Secuenciamiento de ADN por métodos de contenido: formalismos y aplicaciones.

Juan Lucas Bali

16 de Julio, 2004

Índice

1. Objetivos

1.1. Qué es la bioinformática

La bioinformática es una ciencia bastante nueva y hasta incluso necesaria en cualquier tipo de trabajo relacionado con la genética. Esta disciplina es fundamentalmente un intento de incorporar técnicas propias de la computación científica en el análisis de información biológica.

El surgimiento de la misma se debe más que nada al gigantesco incremento en la cantidad de datos biológicos, provenientes de los disitntos secuenciamientos genéticos. Las bases de datos de secuencias genéticas han estado experimentando estos últimos años un crecimiento exponencial, impulsado fuertemente por el proyecto del genoma humano

La masividad de los datos ha vuelto impracticable cualquier tipo de análisis artesanal, se impone la necesidad de introducir un marco de trabajo que sea más abierto a la automatización, y para eso no es posible desdeñar los conceptos que pueden proveernos las disciplinas propias del área de la informática.

Puede ser interesante destacar que no hay practicamente ninguna rama aplicada de la informática que no se pueda aplicar a los problemas actuales del trabajo biológico. Enumeremos algunas de estas disciplinas y su aplicabilidad:

- Algorítmica: es una pieza clave fundamental. Los nuevos mecanismos biológicos estarán fuertemente basados en programas, y estos requieren de algoritmos eficientes y seguros para poder implementarse.

- Estadística: el gigantesco nivel de datos que actualmente se está manejando permite aplicar ideas y conceptos de estimación para poder inferir reglas que respondan a un conjunto importante de la población.

- Bases de Datos: surge de la necesidad de disponer de un mecanismo eficiente y rápido de almacenado y consulta de los datos biológicos. En particular, el estudio de la semejanza y homología de secuencias requiere en general de un análisis de los datos, es preciso disponer de métodos veloces de acceso a los mismos.

- Ingeniería de Software: a medida que los conceptos de software se introduzcan en el área surgen necesidades de organización que trascienden lo meramente algorítmico.

La mayoría de estas disciplinas se han aplicado ya en la bioinformática. Hay ya una larga tradición de investigación en esta áreas, particularmente en la esfera de "pattern recognition", donde diversos investigadores ([BAY95], [BAY94], [LIU01] y varios más) las han aplicado a la decodificación de la informaciòn genética contenida en el ADN de los seres vivos. En este trabajo pretendemos mostrar como el uso de algunas de estas herramientas permite vislumbrar una manera que creemos nóvel para encarar el problema desde el punto de vista teórico. No pretendemos aquí emular o superar el trabajo de los autores antes citados, sino mostrar lo que creemos es un enfoque algo distinto,formalmente riguroso, que seguramente tendrá sus deficiencias, pero que se espera permita avanzar en el desarrollo de estas técnicas merced a su expresión formal del problema. Sería necesario profundizar esta línea de trabajo y complementarla con ideas de otros autores para arribar a una metodología realmente eficiente.

1.2. Búsqueda de genes

Uno de los problemas más importantes en el área de la bioinformática y de la genética en general es la búsqueda de genes. Antes que nada, vamos a indicar con precisión que es lo que se entiende por gen y, en particular, como se lo tratará en el presente trabajo.

Primera Definición (Mendel) :
"Un gen es la unidad básica a través del cual las características hereditarias se transmiten de un padre a su retoño."

Segunda Definición (Moderna/Molecular):
"Un gen consiste de una secuencia de ADN que codifica y dirige la síntesis de una proteína, o en algunos casos de tARN, rARN o algún otro ARN estructural."

Ambas defincciones son diferentes en cuanto a su nivel y no está claro que en la realidad sean efectivamente equivalentes. Nos inclinaremos por la segunda definición no porque nos parezca más acertada sino por ser más propia y adaptable con lo que se estudiará aquí.

A medida que se fueron perfeccionando los métodos de secuenciamiento de ADN, la cantidad de datos genéticos ha sufrido un incremento colosal a tal punto que las bases de datos más importantes (como puede ser GenBank)

deben actualmente albergar datos genéticos de docenas de especies. Pensando que en el ser humano la cantidad de bases que componen su código genético se mide en el orden de los 3000 millones de bases, el problema se vuelve claro.

Más aún, tenemos una dificultad adicional: la mayoría de esos datos (por ahora) no nos interesan. En efecto, corresponden a datos que no tienen injerencia alguna sobre el comportamiento del ser vivo, o por lo menos eso es lo que se cree hasta el momento. En promedio, los organismos eucariotas (como el ser humano y en general todos los organismos de un apreciable nivel de complejidad) disponen de una porción muy reducida (se estima entre un 3 y un 5 por ciento) que se utiliza para la fabricación de proteínas. Se desea normalmente estudiar únicamente aquellas porciones que efectivamente sirven para la generación de secuencias proteicas, lo que normalmente en la literatura del ramo son denominados los *genes* del ADN, y claramente un método manual es ineficiente y muy propenso a errores.

1.3. ¿Por qué buscamos los genes?

Primero y principal, porque es fundamental para proseguir con el estudio genético de un genoma secuenciado. Si es cierto que lo único que realmente tiene apreciación fenomenológica directa sobre el individuo son aquellas partes que se encargan de la generación de proteínas, es primordial poder ubicar a los mismos dentro de una secuencia.

Una vez que sabemos cuales son los genes, podemos dedicarnos a:

- Determinar la secuencia primaria de proteínas.
- Estudiar aspectos de evolución genómica concentrando esfuerzos únicamente en estas porciones especiales.
- Efectuar distintos tipos de comparaciones entre especies.
- Poder determinar la función de todas las proteínas.
- Obtener patrones comunes de expresión de las proteínas.
- Encontrar y entender las componentes genéticas de una enfermedad.
- Determinar procedimientos para la confección de medicamentos especializados en base de la función del gen.

1.4. Background biológico

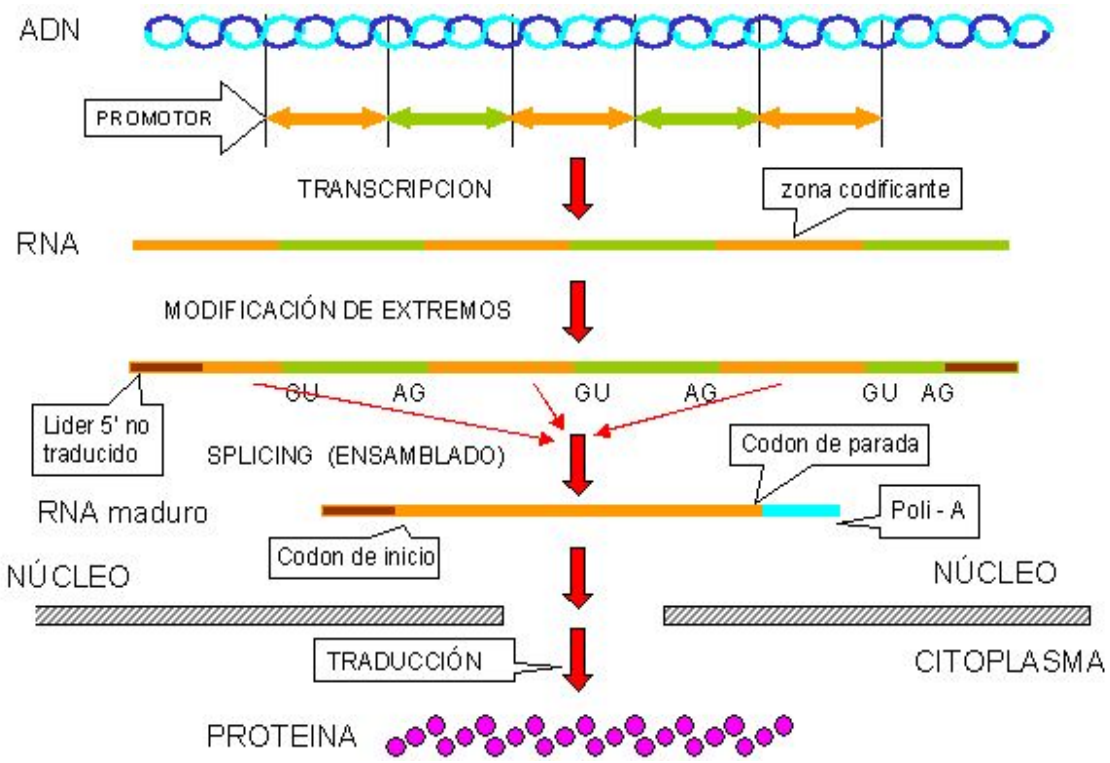
La *expresión genética* es la forma a través del cual una secuencia de ADN genera una proteína, involucrando dos pasos: la *transcripción* y la *traducción*. La transcripción produce un mRNA (RNA mensajero) utilizando como base a la secuencia de ADN, siendo la primera un complemento de la tira de ADN que se usó. El próximo paso, la traducción, sintetiza la proteína en base a la información codificada en el mRNA. Este último paso es efectuado por elementos sub-celulares denominados ribosomas.

La transcripción se lleva a cabo desde la punta 5' a la punta 3' de la tira de ADN copiado, dirección que se denomina como *downstream* mientras que la dirección opuesta se la llama *upstream*. La enzima que lleva a cabo la transcripción, ARN polimerasa, empieza la transcripción unas bases arriba de la región que codifica realmente la proteína y termina la transcripción algunas bases después de la región codificante. Estas regiones delimitantes de la porción codificante del ADN se transcriben en el ARN pero no se codifican en ninguna proteína. Se las denomina *regiones sin traducción* (UTR, del inglés *untranslated regions*).

Las moléculas polimerasa ARN empiezan la transcripción reconociendo y acoplándose a *regiones promotoras*, regiones que controlan el flujo de transcripción.

Las proteínas se encuentran compuestas por aminoácidos. Los ribosomas producen estos aminoácidos a través de la decodificación de la información contenida en los mRNA. Cada triplete de bases en un mRNA es un comando para el ribosoma, denominado *codón*. Existen 64 codones posibles pero únicamente tenemos 20 aminoácidos distintos, la razón de esto es que hay varios codones que codifican el mismo aminoácido. Este mapeo entre codones y aminoácidos se denomina *código genético*

Uno de los codones, denominado el *codón de inicio*, indica el comienzo de la traducción (así también como la codificación para el aminoácido Metonina). Tres de los codones indican el fin de la traducción y se los denomina *codones de parada*. El ribosoma empieza su recorrido en el mRNA deslizándose desde la punta 5' a la punta 3'. Cuando se detecta un codón de inicio, el ribosoma empieza a generar aminoácidos en base de la información del mRNA. El proceso se detiene cuando el ribosoma localiza un codón de parada.



1.5. Estructura genética

La estructura genética varía fuertemente entre los organismos procariotas y los eucariotas, a tal punto que los mecanismos de detección suelen presentar fuertes diferencias para cada caso. La principal diferencia que define ambas familias de organismos es el hecho de que las células procariotas no cuentan con un núcleo. A nivel genético las diferencias son también bastante notorias. Para empezar, en los procariotas la mayor parte de la secuencia de ADN se codifica en proteínas. Además, cada gen es una tira continua de bases, sin interrupciones en el medio.

Este fenómeno de las procariotas no se da en las eucariotas. De hecho, en la mayor parte de los casos la región de ADN codificante no es continua. Estas regiones suelen estar definidos como una intercalación de *exones* e *intrones*. Durante la transcripción, tanto el exón como el intrón son codificados en el ARN , según orden de aparición. Más adelante, se lleva a cabo un proceso de *splicing* a través del cual se descartan todos los intrones de la secuencia. Los segmentos restantes de ARN, los exones, son ligados para formar así la

tira de ARN maduro.

La estructura de un típico gen multi-exón es la siguiente: se empieza con una región promotora, seguida por una región que se transcribe pero que no se traduce, el 5' UTR. A continuación nos llega el exón inicial junto con su codón de inicio. A partir del exón inicial, llega una serie intercalada de exones e intrones, finalizadas por un exón de de terminación con un codón de parada. Luego viene otra región no codificante, el 3' UTR. Finalizando el gen eucariota, entrontramos una señal de polidenilación (polyA) : el nucleótido adenina (A) repetido varias veces.

Los límites entre los intrones y los exones (*splice sites*) se señalan con una secuencia corta (de dos bases) específica. El final 5'(3') de un intrón(exón) se lo denomina sitio *donor*, y el final 3'(5') de un intrón(exón) se lo denomina sitio *aceptor*.

1.6. Detección de genes

Ya se ha mencionado la importancia que tiene la detección de genes en una secuencia de nucleótidos. Además, hemos comentado las dificultades adicionales que presenta el proceso de reconocimiento en organismos eucarióticos. Resumimos las dificultades del mismo:

- los genes no suelen ser contiguos. Eso es, entre dos genes suelen haber grandes regiones intergénicas que no codifican proteínas.
- los genes no suelen ser continuos. Una secuencia proteínica no necesariamente se encuentra especificada por una secuencia continua de ADN. Es normal que los genes se encuentren partidos en un número de fragmentos codificantes, conocidos como exones. Las regiones que se encuentran entre estos exones se los denomina intrones.

En general, la mayor parte del ADN de los organismos eucariotas complejos se compone de regiones intergénicas e intrones, en donde ninguna de ellas participa en la codificación de aminoácidos.

A la hora de intentar inferir los genes contenidos en una secuencia debemos encarar fundamentalmente dos problemas:

- detecar las regiones exónicas dentro de un gen.
- determinar, en función de los exones deducidos, como se componen los genes.

En este trabajo se estudiará y ejemplificará lo primero.

Enunciamos a continuación una breve clasificación de los distintos métodos utilizados para este fin.

1.6.1. Métodos basados en señales

Se intenta detectar aquellas subsecuencias que disponen de un significado especial que permita decidir los límites entre las distintas regiones. Enumeramos a continuación las más conocidas:

- Codones de Inicio: el codón ATG marca el inicio de una zona de codificación de proteínas.
- Codones de Parada: en el código genético estándar, estos son el TAA, el TAG y el TGA.
- Sitios de corte ("splice junctions") : determinan los bordes entre un intrón y un exón. El lateral inicial se lo denomina "donor" y suele estar representado por el AG. El lateral final se lo denomina "aceptor" y suele aparecer como GT.

Un método muy primitivo de detección de potenciales exones sería marcar en la secuencia todas las apariciones de estas señales especiales. En función de las mismas determinaríamos los exones, que podemos agrupar en cuatro categorías:

- exones de inicio(INIT): el primer exón de un gen. Por un lateral se delimita con el codón de inicio ATG y por el otro lateral figura un splice junction GT.
- exones interiores(INTR): se delimitan en ambos laterales por splice junctions, un AG en un lado y un GT en el otro.
- exones de terminación (TERM): por un lateral se delimita con un splice junction (el AG) y por el otro con un codón de parada.
- exones simples(SNGL): es un caso de un gen continuo, sin intrones en el medio. Por un lado se tiene el codón de inicio y en el otro lado el de parada.

El problema de aplicar directamente este método es que las señales son muy simples y es factible que las mismas aparezcan numerosas veces y por cuestiones ajenas a la delimitación exónica.

En efecto, el nivel de conocimiento que se tiene en la actualidad de las señales es muy limitado. En la práctica, resulta difícil distinguir las señales que realmente cumplen un rol dentro de la maquinaria genética de aquellos que no cumplen rol alguno.

Por este motivo es que los métodos basados en señales suelen implementarse junto con algún otro esquema de detección. Es decir, se utilizan para complementar a algún otro mecanismo.

Dentro del contexto de la búsqueda de señales es recurrente el uso del término *motif* para designar un elemento conservado en un alineamiento de secuencias que usualmente se correlaciona con una función particular. El hecho de que se conserve en varias secuencias ya es un aspecto importante que permite adjuntar al mismo un valor predictivo de cualquier aparición posterior de tal región estructural/funcional en alguna nueva secuencia.

1.6.2. Métodos basados en contenido

En estos métodos se procura estudiar el posible comportamiento estadístico que caracteriza las regiones codificantes para poder así distinguirlas de aquellas que no lo son. Enumeramos a continuación algunos esquemas simples de reconocimiento por contenido (ver [GUI99]):

- Utilizacicón de codones: sabiendo la frecuencia de aparición de cada gen dentro de una familia dada, podemos utilizar estos valores para inferir si una secuencia se adecua o no a las mismas. Es posible también hacer algunos modificaciones sobre el mismo en donde se considere más de un codón.

- Utilización de aminoácidos: es como el anterior pero se consideran en vez la frecuencia de utilización de los aminoácidos en vez de los codones.

- Preferencia de codón: sabemos que en muchos casos un aminoácido se encuentra codificado por varios codones. Se asume entonces que en una familia/especie dada existe una preferencia a elegir algún codón específico a la hora de codificar.

- Distribución de bases en cada posición: se trabaja asumiendo conocida la distribución de los nucleótidos en cada una de las tres posiciones del codón.
- Modelos de Markov: se asume que la probabilidad de una base depende de la base ubicada en la posición anterior. Se puede generalizar para que involucre más de una base.

El presenta trabajo estará enfocado principalmente sobre el último punto.

Los métodos basados en contenido por sí solo tampoco resultan exactos, suelen tener problemas para distinguir regiones que se encuentren ubicadas cerca de alguna frontera. Esto se puede potenciar si se considera que los exones suelen ser reducidos en longitud.

1.6.3. Métodos basados en Homología

Fundamentalmente, se intenta ver si la secuencia dada tiene algún parecido con otra secuencia ya almacenada y clasificada. No es raro que se trabajo a nivel proteico debido a que se suele conocer más sobre las proteinas que sobre los genes. Es preciso tener acceso a una base relativamente importante de secuencias ya anotadas con anterioridad. Esto genera una fuerte dependencia con los datos ya existentes sobre el secuenciamiento pero es en la práctica muy útil cuando se trabaja con genes pertenecientes a una misma familia ya estudiada.

1.7. Conceptos e ideas que se suelen utilizar en la predicción de genes

Se enumeran a continuación algunas ideas que normalmente suelen ser integradas en diversas herramientas de detección de genes.

1.7.1. Análisis de discriminación lineal (LDA) y cuadrática (QDA)

Es un método de reconocimiento de patrones utilizados para la clasificación de muestras de distintas clases. Este método cae dentro del paradigma de aprendizaje supervisado. Se asume que las muestras responden además a una serie de medidas y en función de las mismas el LDA permite la definición de un hiperplano que mejor separa ambas clases. QDA utiliza para la separación una superficie cuadrática en vez de un hiperplano.

Se exhibe a continuación (figura 1) un ejemplo en el cual dado una región con elementos de dos clases diferentess se efectúa tanto un LDA (recta L) como un QDA (curva Q). Es interesante notar como la curva cuadrática puede en este caso separar correctamente ambas regiones en función de los puntos.

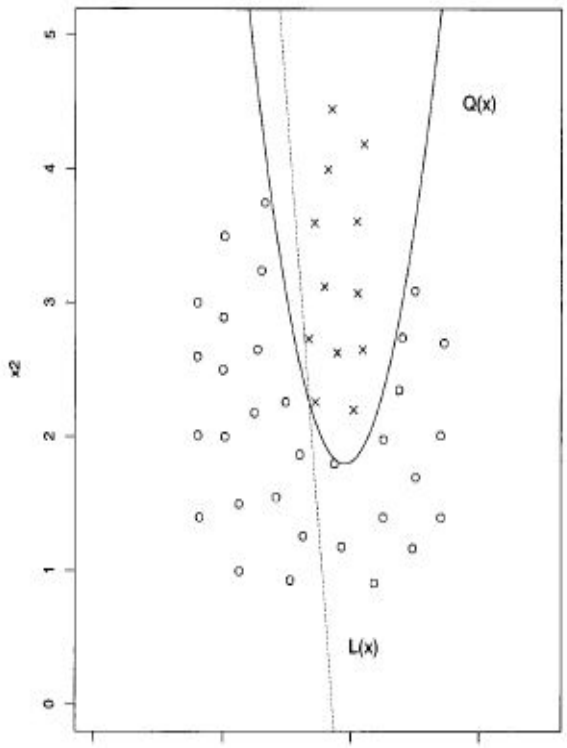


Figura 1: LDA y QDA

1.7.2. Médidas de codificación de hexómeros

Algunos métodos interpretan las secuencias como sucesiones de "palabras" (así llamadas puesto que los nucleótidos no son independientes uno de otro) de longitud k (k-tuplas); las 6-tuplas se las llama hexómeros. La frecuencia de hexómeros ubicados dentro de un marco apropiado de lectura ha sido utilizado tradicionalmente como un mecanismo para la discriminación de regiones codificantes de aquellas que no lo son. Esto es así puesto que algunas "palabras" suelen ser más frecuentes que otras en algún tipo de ADN.

1.7.3. Métodos de matrices pesadas (WMM) y de arreglos pesados(WAM)

Se usan para la calificación de un sitio potencialmente poseedor de una señal. En un WMM, un puntaje s(x,b) es asignado en la posición x para cada base b de forma tal que el puntaje total del sitio se puede calcular como la suma de los puntajes de todas las posiciones que lo componen. En el WAM, el puntaje se asigna sobre palabras y no sobre sitios.

Estos mecanismos se suelen utilizar para la detección de algún *motif* particular. La idea es disponer de un WMM/WAM que lo represente e ir recorriendo la secuencia hasta encontrar una región en donde el puntaje sobre el WMM/WAM sea lo suficientemente alto.

1.7.4. Redes neuronales

Una red neural consiste en un gran número de elementos simples de procesamiento, las neuronas. Una de sus características más importantes es su naturaleza adaptiva en donde el aprendizaje por ejemplo suplanta la programación convencional a la hora de resolver problemas. Son fuertemente aplicables en situaciones en donde la intuición general sobre el problema es escasa pero sin embargo se cuenta con un gran número de datos de entrenamiento para ejercitar el sistema. Una red neuronal se caracteriza por sus patrones conectivos (la arquitectura), el mecanismo de determinación de los pesos sinápticos (el aprendizaje) y una función de activación final que termina de efectuar la clasificación.

El ejemplo más simple de red neuronal es el *perceptron*, en donde no existen capas intermedias, tan sólo tenemos una capa de entrada y una de salida. Las ventajas de este modelo es que es simple de entrenar y es poco susceptible a los efectos de sobreespecialización producto de las (supuestas) correlaciones de alto orden de los datos. Su desventaja es su limitación a la hora de distinguir clases en función de las variables (o características) del elemento que se pretende clasificar. El modelo de perceptrón simple presenta un fuerte correlato con el mecanismo de LDA, de hecho se basa esencialmente en el ajusta progresivo de un hiperplano en el espacio de los elementos con el fin de reducir el error de clasificación.

A la hora de clasificar secuencias es posible utilizar una red neuronal alimentada por alguna serie de características o medidas, como la longitud del potencial exón o un cierto puntaje asignado sobre la porción de la se-

cuencia en virtud de alguna determinación estadística previa. Un atractivo especial del método es que permite integrar de forma elegante información proveniente de distintas fuentes, permitiría por ejemplo una combinación de un estudio por señales (proveniente eventualmente por un WMM/WAM) con otro inherentemente estadístico, como un análisis por procesos de Markov o una medida de codificación de hexómeros.

1.7.5. Modelos ocultos de Markov (HMM)

Los modelos ocultos de Markov representan a un sistema como un conjunto de estados discretos y unas transiciones entre esos estados, con una probabilidad asignada para cada transición. Los modelos se dicen "ocultos" cuando uno o más estados no se pueden ser observados directamente. Es decir, cuando dichos estados se activan emitirán a su vez un símbolo con una probabilidad dada. Es posible modelar un problema de reconocimiento de genes mediante este modelo definiendo algunos estados propios de las regiones codificantes y otros para las no codificantes. Cada estado emitirá un símbolo con alguna probabilidad pero variará la misma en función de si el estado es o no codificante. Esto resulta tan solo en una primer aproximación, es posible efectuar numerosas mejoras sobre las mismas incluyendo estados para modelar por ejemplo codones de parada o inicio. Sin embargo, modelos complejos incurren a su vez en un costo mayor a la hora de entrenar, con todo lo que eso implica (por ejemplo, problemas de sobreespecialización).

En el siguiente gráfico (figura 2) tenemos un ejemplo simple de HMM para el reconocimiento de regiones codificantes en una secuencia.

Para poder determinar los genes en una secuencia se determina, junto con el HMM definido y mediante un algoritmo de análisis de rutas (por ejemplo el algoritmo de Viterbi que dado una secuencia determina la ruta de estados más probable), los estados recorridos y de esa forma se determinar para cada posición el estado que lo generó y por ende si era o no una región codificante. Los HMM están tratado con profundidad en [RAB89] y, con una aplicación a la bioinformática, en [DUR98].

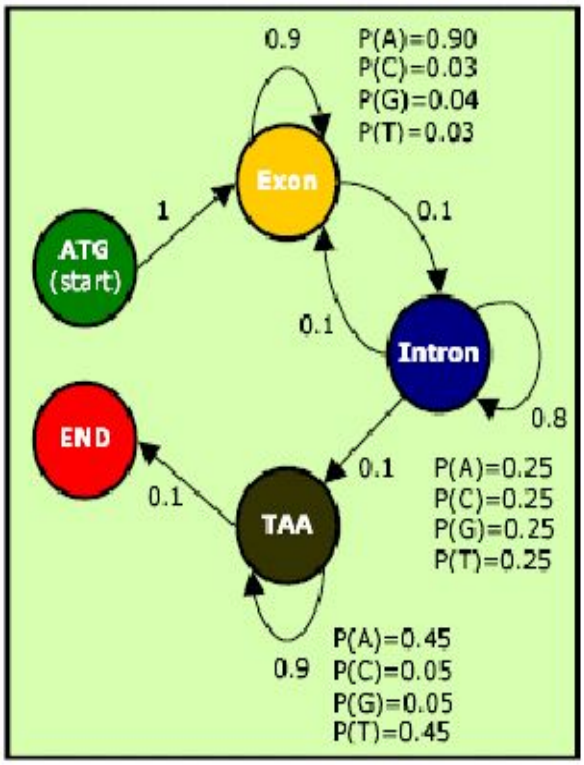


Figura 2: HMM simple

1.8. Breve repaso de algunas herramientas utilizadas para la detección de genes [BAX01]

1.8.1. GRAIL

Grail es una de los primeras herramientas desarrolladas para la detección de genes y todavía se encuentra en desarrollo. Existen actualmente varias versiones de la misma pero distinguimos tres en particular.

- GRAIL1: se basa en un esquema de reconocimiento de potencial codificante mediante una red neuronal con una ventana de tamaño fijo (100 sitios). Para la evaluación del potencial codificante no recurre a información adicional como los "splice junctions" o los codones de inicio y parada. Se termina evaluando la "calidad exónica" mediante un puntaje

- GRAIL1A: incorpora mejoras a la versión anterior, examina las regiones inmediatamente adyacentes a las regiones con potencial codificante. Por otra parte efectúa ya un intento de optimizar los límites de una región codificante. En efectividad, es mejor que el GRAIL1 para la detección de exones verdaderos y la eliminación de falsos positivos.

- GRAIL2: esta versión ya empieza a utilizar ventanas de tamaño variable. Además, incorpora en su análisis información genómica de contexto como los "splice junctions", las señales PolyA y los codones de incio/parada.

Se muestra en la figura3 un esquema de la red neuronal utilizada por GRAIL2 para el procesamiento de diversos indicadores:

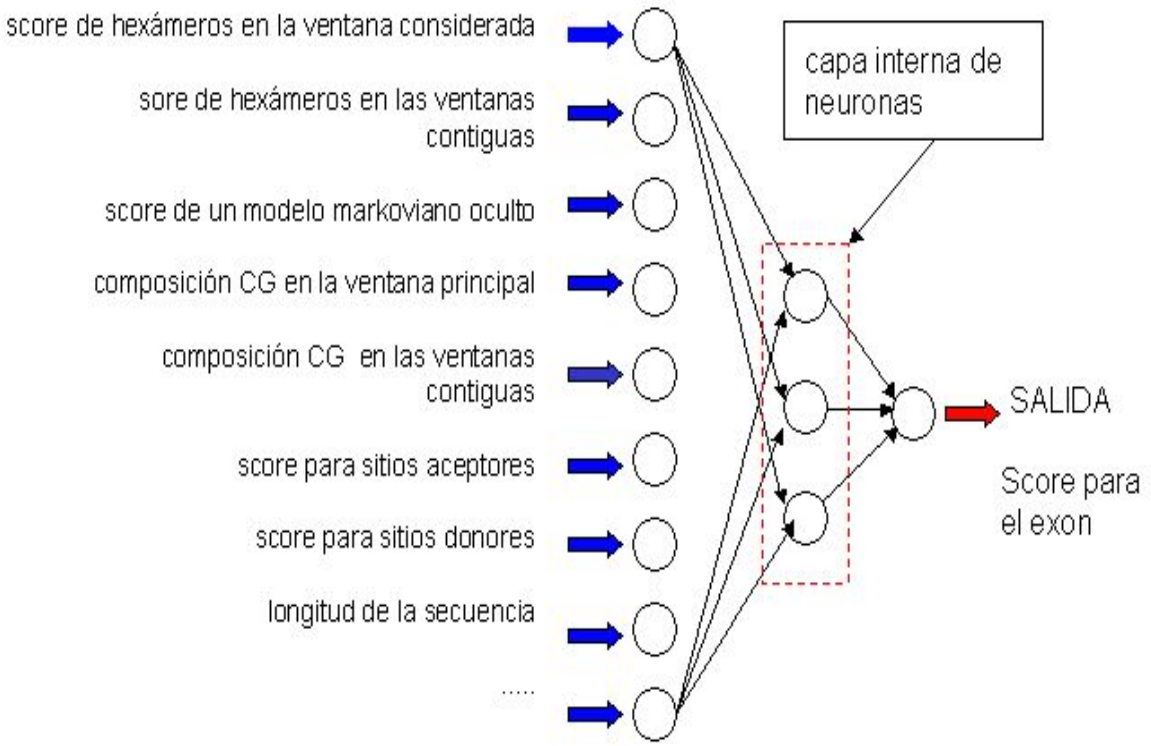


Figura 3: Red del GRAIL2

1.8.2. MZEF - Michael Zhang's Exon Finder ([ZHA97])

Esta herramienta se encuentra limitada a la detección de exones, o sea no se puede a partir del mismo determinar la estructura del gen. Para la discriminación entre regiones codificantes y no codificantes recurre principalmente a un análisis por discriminación cuadrática (QDA). En ese sentido resulta en una mejora sobre herramientas anteriores que recurrían a un análisis por discriminzación lineal (LDA), típicamente como resultado de un estudio mediante un modelo de perceptrón simple.

Para discriminar entre ambas regiones, MZEF efectúa la discriminación sobre diferentes medidas:

- longitud del exón
- transiciones intrón-exón
- puntajes por puntos de ramificación
- puntajes por sitios 3' y sitios 5'
- puntaje del exón

1.8.3. GeneParser ([SNY93])

Se basa en la aplicación de un esquema de programación dinámica para la identifiación de exones e introns en secuencias de ADN. En primer lugar se cuantifica con un puntaje las secuencias de interés en base a su información relevante de señales (como los "splice junctions") junto con algunas medidas de contenido, como un análisis de frecuencia de hexómeros, utilización de codones y asimetrías periódicas. La información así recolectada es organizada para ser procesada por un algoritmo de programación dinámica que fuerza que los introns y exones no se pisen entre sí y determina de esa manera la combinación de exones e introns con mayor puntaje. El puntaje para los distintos procedimientos se determina por una red neuronal entrenada para maximizar el número de predicciones correctas.

1.8.4. GeneID

GeneId se implementó pensando en una estructura jerárquica. En el primer paso, los "splice sites", y los codones de inicio/parada son predecidos y cal-

ificados sobre la secuencia mediante un WMM. En el segundo paso, se construyen los exones a partir de este sitio. Los exones se califican como la suma de los puntajes de los sitios que lo defininen más el logaritmo de una verosimilitud de pertenecer a una región codificante, calculada en función de un modelo de Markov (normalmente de orden 5). Finalmente, del conjunto de exones predecidos se ensambla una estructura genética maximizando la suma de los puntajes de los exones constituyentes.

Conceptualmente, se tiene una función de puntaje S y las siguientes ecuaciones:

$S(gen) = s(exon_1) + \ldots + s(exon_n)$
$s(exon) = s(contenido) + s(sitios)$
$s(sitios) = WMM$
$s(contenido) =$ Markov de orden 5

1.8.5. Procrustes

A diferencia de los mecanismos antes explicados, Procrustes es esencialmente un método de búsqueda por homología. Procrustes recurre a un acercamiento basado en la similitud para el reconocimiento de genes. Se asume que la cantidad de genes ya estudiados es tan grande que es muy probable que un gen recientemente secuenciado tenga un homólogo analizado lo suficientemente parecido.

Dado una secuencia genómica y un conjunto de exones candidatos, se procede a explorar todos los ensamblajes posibles de exones y se busca una cadena de los mismos con el mayor parecido con una proteína objetivo relacionada. El conjunto de exones candidatos se construye seleccionando todos los candidatos entre sitios aceptores y donores, seguido de una filtración basada en un estudio de contenido.

1.9. Reconocimiento de patrones

Ya se ha comentado sobre el concepto de *motif* cuando se discutió el reconocimiento basado en señales. Se pasa a continuación a profundizar un poco más las ideas en este sentido.

Interesa en la biología la obtención de patrones que se conserven sobre distintas secuencias, esto es así puesto que es probable que representen objetos que desempeñen alguna función biológica. O, por otro lado, su presencia podría ayudar a determinar la cercanía (o no) de algún otro objeto de interés.

Por ejemplo, para detectar regiones codificantes es bueno saber, por ejemplo, los sitios en donde es probable que se presente alguna señal relevante (un aceptor por ejemplo).

En general tendremos dos problemas. Por un lado nos interesará, conociendo las características estadísticas de un *motif* o patrón, detectar la presencia del mismo en una secuencia dada (el problema de *localización* de patrones). Por otro lado, vamos a querer deducir patrones útiles en función de una serie de secuencias, lo que se puede definir como el patrón de consenso (el problema de *extracción* de patrones). Muchas veces es ésto lo que interesa, dado un conjunto de secuencias determinar si existe un patrón que se conserve en un cierto grado sobre todas las secuencias, por lo que tendrían una fuerte chance de representar una entidad biológica interesante. Otro aspecto a destacar a tener en cuenta es el medio a través del cual se representará el *motif*. Una posibilidad sería utilizar una expresión regular, que en su exponente más simple sería directamente una secuencia (el consenso). Admitiendo un cierto grado de variabilidad cuantificable, se suelen utilizar otros modelos, como los WMM/WAM y hasta inclusive los modelos ocultos de Markov.

En la actualidad existen esencialmente dos estrategias para hallar *motifs* en el ADN: examinar la sobrerepresentación de todas las posibles n-uplas o bien recurrir a un proceso iterativo que actualice una matriz de probabilidades del motif. El primero se basa en el cálculo de la frecuencia de cada *motif* de un cierto tamaño (n) basado en la distribución de la secuencia de entrada, para luego detectar aquellas n-uplas que son más abundantes en la entrada de lo que se esperaba (ejemplo [HAM02]). La segunda estrategia emplea una matriz de probabilidad para el *motif*, especificando la probabilidad de cada letra para cada posición. Un procedimiento iterativo, implementado a través de un algoritmo de maximización de la expectativa o bien un muestreo de Gibbs, se utiliza para mejorar sucesivamente dicha matriz hasta que se converge. Ejemplos de métodos que siguen esta línea de resolución son el MEME ([BAY95], [BAY94]) y el BioProspector([LIU01]).

1.10. La idea en el trabajo

El enfoque del sistema de detección propuesto en este trabajo está fuertemente vinculado con el análisis del contenido de la secuencia.

Se presenta un modelo estadístico que permita la clasificación de distintas partes de una cadena de símbolos, que en principio nada tiene que tener que ver con los nucleótidos. Se introduce el concepto de "textura" como el de

un esquema de generación estocástica de secuencias. En ese sentido se define formalmente esta idea junto con algunas otras definiciones auxiliares. Inmediatamente después se estudia el problema de la detección de texturas en una secuencia. Es decir, dada una secuencia y una serie de texturas nos interesa saber qué textura pudo haberla generado. Debido a la naturaleza estocástica de las mismas, la respuesta nunca puede ser certera. Dado que se esperan errores introducimos una medida que permita medir, cuantitativamente, el nivel del mismo.

Durante este estudio surge un concepto que nos permite englobar a todos los métodos de reconocimiento a través de una única fórmula de energía. Así como se expone un tratamiento formal para la generación de secuencias por texturas, lo mismo se hace para lo relacionado al reconocimiento de las mismas. Se plantea entonces la estrecha relación entre un método de reconocimiento y una expresión enérgética que dependa de las suposiciones hechas en el reconocimiento.

Para darle un contenido práctico al trabajo se estudia la aplicabilidad de estas ideas para un caso concreto, el reconocimiento de regiones significativas en una secuencia de nucleótidos. Más en particular, se ve como es posible definir un predictor de genes.

Nos proponemos dar tan solo una primera aproximación al problema. Es muy difícil implementar un mecanismo adecuado de detección, ya que el comportamiento genético presenta fuertes variaciones entre las familias genéticas y entre las especies. Difícilmente se pueda establecer una regla universal que las cubra a todas ellas. Los conceptos de textura y de reconocimiento estadístico no son nuevos en el área general de la computación. Son ampliamente estudiados en el campo del reconocimiento digital de imágenes, y es de ese ramo del cual se extraen y aplican buena parte de las ideas. Sin embargo, será preciso adaptar las mismas a un contexto específico y propio de la bioinformática y, en especial, del reconocimiento genético.

1.11. Las dos dimensiones

El punto fuerte, y que se destaca durante este trabajo, gira alrededor de una idea de "doble dimensionalidad". Una de las dimensiones consiste en el juego de texturas que representarán cada una de las partes que vamos a querer distinguir. En la otra dimensión se tiene una función "energía" que cuantifica de alguna manera la calidad de una predicción dada, eventualmente acompañado por un esquema de maximización funcional para poder procesar

a la misma.

Debe resaltarse la fuerte separación que es posible efectuar entre ambas partes, al punto tal de que es factible el desarrollo en paralelo de cada una de ellas para una posterior integración de ambas en la forma de reconocedor estadístico. Hilando más fino la definición podríamos a su vez separar el esquema de minimización de la definición de la energía y tener así una tercer dimensión, también paralelizable junto con el resto. Se opta por no seguir esta alternativa pues en general el esquema de minimización dependerá fuertemente de la función de energía, a veces ni siquera será necesario contar con esquemas sofisticados.

Este es el enfoque principal del presente trabajo. En efecto, empezaremos definiendo formalmente la noción de textura junto con una serie de propiedades deseables entre ellas. A continuación, consideraremos la energía y analizaremos algunos ejemplos que podrían usarse para este fin, cada una de ellas con sus ventajas y desventajas propias.

1.12. Estructura del trabajo

En el trabajo empezaremos enfocándonos en una de las dimensiones, referidas a las texturas y sus propiedadades. El capítulo 2 definirá distintos conceptos sobre las mismas mientras que en el capítulo 3 se estudiará la relación entre distintas texturas desde un punto de vista estadístico.

El capítulo 4 ya empezará a tratar el tema de la segunda dimensión y empieza en particular definiendo un método simple de detección de texturas para luego proseguir, en el capítulo 6, con un mecanismo más complicado que introduce algunas nociones de estadística bayesiana.

Todos los capítulos estarán provistos de una serie de ejemplos con el fin, por un lado, de aclarar las ideas que se introducen y, por otro, el de demostrar utilidades prácticas de la teoría que se expone. Los ejemplos a tratar consisten fundamentalmente en el reconocimiento de regiones codificantes en secuencias de ADN.

1.13. Alcance del trabajo

No se pretende la construcción de una nueva herramienta de detección de genes. Para tal fin sería necesario incorporar distintos tipos de fuentes de información (señales, contenidos, etc) que actualmente se manejan en forma limitada. Tampoco se pretende demostrar (directamente) como se puede

mejorar las herramientas ya construidas. La idea principal es introducir un formalismo teórico y abstracto que permita englobar los distintos métodos de detección a través mecanismos estadísticos en un marco de trabajo unificado. Se estudia y ejemplifica las distintas partes que conformarían un detector "formal" de secuencias por contenido.

Si bien nada impide la aplicación de estos mecanismos a otros ámbitos se recomienda cierta prudencia en el tratamiento de los mismos en contextos en donde no prime el factor estadístico.

2. Definiciones Básicas

2.1. Introducción

Ya hemos mencionado brevemente la idea subyacente en este trabajo, ahora proveeremos a la misma de una base formal. Varios de los métodos expuestos trabajan fuertemente con la idea de la segmentación y para eso es preciso definir alguna propiedad que permita distinguir precisamente estos segmentos o fracciones.

2.2. Alfabetos y Secuencias

Un "símbolo" es una entidad abstracta que no se definirá.

Definición 2.1. Un *alfabeto* es un conjunto finito de símbolos, normalmente notado como Σ.

Ejemplo 2.1. *El conjunto* $\Sigma = \{A, C, T, G\}$ *nos define el alfabeto de los nucleótidos.*

Definición 2.2. Una *cadena* es una secuencia finita de símbolos.

Notación 2.1. • λ *representa a la cadena vacía.*

• S^n *representa el conjunto de cadenas de longitud n, dado un alfabeto fijo.*

• S^+ *representa el conjunto de cadenas de longitud mayor o igual que 1.*

• S^* *representa el conjunto de cadenas de longitud mayor o igual que 0.*

2.3. Texturas

El principal móvil del trabajo consiste en el reconocimiento de patrones estadísticos en distintas secuencias. O sea, queremos identificar aquellas partes de una secuencia que responden en algún sentido a un comportamiento estadísticamente similar. Este concepto así definido puede resultar ambiguo y por lo tanto requiere de una formalización. En un principio usaremos definiciones muy amplias que en general pueden llegar a resultar poco prácticas para ser utilizadas directamente pero que permiten entender un poco mejor el problema desde un punto de vista teórico e ignorando posibles particularidades.

Fundamentalmente, las texturas se podrán pensar como máquinas de generación estocástica de secuencias. Una textura en su sentido más amplio es un "algo" que fabrica una secuencia dada con una cierta probabilidad. Es por eso que la siguiente definición de textura puede llegar a tener un parecido no casual con el de distribución.

Definición 2.3. Una *textura* es una función $T : S^* \rightarrow [0, 1]$ que cumple las siguientes condiciones:

- $\forall\ U \subseteq S^*$ finito vale que $\sum_{u \in U} T(u) \leq 1$.
- $\forall\ \varepsilon > 0$, $\exists\ U \subseteq S^*$ finito tal que $\sum_{u \in U} T(u) \geq 1 - \varepsilon$.

Notación 2.2. *Dado un alfabeto fijo se notará como $\mathcal{T}$ al conjunto universal de texturas.*

Hemos definido una textura de forma tal que emule el comportamiento de una distribución discreta. El segundo punto es necesario en el caso de que se consideren texturas infinitas (eso es, texturas capaces de generar infinitas secuencias). En principio, esto es completamente posible pero en general se reducirá todo al caso finito, más tratable y el único que se usa en los ejemplos y experimentos.

Definiremos ahora un concepto que nos resultará útil en el futuro y que nos dice cuáles son las secuencias que realmente importan para una textura dada. El soporte consiste en las secuencias cuya probabilidad de ser generadas por una textura es mayor que 0, eso es, se exluyen las secuencias que no pueden ser generadas por la textura.

Definición 2.4. Dada una textura T, se define el *soporte* de T como:

- $Sop(T) = \{s \in S^* / T(s) \neq 0\}$.

Si Sop(T) es un conjunto finito, diremos que T es una *textura finita*.

Lema 2.1. *Si T es una textura finita, luego $\sum_{u \in Sop(T)} T(u) = 1$.*

En definitiva, en los casos finitos la textura se convierte en una función de distribución discreta.

Se ha estado mencionando que la textura tendrá una connotación probabilística, algo que todavía no se ha terminado de relacionar con nuestra idea formal de textura. Es preciso entonces introducir el concepto de realizaciones en nuestro modelo. Si pensamos a las texturas como máquinas de generación de secuencias, las realizaciones pasarán a ser las secuencias generadas propiamente dichas.

Una función de realización consiste en una aplicación del espacio de las texturas al espacio de las secuencias. Su definición es estocástica en el sentido en que no se define directamente sino mediante probabilidades, por lo que no es una "función" en el sentido estricto.

Definición 2.5. La *función de realización* es una $\sigma : \mathcal{T} \to [0,1]$ tal que $P(\sigma(T) = s) = T(s)$.

Definición 2.6. Dada una función de realización σ, $s \in S^*$ se dice una *realización* de T (y se nota como $T \to s$) si $\sigma(T) = s$.

En conclusión, una textura nos define un esquema probabilístico de generación de secuencias. Dada una textura T, la probabilidad de que dicha textura genere una secuencia s será $T(s)$.

Debido a que varios de los métodos efectúan análisis locales utilizando ciertas ventanas de trabajo es entonces conveniente disponer de una forma para restringir la aplicabilidad de una textura a secuencias de una determinada longitud. Las *proyecciones* son las funciones encargadas de efectuar dichas restricciones.

Lo que se hace es restringir la textura original a secuencias de longitud r y luego se procede a renormalizar el resultado para que siga siendo una textura.

Definición 2.7. Una *r-proyección* es una función $\pi_r : \mathcal{T} \to \mathcal{T}$ tal que

$$\pi_r(s) = \begin{cases} 0 & \text{si} \quad |s| \neq r; \\ \frac{T(s)}{\sum_{t,|t|=r} T(t)} & \text{si} \quad |s| = r. \end{cases}$$

Es inmediato ver que $\pi_r(T)$ es una textura finita $\forall T \in \mathcal{T}$ puesto que es distinto de cero únicamente en aquellas secuencias que tienen longitur r.

En general, no trabajaremos directamente con la proyección. La idea de esta definición es que nos permite definir una familia de texturas y, en función de algún parámetro de tamaño, poder restringir a través de la proyección la familia para poder así trabajar con porciones limitadas que denominaremos "ventanas".

Definición 2.8. Dado una secuencia S, una posición n y un entero r, definimos la *ventana* centrado en n y de radio r a la subsecuencia de S comprendida entre las posiciones $n - r$ y $n + r$.

Las ventanas conforman secuencias de longitud 2r + 1, y se usarán frecuentemente para recorrer secuencias y efectuar análisis locales en las mismas. Normalmente proyectaremos las texturas mediante π_{2r+1} para analizar una ventana en particular.

Así como tenemos las texturas que son máquinas de generación de secuencias, podemos definir una clase superior encima de las texturas que denominaremos "plantillas". Nos encontraremos en numerosas situaciones en donde contaremos con un juego de parámetros que intentarán definirme una textura dada pero resulta que dichos parámetros bien pueden servir para definir otra textura pero que fabrique secuencias de distinta longitud.

La idea de la plantilla será la de abstraer en su interior todos estos parámetros y poder, pasándole un natural, armar una textura capaz de generar secuencias de una longitud fija.

Definición 2.9. Una *plantilla* sobre un alfabeto Σ de n símbolos es una función $M : \mathbb{N} \to \mathcal{T}$ tal que:

- $M(r)(s) = 0$ *si* $|s| \neq r$
- $\sum_{s \in \Sigma^r} M(r)(s) = 1$

Todo esto se puede resumir diciendo que $\pi_r(M(r)) = M(r)$. Se suele notar $M(r)$ como M_r.

Luego, una plantilla evaluada en un natural me devuelve una textura que únicamente genera secuencias cuya longitud es igual al natural

Como hecho anecdótico podemos decir que es posible "emular" a las plantillas mediante el uso de texturas y proyecciones.

Teorema 2.1. *Dada una plantilla M se puede definir una textura T tal que $\pi_r(T) = M_r$.*

Demostración. Supongamos M una plantilla.

Definimos T de la siguiente forma:

- $T(\alpha_1) = \frac{M_1(\alpha_1)}{2}$, lo que implica $\sum_{\alpha_1 \in \Sigma} T(\alpha_1) = \frac{1}{2}$
- $T(\alpha_1\alpha_2) = \frac{M_2(\alpha_1\alpha_2)}{4}$, lo que implica $\sum_{\alpha_1\alpha_2 \in \Sigma^2} T(\alpha_1\alpha_2) = \frac{1}{4}$

y en general,

- $T(\alpha_1 \dots \alpha_r) = \frac{M_r(\alpha_1\alpha_2\dots\alpha_r)\dots}{2^r}$, lo que implica $\sum_{\alpha_1\alpha_r \in \Sigma^r} T(\alpha_1 \dots \alpha_r) = \frac{1}{2^r}$

Veamos primero que T así definida conforma una textura. Todos los valores son positivos y además vale lo siguiente:

$$lim_{r->\infty} \sum_{1\leq k\leq r} \sum_{\alpha_1\dots\alpha_k \in \Sigma^k} T(\alpha_1 \dots \alpha_k) = lim_{r->\infty} \sum_{1\leq k\leq r} \frac{1}{2^r} = 1.$$

Como esta sucesión es creciente, tenemos entonces que nunca se podrá superar 1, cumpliendo así el primer requerimiento para ser una textura. Pero como además la sucesión tiende a 1, podemos acercarnos a este arbitrariamente y de esa manera se puede siempre escoger un subconjunto finito de cadenas tales que la suma de la probabilidad de las mismas se acerque a 1 tanto como se quiera. Por lo tanto, T es una textura.

Por último, veamos que $\pi_r(T) = M_r$. Sea s una secuencia. Luego, por definición de proyección,

$$\pi_r(s) = \begin{cases} 0 & \text{si} \quad |s| \neq r; \\ \frac{T(s)}{\sum_{t,|t|=r} T(t)} & \text{si} \quad |s| = r. \end{cases}$$

Luego, si s no tiene longitud r su evaluación en T devuelve 0, coincidiendo con el de M_r. Si su longitud es r, nos sale que

$$\pi_r(s) = \frac{T(s)}{\sum_{t,|t|=r} T(t)}$$

y por definición de T esto nos queda

$$= \frac{\frac{M_r(s)}{2^r}}{\sum_{t,|t|=r} T(t)} = \frac{\frac{M_r(s)}{2^r}}{\frac{1}{2^r}} = M_r(s)$$

y tenemos así lo que queríamos probar.

□

Este resultado nos permite establecer una equivalencia entre texturas y plantillas. Tal es así que en cierto sentido la definición de plantillas se volvería innecesaria. Sin embargo, se prefirió definir la misma puesto que en cierto aspecto las definiciones posteriores son más fáciles de ser expresados mediante las plantillas antes que directamente sobre texturas.

2.4. Familias de Texturas

2.4.1. Procesos temporales de Markov

Vamos a definir una serie de familias de texturas que serán fuertemente utilizados en lo que resta del trabajo.

La primer definición tiene su análogo con los procesos Markovianos, en donde se la variable aleatoria intenta emular un proceso temporal en el cual los eventos presentes dependen únicamente del pasado reciente.

Tal analogía la podemos llevar en el estudio de secuencias, si uno asume que existe una especie de dependencia "temporal" entre los símbolos de una secuencia. La primer posición de la secuencia define el evento inicial y el resto se van obteniendo en función de su (o sus) posición (o posiciones) anterior(es).

Definición 2.10. Sea $n \in \mathbb{N}$. Decimos que una plantilla M es *Markov-n* temporal si se cumple la siguiente condición:

• $\forall\ a_1, \ldots, a_n\ \in \Sigma$, existen únicos $t_{a_1 \ldots a_{n-1} -> a_n}$ positivos tales que $\forall\ s \in S^*, M_r(sa_1 \ldots a_n)\ = M_{r-1}(sa_1 \ldots a_{n-1})\, t_{a_1 \ldots\ a_{n-1} -> a_n}$ con $r = |\ sa_1 \ldots a_n\ |$. Además, vale que $\forall\ a_1, \ldots, a_{n-1}\ \in \Sigma,\ \sum_{a_n \in \Sigma} t_{a_1 \ldots\ a_{n-1} -> a_n} = 1$.

En el caso n = 2, diremos que la plantilla es una *cadena simple de Markov*.

Los valores $t_{\alpha -> c}$ se denominan las probabilidades de transición y representan la probabilidad de que dado una cadena α el próximo caracter sea una c.

Para definir una plantilla Markov-1 basta definir la probabilidad de que ocurra cada elemento del alfabeto.

Esto se deduce del hecho de que, si $s = a_1 a_2 \ldots a_n$, luego $M_n(s) = M_n(a_1 \ldots a_n) = M_1(a_1) \ldots M_1(a_n)$.

La definición de una textura Markov-2 (o sea, una cadena simple de Markov) requiere lo siguiente:

- Saber la probabilidad inicial del primer elemento.
- Para todo par a,b de elementos del alfabeto, saber cual es la probabilidad de que ocurra a sabiendo que antes ocurrió b.

En efecto, se deduce que $M_n(a_1 \ldots a_n) = M_1(a_1) \prod_{i=1}^{n-1} t_{a_{i-1} a_i}$.

Podemos entonces representar la plantilla de una cadena simple de Markov de la siguiente forma (suponiendo el alfabeto genético).

El primer vector nos demarca las probabilidades iniciales para el primer símbolo de la secuencia.

Símbolo	Probabilidad
A	π_a
C	π_c
T	π_t
G	π_g

Esta matriz nos representa las probabilidades de transición. t_ab sería la probabilidad de obtener un b dado que el símbolo anterior es una a.

–	A	C	T	G
A	t_{aa}	t_{ac}	t_{at}	t_{ag}
C	t_{ca}	t_{cc}	t_{ct}	t_{cg}
T	t_{ta}	t_{tc}	t_{tt}	t_{tg}
G	t_{ga}	t_{gc}	t_{gt}	t_{gg}

Debido a la cadencia unidireccional del proceso, a estas plantillas se las denominará plantillas Markov temporales o Markov-t, para diferenciarlas de las espaciales que pasaremos a comentar más adelante. Notemos que para poder definir una plantilla de cadena simple de Markov fueron necesarios $|\Sigma|$ (4 en el caso de ADN) parámetros para las probabilidades iniciales más $|\Sigma^2|$ (16 en el caso de ADN) parámetros para las transiciones. Es fácil ver que, en general, para una cadena Markov-n se precisarán $|\Sigma|^n$ (4^n para el ADN) parámetros para el comienzo y $|\Sigma|^{n+1}$ (4^{n+1} para el ADN) parámetros para las transiciones. El crecimiento exponencial de esto representa la primer dificultad importante contra el uso de cadenas de Markov de orden indiscriminado.

2.4.2. Procesos espaciales de Markov

El objetivo de los procesos espaciales es el de generalizar el concepto de las cadenas de Markov. Una forma simple de generalizar un proceso Markoviano de un orden fijo es incrementando precisamente ese orden. A continuación mostraremos otra posible generalización que tenga en cuenta los dos posibles sentidos en los que se puede leer una secuencia.

Para poder lidiar con la cadencia unidireccional naturalmente impuesta por la cadena de Markov usaremos un concepto similar al de los campos

Markovianos, ampliamente utilizados en procesamiento digital de imágenes. Lo que haremos es establecer dependencias entre un nodo y todos sus vecinos, ya no nos restringiremos a los vecinos de un único lado.

Antes de exhibir la definición formal que utilizaremos es prudente aclarar que, como se menciónó en el párrafo anterior, el concepto a ser utilziado es similar pero NO es igual al de los campos Markovianos. La diferencia radica fundamentalmente en que en la teoría de campos Markovianos lo normal es definir las probabilidades condicionales y a partir de ella aproximar la probabilidad conjunta. El camino aquí será el inverso, definiremos la probabilidad conjunta y a partir de aquella se deducirá una expresión simple para la probabilidad condicional que facilite la estimación. O sea, dado una secuencia daremos una fórmula explícita para el cálculo de la probabilidad de ser generada por la secuencia.

Definición 2.11. Sea $n \in \mathbb{N}$. Decimos que una plantilla M es *Markov-n* espacial si se cumple la siguiente condición:

• $\forall\ a_1, \ldots, a_n\ \forall\ b_1, \ldots, b_n\ c \in \Sigma$, existen únicos $t_{a_1 \ldots a_n, b_1 \ldots b_n -> c}$ tal que $\forall\ s, t \in S^*, M_r(sa_1 \ldots a_n\ c\ b_1 \ldots b_n t)$

$= M_{|\ sa_1 \ldots a_{n-1}\ |}(sa_1 \ldots a_{n-1}) t_{a_1 \ldots a_n, b_1 \ldots b_n -> c} M_{|\ b_2 \ldots b_n\ |}(b_2 \ldots b_n t)$.

con $r = |sa_1 \ldots a_n c b_1 \ldots b_n|$.

Además, vale que $\forall\ a_1, \ldots, a_n, b_1, \ldots, b_n\ \in \Sigma,\ \sum_{c \in \Sigma} t_{a_1 \ldots\ a_n, b_1 \ldots\ b_n -> c} = 1$.

Notemos que en el caso n=1, tendremos que:

$M_r(a_1 \ldots a_n) = i_{a_1} t_{a_2, a_4 -> a_3} \ldots t_{a_{n-2}, a_n -> a_{n-1}} d_{a_n}$ siendo i y d dos vectores estocásticos que definen la probabilidad izquierda y derecha de un elemento del borde.

El proceso espacial de Markov de primer orden involucra para cada sitio interno sus dos vecinos. En la cadena simple únicamente participaba uno de los vecinos. Esto implica un aumento en el nivel expresivo de la textura pero con la contrapartida de involucrar una mayor cantidad de parámetros y por ende una mayor complejidad. Para definir las transiciones se precisarán $|\Sigma|^{2n+1}$, en comparación con los $|\Sigma|^{n+1}$ parámetros necesarios para la cadena.

La cuestión radica en que el verdadero competidor de un proceso espacial de orden n no será un proceso temporal del mismo orden sino uno de orden $2n$. de esa forma involucraría la misma cantidad de elementos a la hora de determinar un sitio.

Interesará comparar la performance de ambos mecanismos, especialmente teniendo en cuenta que las cadenas de Markov ya han sido comentadas dentro

de la rama de la predicción computacional de genes.

2.5. Características de una textura

Hay dos aspectos que interesaran fuertemente de una textura. Por un lado, es la posibilidad de poder calcular fácilmente la probabilidad de una secuencia dada. Otro aspecto tiene que ver con la factibilidad de poder obtener realizaciones de forma sencilla. Eso es, queremos ver si para la textura dada existe un procedimiento algorítmico "simple" que simule lo mejor posible el comportamiento de una función de realización. O sea, un procedimiento que dado una secuencia s la genere con una probabilidad de $T(s)$. En tercer lugar vamos a querer ver que tan fácil es poder deducir texturas en función de un conjunto de secuencias del cual se sabe, ya a priori, a qué textura pertenecen.

En general, con las texturas provenientes de plantillas Markov-t todos estos puntos son relativamente simples de resolver.

El primer punto ya lo habíamos resuelto antes, prácticamente se desprende directamente de la definición.

El segundo punto también es simple. Ilustraremos el caso de una cadena simple de Markov. Tenemos las probabilidades iniciales de cada elemento, luego obtenemos el elemento inicial en función de dicha distribución. El cálculo de los próximos elementos se hace obteniendo una distribución con los factores de transición y obtener una realización para lo que sería el próximo elemento.

El tercer punto también es fácil de resolver puesto que los factores de transición se pueden asociar rápidamente con probabilidades condicionales, tal como muestra el siguiente resultado:

Teorema 2.2. *Dado M una plantilla cadena simple de Markov y $X_1, \ldots, X_r$ un conjunto de variables aleatorias en donde X_i denota el símbolo de la posición i-ésima de una secuencia, tenemos que:*

$P(X_i = a | X_{i-1} = b) = t_{b->a}$

Demostración. Por definición de probabilidad condicional, vale que:

$$P(X_i = a | X_{i-1} = b) = \frac{P(X_i=a, X_{i-1}=b)}{P(X_{i-1}=b)}$$

Por el principio de partición, tenemos que:

$P(X_i = a, X_{i-1} = b) = \sum_{y_1,\ldots,y_{i-2}\in\Sigma} P(X_1 = y_1, \ldots, X_{i-2} = y_{i-2}, X_{i-1} = b, X_i = a)$.

Por ser una cadena simple de Markov, vale entonces que esto último es

$\sum_{y_1,\ldots,y_{i-2}\in\Sigma} i_{y_1} t_{y_1->y_2} \ldots t_{y_2->b} t_{b->a}$.

De la misma manera, se obtiene lo siguiente:

$P(X_{i-1} = b) = \sum_{y_1,\ldots,y_{i-2}\in\Sigma} i_{y_1} t_{y_1->y_2} \ldots t_{y_2->b}$.

Entonces, sacando del numerador factor común $t_{b->a}$ y cancelando el resto nos queda:

$P(X_i = a|X_{i-1} = b) = t_{b->a}$

□

Luego, para obtener las transiciones basta utilizar un método que permita estimar las probabilidades condicionales. Un mecanismo posible sería un análisis frecuentista sobre los pares de símbolos que surjan en la secuencia.

Por desgracia, esta propiedad que permite una estimación relativamente simple de los parámetros que caracterizan a una cadena de Markov se pierde en los procesos espaciales. En efecto, la dificultad del problema deviene del hecho en que la relación entre la probabilidad condicional y la probabilidad conjunta no es fácil de establecer en este tipo de distribuciones. En la teoría de los campos espaciales de Markov (Markov Random Fields) usualmente se toma como punto de partida la probabilidad condicional en la definición del campo. En el presente trabajo lo hemos hecho a partir de una probabilidad total, con la contrapartida de perder una caracterización elegante de una probabilidad condicional. Recomendamos [KIN80] para profundizar sobre el tema.

En ese sentido, no es posible establecer una relación directa entre $P(X_i = a|X_{i-1} = b, X_{i+1} = c)$ y los valores de transición que caracterizan a la textura. A pesar de eso se utilizará estos valores de transición como una primera aproximación de la probabilidad condicional. Los resultados prácticos en general tienden a validar esta suposición simplificadora.

2.6. Ejemplos

El siguiente ejemplo está extraido de [DUR98]. Este ejemplo será usado con bastante frecuencia en el resto de este trabajo, siempre con diferentes motivos.

Un problema típico en el campo de la bioinformática es el reconocimiento de las islas CpG. Podemos asumir por un tiempo que es posible distinguir estas regiones por el hecho de que responden a un patrón estadístico que los distingue de otra región. Para simplificar aún más el problema, asumimos que ese comportamiento está dictado por un patrón markiviano simple. Eso es, cada región responde a una textura definida por una cadena de Markov.

Mostraremos a continuación las matrices que nos caracterizarán las transiciones de cada una de las texturas en juego.

Textura CpG positiva: demarca una región contenida dentro de una isla CpG.

+	A	C	T	G
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
T	0.161	0.339	0.375	0.125
G	0.079	0.355	0.384	0.182

Textura CpG negativa: demarca una región contenida fuera de una isla CpG.

–	A	C	T	G
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
T	0.248	0.246	0.298	0.208
G	0.177	0.239	0.292	0.292

Cabe destacar los siguientes hechos:

- En la textura positiva las transiciones C a G son más probables que en la negativa
- Las transiciones a C son más probables en la textura positiva.

Con estas texturas ya podemos generar realizaciones, basta seguir la regla enunciada antes para procesos temporales. La cuestión de cómo determinar

la textura que mejor se acomoda a una secuencia se tratará en lo que viene a continuación.

3. Reconocimiento y Competitividad

3.1. Introducción

Ya hemos definido nuestro modelo teórico, que como parte principal introduce la idea de textura y sus derivados. Hemos, además, definido una serie de familias de texturas que se utilizarán durante este trabajo.

Básicamente, lo que se tiene ahoa es una forma de obtener secuencias ("realizaciones") en función de una textura, lo interesante (y lo que en última instancia es lo que nuclea este trabajo) es el proceso inverso, dado una secuencia poder determinar cual fue la textura que lo generó.

Vamos a introducir algunas propiedades que permiten la comparación de distintas texturas con los fines de estudiar su "distinguibilidad". Vamos a querer en general que las texturas que se usen sean lo suficientemente distintas como para facilitar la tarea de distinguirlas en una secuencia.

3.2. Máxima Verosimilitud

Supongamos el siguiente problema: disponemos de varios modelos $\theta_1 \dots \theta_n$ y un evento s. Nos interesa deducir el modelo que mejor explica al evento s.

El estimador de máxima verosimilitud es una manera para poder responder a esta pregunta, se basa fundamentalmente en determinar el modelo que tiene la mayor probabilidad de generar el evento s. Más formalmente, definimos la función de *verosimilitud* $L : Modelos \rightarrow [0, 1]$ de la siguiente forma $L(\theta) = P(s \mid \theta)$.

Se trata entonces buscar el modelo θ que maximiza la función de verosimilitud. Definimos así el estimador de máxima verosimilitud, y se notará $\theta^M V$ ("máxima versosimilitud"), como $\theta^{MV} = argmax_\theta L(\theta) = argmax_\theta P(s|\theta)$.

Un ejemplo sencillo nos permitirá ilustrar el funcionamiento del método.

Consideremos la siguiente secuencia: AA, las texturas que se disputan la misma son:

Sec.	T_1	T_2
AA	0.25	1.0
AB	0.25	0.0
BA	0.25	0.0
BB	0.25	0.0

Obtengamos el MV, para eso es preciso evaluar $L(T_1)$ y $L(T_2)$. $L(T_1) = 0{,}25$ y $L(T_2) = 1{,}0$, luego T_2 es el modelo que maximiza la verosimilitud y por lo tanto $\theta^M V = T_2$. Eso significa que dado la secuencia AA, lo más probable es que la misma haya sido generada por la textura T_2.

3.3. Primeras extensiones

Aplicar MV sobre una única secuencia puede a veces dar resultados no del todo satisfactorios. Bien puede ocurrir que la secuencia ganadora no mantenga un nivel de consenso muy alto en comparación con el resto.

Una manera de corregir estos problemas es utilizando más de una secuencia. Eso es, tenemos varias secuencias $(s_1, \ldots, s_n)$ y sabemos que todas ellas fueron generadas con una misma textua. Nuestra función de verosimilitud pasa a ser:

$$L(\theta) = P(s_1, \ldots, s_n \mid \theta) \tag{1}$$

Si asumimos independencia entre las secuencias tenemos:

$$L(\theta) = P(s_1, \ldots, s_n \mid \theta) = P(s_1 \mid \theta) \ldots P(s_n \mid \theta) \tag{2}$$

Se busca entonces la textura que maximice esta función.

Esencialmente, el método es igual al anterior, tenemos un conjunto de datos (*data set*) y buscamos el modelo que mejor lo satisfaga.

El hecho de disponer de varias secuencias que hayan sido generadas con la misma textura puede resultar algo fuerte en un principio. Normalmente, se dispone de una secuencia que se desea segmentar. Este caso se podría presentar si disponemos de secuencias que cuentan con algún nivel de homología, probablemente pertenecientes a una misma familia. En ese caso se podría llegar a volver más aplicable el método.

En general, la estimación de máxima verosimilitud es óptima cuando el conjunto de datos es grande, en ese caso la textura MV tenderá a la textura verdadera.

Sin embargo, este método cuenta con una desventaja: no contempla la incorporación de información *a priori* sobre los modelos en su formulación. Por ejemplo, si se tiene fuerte evidencia de la rareza de una textura sería entonces bueno penalizar de alguna manera a dicha textura. La cuestión de la introducción de la información *a priori* se trata en el capítulo 6 mediante el uso de estadística Bayesiana.

Otro problema que tiene es que no es capaz de distinguir semejanzas entre los modelos. O sea, si contamos con varios modelos muy similares entre sí y que satisfacen el dataset en algún sentido pero además tenemos un modelo aislado que representa mejor a los datos, luego el método nos determinará como ganadora a este último modelo. ¿Hasta que punto eso es un buen resultado?

Ejemplifiquemos el problema que esto planteando con un ejemplo "electoral". Supongamos una que se hace una elección entre 6 candidatos, 5 de ellos de tendencias propias de la izquierda y el restante un claro defensor de la derecha ortodoxa. Los cinco candidatos de izquierda obtienen cada uno un 15 % de los votos, totalizando en su conjunto el 75 %, el candidato restante obtiene el 25 %. En una elección sin instancias posteriores ganaría el candidato de derecha, pese a ser diametralmente opuesto a sus opositores, que en conjunto lo podrían derrotar ampliamante.

El modelo MV no permite resolver estas cuestiones, la semenjanza entre modelos es algo que no se puede expresar dentro del mismo.

3.4. Competitividad

Vamos a estudiar ahora otra cuestión importante y que consistire, como veremos cuando expliquemos los métodos, uno de los principales obstáculos de los métodos de estimación.

Varias preguntas que nos podemos hacer: ¿Es justa la estimación de máxima verosimilitud? ¿Todas las texturas se encuentran representadas por igual? La respuesta a estas preguntas por desgracia es negativa, el modelo MV no es justo y favorece a algunos modelos por encima de otros, eso suponiendo completo desconocimiento de los datos de estimación.

Nos interesa diferenciar en particular los casos de deducciones erróneas, o sea predecir que una secuencia fue generada con tal textura cuando en realidad había sido generada con tal otra textura. La segudna textura está, en algún sentido, "agrediendo" a la primera, se apropia de elementos que no le corresponden o, en todo caso, que se suponía pertencen a la primera.

Eso puede ocurrir con algunas texturas, pero bien podría ser que ocurra con muchas texturas (no con todas, eso ya lo resolveremos más adelante). Cuanto mayor sea esto, más grande será la intromisión de la otra textura y por lo tanto peor será la estimación. Antes de intentar resolver este problema vale la pena tratar de cuantificarlo para estudiar la magnitud del mismo.

Dado dos texturas, T_1 y T_2, y la secuencia s diremos que s es una secuencia "mala" para T1 si $P(s \mid T_2) \geq P(s \mid T_1)$. Si T_1 es capaz de generar muchas secuencias "malas", eso indicará que buena parte de sus realizaciones se detectarán como pertenecientes a la textura T_2.

El grado de interferencia o intromisión se puede medir utilizando una función que definiremos a continuación.

Definición 3.1. Sean T_1 y T_2 texturas, siendo además T_2 una textura finita. Luego, se define la "intromisión" de T_1 a T_2 como una función $I : \mathcal{T}x\mathcal{T} \rightarrow [0, 1]$ dada por:

$I(T_1, T_2) = \sum_{s \in\ Sop(T_2)} P(s\ |T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) > P(s \mid T_2) \\ 0 & \text{si no} \end{cases}$

Se lo notará además como $I(T_1\ en\ T_2)$.

Esta suma pesada por la probabilidad sobre las secuencias "malas" generadas por T_1 nos cuantifica esa magnitud que buscábamos. ¿Cómo es que obtenemos esta fórmula? Es posible inferirla de la siguiente forma:

Queremos saber cuál es la probabilidad de que una secuencia generada con T_2 sea reconocida como proveniente de T_1. Eso es, buscamos:

I(T_1 en T_2) = P(una secuencia S generada con T_2 se reconozca como de T_1)

(teorema de partición)

$\Rightarrow$) I(T_1 en T_2) = P(S sea reconocido como de $T_1 \ \cap S = s_1$ en T_2) + ... + P(S sea reconocido como de $T_1 \ \cap S = s_n$)

$\Rightarrow$) I(T_1 en T_2) = $\sum_{s \in Sop(T_2)}$ P(S sea reconocido como de $T_1 \ \cap S = s$ en T_2)

$\Rightarrow$) I(T_1 en T_2) = $\sum_{s \in Sop(T_2)}$P(S sea reconocido como de $T_1|S = s$)P(S=s en T_2)

$\Rightarrow$) I(T_1 en T_2) = $\sum_{s \in Sop(T_2)}$P(s sea reconocido como de T_1)P(S=s en T_2)

(P(s sea reconocido como de T_1) vale 1.0 si P(s $|T_1$) > P(s $|T_2$))

$\Rightarrow$) $I(T_1, T_2) = \sum_{s \in\ Sop(T_2)} P(s\ |T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) > P(s \mid T_2) \\ 0 & \text{si no} \end{cases}$

Notemos que se está haciendo una suposición en este enunciado: dada una secuencia y dos texturas que la generan con la misma probabilidad, se asume que la secuencia que la generó es la "ganadora" en la selección por máxima verosimilitud, a pesar de que las probabilidades coinciden. Eso

se da por el hecho de utilizar el $>$ en vez de $\geq$ en la definición, dándonos una definición que denominaremos "optimista". Un acercamiento "pesimista" podría establecer que cuando las probabilidades coinciden, nos quedamos con la otra secuencia, eso es, utilizamos $\geq$ en vez de $>$.

Notaremos a la primera definición como $I_{opt}(S\ en\ T)$ y a la segunda como $I_{pes}(S\ en\ T)$. Se puede ver que $I_{opt}(S\ en\ T) \leq I_{pes}(S\ en\ T)$. Eso es, la definición pesimista tenderá a darnos mayores valores de intromisión.

La definición pesimista puede llegar a resultar a ser excesivamente pesimista, habrá casos en donde es mejor inclinarse por los resultados teóricos arrojados por la definición optimista. En particular, cuando las texturas en cuestión nunca generan una misma secuencia con la misma probabilidad no existen conflictos de decisión y se puede entonces asumir el comportamiento "bueno" del sistema.

Pero la definición optimista adolece de una falla tal vez algo más radical: tiene el efecto algo desagradable de que $I(T\ en\ T) = 0$, indicando que entre dos texturas iguales no hay interferencia, un resultado algo anómalo. Más adelante definiremos una noción de distancias entre texturas dada por una intromisión mutua y no nos vamos a poder permitir este problema.

Por esta cuestión hemos decidido que, a menos que se aclare expresamente lo contrario, la definición a utilizar será la pesimista, nos inclinamos así por una opción más conservadora.

Ejemplo 3.1. *Dadas las siguientes texturas:*

T_1	–
AA	*0.25*
AB	*0.25*
BA	*0.25*
BB	*0.25*

T_2	
AA	*1.0*

Vale que $I(T_1\ en\ T_2) = 0$. *O sea, cualquier secuencia generada por* T_2 *será siempre clasificada como proveniente de* T_2. *Sin embargo,* $I(T_1\ en\ T_2) =$ 0,25*, lo que indica que existe un 25como proveniente de* T_2. *Marca esto una clara debilidad de la primer textura respecto de la segunda.*

Definición 3.2. Dado T_1 y T_2 texturas, se dice que T_1 es *impermeable* respecto de T_2 si $I(T_2\ en\ T_1) = 0$.

En el ejemplo anterior, la textura constante era impermeable respecto de la uniforme.

Definición 3.3. Dado T una textura, T se dice *inconfundible* si $\forall$ M textura distinto de T vale que $I(M\ en\ T) = 0$.

Las texturas inconfundibles nunca podrían ser incorrectamente clasificadas, aunque esto no implica que la textura no se entrometa en regiones de otras texturas. Si tuviésemos todas texturas inconfundibles la tarea se solucionaría bastante pero por desgracia son muy limitadas. De hecho, es fácil caracterizarlas por completo tal como se enuncia a continuación.

Lema 3.1. *T textura es inconfundible* $\Leftrightarrow |Sop(T)| = 1$.

Demostración $\Rightarrow$) . Sea T una textura inconfundible, supongamos que $|Sop(T)| > 1$.

$\Rightarrow$ Existen $s_0, t_0 \in Sop(T)$, por lo tanto $T(s_0)$ y $T(t_0)$ son ambos mayores que 0.

$\Rightarrow$ Existe $s_0 \in Sop(T)$ tal que $T(s_0) = r < 1$.

Definimos una textura M constante que vale 1 en s_0 y 0 en el resto.

Vale entonces que I(M en T) $= P(s_0|T) * 1 = r > 0$.

Luego, T no es inconfundible, absurdo.

□

Demostración $\Leftarrow$) . Supongamos $|Sop(T)| = 1$, luego $\forall s \in Sop(T)$ vale que $T(s) = P(s|T) = 1$, nadie lo puede superar pues 1 es el máximo posible (excepto la textura misma).

□

Esto es, las únicas texturas inconfundibles son las constantes.

Es un caso extremadamente limitado y muy trivial como para servirnos. La única forma de que no haya "interferencias" en un conjunto de texturas es que cada una de ellas genere secuencias distintas. Esos casos son muy simples, no requieren ningún tipo de estimación, basta con elegir la secuencia adecuada en función de la textura.

Por lo tanto, la intromisión resultará ser un mal necesario. No sería entonces mala idea ir previende el nivel de intromisión antes de efectuar ninguna estimación.

Cabe aclarar que este tipo de fenómeno es especialmente característico con texturas reducidas. Eso es, texturas que produzcan pocas secuencias. Se espera que introduciendo variedad este efecto se vea reducido.

3.5. Independencia

Hasta ahora, tenemos definido una medida entre texturas que nos permite determinar el grado de interferencia entre las mismas. El problema es su eventual asimetría, si queremos definir algo más cercano a una distancia este es un elemento que no se puede ignorar.

Definición 3.4. Se define el *nivel de dependencia*, $dep : \mathcal{T}x\mathcal{T} \rightarrow \mathbb{R}$ como $dep(T, M) = (I(T\ en\ M) + I(M\ en\ T))/2$. Cuando $dep(T, M) = 0$, diremos que T y M son *independientes*.

O sea, dos texturas se dirán independientes si ambas son impermeables entre sí. Cuando estamos ante un conjunto de texturas que son independientes dos a dos, el proceso de reconocimiento por máxima verosimilitud será perfecto. De alguna manera, estos serían los casos ideales.

Pero por desgracia, las texturas independientes, como ocurría con el caso de las inconfundibles, admiten una caracterización bastante trivial y, por ende, poco práctico.

Como en el caso de la intromisión, podemos definir una dependencia pesimista y otra optimista, dependiendo de la función de intromisión que se utilice para el cálculo de la dependencia.

Lema 3.2 (pesimista). *Dados dos texturas, T_1 y T_2, vale que: T_1 y T_2 son NO independientes $\Leftrightarrow$ $\exists s \in S^*/T_1(s) \neq 0, T_s(s) \neq 0, T_1(s) \geq T_2(s)$ o bien $T_2(s) \geq T_1(s)$.*

Demostración $\Rightarrow$) . $dep(T_1, T_2) = r > 0$, luego vale que $I(T_1\ en\ T_2) > 0$ o bien $I(T_2\ en\ T_1) > 0$.

Supongamos el primer caso, luego vale que en la sumatoria que define la I existe algún término no nulo. Pero eso únicamente puede ocurrir cuando existe un s en donde valga que $T_1(s) \geq T_2(s)$, y llegamos a lo deseado.

□

Demostración $\Leftarrow$) . Si existe s tal que $T_1(s) \geq T_2(s)$ y en donde $T_2(s) \neq 0$, luego $I(T_1\ en\ T_2) > 0$ y se obtendrá lo desado. □

La versión optimista es similar a la anterior, pero cambiando el $\geq$ por un $>$.

Lema 3.3 (optimista). *Dados dos texturas, T_1 y T_2, vale que: T_1 y T_2 son NO independientes $\Leftrightarrow$ $\exists s \in S^*/T_1(s) \neq 0, T_s(s) \neq 0, T_1(S) > T_2(s)$ o bien $T_2(S) > T_1(s)$.*

En definitiva, para que dos texturas sean independientes será prácticamente necesario que los conjuntos de secuencias generadas por ambas sean mutuamente excluyentes, siendo esto en general un requisito muy fuerte. La independencia será un ideal para la mayor parte de los casos prácticos, habrá que conformarse con algún nivel de dependencia entre las texturas. Ahora bien, podemos plantearnos varias preguntas:

- ¿Cuánta dependencia podemos esperar de un par de texturas?
- ¿Cómo puedo definir texturas útiles que reduzcan la dependencia?

El siguiente resultado arrojará ideas para intentar resolver la primer pregunta, aunque sea parcialmente. Además, se destaca aquí la diferencia sustancial que se obtiene en la utilización de cada tipo de dependencia/intromisión.

Proposición 3.1. *Dados T_1 y T_2 dos texturas:*

- $dep_{opt}(T_1, T_2) < 0{,}5$
- $dep_{pes}(T_1, T_2) \leq 1$

Caso optimista. $2dep(T_1, T_2) = I(T_1 \text{ en } T_2) + I(T_2 \text{ en } T_1) =$

$$= \sum_{s \in \ Sop(T_2)} P(s \mid T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) > P(s \mid T_2) \\ 0 & \text{si no} \end{cases} +$$

$$+ \sum_{s \in \ Sop(T_1)} P(s \mid T_1) \begin{cases} 1 & \text{si} \quad P(s \mid T_2) > P(s \mid T_1) \\ 0 & \text{si no} \end{cases} =$$

$$= \sum_{s \in \ Sop(T_2) \cup Sop(T_1)} P(s \mid T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) > P(s \mid T_2) \\ 0 & \text{si no} \end{cases} +$$

$$+ P(s \mid T_1) \begin{cases} 1 & \text{si} \quad P(s \mid T_2) > P(s \mid T_1) \\ 0 & \text{si no} \end{cases} =$$

$$= \sum_{s \in \ Sop(T_2) \cup Sop(T_1) / P(s|T_1) \neq P(s|T_2)} min \{P(s \mid T_1), P(s \mid T_2)\}$$

$$\leq \sum_{s \in \ sop(T_1)} P(s \mid T_1) \ = 1$$

Ahora bien, ¿puede valer la igualdad? Si fuesen iguales, eso significa que el mínimo entre ambos valores coincide siempre con la probabilidad de que lo genere T_1. Luego, tendríamos los siguientes casos:

- no hay elementos que cumplan $P(s|T_1) \neq P(s|T_2)$, luego ambas texturas son iguales. Pero entonces la dependencia daría 0 pues nunca se daría el caso de $P(s|T_1) > P(s|T_2)$
- existen elementos en donde $P(s|T_1) > P(s|T_2)$ y en el resto son iguales. Pero entonces,la textura T_1 tendrá siempre mayores pesos que T_2, nunca podría sumar 1 pues es T_2 lo hace y T_1 es siempre mayor que T_2. Saldría que T_1 no es textura.

En definitiva, obtenemos que el $\leq$ es en realidad un $<$ y por ende vale que $2dep_{opt}(T_1, T_2) < 1$, luego $dep_{opt}(T_1, T_2) < 0{,}5$.

□

Caso pesimista. $2dep(T_1, T_2) = I(T_1 \ en \ T_2) + I(T_2 \ en \ T_1) =$
$= \sum_{s \in \ Sop(T_2)} P(s \ |T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) \leq P(s \mid T_2); \\ 0 & \text{si} \quad no \end{cases} +$
$+ \sum_{s \in \ Sop(T_1)} P(s \ |T_1) \begin{cases} 1 & \text{si} \quad P(s \mid T_2) \leq P(s \mid T_1); \\ 0 & \text{si} \quad no \end{cases} =$
$= \sum_{s \in \ Sop(T_2) \cup Sop(T_1)} P(s \ |T_2) \begin{cases} 1 & \text{si} \quad P(s \mid T_1) \leq P(s \mid T_2); \\ 0 & \text{si} \quad no \end{cases} +$
$+ P(s \ |T_1) \begin{cases} 1 & \text{si} \quad P(s \mid T_2) \leq P(s \mid T_1); \\ 0 & \text{si} \quad no \end{cases} =$
$\leq \sum_{s \in \ Sop(T_1) \cup s \in \ Sop(T_2)} P(s \mid T_1) + P(s \mid T_2) \ =$
$= \sum_{s \in \ Sop(T_1)} P(s \mid T_1) \ + \sum_{s \in \ Sop(T_2)} P(s \mid T_2) = 1 + 1 = 2.$
La igualdad ahora sí puede valer, si ambas texturas son iguales la dependencia pesisimista dará 1. .

□

Notemos otro significado importante de la dependencia: me mide el nivel general de error de la estimación por MV. Supongamos dos texturas y una serie de secuencias generadas cada una por alguna de las dos texturas con igual probabilidad. Luego, la dependencia nos estará dando una estmimación de la proporción de secuencias que serán incorrectamente clasificadas.

3.6. Cálculo de la dependencia

Ya hemos definido el concepto de intromisión, dependencia y se establecieron algunas cotas para los mismos. Ahora bien, para poder estar seguros sobre la resistencia del método frente a una situación en particular es preciso

saber hasta cuando puede llegar a valer esta dependencia. Valores muy altos deberían indicarnos que el método puede llegar a efectuar una cantidad numerosa de predicciones incorrectas.

El problema radica fundamentalmente en el hecho de que las definiciones no proveen un método eficiente o rápido para calcular estos valores. Si suponemos texturas finitas que genera secuencias de hasta longitud n con un alfabeto de r símbolos, potencialmente podríamos tener r^n secuencias distintas que considerar. A veces nuestra textura podría estar definido en forma tabular, esto es, dada cada secuencia que genera sabemos su probabilidad. Sin embargo, no hay que esperar que esto ocurra con frecuencia. Lo que en general tendremos son fórmulas que permiten calcular la probabilidad en función de la secuencia, y habrá incluso casos en donde ni siquiera se contará con esto.

En definita, a falta de un buen método analítico nos debemos conformar con aproximaciones numéricas basadas en muestreo de datos. Eso es, dado una textura T generamos una cantidad importante de secuencias con esa textura y luego se procede a estudiar, dado cada secuencia, la probabilidad de que haya sido generada por T contrastada con la probabilidad de haber sido generada por otra textura, llamémosla M. Estudiando la proporción de secuencias correctamente deducidas obtendríamos así la intromisión de M en T. Invirtiendo los roles de las texturas tendríamos una aproximación para la intromisión de T en M. Enunciemos esto en forma de un algoritmo:

Algoritmo 3.1. *Cálculo de la intromisión entre dos texturas.*

Entrada: dos texturas T_1, T_2 y un valor entero n especificando el tamaño de la muestra.

Salida: aproximación a de $I(T_2$ en $T_1)$.

1. *Generar n secuencias con la textura T_1*
2. *Inicializamos el contador en 0*
3. *Para cada secuencia, calcular la probabilidad de ser generada por T_1 y por T_2*
4. *Si estamos en el modo optimista, incrementamos el contador en en uno por cada secuencia s en donde $P(s \mid T_2) > P(s \mid T_1)$*
5. *Si estamos en el modo pesimista, incrementamos el contador en en uno por cada secuencia s en donde $P(s \mid T_2) \geq P(s \mid T_1)$*

6. *Devolvemos* $\frac{contador}{n}$

La dependencia se aproxima sumando $I(T_2\ en\ T_1)$ y $I(T_1\ en\ T_2)$.

Ejemplo 3.2. *Empezamos con un ejemplo sencillo. Será con las mismas texturas que utilizamos en los ejemplos anteriores. Recordamos que* $I(T_1\ en\ T_2) = 0$, $I(T_1\ en\ T_2) = 0{,}25$ *y* $dep(T_1, T_2) = 0{,}15$. *Como no existe coincidencia de probabilidades entre ambas texturas, poco importa si se usa el método optimista o pesimista.*

Mostramos en esta tabla las aproximaciones obtenidas en función del tamaño muestral.

Tamaño	$I(T_1\ en\ T_2)$	$I(T_2\ en\ T_1)$
10	*0.0*	*0.2*
100	*0.0*	*0.28*
300	*0.0*	*0.23*
1000	*0.0*	*0.269*
10000	*0.0*	*0.2452*
100000	*0.0*	*0.24974*
1000000	*0.0*	*0.24974*
10000000	*0.0*	*0.250839*

Ejemplo 3.3. *Consideremos ahora un ejemplo más práctico, tomando en cuenta texturas más realistas. Supongamos que nos proponemos a identificar las zonas de una secuencia de ADN que tienen buenas chances de estar dentro de una isla CpG. Nos proponemos a calcular el factor de intromisión entre ambas texturas, pudiendo así obtener la dependencia y sabiendo así hasta cuanto podemos esperar equivocarnos.*

La dependencia variará fuertemente con el tamaño de la muestra, es de esperar que secuencias más grandes generen smayor significancia estadística sobre las estimaciones y, por lo tanto, reduzca la dependencia entre ellas. Veamos como se verifica esto con el ejemplo de las islas CpG. Usamos el modelo pesimista, de todas formas las diferencias son mínimas puesto que es muy raro una coincidencia completa en la probabilidad, único punto en donde ambas definiciones difieren.

Mostramos los resultados obtenidos en el cálculo de la intromisión, con un tamaño muestral de 10000 secuencias y con distintos tamaños de ventana:

Tamaño	$I(T_1$ *en* $T_2)$	$I(T_2$ *en* $T_1)$	$dep(T_1, T_2)$
10	*0.1932*	*0.2884*	*0.2408*
20	*0.1257*	*0.1689*	*0.1473*
30	*0.0823*	*0.1051*	*0.0937*
40	*0.054*	*0.0782*	*0.0661*
50	*0.0398*	*0.0477*	*0.04375*
75	*0.0139*	*0.0199*	*0.0169*
100	*0.0065*	*0.0052*	*0.00585*
200	*4.0E-4*	*3.0E-4*	*3.5E-4*

Se confirma la hipótesis, cuanto mayor el tamaño de la secuencia menor será la dependencia. Esto nos indica que la calidad de los métodos basados en MV depende fuertemente del tamaño de las secuencias en cuestión. Como veremos más adelante, cuando se introduzcan los métodos de "ventanado", lo que pasará a importar será el tamaño estimado de las texturas. En general, convienen texturas grandes y, por ende, marcadas de fuerte contenido estadístico.

3.7. Aspectos topológicos de la dependencia

Disponemos de un concepto de dependencia entre texturas que de alguna manera mide la "mezcla" entre las mismas al que denominamos dependencia. Es razonable entonces pensar en que es posible, por lo tanto, definir una distancia entre estas que haga uso de esta noción. Las ventajas de disponer de una métrica son muchas. Podremos entonces hablar de métrica e introducir conceptos propios de la topología y el estudio de los espacios métricos. Sin embargo, no se ahondará en detalles en lo que respecta a estos puntos, nos limitaremos a mostrar que es posible utilizar conceptos de los estudios topológicos con la dependencia.

Se utiliza la definición de distancia normalmente utilizada en la teoría de los espacios métricos. Repetimos a continuación la definición de un espacio métrico.

Aclaración: en lo que resta en esta sección se asumirá que las texturas son finitas.

Definición 3.5. Dado X un conjunto y d una función de $X \times X$ en $\mathbb{R}_{\geq 0}$ se dice que d es una *distancia* si cumple las siguientes propiedades:

- $\forall\ x \in X,\ d(x,x) = 0$

- $\forall\ x, y \in X,\ d(x,y) = 0 \Rightarrow x = y$
- $\forall\ x, y \in X,\ d(x,y) = d(y,x)$
- $\forall\ x, y, z \in X,\ d(x,z) \leq d(x,y) + d(y,z)$

Procedemos a definir una distancia utilizando la dependencia antes armada.

Teorema 3.1. *La función* $d : \mathcal{T} \times \mathcal{T} \to \mathbb{R}_{\geq 0}$ *definida por*
$d(T,M) = 1 - dep_{pes}(T,M)$
es una distancia en $\mathcal{T}$*.*

Antes de probar este resultado nos será útil definir algunas otras funciones que simplificarán un poco la notación.

Definición 3.6. Dado una secuencia s, definimos la *dependencia local* como una función $dep_s : \mathcal{T} \times \mathcal{T} \to \mathbb{R}_{\geq 0}$ dada por

$$dep_s(T,M) = \begin{cases} \frac{T(s)}{2} & \text{si} \quad M(s) > T(s); \\ \frac{M(s)}{2} & \text{si} \quad T(s) > M(s); \\ T(s) \text{si} & T(s) = M(s) \end{cases}$$

Se hace notar que en virtud de esta definición podemos reescribir la dependencia como

$dep(T,M) = \sum_{s \in Sop(T) \cup Sop(M)} dep_s(T,M)$.

Otro detalle interesante de la dependencia local es que $dep_s(T,M) \leq min(T(s), M(s))$.

Por último, como en un elemeto fuera del soporte la textura se evalúa en 0, tenemos que vale la siguiente igualdad:
$dep(T,M) = \sum_{s \in Sop(T) \cup Sop(M)} dep_s(T,M) = \sum_{s \in Sop(T)} dep_s(T,M)$

$= \sum_{s \in Sop(M)} dep_s(T,M) = \sum_{s \in Sop(T) \cap Sop(M)} dep_s(T,M)$.

demostración del teorema, d es una distancia. 1) Veamos que $\forall T, d(T,T) = 0$.

$d(T,T) \;=\; 1 - dep(T,T) \;=\; 1 - \sum_{s \in Sop(T)} dep_s(T,T).$

$= 1 - \sum_{s \in Sop(T)} dep_s(T,T) \;=\; 1 - \sum_{s \in Sop(T)} T(s) \;=\; 1 - 1 \;=\; 0.$

2) Sean T_1 y T_2 texturas y supongamos $d(T_1, T_2) = 0$. Eso implica $1 - \sum_{s \in Sop(T_1) \cup Sop(T_2)} dep_s(T_1, T_2) \;=\; 0.$

Supogamos que $T_1 \neq T_2$. Luego, $\exists s_0$ tal que $T_1(s_0) \neq T_2(s_0)$.
Podemos ademas suponer que $T_1(s_0) > T_2(s_0)$, lo que implica que $dep_{s_0}(T_1, T_2) = T_2(s_0)$.
Luego, podemos decir que $dep_{s_0} < min(T_1(s_0), T_2(s_0))$ y en el resto de los puntos podemos afirmar que es menor o igual.
Por lo tanto,

$\sum_{s \in Sop(T_1) \cup Sop(T_2)} dep_s(T_1, T_2) < \sum_{s \in Sop(T_1)} T_1(s) = 1$

Pero esto a su vez implica que $d(T_1, T_2) = 1 - dep(T_1, T_2) > 0$, contradiciendo lo supuesto. El absurdo provino de suponer que $T_1 \neq T_2$, luego $T_1 = T_2$.

3) La simetría es un hecho claro puesto que la dependencia es también simétrica.

4) La desigualdad triangular es probablemente la parte más difícil de la demostración. Supongamos T_1, T_2 y T_3 texturas finitas, queremos ver que $d(T_1, T_3) \leq d(T_1, T_2) + d(T_2, T_3)$.

Para facilitar la escritura definimos $X = Sop(T_1) \cup Sop(T_2) \cup Sop(T_3)$.

Reemplazando cada cosa por su definición, tenemos que:

$1 - dep(T_1, T_3) \leq 2 - dep(T_1, T_2) - dep(T_2, T_3)$

$\Leftrightarrow 1 \geq dep(T_1, T_2) + dep(T_2, T_3) - dep(T_1, T_3)$
$\Leftrightarrow 1 \geq \sum_{s \in X} dep_s(T_1, T_2) + dep_s(T_2, T_3) - dep_s(T_1, T_3)$

Notemos que es posible particionar el dominio X en una serie de conjuntos, en función de la textura que prime en la secuencia dada. Podemos definir:

X_{123}	$s \in X / T_1(s) = T_2(s) = T_3(s) = T_{123}(s)$
$X_{1>23}$	$s \in X / T_1(s) > T_2(s) = T_3(s) = T_{23}(s)$
$X_{23>1}$	$s \in X / T_2(s) = T_3(s) = T_{23}(s) > T_1(s)$
$X_{2>13}$	$s \in X / T_2(s) > T_1(s) = T_3(s) = T_{13}(s)$
$X_{13>2}$	$s \in X / T_1(s) = T_3(s) = T_{13}(s) > T_2(s)$
$X_{3>12}$	$s \in X / T_3(s) > T_1(s) = T_2(s) = T_{12}(s)$
$X_{12>3}$	$s \in X / T_1(s) = T_2(s) = T_{12}(s) > T_3(s)$
$X_{1>2>3}$	$s \in X / T_1(s) > T_2(s) > T_3(s)$
$X_{1>3>2}$	$s \in X / T_1(s) > T_3(s) > T_2(s)$
$X_{2>1>3}$	$s \in X / T_2(s) > T_1(s) > T_3(s)$
$X_{2>3>1}$	$s \in X / T_2(s) > T_3(s) > T_1(s)$
$X_{3>1>2}$	$s \in X / T_3(s) > T_1(s) > T_2(s)$
$X_{3>2>1}$	$s \in X / T_3(s) > T_2(s) > T_1(s)$

Podemos reescribir la fórmula en función de esta partición y nos quedará:

$$\Leftrightarrow 1 \geq \sum_{s \in X_{123}} T_{123}(s) + \sum_{s \in X_{1>23}} T_{23}(s) + \sum_{s \in X_{23>1}} T_{23}(s) +$$

$$\sum_{s \in X_{2>13}} 0 + \sum_{s \in X_{13>2}} T_2(s) - T_{13}(s) + \sum_{s \in X_{3>12}} T_1 2(s) +$$

$$\sum_{s \in X_{12>3}} T_{12}(s) + \sum_{s \in X_{1>2>3}} \frac{T_2(s)}{2} + \sum_{s \in X_{1>3>2}} T_2(s) - \frac{T_3(s)}{2} +$$

$$\sum_{s \in X_{2>1>3}} \frac{T_1(s)}{2} + \sum_{s \in X_{2>3>1}} \frac{T_3(s)}{2} + \sum_{s \in X_{3>1>2}} T_2(s) - \frac{T_1(s)}{2}$$

$$\sum_{s \in X_{3>2>1}} \frac{T_2(s)}{2}$$

Todos estos términos se pueden acotar por $T_2(s)$. Nos queda entonces que $\leq \sum_{s \in X} T_2(s) \; = \; 1$
y llegamos así a lo que se quería probar.

□

Tenemos entonces definida una métrica entre las texturas por medio de una distancia definida en base de la dependencia. Es natural entonces poder estudiar distintas propiedades topológicas que se pueden inferir del mismo, pero no se profundizará más en detalle sobre este tema en lo que resta del trabajo. Se deja abierta la posibilidad como una futura via de investigación.

4. Método MV de Ventaneado

4.1. Descripción del método

Hasta ahora, hemos supuesto problemas de clasificación sobre secuencias generadas por una única textura. El punto es que este es un escenario sumamente irrealista y poco práctico. En el caso general, trabajaremos con secuencias que bien pueden contener más un rasgo estadístico importante, dependiendo de en que porción se la mire. Por ejemplo, supongamos que tenemos una secuencia extraida de algún cromosoma particular de un ratón y queremos extraer las porciones codificantes de la misma. Es probable que el comportamiento estadístico de una parte codificante no coincida con el comportamiento en una parte no codificante (como puede ser un intrón) y, por lo tanto, es muy probable que distintos patrones estadísticos convivan en la secuencia.

La idea es ahora extender un poco más la idea a casos en donde se evidencie la presencia de más de una textura. Es decir, suponemos ahora secuencias en donde cada porción es generada por una textura distinta. El problema consistiría ahora en, dado una secuencia y una serie de texturas, asignar a cada posición de la secuencia la textura que mejor se adecúa para esa posición en función de algún criterio a determinar.

Queremos trasladar los conceptos de clasificación "monotextural" que se introdujo en la sección anterior, por lo que consideraremos la estimación de máxima verosimilitud. A grandes rasgos el método consiste en recorrer la secuencia posición por posición con una ventana de tamaño predeterminado. Nos paramos en una posición, extendemos la ventana y de ahí determinamos cuál es la textura que maximiza la probabilidad de que la subsecuencia contenida en la ventana haya sido generada por esa textura. Es decir, aplicamos un proceso de máxima verosimilitud en cada posición de la secuencia.

En el siguiente recuadro detallamos el algoritmo en su totalidad.

Algoritmo 4.1. *Ventaneado MV.*

Entrada: una secuencia S, un tamaño máximo de ventana r y un conjunto de texturas $T_1, \ldots, T_n$.

Salida: una asignación de texturas para cada posición de la secuencia de forma tal de maximizar la verosimilitud en cada parte de la secuencia.

1. *Establecemos* $i \leftarrow 0$.

2. *Inicializamos un arreglo de asignaciones a_i de la misma longitud que la secuencia.*

3. *Mientras i no supere la longitud de la secuencia ...*

4. *Extendemos una ventana de centro i y radio r. Si la ventana sobrepasa algún límite de la secuencia se la recorta para que quepe. Obtenemos la subsecuencia s_i contenido en la ventana.*

5. *Buscamos la textura T_j que maximice $P(s_i \mid T_j)$. Llamamos j_i al índice de la textura que verifica este máximo.*

6. *Establecemos $a_i \leftarrow j_i$.*

7. *Incrementamos i en 1.*

8. *Devolvemos el arreglo de asignaciones a_i.*

4.2. Analogías Físicas: la Energía

Así como la idea de los rasgos estadísticos fue abstraido en el concepto de "textura", también podemos efectuar una movida similar con los métodos encargados de reconocer a las mismas, y aquí entrará en juego la energía.

Es posible pensar que todos los métodos lo que en última instancia hacen es minimizar una función de energía definida en base a la asignación de texturas de una secuencia. Los métodos tienen un objetivo común: minimizar esta energía. La cuestión es que la definición que se use para esta "energía" nos terminará indicando el mecanismo de texturado.

Toda función de energía debe medir la calidad de una asignación en base a algún criterio en particular. En ese sentido, las distintas metodologías de texturado tienen dos cometidos: en primer lugar arman una función de energía, eso es, definen un criterio de optimalidad de asignación de texturas. Cuanto menor es la energía, mejor es el esquema de asignación en función del método que se esté usando. Sin embargo, no alcanza con definir la energía, también es preciso indicar algún método para poder minimizar la misma. Habrá casos en que esto será una tarea sencilla y otros en donde adquiere el cáriz de un problema de optimización combinatoria, debiendo entonces recurrir a una serie de heurísticas de minimización funcional.

Resumimos todo lo enunciado con la siguiente definición:

Definición 4.1. Dada una secuencia s de longitud r y un conjunto de texturas $T_1, \ldots T_n$, un *método de texturado* se define como el par (E,M) en donde

- $E : Ix \ldots xI \rightarrow \mathbb{R}$ es una función *energía* que dada una asignación de texturas para cada posición nos devuelve un número.
- M es un esquema de minimización funcional.

donde $I = \{ 1, \ldots n\}$.

Para el caso de ventaneado en MV la función de energía que se puede utilizar es

$E(a_1, \ldots a_r) = E_1(a_1) + \ldots + E_r(a_r)$

donde los E_i son funciones locales de energía, definidos como:

$E_i(a_i) = -log(P(\{ s_{i-v} \ldots s_{i+v}\} \mid \text{ la textura } a_i))$

donde v es el tamaño de la ventana (ajustada en caso de sobrepasar el borde de una secuencia) y los s_j son el símbolo ubicado en la posición j.

Cabe destacar que la asignación que se efectúe en la posición i no influye en la energía de cualquier otra posición. Luego, para minimizar la energía bastará minimizar cada una de las energías locales. Eso se consigue escogiendo la textura que mejor se adecúa en esa posición, tal como expresa el método expresado en el inciso anterior. Estamos así en un caso en donde el esquema de minimización es simple y directo.

¿Por qué se usa el logaritmo? En primer lugar, porque normalmente se estarán utilizando texturas de caracter Markoviano en donde existe una fuerte tendencia a los factores multiplicativos. El logaritmo tiene la propiedad de convertir los productos en sumas y de esa forma es posible reducir el impacto de los errores, provenientes en gran medida por el hecho de que esos factores podrían llegar a reducir a la probabilidad a valores muy cercanos a cero, corriéndose el riesgo de un *underflow*.

Por supuesto, estamos asumiendo que calcular la probabilidad es una tarea sencilla, y esto no tiene porque ser así. En el caso de las texturas markovianas, es una tarea sencilla, es simplemente aplicar una serie de multiplicación con distintos factores pero bien podría ser que se utilice una textura más complicada (como podría ser un campo markoviano) en donde ya no es tan claro el cálculo de este número. Es más, no sería raro que en algunos casos uno debe contentarse con obtener aproximaciones numéricas a la energía.

4.3. Ejemplos

Mostraremos algunos ejemplos con el fin de ilustrar la aplicabilidad del método a varios problemas clásicos. Se hace notar que en cada ejemplo calcularemos la dependencia y veremos como son los resultados. Es importante destacar que el tamaño de la ventana termina definiendo en última instancia el juego de texturas que se utilizará y por ende nos dará la dependencia que se puede esperar.

4.3.1. Ejemplo 1

La idea de este ejemplo es mostrar, por un lado, el funcionamiento del método y por otro lado empezar a definir el formato general de los ejemplos de métodos de ventaneado.

El radio de la ventana será de 50 (eso es, subsecuencias de 101 nodos). Los valores de interferencia para ambas texturas son:

$I(T_1\ en\ T_2)$	$I(T_2\ en\ T_1)$	$dep(T_1, T_2)$
0.0074	0.0070	0.0072

Generaremos una realización de 250 símbolos con la textura CpG+ y le concatenaremos otra realización de 250 símbolos armada con la textura CpG-. La realización obtenida es:

GCTATTCATG TTCAAATAGC CCTTACCCCC TACCGCTTTT
CCCCTTTTTC ATTCACCTTT CGTCGCCTTT CTTTTTTTCA
GCCTGCATGG CACCTGCCCA CTCACTTATC CCCCCCACGG
ATCGCAAAAA CCGCTTGCAC TTATTCTTCG TTCATCCCCG
CGTGCCATTT ATTACATTCC TTGCAGTGGC TTGCCTCAAT
CCATTTCCTA TCTGTGCACT TGTCCTGCCT GTCCGCCTCT
TATGTGCATC GCGGTGGGAG GATTATTTAA TATGCGTTTC
GCCAAAGGTC ATTCATTCAC GTTTTACACT CAAATTTGAT
TCAAATATTC GCAACGCCCG TTTTTCAGGC GTTTATTAGG
ACAATAAGAA CGCATAGGGG TATACCGCGT TGTTCCGTTG
ACGGGTGGGG GGAAAGGTGG CTATCCGGGC AATGTTTGCT
GCGCGGCAGT CGGAGACAAG CCACATCGCG GAGTAAGTAT
TTGCGAGTGC CGGCAACGCT

En la figura 4 de comparación energética. En el eje X tenemos las posiciones de la secuencia y en el eje Y tenemos la diferencia entre la energía local de la posición X con la textura T_1 y la energía local con la textura T_2.

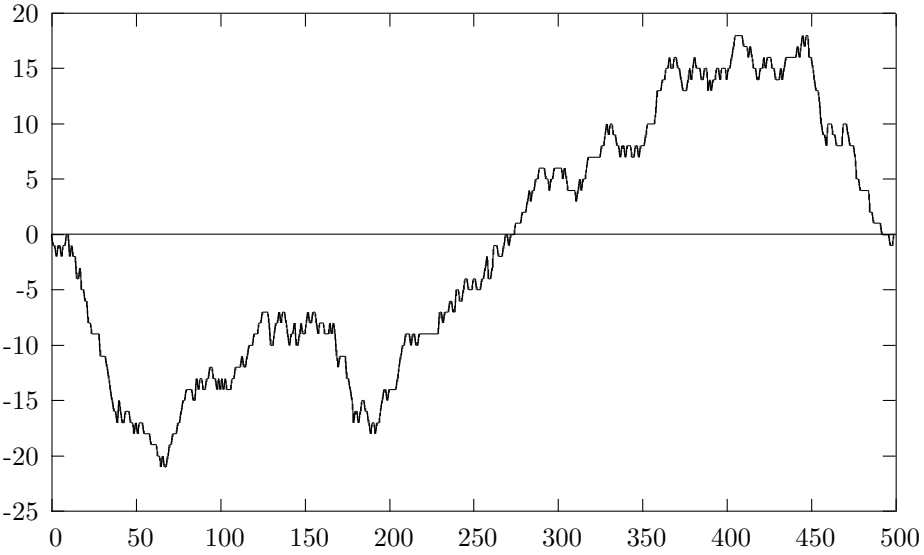


Figura 4: clasificación con radio 50

Recordamos que se intenta minimizar la energía, una diferencia positiva establece que T_1 tiene más energía que T_2 y es, por lo tanto, menos factible que T_2. Luego, los valores negativos favorecen a la primer textura y los positivos a la segunda.

Posiciones incorrectamente clasificadas: 27.

O sea, hay un 5 (por ciento) de las posiciones incorrectamente clasificados. Esto es mayor que el 0.7 (por ciento) predicho por la dependencia pero este fenómeno se debe a los efectos de borde entre texturas, una cuestión que no es tomado en cuenta durante el cálculo de la dependencia. Este problema será discutido en una sección posterior.

4.3.2. Ejemplo 2

Seguimos con las texturas del ejemplo anterior. Ahora suponemos un escenario de intercalación de texturas CpG positivas y negativas. Tenemos una secuencia de 1000 posiciones en donde las texturas se intercalan cada 100 posiciones.

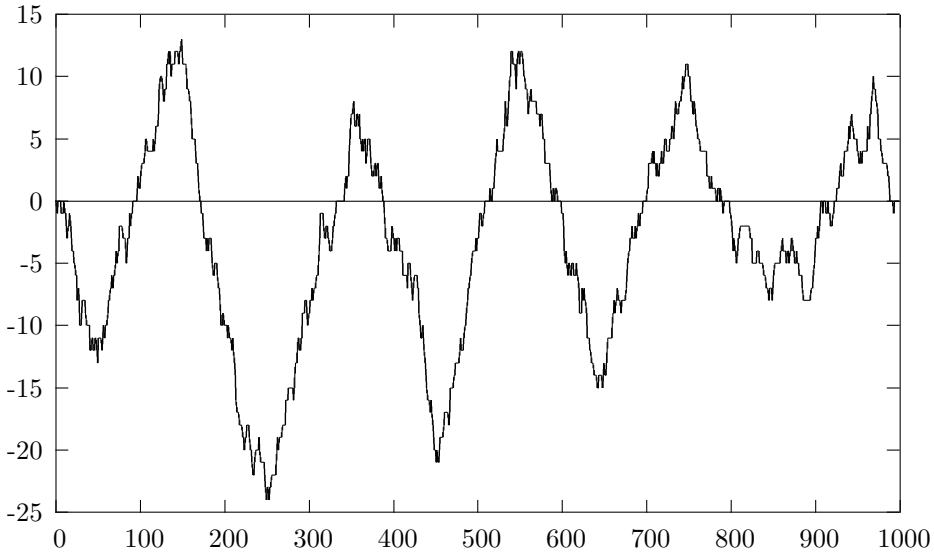


Figura 5: clasificación con radio 50

Con una ventana de radio 50 obtenemos los resultados que se pueden observar en la figura 5.

Posiciones incorrectamente clasificadas: 143.

La distancia de Hamming ha sufrido un fuerte incremento respecto del ejemplo anterior (un porcentaje del 14), a pesar de que el tamaño de la ventana es el mismo y las texturas no han sufrido cambio. Los siguientes factores han sido alterados:

- Se incrementó la cantidad de regiones con texturas. En el ejemplo anterior había 2, ahora tenemos 10.
- El tamaño de cada región se ha visto disminuida de 250 a 100.

4.4. Dependencia y Efectos de Borde

En función de lo expresado en el capítulo anterior, se dedujo que una condición fundamental para el buen desempeño de un método basado en MV

es que la dependencia entre las texturas sea baja. Sin embargo, como veremos ahora y como se ha podido empezar a ver en los ejemplos anteriores, no es este el único factor que determina la bondad del método.

En efecto, la dependencia es en última instancia la probabilidad de clasificar incorrectamente una secuencia generada por una textura. De esa forma, se asume que las secuencias son generadas únicamente por una única textura. Los ejemplos que se mostraron antes y, en definitiva, el rango de aplicabilidad del método incluye principalmente secuencias generadas con más de una textura. Es por eso que en el proceso de texturado empezarán a surgir anomalías de clasificación, concentrados fundamentalmente en las fronteras de las texturas.

En las cercanías de un borde, las ventanas que se arrastren incorporarán partes de las dos texturas colindantes, la deducción tenderá a volverse cada cez menos precisa hasta que se llega al borde, punto en donde, en general, la selección de textura se vuelve aleatoria.

El problema se vuelve mayor cuando las texturas son pequeñas pues en ese caso se incrementará la cantidad de regiones de borde. Una forma de reducir este fenómeno es reduciendo el tamaño de las ventanas. Así, la cantidad de ventanas que incorporen más de una textura será menor. Esta solución serviría para el problema de los bordes pero corre el riesgo de incorporar un nuevo problema: la dependencia. Como se ha mencionado en la sección anterior, disminuir el tamaño de las ventanas tiende a causar que la dependencia entre distintas texturas markovianas se incremente. En efecto, tiene sentido esto puesto que la reducción de las ventanas tiende a disminuir el significado estadístico de las mismas.

En definitiva, tenemos entonces dos problemas fundamentales para los cuales la solución de uno de ellos implicaría una potenciación del otro. El primero son los efectos de borde, cuya solución requiere una disminución del tamaño de la ventana. El segundo de ellos es la dependencia entre las texturas, cuya solución requere un aumento del tamaño de la ventana.

Mostraremos algunos ejemplos gráficos en donde se ilustran estos problemas.

Procedemos a la texturización de una secuencia de 5000 posiciones, con texturas CpG positivas y negativas intercaladas cada 500 sitios.

Con una ventana de radio 3 (eso es, subsecuencias de tamaño 7) se pueden observar los resultados en la figura 6.

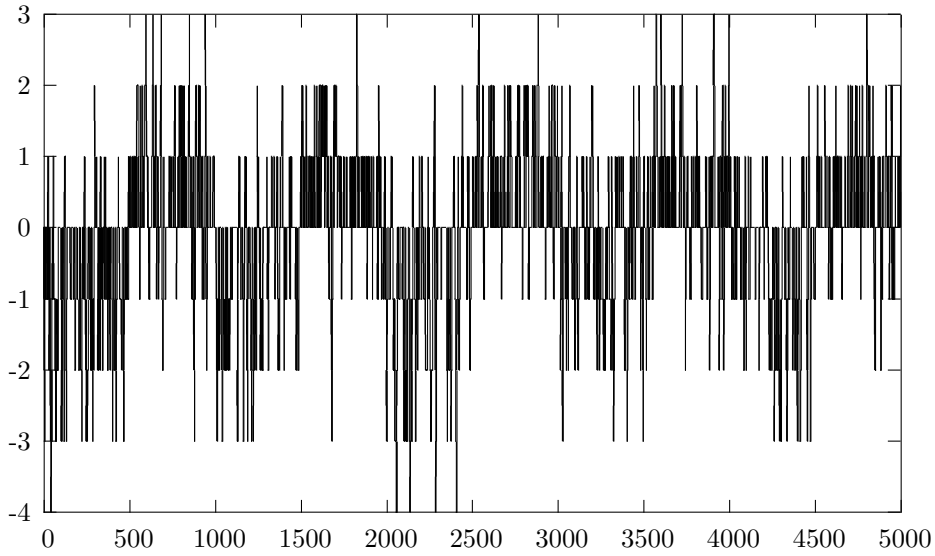


Figura 6: clasificación con radio 3

$I(T_1\ en\ T_2)$	$I(T_2\ en\ T_1)$	$dep(T_1, T_2)$
0.231	0.3494	0.2902

Posiciones incorrectamente clasificadas: 1414 (28 %).

Más que por los bordes aquí el problema viene dado por la fuerte dependencia entre ambas texturas, producto del reducido tamaño de secuencia que se utiliza. Cabe destacar la coincidencia importante entre la dependencia predicha (29 %) y el resultado obtenido (28 %), mostrándonos que en general los problemas surgidos son propios de la dependencia entre las texturas y no por los bordes entre las mismas.

Mostramos ahora en la figura 7 el mismo ejemplo pero con un radio de ventana de 70, para un total de subsecuencias de 141 sitios.

$I(T_1\ en\ T_2)$	$I(T_2\ en\ T_1)$	$dep(T_1, T_2)$
0.0012	0.0027	0.00195

Posiciones incorrectamente clasificadas: 223 (4,5 %).

Se destaca una notable mejoría en los resultados, pero se vuelve a eviden-

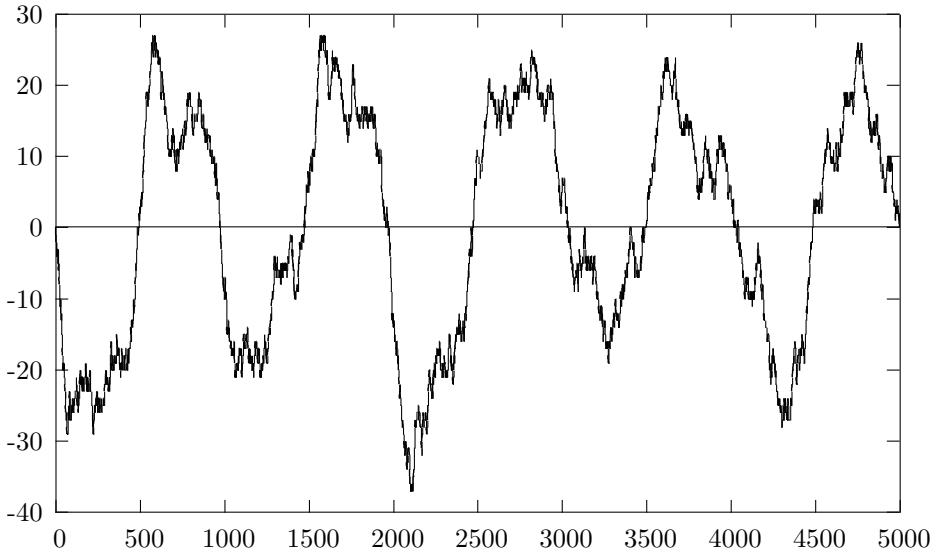


Figura 7: clasificación con radio 70

ciar una diferencia sustancial con lo predicho por la dependencia. Otra vez, causa de los efectos de borde.

Ahora tomamos un radio 500, totalizando 1001 sitios por ventana. Los resultados se pueden apreciar en la figura 8.

$I(T_1\ en\ T_2)$	$I(T_2\ en\ T_1)$	$dep(T_1, T_2)$
0.0	0.0	0.0

Posiciones incorrectamente clasificadas: 2067 (41 %).

Los cálculos de la dependencia ya no tienen correlato alguno con la realidad, los efectos de borde se han impuesto por completo en el sistema, al punto tal de que ya no existen los mismos. El efecto de incrementar el tamaño de ventana es una mezcla entre las texturas. Esto se aprecia bien en el gráfico, varias texturas se han perdido producto de la mezcla. En el caso extremo, con ventanas que cubran toda la secuencia, tendremos que el texturado nos dará igual en toda la secuencia. La textura ganadora será aquella que maximice la probabilidad de obtener la secuencia. Sería, en definitiva, volver al

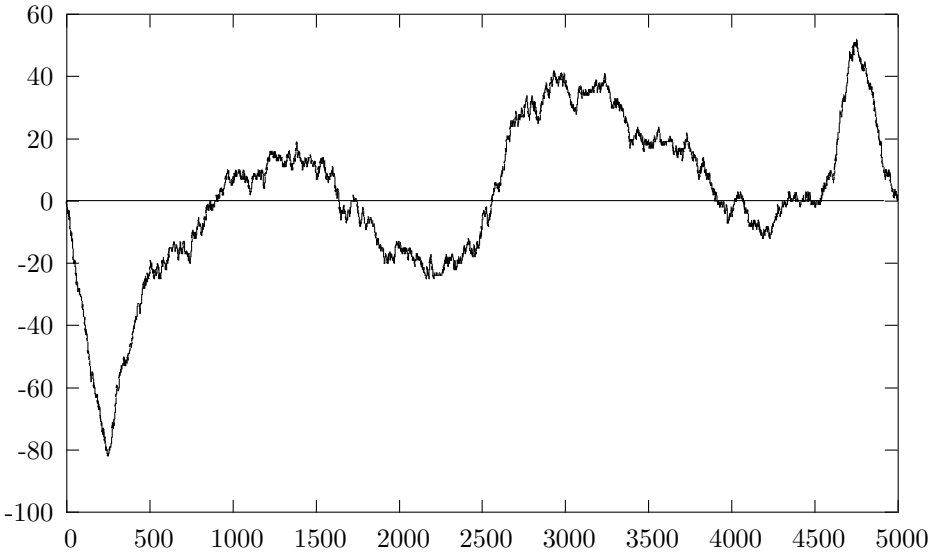


Figura 8: clasificación con radio 500

caso de MV sin ventaneado.

El resultado de incrementar el tamaño de la ventana es imponer un suavizado sobre la curva hasta llegar a la meseta absoluta.

El gráfico de la figura 9 marca la relación existente entre el radio de la ventana y la cantidad de posiciones incorrectas entre el texturado inferido y el real (siempre en el mismo ejemplo).

En los bordes el error ciertamente alto, pero las causas son muy distintas. En el lado izquierdo lo que ocurre es debido a dependencia, en el lado derecho se debe a los efectos de borde y la mezcla de texturas.

En el medio tenemos un compromiso entre ambas anomalías que nos permite obtener resultados aceptables, con errores en varios casos inferiores al 5 %.

En un problema dado uno desearía caer en este caso, por lo que se debe poder determinar estos valores de compromiso. Hay una relación estrecha entre la dependencia de las texturas y el tamaño de las regiones.

En general, la dependencia es un problema que rápidamente se debilita

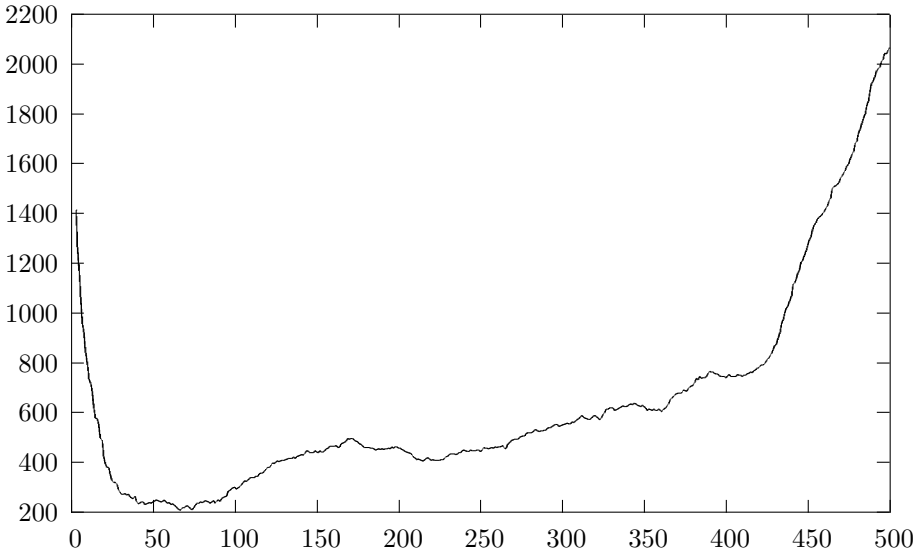


Figura 9: Radio vs Error

al incrementar el tamaño de la ventana, especialmente con texturas de familias Markovianas. Eso nos indica que no es necesario utilizar ventanas muy extensas, que nos podrían llegar a dar conflictos de borde.

Es importante destacar que existe una amplia región en donde el desempeño general es más o menos el mismo. Esto parece indicar que no existiría a priori un tamaño específico que sea óptimo para la predicción, sino que mientras no haya fuertes corrimientos hacia los extremos los resultados serán más o menos los mismos. O, en todo caso, el efecto de la dependencia y los bordes será reducido.

5. Experimentos

5.1. Descripción de los experimentos

En los experimentos que vienen se compararán varios sistemas de reconocimiento de genes, se probará con distintos tipos de texturas y funciones de energía.

Los experimentos incluyen las siguientes categorías:

Para las texturas:

- Cadena de Markov de primer orden
- Cadena de Markov inversa: es una cadena pero en donde las transiciones son de derecha a izquierda
- Proceso Markov espacial de primer orden
- Cadena de Markov de segundo orden
- Cadena de Markov de tercer orden
- Cadena de Markov de cuarto orden
- Cadena de Markov de quinto orden

Para cada combinación el procedimiento será el mismo: se toma un conjunto de secuencias de entrenamiento, se las separa en exones e introns en función del conocimiento de las mismas. Luego, se deducen las texturas que correspondan para los exones e introns de cada caso por medio de un análisis frecuentista, tal como se explica en el capítulo 2. Es interesante incorporar pruebas con cadenas de Markov de diverso orden, principalmente para poder compararlo con el comportamiento del proceso espacial. Recordamos que la cadena de Markov de primer orden requiere 16 parámetros de transición contra los 64 del proceso espacial, 64 también para la cadena de segundo orden y 256 para la de tercer orden. Por lo tanto, tenemos casos con menos, igual y mayor cantidad de parámetros que el proceso espacial. Vamos a ver hasta que punto estos parámetros afectarán el comportamiento de la deducción.

5.2. Calidad de las medidas

Para determinar la calidad de la detección nos limitaremos a un estudio a nivel de nucleótido. El sistema de detección se limita a definir si un nucleótido dado es codificante o no, aspectos como la estructura genética y la composición de exones no serán tratados en este informe. Interesará fundamentalmente saber si una predicción dada para un nucleótido es o no correcta.

Una primera medida podría basarse en determinar el porcentaje de posiciones correctamente clasificadas. Sin embargo, esta medida puede resultar un poco tramposa y no es la que se suele utilizar para medir calidad predictiva. Se debe esto fundamentalmente a que, particularmente en caso de eucariotas, las regiones no codificantes son una amplia mayoría en comparación con las codificantes. Luego, un detector que se limite a decir siempre que el nucleótido no es codificante tendría una performance bastante alta, ciertamente mayor al 50 %. Es interesante notar que el genoma humano tiene tan solo una porción muy reducida que codifica proteinas: un 2 % sobre el total. Por lo tanto, un algoritmo que para cada posición determine que no es codificante tendrá un porcentaje de éxito cercano al 98 %, un valor muy alto pese a la trivialidad del algoritmo.

De ahí la necesidad de introducir medidas que ponderen en alguna forma por la cantidad de posiciones que realmente son o no codificantes.

A nivel de nucleótidos, podemos definir las siguientes medidas:

- Especificidad: Positivos Verdaderos / Positivos Existentes
- Sensibilidad: Positivos Verdaderos / Positivos Predichos
- Coaficiente de correlación(aproximado): una aproximación a la correlación entre las dos variables anteriores.

Estas medidas son ampliamente utilizadas para la calificación y comparación de distintos métodos de predicción de genes. Es posible localizar un análisis de las mismas en [BAX01] y en [GUI96].

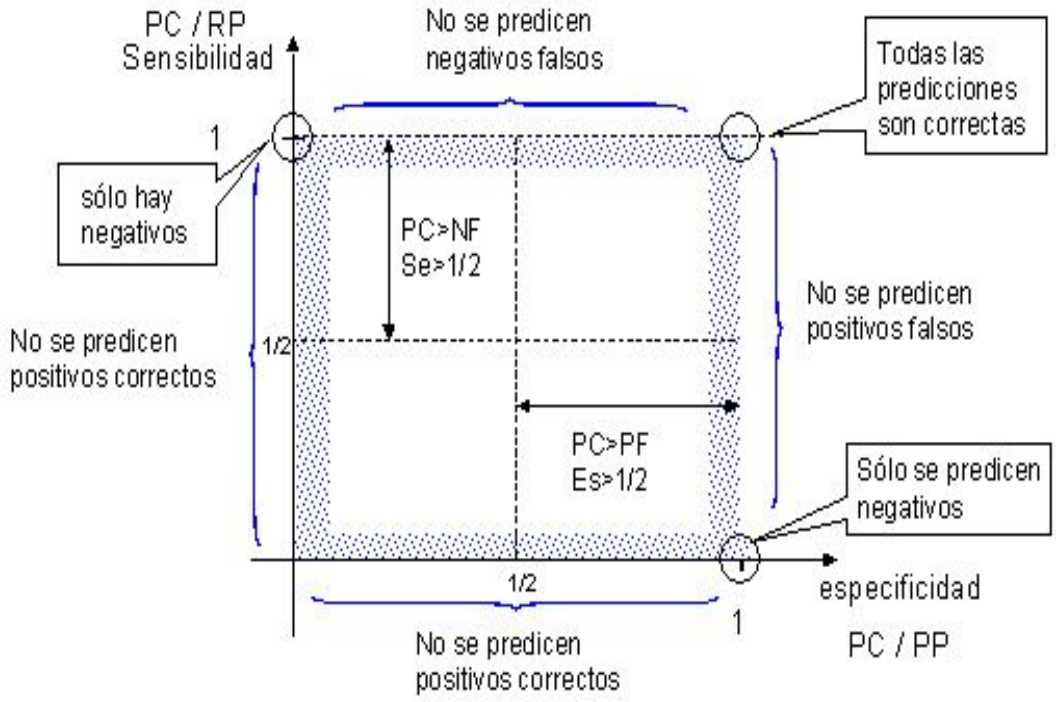
Típicamente utilizaremos las siguientes siglas para denotar los valores que nos interesan:

PV: Positivos Verdaderos (zonas codificantes correctamente clasificadas).
PF: Positivos Falsos (zonas no codificantes incorrectamente clasificadas).
NV: Negativos Verdaderos (zonas no codificantes correctamente clasificadas).

NF: Negativos Falsos (zonas cofificantes incorrectamente clasificadas).
ES: Especificidad (PV / (PV + PF)).
SE: Sensibilidad (PV / (PV + NF)).
CC: Coafieciente de correlación.

El coaficiente de correlación se puede utilizar como medida general de la calidad predicitiva. Es un valor que varía entre -1 y 1. Cuando vale -1 indica que todas las predicciones han sido incorrectas y en 1 que todas las predicciones fueron correctas. Un algoritmo completamente azaroso tendrá un valor que tenderá a acercarse a 0. La fórmula que se usa para el mismo es:

$$CC = \frac{(PV*NV)-(NF*PF)}{\sqrt{(PV+NF)*(NV+PF)*(PV+PF)*(NV*NF)}}$$



Como comentario podemos agregar que no es este el único sistema de evaluación utilizado en la predicción de genes. El sistema aquí presentado trabaja a nivel de nucleótido, existen otros modelos que califican en función de los exones completamente codificados. Nuestro sistema no detecta exones

enteros, con sus bordes, sino que indica el potencial de que una posición dada pertenezca o no a un exón. De ahí que es más razonable la utilización de un mecanismo a nivel de nucleótido antes que a nivel de exón.

5.3. Datos de prueba y texturas

Todas las pruebas ese efectuarán sobre un conjunto especial de datos denominados "Guigo Data Set", utilizados en [GUI96] para la evaluación de una serie de programas de detección genética. Cada una de estas secuencias codifica una proteína en un único sentido.

Se tomarán parte de esas secuencias para "entrenar" el sistema. El entrenamiento consiste en armarse una serie de texturas codificante y no codificante para cada tipo. Para las pruebas que vienen a continuación se ha optado por una veintena de secuencias escogidas sin ningún criterio particular.

De las texturas obtenidos es posible calcular algunas propiedades que enumeraremos a continuación:

5.3.1. Dependencia

Dado un tipo de texturas (cadena de orden 1, proceso espacial, etc) calcularemos la dependencia pesimista existente entre la versión codificante y la no codificante, según se definió en el capítulo 3. Conviene recordad la definición de la dependencia: dado una secuencia generada con cualquiera de las dos texturas determina la probabilidad de ser incorrectamente clasificada. Valores altos de dependencia implican texturas parecidas.

Se aclara que en esta experiencia se ha debido obviar el caso del proceso espacial de Markov debido a la dificultad para generar realizaciones con la misma.

El tamaño de la ventana es de 101 posiciones (50 de radio) y se efectuaron 10000 iteraciones para el cálculo.

Tipo	I(Codif. en No Codif.)	I(No Codif. en Codif.)	Dependencia
Cadena-1	0,0667	0,06	0,06335
Cadena-2	0,0491	0,0418	0,04545
Cadena-3	0,0393	0,0326	0,03595
Cadena-4	0,0106	0,0104	0,0105
Cadena-5	0,0	0,0	0,0

De los resultados se desprende en primer lugar que la dependencia disminuye con el orden de la cadena. Es razonable puesto que al aumentar el orden se está incorporando un mayor número de información específica de cada tipo de secuencia y se vuelve entonces más factible poder distinguir entre ellas. Esto debería redundar a favor de los métodos de orden mayor a la hora de efectuar predicciones. Sin embargo, hay que tener en cuenta que no es este el único factor en juego, también se debe tener en cuenta la calidad del conjunto de entrenamiento y los efectos de borde.

Un detalle interesante es que la intromisión que efectúan las texturas codificantes es mayor que el de las no codificantes. No debe ser un hecho accidental, el hecho de que la texturas no codificantes sean más cercanas a distribuciones uniformes debe jugar algún papel. Las próximas mediciones permitirán avanzar un poco más en este sentido.

5.3.2. Aleatoriedad

Otro parámetro interesante para estudiar es el grado de "desorden" de las texturas, en analogía a las nociones de entropía. Los conceptos desarrollados sobre la dependencia entre texturas permite disponer de una manera de medir el grado de desorden de una textura, calculando la distancia entre ella y una textura basada en una distribución uniforme sobre secuencias de un tamaño fijo, en donde la entropía es máxima.

En las siguientes pruebas se calcula, con ventanas de radio 50 y 10000 secuencias, la distancia entre cada textura y la distribución uniforme.

Tipo	Dependencia codificante	Dependencia no codificante
Cadena-1	0.06225	0.04575
Cadena-2	0.04505	0.0417
Cadena-3	0.02495	0.0388
Cadena-4	0.00645	0.0309
Cadena-5	0.0	0.0124

Para las últimas texturas se confirma el hecho de que las secuencias no codificantes suelen presentar más desorden (o entropía) que las secuencias codificantes. Este hecho se va destacando cada vez más a medida que se incrementa la cantidad de parámetros en juego y, por lo tanto, se incorpora más información de las secuencias dentro de las texturas. Se nota además que este caos no es muy visible y que requiere medidas finas para poder detectarlo, puesto que en las texturas más simples la situación es bastante más pareja.

Probablemente medidas simples de entropía no sean capaces de distinguir variaciones significativas entre secuencias codificantes y no codificantes.

5.3.3. Reducciones de orden

Se intentará estudiar ahora el aporte que hacen los parámetros adicionales a cada orden. Es cierto que toda cadena de primer orden es a su vez una cadena de segundo orden pero la recíproca no es cierta. Sin embargo, puede ocurrir que debido a las circunstancias de las secuencias utilizadas para el entrenamiento si se verifique este hecho, o tal vez se verifique hasta cierto punto. La idea será entonces medir el grado de ocurrencia del mismo. Para eso se deberá exponer un modelo que permita cuantificar este grado de dependencia de orden.

Consideremos T una textura cadena de Markov de orden n, queremos cuantificar el nivel de error que se tiene al reducir la misma a una cadena de orden $n-1$. Debemos inferir una serie de nuevas probabilidades de transición en función de las existentes, y para mantener la consistencia con la textura original esto se puede lograr definiendo un T' de la siguiente forma:

$$\text{Dado } S \in \Sigma^{n-1}, y \in \Sigma \text{ definimos } T'(S->y) = \sum_{x\in\Sigma} T(xS->y)/|\Sigma|.$$

El grado de error estará dado por aquellas circunstancias para las que dado un S, un x, un y y un z la diferencia entre $T(xS->z)$ y $T(yS->z)$ es grande. Podemos entonces cuantificar el nivel de error de reducción de orden mediante:

$$E(T) = \sum_{S\in\Sigma^{n-1}, x\in\Sigma, y\in\Sigma, z\in\Sigma} |T(xS->z) - T(yS->z)|.$$

Se muestra a continuación el nivel de error de reducción de orden para todas las cadenas de Markov. El error en las texturas codificantes se nota como E_+ y en las no codificantes como E_-.

Tipo	E_+	E_-
Cadena-1	0,0593	0,0540
Cadena-2	0,0407	0,0290
Cadena-3	0,0515	0,0209
Cadena-4	0,0748	0,0320
Cadena-5	0,133	0,0598

De estos resultados se desprenden algunas observaciones. Es posible apreciar en una primer instancia que el error por reducción es mayor en las texturas codificantes que en las no codificantes. Es razonable que esto sea así puesto que las texturas intrónicas o no codificantes suelen presentar mayor nivel de entropía y, por lo tanto, mayor cercanía a una distribución uniforme, reducible siempre a una cadena de orden 0.

5.4. Experimentos

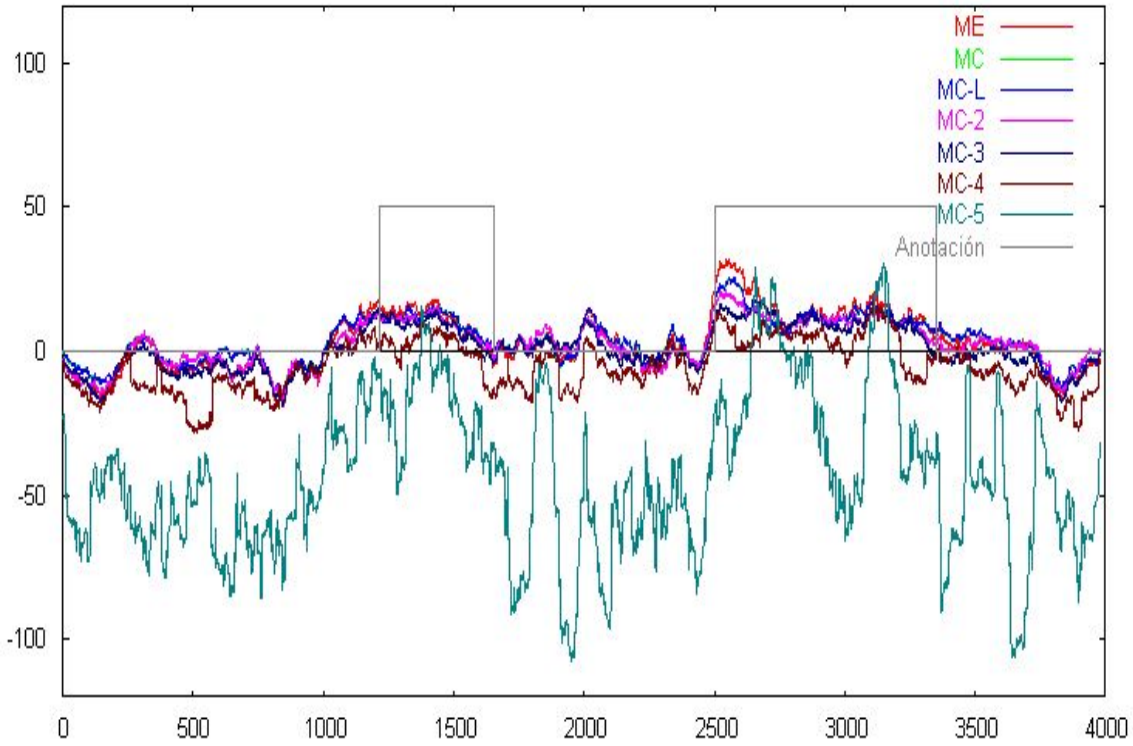
5.4.1. Pruebas Ilustradas

En esta prueba se procederá a aplicar el método de MV ventaneado a una secuencia en particular (MMU02278 del Guigo Data Set), con el fin de determinar las zonas codificantes y no codificantes de la misma. Las texturas a utilizar provienen del data set antes definido y se tendrá un par de texturas (codificante y no codificante) para cada tipo (Proceso Temporal de Markov, Proceso Espacial de Markov, Proceso Temporal Invertido de Markov, Proceso Temporal de Markov de orden 2,3,4 y 5).

En función de las pruebas antes efectuadas y considerando que el tamaño típico de un exón oscila entre 50 y 100 posiciones, se determinó que una ventana de radio 50 sería adecuada para las pruebas. Es suficientemente grande como para reducir efectos de dependencia y no es tan grande como para que se evidencien fuertes efectos de borde.

En el gráfico que sigue tenemos el resultado de la prueba. En el eje X se tiene cada una de las posiciones de la secuencia. En el eje Y tenemos, para cada tipo, la diferencia entre el log-prob de que la ventana sea codificante menos el log-prob de que no sea codificante. Bajo ese criterio, las regiones positivas corresponderías a zonas codificantes y las negativas a las que no lo son. Mostramos además, mediante un escalón fijo, las regiones que son verdaderamente codificantes en función de la anotación dada.

Las calificaciones obtenidas por las distintas texturas se enumeran en la siguiente tabla:



Tipo	PV	PF	NV	NF	SE	ES	CC
Cadena-1	1302	1333	1346	0	1.0	0.494	0.498
Cadena-L	1302	1335	1344	0	1.0	0.494	0.498
Cadena-2	1302	1219	1460	0	1.0	0.516	0.531
Cadena-3	1268	737	1942	34	0.974	0.632	0.657
Cadena-4	1016	258	2421	286	0.780	0.797	0.688
Cadena-5	257	0	2679	1045	0.197	1.0	0.458
Markov-E	1296	1124	1555	6	0.995	0.535	0.554

(PV = Positivos Verdaderos, PF = Positivos Falsos, NV = Negativos Verdaderos, NF = Negativos Falsos, SE = Sensibilidad, ES = Especificidad, CC = Coaficiente de Correlación)

En general, los resultados favorecen a las texturas que más parámetros incorporan. Se destaca la similitud en los resultados del proceso espacial y la cadena de Markov de orden 2, no del todo irracional considerando que ambos precisan la misma cantidad de parámetros para definirse. Es interesante destacar la pobre performance de la cadena de orden 5, indicando que no

basta simplemente con agregar más parámetros para mejorar la predicción, es probable que esto debe verse sustentado por un conjunto de entrenamiento más sustancioso, implicando a su vez una mayor especialización.

5.4.2. Pruebas "Masivas"

Para poder obtener una visión más objetiva de la realidad es preciso hacer pruebas más abarcativas. Consideramos como "Training Set"las primeras 20 secuencias del Guigo Data Set y procedimos a efectuar una predicción sobre las 550 restantes, a modo de "testing set".

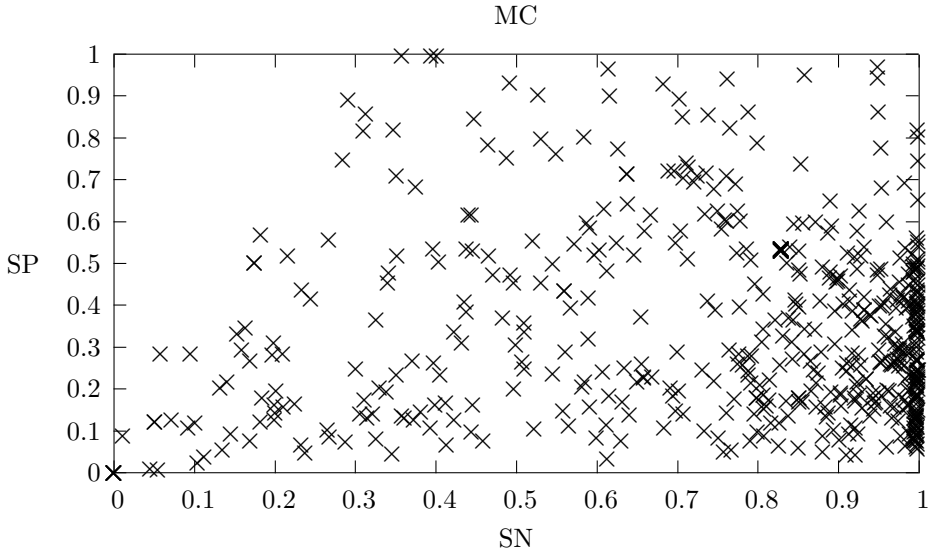
Tipo	Positivos predichos	Negativos predichos	Aciertos
Cadena-1	46,47 %	53,52 %	61,29 %
Cadena-L	46,48 %	53,52 %	61,28 %
Cadena-2	40,39 %	59,61 %	67,70 %
Cadena-3	31,99 %	68,01 %	74,30 %
Cadena-4	18,59 %	81,41 %	81,66 %
Cadena-5	2,80 %	97,20 %	85,32 %
Markov-E	41,64 %	58,36 %	66,26 %

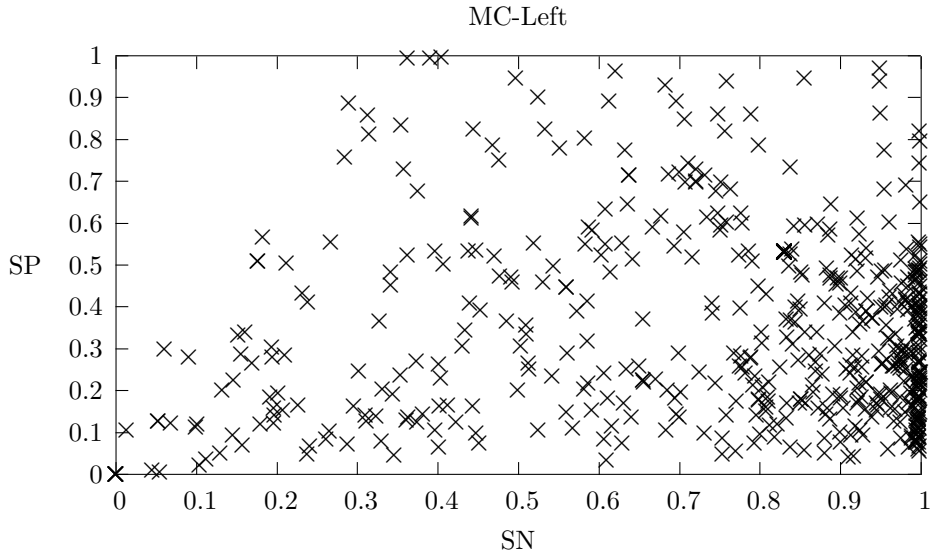
Tipo	TP	TN	FP	FN
Cadena-1	11,51 %	49,78 %	34,96 %	3,75 %
Cadena-L	11,51 %	49,77 %	34,97 %	3,75 %
Cadena-2	11,68 %	56,02 %	28,70 %	3,58 %
Cadena-3	10,77 %	63,53 %	21,21 %	4,48 %
Cadena-4	7,75 %	73,91 %	10,83 %	7,51 %
Cadena-5	1,70 %	83,62 %	1,11 %	13,57 %
Markov-E	11,58 %	64,68 %	30,06 %	3,68 %

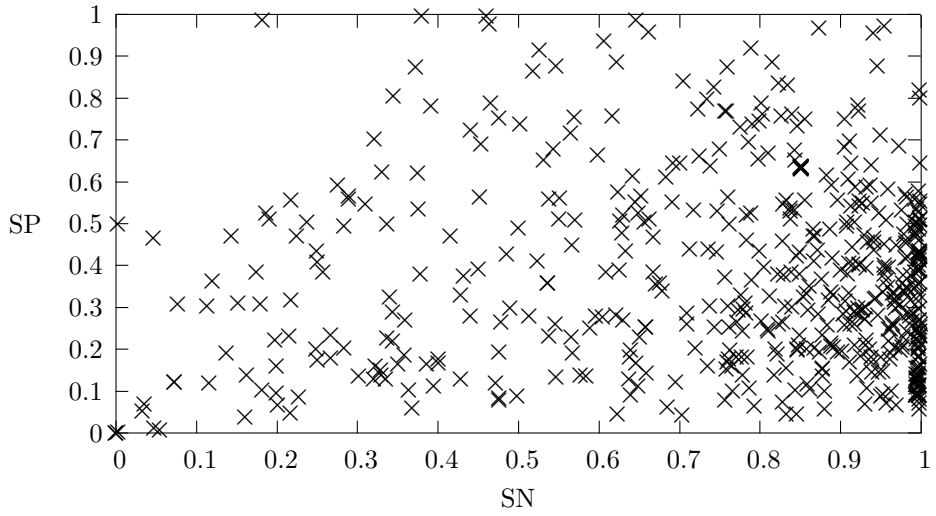
Tipo	SE	ES	CC
Cadena-1	0.748	0.343	0.269
Cadena-L	0.748	0.343	0.269
Cadena-2	0.760	0.396	0.338
Cadena-3	0.700	0.435	0.372
Cadena-4	0.507	0.484	0.349
Cadena-5	0.129	0.469	0.172
Markov-E	0.754	0.378	0.319

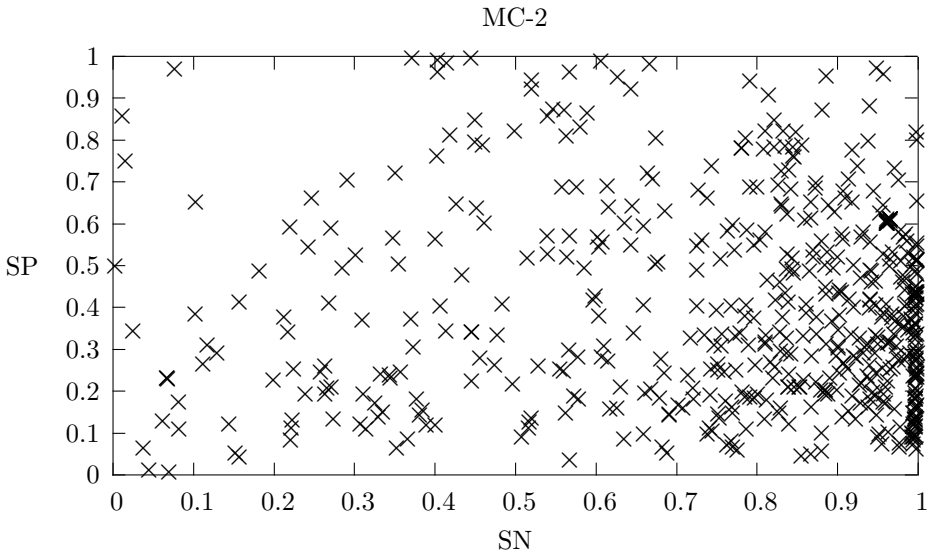
Los resultados masivos corroboran lo expuesto en la subsección anterior, aunque ahora se puede destacar que, a partir de la cadena de Markov 3, se detecta un decrecimiento en la sensitividad junto con un aumento de la especificidad. La calidad neta de la predicción, sin embargo, empeora con los órdenes mayores y confirma la tendencia expuesta en la sección anterior.

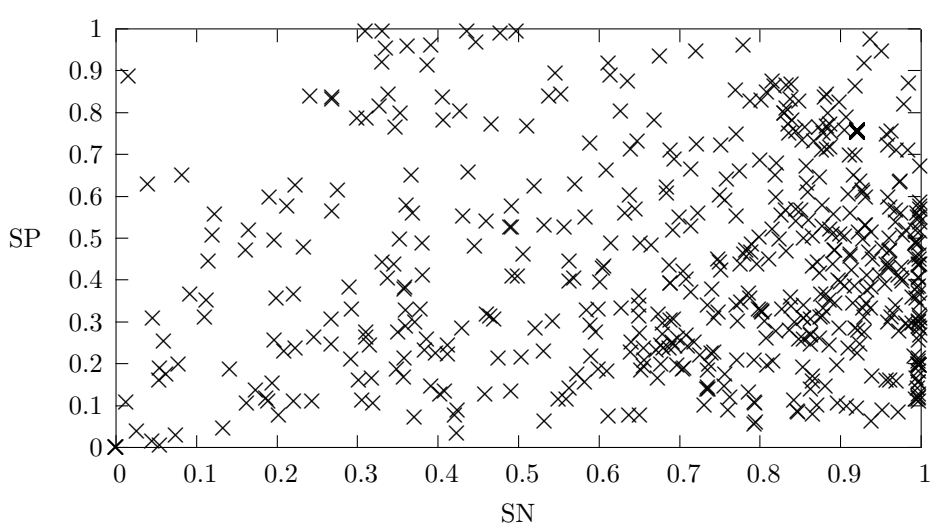
Mostramos a continuación algunos diagramas de dispersión en donde se ilustra la relación entre especificidad y sensitividad de cada secuencia con un punto.

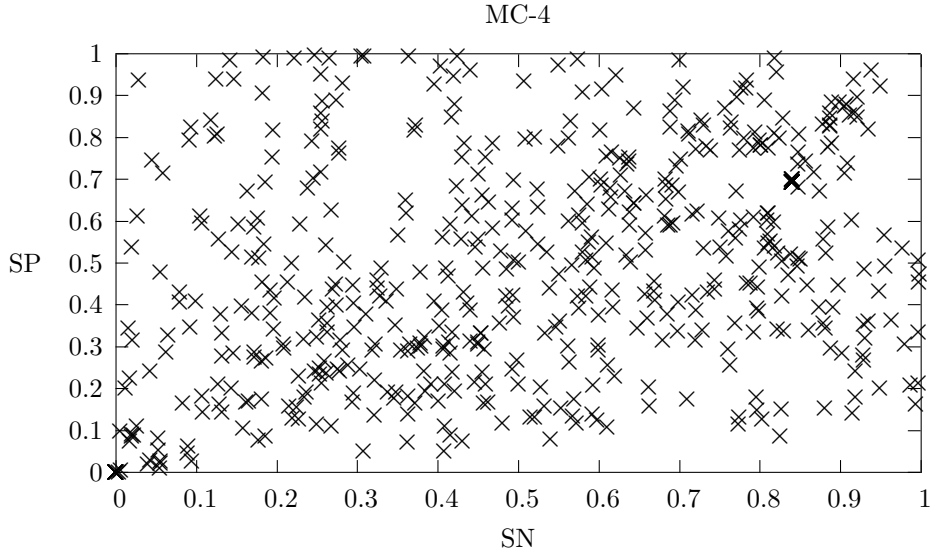


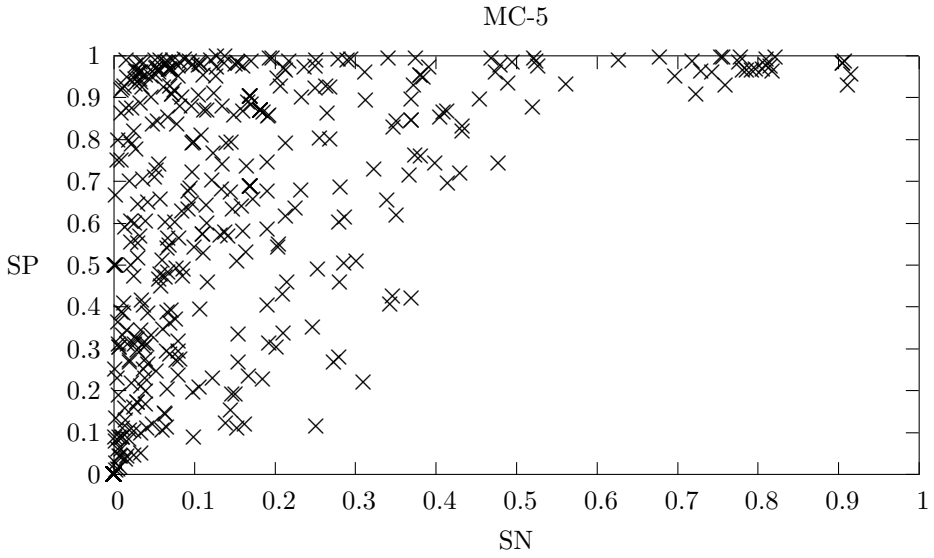

MC-Left
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Espacial
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


MC-2
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


MC-3
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


MC-4
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


MC-5
SP
SN
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Los diagramas confirman la tendencia de los resultados exhibidos antes, un incremento de la especificidad a costa de la sensitividad.

5.4.3. Variaciones: conjunto de entrenamiento

Procedemos a estudiar el efecto general del conjunto de entrenamiento sobre las distintas texturas. Para eso, se repite una prueba parecida a la de la sección anterior pero variando el conjunto de entrenamiento. La idea es estudiar el desempeño de cada tipo de textura ante distintas situaciones, es de esperar que el desempeño varíe de distinta manera para cada una de ellas.

Las pruebas se efectuarán sobre un conjunto del Guigo Data set de 40 secuencias.

Resultados con un set de entrenamiento de 5 secuencias.

Tipo	SE	ES	CC
Cadena-1	0.750	0.360	0.316
Cadena-L	0.932	0.239	0.231
Cadena-2	0.718	0.417	0.378
Cadena-3	0.448	0.497	0.339
Cadena-4	0.053	0.427	0.197
Cadena-5	0.104	0.542	0.204
Markov-E	0.712	0.394	0.348

Resultados con un set de entrenamiento de 10 secuencias.

Tipo	SE	ES	CC
Cadena-1	0.781	0.352	0.321
Cadena-L	0.940	0.245	0.262
Cadena-2	0.777	0.409	0.387
Cadena-3	0.661	0.464	0.399
Cadena-4	0.189	0.493	0.229
Cadena-5	0.076	0.589	0.218
Markov-E	0.769	0.392	0.359

Resultados con un set de entrenamiento de 20 secuencias.

Tipo	SE	ES	CC
Cadena-1	0.668	0.395	0.326
Cadena-L	0.882	0.287	0.307
Cadena-2	0.695	0.449	0.385
Cadena-3	0.665	0.482	0.417
Cadena-4	0.477	0.508	0.359
Cadena-5	0.167	0.586	0.270
Markov-E	0.681	0.426	0.351

Con conjuntos de entrenamiento reducidos se evidencia una fuerte disminución en la calidad de las predicciones, denotando que el entrenamiento es un factor importante. Cabe destacar que este afectó bastante más a las texturas más complejas, reduciendo la distancia entre las cadenas de primer orden y las demás. Es razonable de esperar considerando que al tener más parámetros se espera un entrenamiento más intensivo para poder otorgar valores adecuados para los mismos. En ese sentido, las texturas tienden a asemejarse cuanto menor es el entrenamiento. El otro aspecto que se puede destacar es que un entrenamiento más intensivo tiende a mejorar el desempeño de las texturas más especializadas.

Se sigue destacando la posición del proceso espacial como estando en el medio entre la cadena de Markov de segundo orden y la de primer orden. Parece entonces que existe mayor correlación hacia un lado que en ambos sentidos.

6. Refinamiento del método: ventaneado con clustering

6.1. Limitaciones del método de ventaneado

El método de ventaneado expresado en el capítulo anterior funcionaba en general cuando el tamaño de las regiones era lo suficientemente grande como para poder establecer diferencias importantes entre las texturas en juego y, por lo tanto, disminuir la dependencia. Los problemas que provienen de utilizar tamaños muy grandes de ventana fueron discutidos antes y tienen que ver con el hecho de que se evidencia un efecto de "mezcla" en los bordes. Esto se vuelve particularmente importante cuando se consideran texturas pequeñas, en donde los bordes dominan en algún sentido todo el problema. Para reducir este efecto una solución es reducir el tamaño de la ventana, reintroduciendo un nuevo problema que es la disminución de la significancia estadística y el incremento de la dependencia. En la práctica, esto termina resultando en texturados con una fuerte presencia de regiones "intrusas". Eso es, texturizaremos regiones en donde aparecerán pequeñas (o no tan pequeñas!) "intrusas" islas clasificadas como provenientes de otra textura.

Queremos penalizar de alguna manera aquellos texturados que generen este tipo de soluciones, en especial si uno asume que las regiones suelen ser continuas y no se verán interrumpidas por islas intrusas en su interior. Claro está que esto es una suposición adicional que se debe introducir al problema y será preciso evaluar la factibilidad de la misma en una situación particular. Bien puede ser que estas islas sean lo esperado para algún caso especial.

Fundamentalmente, lo que se quiere es tomar el modelo anterior de ventaneado, definido por una función de energía y un esquema de minimización, y adaptarlo de forma tal de poder introducir estos nuevos parámetros en el mismo. Se puede pensar a esto como una información "a priori" que tenemos de nuestro modelo, enunciada informalmente de la siguiente manera:

"Las regiones texturadas suelen ser grandes y no se verán interrumpidas por subregiones provenientes de otra textura."

o bien
"La interpenetración de texturas es baja."

Es preciso introducir un nuevo mecanismo de inferencia que permita considerar estos datos adicionales en el modelo, algo que el esquema de máxima verosimilitud no hace. Haremos uso de algunos conceptos de estadística Bayesiana, detallada brevemente en la próxima sección.

6.2. Principios de estimación Bayesiana

El modelo de Bayes permite la introducción de información adicional sobre el modelo en cuestión que va más allá de los datos utilizados para la inferencia. En el procedimiento estándar por MV lo que se intenta es buscar dentro del espacio de parámetros aquel que maximice $P(\ D \mid \theta\ )$, siendo D el conjunto de datos y θ una asignación de parámetros. En el procedimiento Bayesiano se hace precisamente lo inverso, se considera $P(\ \theta \mid D\ )$ y es eso lo que, por lo menos en el modelo más clásico, se intenta maximizar.

Es posible calcular la probabilidad de un conjunto particular de parámetros θ dado una serie de datos utilizando el teorme de Bayes, obteniendo:

$$P(\ \theta \mid D\ ) = \frac{P(\ \theta\ )P(\ D \mid \theta\ )}{\sum_{\theta'} P(\ \theta'\ )P(\ D \mid \theta'\ )}.$$

La información "a priori" sobre el problema cae fundamentalmente en el término $P(\ \theta\ )$, en donde se expresa una distribución sobre los parámetros. Este término se lo suele denominar *el prior*. Es normal utilizar una distribución constante para todos los parámetros, aunque en la práctica esto tiene como resultado un proceso de estimación de máxima verosimilitud. Cuando no se tiene ningún tipo de información inicial sobre los parámetros, esta es la única opción razonable.

En algunos casos se dispone de información adicional sobre los parámetros, y dicha información la expresaremos en el término $P(\ \theta\ )$. Podremos, así, agregarle más fuerza a algún conjunto particular de parámetros incrementando su correspondiente valor de $P(\ \theta\ )$. Haremos esto cuando contamos con alguna evidencia ajena a los datos de estimación que nos permita inclinarnos por esa configuración de parámetros.

Hasta ahora lo único que se hizo es exponer una fórmula alternativa a la sugerida por el MV, pero no hemos mencionado como se efectúan las estimaciones. Un método comunmente utilizado es maximizar el valor de $P(\ \theta \mid D\ )$ en función de θ , y quedarnos con la configuración de parámetros que realice el máximo. Esta estimación se la denomina "maximización *a posteriori* " o, en función de sus siglas, estimación MAP. Cabe aclarar que la expresión

que figura en el denominador de la fórmula no depende del valor específico de θ , lo que nos permite obviar el mismo en el proceso de maximización. Así, la estimación MAP (simbólicamente θ^{MAP}) corresponde a maximizar la función de verosimilitud multiplicado por el prior. Si el prior es una función constante, luego la estimación MAP es equivalente a la estimación de máxima verosimilitud ($\theta^{MAP} = \theta^{MV}$).

6.3. Aplicación Bayesiana al modelo de texturado

Aplicaremos el modelo de estimación MAP en nuestro trabajo de texturado. Para eso, recordamos el papel de cada uno de los parámetros de la formulación de estimación y su correspondiente par en el problema de texturado.

- D es el conjunto de datos que se usarán para la estimación. En este caso, D será la secuencia que queremos texturar.
- θ es el conjunto de parámetros que se desea calcular. En este caso será una función de asignación entra cada posición de la secuencia y el conjunto de texturas que se considerarán.

Se empieza con un conjunto de texturas $T_1, \ldots, T_n$, una secuencia $y_1, \ldots, y_m$ (que será nuestro D) y se fija un tamaño de ventana r. El θ será una secuencia $l_1, \ldots, l_m$ de elementos pertenecientes al conjunto $\{ 1, \ldots n\}$, en donde se especifica en cada posición la textura que se asignará.

El valor $P(\ D \mid \theta\ )$ es igual al utilizado en el capítulo anterior, eso es:

$$P(y_1, \ldots, y_m|\ l_1, \ldots, l_m) = \prod_{i=1,\ldots,m} P(\ y_{i-r}, \ldots, y_i, \ldots, y_{i+r}|l_i).$$

O lo que es lo mismo, para utilizar nuestra definición de textura:

$$P(y_1, \ldots, y_m|\ l_1, \ldots, l_m) = \prod_{i=1,\ldots,m} \pi_r(T_i(\ y_{i-r}, \ldots, y_i, \ldots, y_{i+r})).$$

Nota: acercándonos a los bordes las ventanas empezarán a irse más allá de la secuencia. Tal como se hizo antes, se ajusta el tamaño r de la ventana en esos puntos de forma tal de que la misma siempre quepe dentro de la secuencia.

Hasta ahora, todo sigue igual que antes. Es en el término prior en donde se evidenciarán los cambios. Se postulará un término que nos permita penalizar

situaciones de interpenetración de texturas para así poder reducir los casos en que en una ventana entran en juego diversas texturas diferentes. Esto se expresará en un término que involucre un costo a los cambios de textura.

Dada una posición x, definimos $\mathcal{N}_r(x)$ como la ventana de índices de centro x y radio r intersecado con la secuencia $y_1, \ldots, y_m$. Es preciso pedir esto último para asegurarnos de caer siempre dentro de la secuencia original.

Dado una posición p de la secuencia y una ventana r, se postula para la probabilidad $P(\{l_q\}\ , q \in \mathcal{N}_r(p))$ una distribución de Gibbs con una energía que depende del número de texturas distintas en el entorno $\mathcal{N}_r(p)$

$$P(\{l_q\}\ , q \in \mathcal{N}_r(p)) = \frac{e^{-U_\beta(p)}}{Z}$$

con

$$U_\beta(p) = -\beta \sum_{q \in \mathcal{N}_r(p)} \delta_{l_q, l_p}$$

en donde β es un número positivo y $\delta_{l,l'}$ es la delta de Kroneker. El denominador Z es una constante de normalización que no es preciso calcular a los fines de la maximización. $U_\beta(p)$ será la *energía de clustering* de la configuración en la posición p.

Puede parecer en un principio algo arbitraria esta elección y en cierta medida esto es así. Se escogió esta expresión puesto que cuando hagamos el planteo energético esto quedará definido de una forma bastante simple y en donde queda bien clara la idea de la penalización por cambios de textura. Nótese que la probabilidad aumenta a medida que la cantidad de sitios que compartan una misma textura sea mayor. Cuanto más grande es el término β, mayor será la penalización debido a cambios de textura. Valores grandes de β tienden a disminuir las interpenetraciones entre las texturas, tendiendo a juntar y homogeneizar las regiones de la secuencia de forma tal que respondan a una única textura. Por este motivo el coeficiente β se lo denominará de "clustering".

Introduciremos la distribución de Gibbs para los cálculos original de verosimilitud. Cabe aclarar que no hay cambios en el valor final del mismo salvo denominadores constantes que no afectan el proceso de maximización. La idea es disponer de una notación unificada para todos los términos que nos ayude a mantener la convención de energía que se fue utilizando durante el trabajo. Las distribuciones de Gibbs se prestan naturalmente para estas tareas.

Planteamos entonces que

$$P(\{\ y_q\}\, q \in \mathcal{N}_r(p)|\ l_p\ ) = e^{-U_Y(p)}$$

con

$$U_Y(p) = -\ln(P(\{\ y_q\}\ , q \in \mathcal{N}_r(p)|\ l_p\ ).$$

La fórmula no es cerrada, pues no es la intención de este planteo redefinir la probabilidad, que es algo propio de la textura. Simplemente se intenta destacar la presencia de un factor de energía, tal como aparecía en el planteo de clustering. $U_Y(p)$ será la *energía interna* de la configuración en la posición p.

Podemos entonces aplicar la regla de Bayes y tenemos que

$$P(\mathbf{L}_p|\ \mathbf{Y}_p\ ) = \frac{P(\mathbf{L}_p)P(\mathbf{Y}_p|\ l_p\ )}{Z'}.$$

con

$$\mathbf{L}_p = \{\ l_q\}\ , q \in \mathcal{N}_r(p)$$

y

$$\mathbf{Y}_p = \{\ y_q\}\ , q \in \mathcal{N}_r(p)\ .$$

Reemplazando cada término por lo expresado antes, obtenemos que

$$P(\mathbf{L}_p|\ \mathbf{Y}_p\ ) = \frac{e^{-U_{tot}(p)}}{Z'}$$

donde la energía total $U_{tot}(p) = \ U_Y(p) + U_\beta(p)$. Eso es, la energía total será la suma de la energía de clustering y la energía interna.

En definitiva, esa será la energía que vamos a querer minimizar en cada posición de la secuencia. Reexpresado, eso es

$$U_{tot}(p) = \beta \sum_{q \in \mathcal{N}_r(p)} \delta_{l_q, l_p} - \ln(P(\mathbf{Y}_p|\ l_p\ )).$$

Queda claro entonces que el clustering efectivamente contribuye a penalizar situaciones en donde exista mezcla de texturas.

Para el caso en que se tiene texturas de tipo Markov temporal, la fórmula sería:

$U_{tot}(p) = \beta \sum_{q \in \mathcal{N}_r(p)} \delta_{l_q,l_p} - \ln(i(y_{p-n})t(y_{p-n}y_{p-n+1}) \dots t(y_{p+n-1}y_{p+n}))$.

siendo $i(x)$ la probabilidad inicial y $t(xy)$ la probabilidad de transición.

Para los procesos espaciales de Markov, nos queda:

$U_{tot}(p) = \beta \sum_{q \in \mathcal{N}_r(p)} \delta_{l_q,l_p} - ln(i(y_{p-n})t(y_{p-n}y_{p-n+2} -> y_{p-n+1}) \dots t(y_{p+n-2}y_{p+n} -> y_{p+n-1})d(y_{p+n})$.

siendo $i(x)$ la probabilidad izquierda, $d(x)$ la probabilidad derecha y $t(xy -> z)$ la probabilidad de transición.

Tenemos entonces definida una energía local a la posición p. Podemos definir una energía que abarque el total de la secuencia considerando la suma de las energías locales.

$\mathbf{U}(\ y_1, \dots, y_m\ ) = \sum_{i=1,\dots,m} U_{tot}(i)$.

Cumplimos así uno de los requisitos necesarios para definir un método de texurado: ya tenemos una función de energía que se desea minimizar. Lo que nos falta ahora es armar un esquema de minimización funcional para la misma, tema que se trata en la próxima sección.

6.4. Expresión energética y esquemas de optimización

Como se ha mencionado, la expresión energética que nos ha surgido incurre en un problema de optimización combinatoria, teniendo que recurrir a algún tipo de heurística.

Tenemos entonces por un lado el hecho de se debe identificar un máximo de la función pero por el otro se tiene una cuestión más de fondo que afecta directamente a la función de energía antes que al método utilizado para maximizar la misma: ¿es el máximo de esa función la solución verdadera? Eso es, si la función de energía no es correcta luego no tendría que haber correlato alguno entre su máximo y la solución verdadera.

Tenemos entonces dos posibles "anomalías" con las que lidiar. Una solución que no llegue hasta el mínimo de la función de energía se denominará un "subóptimo". En este caso, lo que ocurre es que el esquema de minimización demostró ser insuficiente. Una forma de verificar si una solución es subóptima es comparándola con la energía de la solución verdadera.

Una solución cuya energía sea menor que la energía de la solución verdadero se denominará un "superóptimo". En este caso, la falta no estaría en el mecanismo de minimización sino en la función de energía. Para el caso del ventaneado con clustering, esto tal vez sea indicativo de una corrección en el factor de clustering.

Por supuesto, todos estos hechos son verificables cuando se dispone de la solución verdadera. En la práctica esta no lo sabremos pues precisamente el cometido de esto es encontrarla. Sin embargo, puede servir como una etapa de entrenamiento practicar el método con secuencias ya conocidas de forma tal de definir una buene serie de parámetros para la energía.

6.5. Experimentos

Se muestra a continuación los resultados de haber aplicado este método para texturar algunas secuencias. Se considerará un gen (BRHOX22) y sobre el mismo se ensayará un proceso de reconocimiento con distintos valores de clustering. Para cada uno de ellos se mostrará los resultados en función de los medidores definidos en el capítulo anterior (sensitividad, especificidad y coeficiente de correlación) y un valor de energía en comparación con la energía de la solución verdadera. Esto nos permitirá determinar la naturaleza de la solución (subóptimo o superóptimo).

Se estudiará con un mecanismo simple de optimización, en este caso una búsqueda local.

Clustering	0.0	0.01	0.05	0.1	0.2
Energía S.V.	-287095	-287132	-287281	-287467	-287839
Energía S.O.	-286072	-286173	-286420	-286788	-288461
Iteraciones	1740	1800	1800	1950	1680
SN	0.689	0.658	0.703	0.808	—
SP	0.696	0.715	0.763	0.787	—
CC	0.616	0.612	0.670	0.746	—

En el caso de clustering 0.2, se obtuvo una solución en donde todas las posiciones se clasificaban como no codificantes (de ahí que no tiene sentido marcar sus clasificaciones). Llega un punto en donde el nivel de clustering conlleva a generar soluciones en donde todas las posiciones se clasifican con la misma provenientes, después de todo estas soluciones maximizan la contribución energética del clustering.

Los valores energéticos obtenidos no se diferencias mucho del valor de la solución verdadera, indicando que no es probable que se tenga problemas de

superóptimos o subóptimos.

Se destaca los resultados en general positivos del clustering. Las predicciones tienden a mejorar hasta que se llega a un punto de exceso, en el cual muy rápidamente se cae en una solución uniforme.

6.6. Comentarios sobre el clustering

Es interesante destacar los resultados positivos esta nueva energía. Sin embargo, es prudente aclarar que la misma además adolece de las siguientes fallas:

- A diferencia del método MV de ventaneado, esta función de energía no es trivialmente minimizable, es preciso un mecanismo de optimización.
- Es extremadamente frágil. Los resultados mejoran a los obtenidos mediante una aplicación del MV pero únicamente con un rango especial de valores de clustering. En efecto, pasarse apenas del valor óptimo implica en general caer sobre una solución trivial(todas las posiciones clasificadas con una única etiqueta).

Respecto del primer punto se puede mencionar que las experiencias efectuadas en general resultaron positivas pese a que se utilizó un esquema rudimentario de optimización, basado en una búsqueda local. En efecto, los resultados energéticos típicamente eran mejores que los de la solución verdadera. Habría que ver hasta que punto esto es así y, en caso contrario, habría que determinar un mecanismo adecuado de optimización.

El segundo punto es probablemente el más delicado y el que mayores problemas amontona sobre el modelo energético bayesiano. Es importante que el rango de clustering adecuado no varíe mucho entre distintas secuencias.

Por otro lado, sería bueno estudiar la diferencia que existe entre la utilización del clustering con respecto a un MV con ventanas grandes. En efecto, hemos demostrado que las ventanas amplias tendían a homogeneizar en algún sentido la clasificación sobre toda la secuencia, un efecto que también se aprecia, aunque a veces un tanto más brusco, con el clustering. ¿Será posible reducir todo problema de clustering a uno de MV de ventaneado? Si bien no es posible asegurarlo pensamos que la respuesta es negativa, el modelo de clustering es, por definición, más complejo que el de MV. Incorpora en su formulación un parámetro nuevo que implica una nueva estrategia de

resolución. En algún sentido, el clustering enriquece al modelo de MV, los parámetros del modelo MV se siguen conservando todavía.

7. Conclusiones y posibilidades futuras

7.1. Resumen del trabajo

Hemos definido un marco general de trabajo con la intención de comprender aspectos puramente estadísticos en el reconocimiento de ADN. Dicho marco está compuesto fundamentalmente por las siguientes características:

- Textura: una textura representa un esquema de generación estocástica de secuencias. En el caso general tendremos una numerosa cantidad de texturas pero dentro del contexto de esta sección y en el estudio particular de una secuencia tendremos únicamente dos, la textura codificante o exónica y la textura no codificante o intrónica.

- Energía: la función de energía es una medida de calidad de la asignación que nosotros efectuemos sobre una secuencia. Una parte importante del trabajo de desarrollar un reconocedor de genes está en la definición de una función de energía que sea útil y práctica, tanto a nivel de efectividad como en performance.

- Esquema de Minimización: dentro de esta categoría involucramos cualquier heurística de minimización funcional que se utilizará para poder minimizar la función de energía.

Tenemos entonces definido una "triada" que, en su conjunto, nos terminan armando un sistema de reconocimiento. Puede ser especialmente útil pensar a estos tres conceptos por separado. Eso es, cada parte se define con aquel formalismo o metodología que mejor le parezca, independientemente del resto. Eliminamos así un acoplamiento fuerte en nuestro sistema, quedará este partido en tres subsistemas todos relativamente independientes entre sí. Las ventajas de esto son claras, podemos así apuntar al diseño e implementación en paralelo. Además, al alterar alguna de estas partes no tendríamos que tocar el resto de los componentes.

La desventaja de esta visión es que a veces puede llegar a incurrir en costos de performance, por ejemplo si la función de energía es relativamente simple, como es el caso del ventaneado simple MV, no tendrá mucho sentido utilizar una heurística complicada de minimización. Es por eso que esta separación tal vez deba ser vista más como una forma de ordenar los conceptos antes que un verdadero patrón de diseño.

Esta división ha sido puesta en práctica en el programa desarrollado para todos los ejemplos anteriores y se destaca en los mismos el potencial de reusabilidad producto de esta separación del problema. Por ejemplo, los módulos de texturas han podido ser utilizados en todos los ejemplos sin necesidad de incurrir en cambios o adaptaciones de los mismos.

A fin de dotar de un caracter formal al desarrollo se ha llevado a cabo un basamento teórico sobre los conceptos antes mencionados, enfocando principalmente el tema de la textura. Se exhibieron distintos tipos de texturas y dos funciones de energía, los cuales demostraron su aplicabilidad en el reconocimiento estadístico.

Este último punto fue evaluado mediante una prueba estricta que incluía el reconocimiento de un número importante de secuencias. El paquete de prueba es un set normalmente utilizado para la evaluación de métodos de reconocimiento de secuencias. Los resultados obtenidos no descartan un estudio más profundo sobre esta alternativa de estudio.

Cabe aclarar que el modelo desarrollado es efectivo en la medida de que se lo entrene con datos efectivos. Las experiencias han demostrado la imprudencia que significa utilizar texturas complicadas si no se cuentan con los datos adecuados como para sustentarlas.

7.2. Alcance del modelo: limitaciones

Hacemos destacar una limitante de este modelo y es que, por ahora, su aplicabilidad se restringe a la detección por métodos basados en el contenido, eso es, métodos estadísticos. Para poder terminar de armar un buen proceso de detección es preciso definir con precisión los bordes entre las regiones y es difícil que esto se consiga exclusivamente por métodos estadísticos puesto que los bordes se suelen identificar por patrones muy específicos de símbolos.

Por ese motivo se debería incorporar en alguna forma métodos basados en la detección por señales. Eso quiere decir que será preciso identificar posiciones en donde haya, por ejemplo, codones de inicio, "splice junctions", codones de parada, promotores, etc. Disponiendo datos de contenido y de señales es factible la reconstrucción del gen. El primero de ellos marcaría a grandes rasgos la presencia de potenciales zonas codificantes y el segundo terminaría de marcar bien los bordes.

Habría que ver como combinar un modelo de detección de señales junto con el presentado para terminar de conformar un predictor genético.

7.3. Nuevas líneas de investigación

Varios aspectos han sido dejados sin un desarrollo detallado lo cual abre nuevos caminos de investigación.

Por el lado de los formalismos y la construcción teórica sería interesante seguir estudiando las propiedades topológicas del espacio de las texturas, partiendo como base de la distancia que se definió y se demostró como tal en el capítulo 3.

Por el lado del clustering quedaría pendiente resolver los temas planteados al final del capítulo, que enumeramos someramente a continuación:

- La complejidad de resolución
- La determinación de los valores de clustering
- La factibilidad de simular el clustering con ventanas grandes

Si se desea un mecanismo efectivo de reconocimiento es preciso salvar las limitaciones planteadas en la sección anterior. Fundamentalmente es preciso ver como integrar aspectos propios del reconocimiento de señales en el modelo aquí planteado. No creemos que sea prudente definir las señales como texturas, estas últimas tienen una función principalmente estadística. Pensamos que es mejor que este método se combine en algún sentido con un esquema de detección de secuencias y de esa forma, juntos pero también separados, resuelva cada uno las falencias del otro.

La naturaleza, si se quiere, "enciclopedista" del reconocimiento de secuencias requiere seguramente que, además de lo expuesto antes, el esquema incorpore de alguna manera de un una vía de integración y comparación con datos provenientes de los repositorios genéticos actualmente en uso, como los proveen NCBI y otros organismos.

8. Agradecimientos

Agradezco al doctor Roberto Perazzo por haberme sugerido este problema y haberme guiado en su desarrollo y a Sergio Lew por valiosas discusiones. A Norma Paniego del INTA por habernos provisto parte del material utilizado en esta tesis.

Referencias

[BAX01] Baxevanis A., Ouellette F.: *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Second Edition*, Wiley, 2001

[BAY95] Bayley T.: *Discovering motfs ind DNA and protein sequences, the approximate common substring problem*, Thesis Ucal San Diego, 1995

[BAY94] Bayley T., Elkan C.: *Fitting a texture model by expectation maximization to discover motifs in biopolymers*, Proc. Second Int. Conference on Int. Systems for Mol. Biology pp 28-36, 1994

[BOR93] Borodovsky M., McIninch J: *GeneMark: parallel gene recognition for both DNA strands*, Computers & Chemistry Vol. 17 No. 19 pp. 123-133, 1993

[CRO83] Cross G., Jain A.: *Markov Random Field Texture Models*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol PAMI-5 Nbr 1, 1983

[DUR98] Durbin R., Eddy S., Krogh A., Mitchison G.: *Biological sequence analysis*, Cambridge University Press, 1998

[GUI99] Guigó R.: *DNA Composition, Codon Usage and Exon Prediction*, Genetic Databases 53-80, Academic Press, 1999

[GUI96] Guigó R., Burset M.: *Evaluation of Gene Structure Prediction Programs*, Genomics 34:353-357, 1996

[HAM02] Hampson S., Kibler D., Baldi P.: *Distribution patterns of over-represented k-mers in non-coding yeast DNA*, Bioinformatics Vol 18 no. 4 pp 513-529, 2002

[LAN00] Lanctot K., Ming Li, En-Hui Yang: *Estimatig DNA sequence entropy*, Symposium on Discrete Algorithms 409-418, 2000

[LIU01] Liu X. , Brutlag D., Liu J.: *BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes*, Proc. Pac. Symposium Biocomputing 127-38, 2001

[KIN80] Kindermann R., Snell L.: *Markov Random Fields and Their Applications*, American Mathematical Society, 1980

[MOU01] Mount D.: *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, 2001

[RAB89] Rabiner. L. R. : *A tutorial on HMM and selected applications in speech recognition*, Proc. IEEE, Vol. 77, No. 2, pp. 257-286, 1989

[SAL98] Salzberg S., Delcher A., Kasif S., White O.: *Microbial gene identification using interpolated Markov models*, Nucleic Acids Research 26:2 pp 544-548, 1998

[SNY93] Snyder E., Stormo G.: *Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks*, Nucleic Acids Research Vol 21 No 3, 1993

[ZHA97] Zhang M. Q.: *Identification of protein coding regions in the human genome by quadratic discriminant analysis*, Analysis. Proc. Natl. Acad. Sci.USA 94:565-568, 1997