683 -1685 -1687 -1689 -1691 -1693 -1695 -1697 -1699 -1701

A -1674 -1676 -1678 -1680 -1682 -1684 -1686 -1688 -1690 -1692 -1694 -1696 -1698

F -1671 -1673 -1675 -1677 -1679 -1681 -1683 -1685 -1687 -1689 -1691 -1693 -1695

A -1668 -1670 -1672 -1674 -1676 -1678 -1680 -1682 -1684 -1686 -1688 -1690 -1692

L -1665 -1667 -1669 -1671 -1673 -1675 -1677 -1679 -1681 -1683 -1685 -1687 -1689

C -1662 -1664 -1666 -1668 -1670 -1672 -1674 -1676 -1678 -1680 -1682 -1684 -1686

L -1659 -1661 -1663 -1665 -1667 -1669 -1671 -1673 -1675 -1677 -1679 -1681 -1683

H -1656 -1658 -1660 -1662 -1664 -1666 -1668 -1670 -1672 -1674 -1676 -1678 -1680

H -1653 -1655 -1657 -1659 -1661 -1663 -1665 -1667 -1669 -1671 -1673 -1675 -1677

L -1650 -1652 -1654 -1656 -1658 -1660 -1662 -1664 -1666 -1668 -1670 -1672 -1674

I -1647 -1649 -1651 -1653 -1655 -1657 -1659 -1661 -1663 -1665 -1667 -1669 -1671

S -1644 -1646 -1648 -1650 -1652 -1654 -1656 -1658 -1660 -1662 -1664 -1666 -1668

A -1641 -1643 -1645 -1647 -1649 -1651 -1653 -1655 -1657 -1659 -1661 -1663 -1665

I -1638 -1640 -1642 -1644 -1646 -1648 -1650 -1652 -1654 -1656 -1658 -1660 -1662……

[1] "Alineamiento optimo secuencia 1"

M V L H S H Q L T Y A G I A F A L C L H H L I S A I E V P L D S N I Q S E L P Q P P T I T K Q S V K D Y I V
D P R D N I F I E C E A K G N P V P T F S W T R N G K F F N V A K D P K V S M R R R S G T L V I D F H G
G G R P D D Y E G E Y Q C F A R N D Y G T A L S S K I H L Q V S R S P L W P K E K V D V I E V D E G A P L
S L Q C N P P P G L P P P V I F W M S S S M E P I H Q D K R V S Q G Q N G D L Y F S N V M L Q D A Q T
D Y S C N A R F H F T H T I Q Q K N P Y T L K V K T K K P H N E T S L R N H T D M Y S A R G V T E T T P S
F M Y P Y G T S S S Q M V L R G V D L L L E C I A S G V P A P D I M W Y K K G G E L P A G K T K L E N F
N K A L R I S N V S E E D S G E Y F C L A S N K M G S I R H T I S V R V K A A P Y W L D E P Q N L I L A P
G E D G R L V C R A N G N P K P S I Q W L V N G E P I E G S P P N P S R E V A G D T I V F R D T Q I G S S
A V Y Q C N A S N E H G Y L L A N A F V S V L D V P P R I L A P R N Q L I K V I Q Y N R T R L D C P F F G
S P I P T L R W F K N G Q G N M L D G G N Y K A H E N G S L E M S M A R K E D Q G I Y T C V A T N I L
G K V E A Q V R L E V K D P T R I V R G P E D Q V V K R G S M P R L H C R V K H D P T L K L T V T W L K
D D A P L Y I G N R M K K E D D G L T I Y G V A E K D Q G D Y T C V A S T E L D K D S A K A Y L T V L A I
P A N R L R D L P K E R P D R P R D L E L S D L A E R S V K L T W I P G D D N N S P I T D Y I V Q F E E D
R F Q P G T W H N H S R Y P G N V N S A L L S L S P Y V N Y Q F R V I A V N D V G S S L P S M P S E R Y
Q T S G A R P E I N P T G V Q G A G T Q K N N M E I T W T P L N A T Q A Y G P N L R Y I V R W R R R D
P R G S W Y N E T V K A P R H V V W N T P I Y V P Y E I K V Q A E N D F G R A P E P E T Y I G Y S G E D
Y P K A A P T D V R I R V L N S T A I A L T W T R V H L D T I Q G Q L K E Y R A Y F W R D S S L L K N L W
V S K K R Q Y V S F P G D R N R G I V S R L F P Y S N Y K L E M V V T N G R G D G P R S E V K E F P T P E
G V P S S P R Y L R I R Q P N L E S I N L E W D H P E H P N G V L T G Y N L R Y Q A F N G S K T G R T L V
E N F S P N Q T R F T V Q R T D P I S R Y R F F L R A R T Q V G D G E V I V E E S P A L L N E A T P T P A S
T W L P P P T T E L T P A A T I A T T T T T A T P T T E T P P T E I P T T A I P T T T T T T T A T A A S T V A
S T T T T A E R A A A A T T K Q E L A Y T K N H V D I A T Q G W F I G L M C A I A L L V L I L L I V C F I K
R S R G G K Y P V R D N K D E H L N P E D K N V E D G S F D Y R S L E S D E D N K P L P N S Q T S L D G
T I K Q Q E S D D S L V D Y G E G G E G Q F N E D G S F I G Q Y T V K K D K E E T E G N E S S E A T S P V
N A I Y S L A

[1] "Alineamiento optimo secuencia 2"

M A V S V V Q L M Y A G I A F A L C L H H L I S A I E V P L D S N I Q S E L P Q P P T I T K Q S V K D Y I V
D P R D N I F I E C E A K G N P V P T F S W T R N G K F F N V A K D P K V S M R R R S G T L V I D F H G
G G R P D D Y E G E Y Q C F A R N D Y G T A L S S K I H L Q V S R S P L W P K E K V D V I E V D E G A P L
S L Q C N P P P G L P P P V I F W M S S S M E P I H Q D K R V S Q G Q N G D L Y F S N V M L Q D A Q T
D Y S C N A R F H F T H T I Q Q K N P Y T L K V K T K K P H N E T S L R N H T D M Y S A R G V T E T T P S
F M Y P Y G T S S S Q M V L R G V D L L L E C I A S G V P A P D I M W Y K K G G E L P A G K T K L E N F
N K A L R I S N V S E E D S G E Y F C L A S N K M G S I R H T I S V R V K A A P Y W L D E P Q N L I L A P
G E D G R L V C R A N G N P K P S I Q W L V N G E P I E G S P P N P S R E V A G D T I V F R D T Q I G S S
A V Y Q C N A S N E H G Y L L A N A F V S V L D V P P R I L A P R N Q L I K V I Q Y N R T R L D C P F F G
S P I P T L R W F K N G Q G N M L D G G N Y K A H E N G S L E M S M A R K E D Q G I Y T C V A T N I L
G K V E A Q V R L E V K D P T R I V R G P E D Q V V K R G S M P R L H C R V K H D P T L K L T V T W L K
D D A P L Y I G N R M K K E D D G L T I Y G V A E K D Q G D Y T C V A S T E L D K D S A K A Y L T V L A I
P A N R L R D L P K E R P D R P R D L E L S D L A E R S V K L T W I P G D D N N S P I T D Y I V Q F E E D
R F Q P G T W H N H S R Y P G N V N S A L L S L S P Y V N Y Q F R V I A V N D V G S S L P S M P S E R Y
Q T S G A R P E I N P T G V Q G A G T Q K N N M E I T W T P L N A T Q A Y G P N L R Y I V R W R R R D
P R G S W Y N E T V K A P R H V V W N T P I Y V P Y E I K V Q A E N D F G R A P E P E T Y I G Y S G E D
Y P K A A P T D V R I R V L N S T A I A L T W T R V H L D T I Q G Q L K E Y R A Y F W R D S S L L K N L W
V S K K R Q Y V S F P G D R N R G I V S R L F P Y S N Y K L E M V V T N G R G D G P R S E V K E F P T P E
G V P S S P R Y L R I R Q P N L E S I N L E W D H P E H P N G V L T G Y N L R Y Q A F N G S K T G R T L V
E N F S P N Q T R F T V Q E T D P I S R Y R F F L R A R T Q V G D G E V I V E E S P A L L N E A T P T P A S
T W L P P P T T E L T P A A T I A T T T T T A T P T T E T P P T E I P T T A I P T T T T T T T A T A A S T P A
E T T T T A E R A A A A P T K Q E L A Y T K N H V D I A T Q G W F I G L M C - - A L L V L I L L I V C F I K R
S R G G K Y P V R D N K D E H L G P E D K N V E D G S F D Y R S L E S D E D N K P L P N S Q T S L D G T I
K Q Q E S D D S L L Y Y G E G G E G Q F N E D G S F I G Q Y T V K K D K E E A E G N E S S E A T S P V N
A I Y S L A

[1] "Puntuacion alineamiento optimo"

1252

[1] "Numero de Gaps"

2

La matriz de puntuación utilizada sólo tiene “1” en la diagonal puesto que las secuencias son muy similares, se aplicó el algoritmo de Needleman Wunsch y se encontró la matriz de alineamiento, con 2 como número de gaps, es decir, el número de huecos introducidos con el fin de llevar a cabo el mejor alineamiento. Así que basándonos en el alineamiento global encontrado concluimos que las secuencias que correspondían a la proteína Neurofascin de dos organismos diferentes como lo son el pollo y el ratón coinciden en más del 90%, es decir, dado que comparten características en su mayoría, pertenecen a la misma familia.

Alineamiento local

En ocasiones el alineamiento global no es el más indicado y entonces otra solución sería buscar el mejor alineamiento entre *subsecuencias*. Esto puede darse, por ejemplo, cuando se sospecha que dos secuencias de proteínas pueden tener en común un dominio aunque no parezcan semejantes a simple vista. Así que, se hace lo que se denomina un alineamiento local.

El algoritmo de Smith Waterman, propuesto en 1981 por Temple Smith y Michael Waterman [21], para encontrar alineamientos locales óptimos está muy relacionado con el descrito anteriormente, aunque existen dos diferencias:

- Si el valor en la celda es negativo, entonces tomará el valor 0.
- El alineamiento puede terminar donde sea en la matriz, en lugar de tomar el valor de la esquina inferior derecha. El camino hacia atrás termina cuando llegamos a una celda con valor 0.

Análogamente al algoritmo anteriormente mencionado, describimos el algoritmo de Smith Waterman, para el cual es necesario definir:

- Una función de similitud $S(a_i, b_j)$.
- Las inserciones o deleciones de longitud k se penalizan con un peso W_k
- Se construye una matriz H de $n+1$ filas por $m+1$ columnas de acuerdo a la siguiente relación:

$$H(i,j) = max \left\{ \begin{array}{l} 0, \\ H(i-1, j-1) + S(a_i, b_j), \\ H(i-k, j) - W_k, \\ H(i, j-1) - W_i, \end{array} \right\}$$

  con $i \in 0, 1, ..., m+1$ y $j \in 0, 1, ..., m+1$.

Finalmente, después de construir la matriz H se busca el valor más grande y se recorre el camino hacia atrás hasta encontrar un cero en la diagonal, este camino recorrido nos dará el mejor alineamiento.

Veamos un ejemplo del algoritmo Smith Waterman, donde tomamos las 2 secuencias anteriores y aplicamos la función que aparece en el apéndice C (2. Algoritmo Smith Waterman)

Consideramos las secuencias correspondientes a la proteína denominada Neurofascin, primero del pollo y luego del ratón y aplicamos el algoritmo de Smith Waterman utilizado para alineamientos locales, es decir, toma subsecuencias más pequeñas con el fin de encontrar el mejor alineamiento.

Se muestra un fragmento de la matriz de alineamiento:

		V	Q	F	E	E	D	R	F	Q	P	G	T	W	H	N	H	S	R	Y
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
V	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
V	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Q	0	2	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
L	0	0	2	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	
M	0	0	0	2	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	
Y	0	0	0	0	2	1	0	1	0	0	0	1	0	0	0	0	0	0	1	
A	1	0	0	0	0	2	1	0	1	0	0	0	1	0	0	0	0	0	0	
G	1	1	0	0	0	0	2	1	0	1	1	0	0	1	0	0	0	0	0	
I	0	1	1	0	0	0	0	2	1	0	1	1	0	0	1	0	0	0	0	
A	1	0	1	1	0	0	0	0	2	1	0	1	1	0	0	1	0	0	0	
F	0	1	1	1	1	0	0	1	0	2	1	0	1	1	0	0	1	0	0	
A	0	0	1	1	1	1	0	0	1	0	2	1	0	1	1	0	0	1	0	
L	0	0	0	1	1	1	1	0	0	1	0	2	1	0	1	1	0	0	1	
C	1	0	0	0	1	1	1	1	0	0	1	0	2	1	0	1	1	0	0	
L	0	1	0	0	0	1	1	1	1	0	0	1	0	2	1	0	1	1	0	
H	0	0	1	0	0	0	1	1	1	1	0	0	1	1	2	2	0	1	1	
H	0	0	0	1	0	0	0	1	1	1	1	0	0	2	1	3	2	0	1	
L	1	0	0	0	1	0	0	0	1	1	1	1	0	0	2	1	3	2	0	

I 2 1 0 0 0 1 0 0 0 1 1 1 1 0 0 2 1 3 2

S 4 2 1 0 0 0 1 0 0 0 1 1 1 1 0 0 3 1 3

A 3 4 2 1 0 0 0 1 0 0 0 1 1 1 1 0 1 3 1

I 1 3 4 2 1 0 0 0 1 0 0 0 1 1 1 1 0 1 3 .....

[1] "Alineamiento optimo secuencia 1"

M V L H S H Q L T Y A G I A F A L C L H H L I S A I E V P L D S N I Q S E L P Q P P T I T K Q S V K D Y I V
D P R D N I F I E C E A K G N P V P T F S W T R N G K F F N V A K D P K V S M R R R S G T L V I D F H G
G G R P D D Y E G E Y Q C F A R N D Y G T A L S S K I H L Q V S R S P L W P K E K V D V I E V D E G A P L
S L Q C N P P P G L P P P V I F W M S S S M E P I H Q D K R V S Q G Q N G D L Y F S N V M L Q D A Q T
D Y S C N A R F H F T H T I Q Q K N P Y T L K V K T K K P H N E T S L R N H T D M Y S A R G V T E T T P S
F M Y P Y G T S S S Q M V L R G V D L L L E C I A S G V P A P D I M W Y K K G G E L P A G K T K L E N F
N K A L R I S N V S E E D S G E Y F C L A S N K M G S I R H T I S V R V K A A P Y W L D E P Q N L I L A P
G E D G R L V C R A N G N P K P S I Q W L V N G E P I E G S P P N P S R E V A G D T I V F R D T Q I G S S
A V Y Q C N A S N E H G Y L L A N A F V S V L D V P P R I L A P R N Q L I K V I Q Y N R T R L D C P F F G
S P I P T L R W F K N G Q G N M L D G G N Y K A H E N G S L E M S M A R K E D Q G I Y T C V A T N I L
G K V E A Q V R L E V K D P T R I V R G P E D Q V V K R G S M P R L H C R V K H D P T L K L T V T W L K
D D A P L Y I G N R M K K E D D G L T I Y G V A E K D Q G D Y T C V A S T E L D K D S A K A Y L T V L A I
P A N R L R D L P K E R P D R P R D L E L S D L A E R S V K L T W I P G D D N N S P I T D Y I V Q F E E D
R F Q P G T W H N H S R Y P G N V N S A L L S L S P Y V N Y Q F R V I A V N D V G S S L P S M P S E R Y
Q T S G A R P E I N P T G V Q G A G T Q K N N M E I T W T P L N A T Q A Y G P N L R Y I V R W R R R D
P R G S W Y N E T V K A P R H V V W N T P I Y V P Y E I K V Q A E N D F G R A P E P E T Y I G Y S G E D
Y P K A A P T D V R I R V L N S T A I A L T W T R V H L D T I Q G Q L K E Y R A Y F W R D S S L L K N L W
V S K K R Q Y V S F P G D R N R G I V S R L F P Y S N Y K L E M V V T N G R G D G P R S E V K E F P T P E
G V P S S P R Y L R I R Q P N L E S I N L E W D H P E H P N G V L T G Y N L R Y Q A F N G S K T G R T L V
E N F S P N Q T R F T V Q R T D P I S R Y R F F L R A R T Q V G D G E V I V E E S P A L L N E A T P T P A S
T W L P P P T T E L T P A A T I A T T T T T A T P T T E T P P T E I P T T A I P T T T T T T T A T A A S T V A
S T T T T A E R A A A A T T K Q E L A Y T K N H V D I A T Q G W F I G L M C A I A L L V L I L L I V C F I K
R S R G G K Y P V R D N K D E H L N P E D K N V E D G S F D Y R S L E S D E D N K P L P N S Q T S L D G
T I K Q Q E S D D S L V D Y G E G G E G Q F N E D G S F I G Q Y T V K K D K E E T E G N E S S E A T S P V
N A I Y S L A

[1] "Alineamiento optimo secuencia 2"

M A V S V V Q L M Y A G I A F A L C L H H L I S A I E V P L D S N I Q S E L P Q P P T I T K Q S V K D Y I V
D P R D N I F I E C E A K G N P V P T F S W T R N G K F F N V A K D P K V S M R R R S G T L V I D F H G
G G R P D D Y E G E Y Q C F A R N D Y G T A L S S K I H L Q V S R S P L W P K E K V D V I E V D E G A P L
S L Q C N P P P G L P P P V I F W M S S S M E P I H Q D K R V S Q G Q N G D L Y F S N V M L Q D A Q T
D Y S C N A R F H F T H T I Q Q K N P Y T L K V K T K K P H N E T S L R N H T D M Y S A R G V T E T T P S
F M Y P Y G T S S S Q M V L R G V D L L L E C I A S G V P A P D I M W Y K K G G E L P A G K T K L E N F

N K A L R I S N V S E E D S G E Y F C L A S N K M G S I R H T I S V R V K A A P Y W L D E P Q N L I L A P
G E D G R L V C R A N G N P K P S I Q W L V N G E P I E G S P P N P S R E V A G D T I V F R D T Q I G S S
A V Y Q C N A S N E H G Y L L A N A F V S V L D V P P R I L A P R N Q L I K V I Q Y N R T R L D C P F F G
S P I P T L R W F K N G Q G N M L D G G N Y K A H E N G S L E M S M A R K E D Q G I Y T C V A T N I L
G K V E A Q V R L E V K D P T R I V R G P E D Q V V K R G S M P R L H C R V K H D P T L K L T V T W L K
D D A P L Y I G N R M K K E D D G L T I Y G V A E K D Q G D Y T C V A S T E L D K D S A K A Y L T V L A I
P A N R L R D L P K E R P D R P R D L E L S D L A E R S V K L T W I P G D D N N S P I T D Y I V Q F E E D
R F Q P G T W H N H S R Y P G N V N S A L L S L S P Y V N Y Q F R V I A V N D V G S S L P S M P S E R Y
Q T S G A R P E I N P T G V Q G A G T Q K N N M E I T W T P L N A T Q A Y G P N L R Y I V R W R R R D
P R G S W Y N E T V K A P R H V V W N T P I Y V P Y E I K V Q A E N D F G R A P E P E T Y I G Y S G E D
Y P K A A P T D V R I R V L N S T A I A L T W T R V H L D T I Q G Q L K E Y R A Y F W R D S S L L K N L W
V S K K R Q Y V S F P G D R N R G I V S R L F P Y S N Y K L E M V V T N G R G D G P R S E V K E F P T P E
G V P S S P R Y L R I R Q P N L E S I N L E W D H P E H P N G V L T G Y N L R Y Q A F N G S K T G R T L V
E N F S P N Q T R F T V Q E T D P I S R Y R F F L R A R T Q V G D G E V I V E E S P A L L N E A T P T P A S
T W L P P P T T E L T P A A T I A T T T T T A T P T T E T P P T E I P T T A I P T T T T T T T A T A A S T P A
E T T T T A E R A A A A P T K Q E L A Y T K N H V D I A T Q G W F I G L M C - - A L L V L I L L I V C F I K R
S R G G K Y P V R D N K D E H L G P E D K N V E D G S F D Y R S L E S D E D N K P L P N S Q T S L D G T I
K Q Q E S D D S L L Y Y G E G G E G Q F N E D G S F I G Q Y T V K K D K E E A E G N E S S E A T S P V N
A I Y S L A

[1] "Puntuacion alineamiento optimo"

1252

[1] "Numero de Gaps"

2

Utilizamos nuevamente la matriz de puntuación con “1” en la diagonal, ya que las secuencias son similares, notamos que la matriz de alineamiento local difiere respecto a la correspondiente al alineamiento global, ya que en el alineamiento local los números negativos son reemplazados por “0” como se observa en la matriz mencionada. En este caso en particular, debido a que las secuencias son similares, el alineamiento global y el local son casi el mismo ya que no es necesario hacer un reacomodo.

Así que nuevamente, dado que coinciden en su mayoría y sólo ha sido necesario introducir 2 huecos podemos concluir que estas dos secuencias correspondientes a dos organismos diferentes pertenecen a la misma familia de proteínas.

Perfiles de Modelos Ocultos de Markov (PMOM)

Hasta ahora, nos hemos concentrado en las propiedades intrínsecas de una secuencia, tales como regiones concernientes a islas CpG en el ADN, o en alineación en pares de secuencias. Sin embargo, las secuencias biológicas funcionales se encuentran comúnmente organizadas en familias, por lo cual es útil identificar características estadísticas correspondientes a la familia, de forma que podamos identificar nuevos miembros. Así como la alineación de dos secuencias muestra la relación que hay entre ellas, la alineación de varias secuencias puede mostrar cómo se relacionan las secuencias dentro de una familia.

El cuadro 4.2 muestra la alineación de una pequeño segmento de 7 secuencias de una larga familia de globinas.

Ahora, para capturar esas características, construiremos un modelo probabilístico, el cual es un tipo particular de Modelos Ocultos de Markov adecuado para modelar alineamientos múltiples. Este modelo recibe el nombre de Perfil de Modelos Ocultos de Markov. La topología básica del PMOM consiste en tener tres estados por cada columna del alineamiento múltiple, excepto para las columnas en las que más de la mitad de sus elementos son huecos, como las columnas 4 y 5 del ejemplo anterior. Los estados son:

- Estados de coincidencia M.
- Estados de inserción I.
- Estados de deleción D.

Luego, el modelo que representa la familia de proteínas será el representado en la Figura 4.7.

Como los Modelos Ocultos de Markov, los Perfiles de Modelos Ocultos de Markov tienen como parámetros las probabilidades de emisión y las probabilidades de transición. Suponiendo que estas probabilidades son diferentes de cero, un Perfil de Modelos Ocultos de Markov puede modelar cualquier secuencia de residuos de un alfabeto dado.

Estas probabilidades, junto con la longitud del modelo nos permitirán controlar la forma de la ditribución. La elección de la longitud del modelo corresponde esencialmente a la elección de las columnas del alineamiento que designaremos como estados de coincidencia, de inserción o de deleción. Luego, para estimar los parámetros de probabilidad, usamos las siguientes ecuaciones:

$$p_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

y

$$e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')}.$$

donde k y l son índices sobre los estados, a_{kl} las probabilidades de transición y e_k las probabilidades de emisión, finalmente, A_{kl} y E_k son las frecuencias correspondientes. Cuando se

HBA_HUMAN	...	V	G	A	-	-	H	A	G	E	Y	...
HBB_HUMAN	...	V	-	-	-	-	N	V	D	E	V	...
MYG_PHYCA	...	V	E	A	-	-	D	V	A	G	H	...
GLB3_CHITP	...	V	K	G	-	-	-	-	-	-	D	...
GLB5_PETMA	...	V	Y	S	-	-	T	Y	E	T	S	...
LGB2_LUPLU	...	F	N	A	-	-	N	I	P	K	H	...
GLB1_GLYDI	...	I	A	G	A	D	N	G	A	G	V	...
		*	*	*			*	*	*	*	*	

Cuadro 4.2: Alineación de 7 secuencias de globinas, las columnas marcadas con asterisco serán "coincidencias" en el PMOM

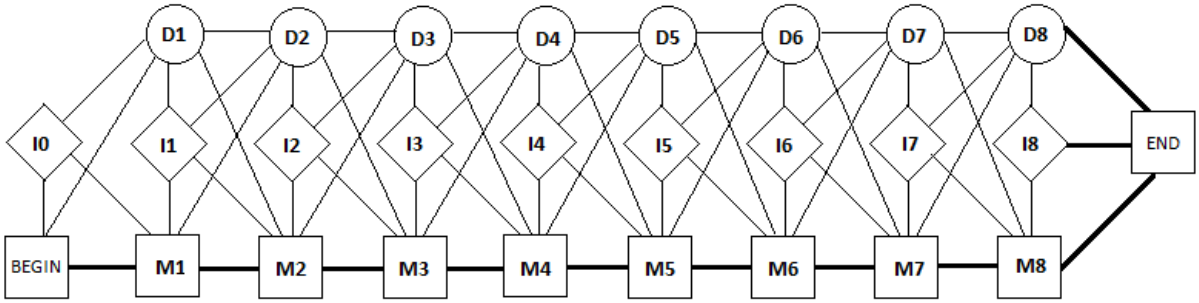


Figura 4.7: Perfil del Modelo Oculto de Markov asociado a la familia de proteínas que se muestra en el cuadro 4.2. Los círculos representan los estados de deleción, los diamantes los estados de inserción y los cuadros los estados de coincidencia

dispone de un número grande quizá no hay problema, sin embargo, si el número es pequeño puede suceder que las probabilidades sean 0, así que para evitar esto se agrega 1 a cada frecuencia. Por ejemplo, para el alineamiento múltiple anterior las frecuencias de aparición en la primera columna son: 5 para V, 1 para F e I, y 0 para los restantes 17 aminoácidos. Al agregar 1 a cada frecuencia se tiene que ahora la frecuencia de aparición es: 6 para V, 2 para F e I, y 1 para los restantes.

Así se determina que $e_{M_1}(V) = \frac{6}{27}$, $e_{M_1}(F) = e_{M_1}(I) = \frac{2}{27}$ y $e_{M_1}(a) = \frac{1}{27}$ para $a \neq V, F, I$.

De manera similar se tiene que en la columna 1 hay 6 transiciones del estado de coincidencia M_1 al siguiente, 1 transición a un estado de deleción y 0 a un estado de inserción. Sumando 1 a cada frecuencia se obtiene $a_{M_1M_2} = \frac{7}{10}$, $a_{M_1D_1} = \frac{2}{10}$ y $a_{M_1I_1} = \frac{1}{10}$.

De forma análoga obtenemos los parámetros restantes:

- $e_{M_2}(G) = e_{M_2}(E) = e_{M_2}(K) = e_{M_2}(Y) = e_{M_2}(N) = e_{M_2}(A) = \frac{2}{26}$, $e_{M_2}(a) = \frac{1}{26}$ para $a \neq G, E, K, Y, N, A$ y $a_{M_2M_3} = \frac{7}{10}$, $a_{M_2D_2} = \frac{2}{10}$ y $a_{M_2I_2} = \frac{1}{10}$.

- $e_{M_3}(A) = \frac{4}{26}$, $e_{M_3}(G) = \frac{3}{26}$, $e_{M_3}(S) = \frac{2}{26}$, $e_{M_3}(a) = \frac{1}{26}$para $a \neq A, G, S$ y $a_{M_3M_4} = \frac{2}{10}$, $a_{M_3D_3} = \frac{7}{10}$ y $a_{M_3I_3} = \frac{1}{10}$.

- $e_{M_4}(A) = \frac{2}{21}$, $e_{M_4}(a) = \frac{1}{21}$ para $a \neq A$ y $a_{M_4M_5} = \frac{2}{10}$, $a_{M_4D_4} = \frac{7}{10}$ y $a_{M_4I_4} = \frac{1}{10}$.

- $e_{M_5}(D) = \frac{2}{21}$, $e_{M_5}(a) = \frac{1}{21}$ para $a \neq D$ y $a_{M_5M_6} = \frac{7}{10}$, $a_{M_5D_5} = \frac{2}{10}$ y $a_{M_5I_5} = \frac{1}{10}$.

- $e_{M_6}(N) = \frac{4}{26}$, $e_{M_6}(H) = e_{M_6}(D) = e_{M_6}(T) = \frac{2}{26}$ $e_{M_6}(a) = \frac{1}{26}$ para $a \neq N, H, D, T$ y $a_{M_6M_7} = \frac{7}{10}$, $a_{M_6D_6} = \frac{2}{10}$ y $a_{M_6I_6} = \frac{1}{10}$.

- $e_{M_7}(A) = e_{M_7}(Y) = e_{M_7}(I) = e_{M_7}(G) = \frac{2}{26}$, $e_{M_7}(v) = \frac{2}{26}$, $e_{M_7}(a) = \frac{1}{26}$ para $a \neq A, V, Y, I, G$ y $a_{M_7M_8} = \frac{7}{10}$, $a_{M_7D_7} = \frac{2}{10}$ y $a_{M_7I_7} = \frac{1}{10}$.

- $e_{M_8}(G) = e_{M_8}(D) = e_{M_8}(E)) = e_{M_8}(P) = e_{M_2}(A) = \frac{2}{26}$, $e_{M_8}(A) = \frac{3}{26}$, $e_{M_8}(a) = \frac{1}{26}$ para $a \neq G, D, A, E, P$ y $a_{M_8M_9} = \frac{7}{10}$, $a_{M_8D_8} = \frac{2}{10}$ y $a_{M_8I_8} = \frac{1}{10}$.

- $e_{M_9}(K) = e_{M_9}(T) = \frac{2}{26}$, $e_{M_9}(E) = e_{M_9}(G) = \frac{3}{26}$,$e_{M_9}(a) = \frac{1}{26}$ para $a \neq K, T, G, E$ y $a_{M_9M_{10}} = \frac{8}{10}$, $a_{M_9D_9} = \frac{1}{10}$ y $a_{M_9I_9} = \frac{1}{10}$.

- $e_{M_{10}}(Y) = e_{M_{10}}(D) = e_{M_{10}}(S) = \frac{2}{27}$, $e_{M_{10}}(V) = e_{M_{10}}(H) = \frac{3}{27}$, $e_{M_{10}}(a) = \frac{1}{27}$ para $a \neq Y, V, H, D, S$.

Ahora, mencionaremos cómo construir un PMOM cuando la longitud de las secuencias es mayor.

Cuando la longitud de las secuencias es mayor, aunque nos centraremos en las probabilidades de emisión, métodos análogos pueden ser usados para estimar las probabilidades de transición. El método más sencillo sería dar las estimaciones de máxima verosimilitud para los parámetros. Cambiaremos un poco la notación de la usada anteriormente. Dadas las

frecuencias observadas c_{ja} del residuo a en la posición j del alineamiento, las estimaciones de máxima verosimilitud para $e_{M_j}(a)$, los parámetros correspondientes del modelo son:

$$e_{M_j}(a) = \frac{c_{ja}}{\sum_{a'} c_{ja'}}.$$

Ahora bien, como en el caso anterior, un problema sería que alguna probabilidad sea cero. Por ejemplo en la Tabla 4.2 sólo V, I y F aparecen en la primera columna. Sin embargo, es probable que otros aminoácidos aparezcan en esa posición en alguna otra secuencia de globinas. La forma más simple de lidiar con este problema es sumar "pseudocounts" a las frecuencias observadas c_{ja}.

Conclusiones y trabajo futuro

El presente trabajo presentó un estudio de secuencias de ADN haciendo uso de Modelos Ocultos de Markov, con el fin de identificar regiones de islas CpG y los estados ocultos subyacentes a la secuencia observada. En primer lugar se descargó el software Bioconductor para acceder a la información correspondiente al genoma del ratón, luego, se usaron como datos de entrenamiento las posiciones de islas CpG del mismo organismo, es decir del ratón, referidas en un artículo [14]. Para aplicar la teoría estudiada respecto a los Modelos Ocultos de Markov [3], se estimaron los parámetros necesarios, es decir la matriz de emisión, transición y probabilidad inicial (veáse Capítulo 3). Luego, se programó el algoritmo de Viterbi en R para encontrar la secuencia de estados ocultos que generó la secuencia observada, es decir, la secuencia que descargamos y que correspondía a una región de isla CpG (veáse Capítulo 4).

Se programaron algunos de los algoritmos de programación dinámica necesarios para resolver los problemas relacionados con los Modelos Ocultos de Markov. El algoritmo de Viterbi, como ya hemos mencionado, se aplicó a datos reales y se comprobó su eficiencia.

Así mismo, dado que parte de la información genética contenida en el ADN codifica en proteínas, se hace mención de estas últimas y se programaron los algoritmos correspondientes al alineamiento por pares de secuencias, al mismo tiempo se hizo notar la diferencia entre los alineamientos globales y locales. Finalmente, se presentó una variante de los Modelos Ocultos de Markov, denominada Perfiles de Modelos Ocultos de Markov utilizada para modelar familias de proteínas y que nos permite inferir ciertas propiedades acerca de los miembros que las componen [10].

Cabe mencionar que R cuenta con funciones diseñadas para encontrar regiones de islas, así como para encontrar la secuencia de estados ocultos, sin embargo en este trabajo se hizo paso a paso para ilustrar claramente el funcionamiento del algoritmo (veáse Apéndice B) de manera que pueda ser utilizado fácilmente para cualquier otro organismo e identificación de otras regiones. De igual manera, se encuentran páginas disponibles de algunos laboratorios para la alineación de secuencias, sin embargo, para el lector que no está tan relacionado con la bioinformática y el manejo de datos a gran escala puede resultar un poco díficil interpretar los resultados, por ello se programaron los algoritmos utilizados para el alineamiento.

En cuanto al trabajo futuro cabe mencionar que los Modelos Ocultos de Markov están siendo ampliamente utilizado en bioinformática, así que los temas a abordar podrían ser variados:

- Se puede estimar la probabilidad de que cierta secuencia pertenezca a alguna región en particular, sea parte de una señal o en el caso de las proteínas de una familia utilizando el algoritmo de avance-retroceso [19].
- Se pueden utilizar Modelos de Markov para estudiar secuencias de ADN y proteínas simultáneamente.
- También se pueden construir árboles filogenéticos basados en Modelos de Markov [10].
- Así mismo, como hemos mencionado anteriormente, los Modelos Ocultos de Markov pueden ayudarnos identificar diferentes regiones dentro del ADN como lo son: motivos, patrones, genes, pseudogenes [11].

Apéndice A

Programación Dinámica

En informática, la programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas. El matemático Richard Bellman inventó la programación dinámica en 1953 que se utiliza para optimizar problemas complejos que pueden ser discretizados y secuencializados.

Para poder emplear programación dinámica, una secuencia óptima debe cumplir la condición de que cada una de sus subsecuencias también sea óptima:

Dado un problema P con n elementos, si la secuencia óptima es $e_1e_2...e_k...e_n$ entonces:

- $e_1e_2...e_k$ es solución al problema P considerando los k primeros elementos.
- $e_{k+1}...e_n$ es solución al problema P considerando los elementos desde $k+1$ hasta n.

En otras palabras: dada una secuencia óptima de soluciones, toda subsecuencia de ella es, a su vez, óptima.

La programación dinámica se aplica cuando la subdivisión de un problema conduce a:

- Una enorme cantidad de problemas.
- Problemas cuyas soluciones parciales se solapan.
- Grupos de problemas de muy distinta complejidad.

El viaje más barato por el río. En un río hay n embarcaderos, en cada uno de los cuales se puede alquilar un bote para ir a otro embarcadero que esté más abajo en el río. Suponemos que no se puede remontar el río. Una tabla de tarifas indica los costes de viajar entre los distintos embarcaderos. Se supone que puede ocurrir que un viaje entre i y j salga más barato haciendo escala en k embarcaderos que yendo directamente.

El problema consistirá en determinar el coste mínimo para un par de embarcaderos.

Vamos a llamar a la tabla de tarifas, T. Así, $T[i,j]$ será el coste de ir del embarcadero i al j. La matriz será triangular superior de orden n, donde n es el número de embarcaderos. La idea recursiva es que el coste se calcula de la siguiente manera:

$$C(i,j) = T[i,k] + C(k,j)$$

A partir de esta idea, podemos elaborar una expresión recurrente para la solución:

$$\begin{cases} 0 \quad si \quad i = j \\ C(i,j) = Min(T[i,k] + C(k,j), T(i,j)) \quad si \quad i < k \leq j \end{cases}$$

Un algoritmo que resuelve este problema es el siguiente, donde T es la matriz de tarifas, origen y destino, los embarcaderos del que se parte y al que se llega respectivamente, y C la matriz en la que almacenaremos los resultados de los costes. La función MenorDelosCandidatos devuelve el menor coste entre dos puntos, utilizando como base la recurrencia anteriormente expuesta.

FUNC Embarcaderos (↓ origen, destino, n: **NATURAL**, ↓ T :**MATRIZ DE NATURALES**): **NATURAL**
Variables C: **MATRIZ DE NATURALES** i, j: **NATURAL Inicio PARA** $i = 1$ **HASTA** n **HACER** $C[i][j] := 0$ **FINPARA PARA** $i = 1$ **HASTA** n **HACER PARA** $j = 1$ **HASTA** n **HACER** $C[i][j]$:=menorDeLosCandidatos(i, j, n, T, C) **FINPARA FINPARA devolver** $C[n][n]$ **Fin**
FUNC menorDeLosCandidatos (↓ origen, destino, n: **NATURAL**, ↓ T, C : **MATRIZ DE NATURALES**): **NATURAL**
Variables temp:**NATURAL Inicio** temp:= MAX_NATURAL **PARA** i =origen+1 **HASTA** n **HACER** temp:= min $(temp, T[origen][i] + C[i][destino])$ **FINPARA devolver** *temp* **Fin**

Apéndice B

1. Algoritmo de Viterbi

> source("http://bioconductor.org/biocLite.R")

> available.genomes()

> library(BSgenome.Mmusculus.UCSC.mm9)

> Mmusculus

> class(Mmusculus)

> seqnames(Mmusculus)

> crom10<-Mmusculus$chr10

> crom10

> length(crom10)

> ubic<-read.table("http://rafalab.jhsph.edu/CGI/model-based-cpg-islands-mm9.txt",header=TRUE)

> head(ubic)

> ubic10<-subset(ubic,ubic[,1]=="chr10")

> dim(ubic10)

> start<-ubic10[,2]

> end<-ubic10[,3]

> class(crom10)

> frec10<-alphabetFrequency(crom10)

> frec10

> crom10<-unmasked(crom10)

> class(crom10)

> frec10_enm<-alphabetFrequency(crom10)

```
> frec10_enm
> cadenamas<-matrix(NA,4366,1)
> for (i in 1:4366)
+ cadenamas[i,]<-as.character(crom10[(start[i]+1):(end[i]-1)])
> juntarmas<-do.call(paste, c(as.list(cadenamas), sep=""))
> juntarmas<-as(juntarmas,"DNAString")
> cant_n<-countPattern("N",juntarmas) # elimino los gaps
> m_n<-matchPattern("N",juntarmas)
> start2<-start(m_n)
> juntarmas[start2]<-NULL
> vmas<-strsplit(gsub("([[:alnum:]]2)", "\ \ 1", juntarmas), " ")[[1]]
> contingisla<-table(vmas)
> contingisla<-matrix(contingmas,4,4,dimnames = list(c("A","C","G","T"),
+ c("A","C","G","T")))
> contingisla<-t(contingisla)
> cadenaocean<-matrix(NA,4367,1) #con gaps
> cadenaocean[1,]<-as.character(crom10[1:start[1]])
> for (i in 1:4365)
+ cadenaocean[i+1,]<-as.character(crom10[end[i]:start[i+1]])
> cadenaocean[4367,]<-as.character(crom10[end[4366]:129993255])
> contingocean<-matrix(contingmenos,4,4,dimnames = list(c("A","C","G","T"),
+ c("A","C","G","T")))
> contingocean<-t(contingocean)
> transi1<-matrix(0,8,8,dimnames =
+ list(c("A+","C+","G+","T+","A-","C-","G-","T-"),
+ c("A+","C+","G+","T+","A-","C-","G-","T-")))
```

```
> transi1[1:4,1:4]<-contingmas
> transi1[5:8,5:8]<-contingmenos
> transi3<-matrix(NA,dim(ubic10)[1],2)
> for (i in 1:dim(ubic10)[1]){
+ transi3[i,1]<-as.character(crom10[(start[i]-1):start[i]])
+ transi3[i,2]<-as.character(crom10[end[i]:(end[i]+1)])
+ }
> inicial<-table(transi3[,1])
> inicim<-matrix(0,4,4,dimnames = list(c("A-","C-","G-","T-"),
+ c("A+","C+","G+","T+")))
> inicim[,2]=c(inicial[1],inicial[2],inicial[3],inicial[4])
> final<-table(transi3[,2])
> final<-matrix(final[-c(4,4)],4,4,dimnames =
+ list(c("A-","C-","G-","T-"), c("A+","C+","G+","T+")))
> final<-t(final)
> transicion[1:4,5:8]<-final
> transicion[5:8,1:4]<-inicim
> transicion<-prop.table(transi1,1)
> transicion
> emision
> prob_inicial
> crom10<-strsplit(gsub("([[:alnum:]]1)",
+ "\ \ 1", crom10), " ")[[1]]
> seq<-crom10[start[3499]:end[3499]]
> y<-c("A","C","G","T") # observaciones
> x<-c("A_+","C_+","G_+","T_+","A_-","C_-","G_-","T_-") # estados
```

```
> n=length(seq)
> gamma_1<- matrix(NA,8,n)
> gamma_2<- matrix(NA,8,n)
> oculta<-numeric(n)
> prob_oculta<-numeric(n)
> # inicializacion
> for (i in 1:8){
+ gamma_1 [i,1]<- log(prob_inicial[i]*emision[i, as.character(seq[1])])
+ gamma_2 [i,1]<-0
+ }
> # induccion
> for (i in 2:n){
+ for (j in 1:8){
+ maxi<-NULL
+ for (k in 1:8){
+ temp<-max(gamma_1[k,i-1]+log(transicion[k,j]))
+ gamma_2[j,i]<-which.max(gamma_1[,i-1]+log(transicion[,j]))
+ maxi<-max(maxi,temp)
+ }
+ gamma_1[j,i]<-maxi+log(emision[j, as.character(seq[i])])
+ }
+ }
> chi=numeric(n)
> chi[n]<-which.max(gamma_1[,n])
> oculta[n]<-x[chi[n]]
> prob_oculta[n]<-max(gamma_1[,n])
```

```
> for (i in n:2){
+ chi[i-1]<-gamma_2[chi[i],i]
+ oculta[i-1]<-x[chi[i-1]]
+ prob_oculta[i-1]<-max(gamma_1[,i]+log(transicion[,chi[i]]))
+ }
> chi
> oculta
```

2. Algoritmo Baum Welch

```
HMMsample <- function(nu, Q, g, n){
  cQ <- t(apply(Q, 1, cumsum));
  cg <- t(apply(g, 1, cumsum));
  x <- array(0, n);
  y <- array(0, n);
  x[1] <- 1+sum(as.numeric(runif(1)>cumsum(nu)));
  y[1] <- 1+sum(as.numeric(runif(1)>cg[x[1],]));
  for (t in 2:n){
    x[t] <- 1+sum(as.numeric(runif(1)>cQ[x[t-1],]));
    y[t] <- 1+sum(as.numeric(runif(1)>cg[x[t],]));
    #remark: it is slightly simpler, but also slightly slower to use:
    # x[t] <- sample(1:k, 1, prob = Q[x[t-1],])
    # y[t] <- sample(1:r, 1, prob = g[x[t],]);

  }
  list(x=x,y=y);
}

HMMfilter <- function(y, nu, Q, g){
  n <- length(y);
  dims <- dim(Q);
  res <- list();
  res$phi <- matrix(nrow = dims[1], ncol = n);
  res$c <- array(0, n);
```

```
  Z <- nu*g[,y[1]];
  res$c[1] <- sum(Z);
  res$phi[,1] <- Z/res$c[1];

  for (t in 2:n){
    Z <- (res$phi[,t-1] %*% Q) * g[, y[t]]
    res$c[t] <- sum(Z);
    res$phi[,t] <- Z / res$c[t];
  }
  res;
}

HMMsmoother <- function(y, Q, g, c){
  n<- length(y);
  dims <- dim(Q);
  beta <- matrix(1, nrow=dims[1], ncol=n);
  for (t in seq(n-1, 1, -1)){
    beta[,t] = Q %*% (g[,y[t+1]] * beta[, t+1]) / c[t+1];
  }
  beta;
}

HMMfilter_C = function(y, nu, Q, g){
  n <- length(y);
  N <- dim(Q)[1];
  M <- dim(g)[2];
```

```
  res <- .C("HMMfilter", as.integer(N), as.integer(M), as.integer(n), as.double(y),
as.double(nu), as.double(Q), as.double(g), as.double(array(0, N*n)), as.double(array(0,
n)));
  list(phi = matrix(res[[8]], nrow = N, ncol = n), c = res[[9]]);
}

HMMsmoother_C = function(y, Q, g, c){
  n <- length(y);
  N <- dim(Q)[1];
  M <- dim(g)[2];

  res <- .C("HMMsmoother", as.integer(N), as.integer(M), as.integer(n), as.double(y),
as.double(Q), as.double(g), as.double(c), as.double(matrix(0, nrow = N, ncol = n)))[[8]];
  matrix(res, nrow = N, ncol = n);
}

HMMbaumwelch <- function(y, nu, tol = 1e-4, maxIt = 100){
  k <- length(nu);
  r <- max(y);
  n <- length(y);
  Y <- matrix(0, nrow=n, ncol=r);   for (t in 1:n){ Y[t, y[t]]=1; }
  Q <- matrix(runif(k*k), nrow=k);  Q <- Q / apply(Q, 1, sum);
  g <- matrix(runif(k*r), nrow=k);  g <- g / apply(g, 1, sum);
  it <- 0; oldQ <- Q-tol; oldg <- g-tol;
  while ((sum(abs((oldQ-Q))) + sum(abs(oldg-g)) > tol) & (it<maxIt)){
    it <- it+1;
```

```
    f <- myfilter(y, nu, Q, g);
    beta <- mysmoother(y, Q, g, f$c);
    post <- f$phi * beta;

    N <- Q * (f$phi[,1:(n-1)] %*% t(beta[, 2:n] * g[, y[2:n]]  / (matrix(1, nrow=k,
ncol=1)%*%f$c[2:n])));
    M <- post %*% Y;

    oldQ <- Q; oldg <- g;
    Q <- N / apply(N, 1, sum);
    g <- M / apply(M, 1, sum);
  }

  res <- list();
  res$Q <- Q;
  res$g <- g;
  res$l <- sum(log(f$c));
  res$it <- it;
  res;
}
```

Apéndice C

1. Algoritmo Needleman Wunsch

```
> NeedlemanWunsch<- function(Sec1, Sec2, Hueco, MSCab, MSVal){
+   NumGaps<-0
+   if (missing(Hueco))
+    Hueco<- 0
+   if (missing(MSVal)) {
+    for(i in 1:length(MSCab)) {
+     for (j in 1:length(MSCab)) {
+       if(i!=j){
+          MSVal<-c(MSVal, 0)
+       } else {
+          if(i==1 & j==1) {
+            MSVal <- 1
+          } else{
+            MSVal<- c(MSVal,1)
+           }
+        }
+      }
+    }
+   }
+
+  MSFilCol <- length(MSCab)
+  NombresF<-NombresC<-MSCab
```

```
+  n<-length(NombresF)
+  print(n)
+  MatrizPuntaje<-matrix(MSVal, MSFilCol, MSFilCol, byrow=TRUE,
dimnames=list(NombresC, NombresF))
+  S1<-c(" ", Sec1)
+  S2<-c(" ", Sec2)
+
+  LongS1<-length(S1)
+  LongS2<-length(S2)
+
+  MatrizAlin<-matrix(data=NA, nrow=LongS2, ncol=LongS1, byrow=FALSE,
dimnames=list(S2, S1))
+  for(i in 1:LongS2){
+    if(i==1){
+       MatrizAlin[i,1]<-0
+    } else {
+      MatrizAlin[i,1]<-Hueco*i-Hueco
+      }
+  }
+  for(j in 1:LongS1){
+    if(j==1){
+       MatrizAlin[1,j]<-0
+    } else {
+       MatrizAlin[1,j]<- Hueco*j-Hueco
+      }
+  }
+  for(i in 2:LongS2){
```

```
+   for(j in 2:LongS1){
+    Elem1<-S1[j]
+    Elem2<-S2[i]
+    for(k in 1:length(NombresF)){
+      if(Elem1==NombresF[k]){
+        PosF<-k
+      }
+  }
+
+   for(l in 1:length(NombresC)){
+    if(Elem2==NombresC[l]){
+      PosC <-l
+    }
+   }
+
+   Diag<-MatrizAlin[i-1, j-1]+MatrizPuntaje[PosF, PosC]
+   Arriba<-MatrizAlin[i-1, j]+Hueco
+   Izq<-MatrizAlin[i, j-1]+Hueco
+   MatrizAlin[i,j]<-max(Diag, Arriba, Izq)
+   }
+  }
+
+  AlineamA<-character()
+  AlineamB<-character()
+  i<-LongS2
+  j<-LongS1
```

```
+  while(i>1 & j>1){
+    Punt<-MatrizAlin[i,j]
+    Diag<-MatrizAlin[i-1, j-1]
+    Arriba<-MatrizAlin[i-1, j]
+    Izq<-MatrizAlin[i, j-1]
+    Elem1<-S1[j]
+    Elem2<-S2[i]
+
+    for(k in 1:length(NombresF)){
+       if(Elem1==NombresF[k]){
+        PosF <- k
+       }
+    }
+    for(l in 1:length(NombresC)){
+      if(Elem2==NombresC[l]){
+        PosC <- l
+      }
+    }
+
+  if(Punt==Diag+MatrizPuntaje[PosF, PosC]){
+    AlineamA<-c(S1[j], AlineamA)
+    AlineamB<-c(S2[i], AlineamB)
+     i<-i-1
+     j<-j-1
+  } else if (Punt==Arriba+Hueco){
+    NumGaps<-NumGaps+1
```

```
+    }
+  }
+  while(i>1){
+    AlineamA<-c("-",AlineamA)
+    AlineamB<-c(S2[i], AlineamB)
+      i<-i-1
+    NumGaps<-NumGaps+1
+   }
+   while(j>1){
+     AlineamA<-c(S1[j], AlineamA)
+     AlineamB<-c("-", AlineamB)
+     j<-j-1
+     NumGAps<-NumGaps+1
+   }
+
+    library("seqinr")
+    dotplot(Sec1, Sec2)
+    print("Secuencia 1")
+    cat(Sec1, "\n\n")
+    print("Secuencia 2")
+    cat(Sec2, "\n\n")
+    print("Matriz de sustitucion")
+    print(MatrizPuntaje)
+    cat("\n")
+    print("Matriz de Alineacion")
+    print(MatrizAlin)
```

```
+    cat("\n")
+    print("Alineamiento optimo secuencia 1")
+    cat(AlineamA, "\n\n")
+    print("Alineamiento optimo secuencia 2")
+    cat(AlineamB, "\n\n")
+    print("Puntuacion alineamiento optimo")
+    cat(MatrizAlin[LongS2, LongS1], "\n\n")
+    print("Numero de Gaps")
+    cat(NumGaps, "\n\n")
+ }
```

2. Algoritmo Smith Waterman

```
> SmithWaterman <- function (Sec1, Sec2, Hueco, MSCab, MSVal) {
+    NumGaps <- 0
+   MSFilCol <- length(MSCab)
+   #Matriz de puntaje
+   NombresF <- NombresC <- MSCab
+   MatrizPuntaje <- matrix (MSCab, MSFilCol, MSFilCol,
+   byrow=TRUE, dimnames=list(NombresC, NombresF))
+   #Convertir valores negativos a 0
+   for (i in 1:length(NombresF)){
+     for(j in 1:length(NombresC)){
+       if(MatrizPuntaje[i,j] < 0){
+          MatrizPuntaje[i,j] <- 0
+       }
+     }
+   }
+
+   #Resultados parciales de cada posible alineamiento
+   S1 <- c("",Sec1)
+   S2 <- c("",Sec2)
+
+   LongS1 <- length(S1)
+   LongS2 <- length(S2)
+
+   MatrizAlin <- matrix(data=NA, nrow=LongS2, ncol=LongS1,
+   byrow=FALSE, dimnames=list(S2, S1))
```

```
+
+   for(i in 1:LongS2){
+     if(i==1)
+     {
+       MatrizAlin[i,1] <- 0
+     }else{
+       MatrizAlin[i,1]<- Hueco*i-Hueco
+       if(MatrizAlin[i,1] < 0)
+       {
+         MatrizAlin[i,1] <- 0
+       }
+     }
+   }
+
+   for(j in 1:LongS1){
+     if(j==1)
+     {
+       MatrizAlin[1,j] <- 0
+     }else{
+       MatrizAlin[1,j] <- Hueco*j-Hueco
+       if(MatrizAlin[1,j] < 0)
+       {
+         MatrizAlin[1,j] <- 0
+       }
+     }
+   }
```

```
+
+   for(i in 2:LongS2){
+     for(j in 2:LongS1){
+       Elem1 <- S1[j]
+       Elem2 <- S2[i]
+         for (k in 1:length(NombresF)){
+           if(Elem1==NombresF[k]){
+              PosF <- k
+           }
+         }
+
+         for (l in 1:length(NombresC)){
+            if(Elem2==NombresC[l]){
+               PosC <- l
+            }
+         }
+
+         Diag <- MatrizAlin[i-1,j-1] +
+         MatrizPuntaje[PosF,PosC]
+         Arriba <- MatrizAlin[i-1,j] + Hueco
+         Izq <- MatrizAlin[i,j-1] + Hueco
+         MatrizAlin[i,j] <- max(Diag,Arriba,Izq)
+       }
+     }
+     #Hallamos el alineamiento óptimo
+     AlineamA <- character()
```

```
+     AlineamB <- character()
+
+     MaxVal <- 0
+
+     for (i in 2:LongS2){
+       for (j in 2:LongS1){
+         if(MaxVal < MatrizAlin[i,j]){
+             MaxVal <- MatrizAlin[i,j]
+             MaxVali <- i
+             MaxValj <- j
+         }
+       }
+     }
+
+     i <- MaxVali
+     j <- MaxValj
+
+     while((i > 1 & j > 1) | (MatrizAlin[i,j] > 0)){
+        Punt <- MatrizAlin[i,j]
+        Diag <- MatrizAlin[i-1,j-1]
+        Arriba <-MatrizAlin[i-1,j]
+        Izq <- MatrizAlin[i,j-1]
+
+        Elem1 <- S1[j]
+        Elem2 <- S2[i]
+
```

```
+        for (k in 1:length(NombresF)){
+          if(Elem1==NombresF[k]){
+          PosF <- k
+          }
+        }
+
+        for (l in 1:length(NombresC)){
+          if(Elem2==NombresC[l]){
+          PosC <- l
+          }
+        }
+
+        if(Punt == Diag + MatrizPuntaje[PosF,PosC])
+        {
+          AlineamA <- c(S1[j],AlineamA)
+          AlineamB <- c(S2[i],AlineamB)
+          i <- i-1
+          j <- j-1
+        }else if(Punt == Arriba + Hueco){
+          AlineamA <- c("-",AlineamA)
+          AlineamB <- c(S2[i],AlineamB)
+          i <- i-1
+          NumGaps <- NumGaps + 1
+        }else if(Punt == Izq + Hueco){
+          AlineamA <- c(S1[j],AlineamA)
+          AlineamB <- c("-",AlineamB)
```

```
+            j <- j-1
+            NumGaps <- NumGaps + 1
+        }
+      }
+
+      while((i > 1) & (MatrizAlin[i,j] > 0))
+      {
+        AlineamA <- c("-",AlineamA)
+        AlineamB <- c(S2[i],AlineamB)
+        i <- i-1
+        NumGaps <- NumGaps + 1
+      }
+
+      while((j > 1) & (MatrizAlin[i,j] > 0))
+      {
+        AlineamA <- c(S1[j],AlineamA)
+        AlineamB <- c("-",AlineamB)
+        j <- j-1
+        NumGaps <- NumGaps + 1
+      }
+
+    library("seqinr")
+    dotPlot(Sec1,Sec2)
+    print("Secuencia 1")
+    cat(Sec1, "\n\n")
+    print("Secuencia 2")
```

```
+    cat(Sec2, "\n\n")
+    print("Matriz de sustitucion")
+    print(MatrizPuntaje)
+    cat("\n")
+    print("Matriz de Alineacion")
+    print(MatrizAlin)
+    cat("\n")
+    print("Alineamiento optimo secuencia 1")
+    cat(AlineamA, "\n\n")
+    print("Alineamiento optimo secuencia 2")
+    cat(AlineamB, "\n\n")
+    print("Puntuacion alineamiento optimo")
+    cat(MatrizAlin[MaxVali,MaxValj], "\n\n")
+    print("Numero de Gaps")
+    cat(NumGaps, "\n\n")
+  }
```

Glosario

- Ácido nucleico. Polímero formado por la repetición de monómeros denominados nucleótidos, unidos mediante enlaces fosfodiéster.

- Alineamiento de secuencias. Es una forma de representar y comparar dos o más secuencias o cadenas de ADN, ARN, o estructuras primarias proteicas para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados.

- Base nitrogenada. Compuesto orgánico cíclico, que incluye dos o más átomos de nitrógeno.

- Deleción. Nombre que recibe la eliminación de símbolos o letras en el alineamiento.

- Dideoxinucleótido. Nucleótidos que carecen de un grupo 3'-hidroxilo (-OH) en la desoxirribosa.

- Enlace covalente. Enlace entre dos átomos se produce cuando estos átomos se unen, para alcanzar el octeto estable, compartiendo electrones del último nivel.

- Enlace fosfodiéster. Es un tipo de enlace covalente que se produce entre un grupo hidroxilo en el carbono 3' y un grupo fosfato en el carbono 5' del nucleótido entrante.

- Electroferesis. Técnica para la separación de moléculas según la movilidad de éstas en un campo eléctrico.

- Grupo hidroxilo. Es un grupo funcional formado por un átomo de oxígeno y otro de hidrógeno.

- Hueco. Separación añadida artificialmente a una secuencia para maximizar su alineamiento con otras.

- Inserción. Nombre que recibe el símbolo o letra insertada entre 2 posiciones de un alineamiento.

- Isla CpG. Región del ADN con al menos 200 pares de bases y un porcentaje del dinucleótido GC mayor del 50% y con una razón del promedio observado/esperado mayor que 60%.

- Monómero (del griego *mono*, 'uno', y *meros*, 'parte'). Es una molécula de pequeña masa molecular que está unida a otros monómeros.

- Nucleótido. Molécula orgánica formadas por la unión de una pentosa), una base nitrogenada y un grupo fosfato.

- Pentosa. Monosacárido formado por una cadena de cinco átomos de carbono que cumplen una función estructural.

- Polímero de nucleótidos. (del griego *poly*: «muchos» y *mero*: «parte», «segmento») son macromoléculas (generalmente orgánicas) formadas por la unión de monómeros.

- Replicación del ADN. Es el mecanismo que permite al ADN duplicarse (es decir, sintetizar una copia idéntica).

- Secuenciación del ADN. Es un conjunto de métodos y técnicas bioquímicas cuya finalidad es la determinación del orden de los nucleótidos (A, C, G y T) en un oligonucleótido de ADN.

- Splicing (también llamado corte y empalme, empalme o ayuste). Proceso que consiste en la unión covalente de dos fragmentos de ADN bicatenario.

Bibliografía

[1] Astbury William. Nucleic Acid. *Symp. SOC. Exp. Bbl* 1(66). (1947)

[2] Bates Andrew. DNA structure. DNA topology. Oxford University Press. ISBN 0-19-850655-4. (2005)

[3] Byung-Jun Yoon. Hidden Markov Models and their Applications in Biological Sequence Analysis Bentham Science Publishersn Ltd. (2009)

[4] Burge C., Karlin S. Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology. (1997)

[5] Bartlett JA, Jia HP, Mitros JP, Schutte BC, Welsh MJ, et al. Discovery of five conserved β-defensin gene clusters using a computational search strategy. PNAS 99 (4) 2129-2133. (2002)

[6] Dhanda J. S., Shyam S. Chauhan. Structural levels of Nucleic Acids and Sequencing. En All India Institute of Medical Sciences. Molecular Biology (Department of Biochemistry edición). New Delhi - 110 029. (2008)

[7] Dahm Ralf. Fiedrich Miescher and the discovery of DNA. Dev Biol 278 (2): 274-288. (2005)

[8] Dahm Ralf. Discovering DNA: Fiedrich Miescher and the early years of nucleic acid researchs. Hum Genet 122 (2): 565-581. (2008)

[9] De Robertis, E.D.P. Biología celular y molecular. El Ateneo. ISBN 950-02-0364-2. (1998)

[10] Durbin R., Eddy S., Krogh A., Mitchison G. Biological Sequence Analysis. Probabilistic Models of proteins and nucleic analysis. Cambridge University Press. (1998)

[11] Ewens J. Warren, Grant Gregory . Statistical methods in bioinformatics. An Introduction. Springer. (2005)

[12] Frommer M, Gardiner-Garden M. CpG islands in vertebrate genomes. J. Mol. Biol. 196(2):261-82. (1987)

[13] Gotoh O. An Improved Algorithm for Matching Biological Sequences. Journal of Molecular Biology 162:705-708. (1982)

[14] Caffo Brian, Irizarry Rafael A., Jaffee Harris A., Wu Hao. Redefining CpG islands using Hidden Markov Models. Biostatistics, 11, 3. (2010)

[15] Levene P. The structure of yeast nucleic acid. J. Biol. Chem. 402(2):415-24. (1919)

[16] Garraway Levi A. and Lander Eric S. Elsevier, Issue 1, 17-37. (2013)

[17] Berg Jeremy, Stryer Lubert, Tymoczko John: versión española por: Prof. Dr. Miguel Ángel Trueba. Bioquímica. Reverté. (2013)

[18] Oliver L. Jose et al. Isocore chromosome maps of eukaryotic genomes. Gene 276, 4756. (2001)

[19] Rabiner R. Lawrence. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". Proceedings of the IEE, Vol. 77, No. 2

[20] Juang B. H., Rabiner R. Lawrence. Fundamental of Speech Recognition. Prentice-Hall International, Inc. (1993)

[21] Smith Temple F., Waterman Michael S. Identification of Common Molecular Subsequences. Journal of Molecular Biology 147:195-197. (1981)

[22] Srinivasan M., Venkata H. P. Control Automation Robotics & Vision (ICARCV), 13th International Conference, pp. 1864-1869, 1'0-12 Dec. 2014. (2014)

[23] The ENCODE Project Consortium. Identification and analysis of functional elements in 1 % of the human genome by the ENCODE pilot project. Nature 447(7146):799-816. (2007)

[24] Crick F. H. C. and Watson J. D. A structure for Deoxyribose Nucleic Acid Nature 171:737-738. (1953)

[25] Welsh Dominic. Codes and Cryptography. Clarendon Press, Oxford. (1988)