Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Análisis del algoritmo MEME para la búsqueda de motivos en cadenas de ADN

Alumno: Salvador Gil López

Tutor: Fernando Díaz Gómez

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Identificación del Trabajo de Fin de Grado

Este trabajo, de título "Análisis del algoritmo MEME para la búsqueda de motivos en cadenas de ADN", ha sido realizado por Salvador Gil López, como Trabajo de Fin de Grado para el grado de Informática de Servicios y Aplicaciones de la Universidad de Valladolid.

1.2. Objetivos del Trabajo del Fin de Grado

Los objetivos básicos de este trabajo son dos:

1 Introducción a la búsqueda de motivos

Con el primer objetivo se pretende definir qué se entiende por un motivo dentro de una secuencia de ADN, justificar el interés que tiene su análisis de la información genética y, presentar una revisión del estado del arte en lo que se refiere a los algoritmos de búsqueda de motivos existentes.

2 Análisis del algoritmo MEME

El segundo objetivo, una vez introducido y motivado el problema de la búsqueda de motivos, se refiere a presentar y detallar el funcionamiento de uno de los algoritmos más competitivos y de uso extendido, como es el algoritmo *Multiple EM for Motif Elicitation* (MEME).

2.1 Ejemplo de ejecución

Dentro del segundo objetivo se encuadra la presentación del paquete software que implementa el algoritmo MEME, aportándose un breve manual de instalación y uso, junto con una serie de ejemplos de ejecución sobre un conjunto de datos real, con el fin de ilustrar y presentar, de forma aplicada, los conceptos teóricos.

1.3. Introducción

Desde la invención de los ordenadores, la idea de aplicar la capacidad de computación de estos a la investigación y a la resolución de problemas ha sido uno de los factores que más ha impulsado su desarrollo. Encontramos los primeros ejemplos de este uso durante la Segunda Guerra Mundial, donde los aliados dedicaron un gran número de recursos para crear ordenadores que les ayudasen a descifrar los mensajes en clave de los alemanes.

El ordenador Colossus se considera el primer ejemplo de ordenador electrónico, cuyo único propósito fue el de acelerar los cálculos para descifrar los mensajes entre Hitler y sus generales del alto mando alemán con sus unidades militares.

Desde el Colossus, –construido en los años 40, que carecía memoria interna, se programaba mediante enchufes e interruptores y funcionaba mediante válvulas de vacío–, hasta nuestros días, la potencia de computación ha crecido exponencialmente. Es muy conocida la Ley de Moore, enunciada por Gordon Moore en 1965, que afirma que la densidad de los transistores en los microchips iba duplicarse cada dos años.

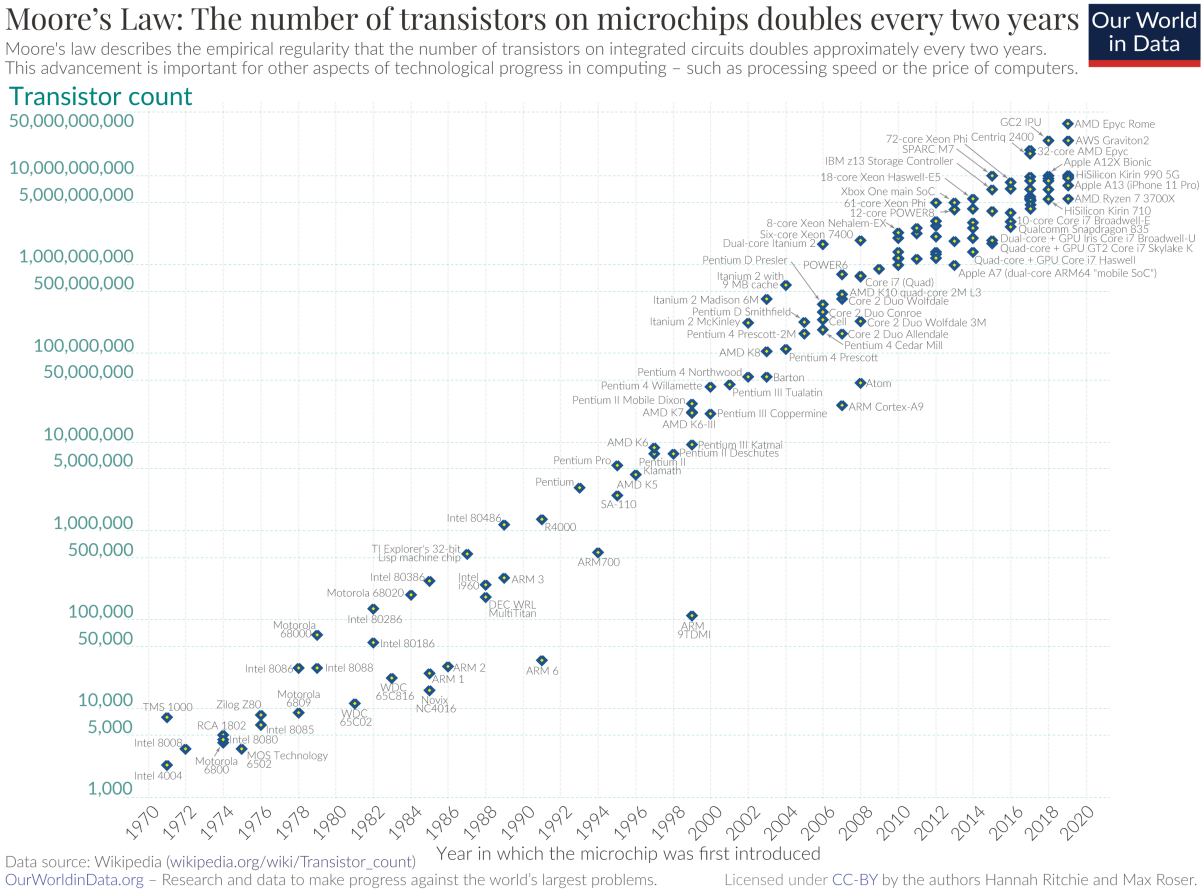


Figura 1.1: Ley de Moore

Este crecimiento exponencial de la capacidad de computación, va de la mano con la aparición de nuevos campos y disciplinas, tanto en la investigación como en la industria, a los que aplicar estas nuevas herramientas computacionales. A veces incluso, como ya ocurrió con la máquina Colossus, es la intención de resolver un determinado problema, la que ha impulsado la creación de mejores y más sofisticados ordenadores. En otras ocasiones, con la mejora de las capacidades de computación se han descubierto nuevas aplicaciones.

Uno de estos campos, que ha tomado una gran relevancia en los últimos años debido, entre otras causas a la gran importancia que la salud tiene en las sociedades actuales, es la bioinformática.

La bioinformática es un área de investigación interdisciplinar entre la biología y las ciencias computacionales que se encarga de almacenar, adquirir, manipular y distribuir, la información biológica que, en gran parte se corresponde con secuencias de Ácido desoxirribonucleico (ADN) y aminoácidos.

Esta disciplina nace hace más de 50 años, a comienzos de 1960, con la aplicación de métodos computacionales al análisis de la secuencia de proteínas. Hoy en día, el concepto de bioinformática es muy diferente gracias a los avances computacionales, los volúmenes de datos que se pueden procesar han aumentado enormemente y por tanto los problemas que se pueden abordar son también muy diferentes a los de entonces.

En los últimos años, ha cobrado relevancia en este ámbito la búsqueda de motivos. Patrones probabilísticos que se repiten a lo largo de una o varias secuencias, la repetición y la conservación de estos patrones hace conjeturar que llevan asociada una función biológica de algún tipo. De momento, este problema no tiene una solución exacta y eficiente y, a lo largo de los años, han surgido diversos algoritmos con estrategias muy diferentes que buscan resolverlo.

En este trabajo se busca proporcionar una explicación detallada de uno de los primeros algoritmos desarrollado con este fin, el algoritmo *Multiple EM for Motif Elicitation* (MEME). La importancia de MEME es significativa, no sólo por ser uno de los primeros en aparecer, sino porque ha dado lugar a muchos otros algoritmos, basados en los principios de este, que mejoran y amplían sus capacidades.

Varios de estos algoritmos derivados de MEME con sus diferentes usos y mejoras, todos ellos enfocados en la búsqueda de motivos, han sido recopilados en una herramienta en línea, conocida como la Suite MEME [12], que es pública y que, a día de hoy, sigue recibiendo actualizaciones.

1.3.1. Motivación

Con el auge de la investigación genómica de los últimos años, e impulsado gracias al esfuerzo de entidades públicas y privadas y que se materializa en logros como la secuenciación del ADN humano perseguido por el *Proyecto Genoma Humano*, aparecen nuevos enfoques para la medicina. Aparece la medicina personalizada, en la que se utiliza el perfil genético del individuo para guiar las decisiones tomadas en relación con la prevención, diagnóstico y tratamiento de la enfermedad. Ya no hay un tratamiento estándar, sino que podemos adaptar los tratamientos a los individuos, desde las dosis del medicamento hasta qué medicamento es más conveniente.

Entender la regulación de la expresión genética se convierte en uno de los nuevos retos. La regulación de la expresión genética determina cómo se expresan los genes, qué factores ambientales intervienen, por qué algunos genes se expresan mejor que otros, etc. El proceso es muy complejo y todavía quedan muchos años hasta poder entenderlo del todo, pero se hacen avances.

Es, en este momento, cuando se descubren los motivos, pequeños patrones probabilísticos que se repiten a lo largo de una o varias secuencias y que suelen ir asociados a cierta función biológica. Por ejemplo, a veces se encuentran marcando un lugar de unión para un factor de transcripción, indicando desde dónde debe empezar a transcribir el ADN en ARN mensajero. Otras aparecen a nivel del ARN participando en procesos como el procesado de ARN mensajero o marcando un lugar de unión para un ribosoma.

Desarrollar buenos algoritmos para descubrir motivos es de gran importancia porque permite reducir enormemente la búsqueda de estas secuencias repetidas, y es una estrategia de cribado computacional, que permite focalizar el descubrimiento de las funciones biológicas asociadas a estas.

1.4. Estructura de la memoria del Trabajo de Fin de Grado

Capítulo 1. Introducción

Primer capítulo donde se hará una introducción al tema de estudio y se expondrán la motivación del trabajo, así como, los objetivos de este.

Capítulo 2. Plan de proyecto

Estimación del coste que supondría llevar a cabo un estudio como este en un ámbito profesional y, una descripción del trabajo realizado, las fases de las que ha consistido y cuánto ha durado cada fase, además se realiza una comparación con respecto a la estimación del coste realizada *a priori*.

Capítulo 3. Estado del arte

Breve repaso al estado del arte de la búsqueda de motivos, un esquema de la forma general que toman los algoritmos y una clasificación de estos, atendiendo al enfoque con el que abordan la búsqueda.

Capítulo 4. Fundamentos y herramientas matemáticas

Exposición de la formulación matemática sobre la que se sustenta el problema de encontrar motivos, y una explicación de los diferentes métodos y herramientas matemáticas que usa el algoritmo MEME en la búsqueda.

Capítulo 5. El algoritmo MEME

En este capítulo se hará una explicación detallada del funcionamiento del algoritmo MEME, de las fases en las que se divide el proceso de búsqueda, de las estructuras de datos auxiliares que usa y cómo se aplican las herramientas matemáticas explicadas en el capítulo anterior.

Capítulo 6. Caso de estudio

Por último, se hará una breve guía de instalación y uso y se presentarán dos ejemplos de ejecución del algoritmo desde la Suite MEME, todo ello con el ánimo de complementar la exposición teórica del algoritmo.

Capítulo 7. Conclusiones y posibles ampliaciones

Se finalizará la memoria con las conclusiones extraídas, una vez realizado el trabajo, y posibles ampliaciones.

Capítulo 2

Plan de proyecto

En este capítulo se hará una estimación del coste de realizar un trabajo de investigación similar de forma profesional, teniendo en cuenta los costes salariales, los costes operativos, los costes de hardware y los costes software. Además, se hará una exposición de las fases en las que ha consistido el desarrollo de este trabajo, se detallará cuál es el trabajo realizado y cuánto ha durado cada una. Por último, se realiza una comparación entre el coste estimado *a priori* y el coste según el número de horas dado en el seguimiento de las fases del proyecto.

2.1. Estimación del coste

A continuación se presenta una estimación *a priori* del coste de realizar este trabajo. Esto no es un presupuesto al uso, debido a que no se conoce exactamente el alcance de este proyecto, sólo la materia, sobre la que hay que investigar para, luego, definir el alcance poco a poco. Para la estimación del coste se toman 360 horas de trabajo, que es lo que debe durar, aproximadamente, una asignatura de 12 ECTS como es el TFG, considerando que 25 horas $\leq$ 1 ECTS $\leq$ 30 horas, se ha tomado el límitesuperior. Se tienen en cuenta cuatro tipos de costes, costes salariales, costes operativos, costes de hardware y costes de software.

Costes salariales

Para los costes salariales se toma el trabajo como si fuese un trabajo de consultoría, dado un salario medio de un consultor en España, que es de $13,21$ euros la hora[1], los costes salariales debido a un solo consultor son de

$$360 \text{ horas } \cdot 13,21 \frac{€}{\text{hora}} = 4755{,}6 \text{ €}.$$

Costes operativos

Los costes operativos son los costes de la conexión a internet que se ha utilizado, por un lado, para acceder a los diferentes trabajos o repositorios donde buscar la información, y por otro, para mantener las reuniones con el tutor.

Dadas las 346 horas de trabajo, y que el coste de la tarifa contratada con el proveedor es de 120 euros al mes, aproximadamente 0,167 euros la hora, los costes operativos son de

$$360 \text{ horas } \cdot 0{,}167 \frac{€}{\text{hora}} = 60{,}12 \text{ €}.$$

[1] https://es.talent.com/salary?job=consultor

Costes de *hardware*

Los costes hardware son los costes de amortización del equipo que se ha utilizado. El equipo es un *Dell Vostro 5490*, con un coste de 900 euros, y asumiendo una vida útil de 5 años los costes de amortización del equipo durante la realización del trabajo son de

$$360 \text{ horas } \cdot \frac{900}{5 \cdot 365 \cdot 24} \frac{€}{\text{hora}} = 7{,}39 \text{ €}.$$

Costes de software

Las herramientas software utilizadas han sido *Visual Studio Code*, *Overleaf*, *Docker*, *Microsoft Teams* y un navegador ordinario, todas ellas gratuitas, por lo que los costes software son de 0 €.

En total el coste estimado es de

$$4755{,}60 \text{ €} + 60{,}12 \text{ €} + 7{,}39 \text{ €} + 0 \text{ €} = 4823{,}11 \text{ €}.$$

2.2. Fases de trabajo, seguimiento del trabajo realizado

2.2.1. Estudio del estado del arte

La primera fase ha consistido en un estudio a fondo del estado del arte de la búsqueda de motivos. Esto ha incluido un estudio de la historia de esta disciplina, cómo surge y con qué intenciones. También se lleva a cabo un estudio de los conceptos biológicos en los que se apoya, con el objetivo de poder comprender a fondo el contexto en el que se desarrolla. Aquí se incluye un estudio de qué es el material genético, cómo se regula su expresión y, la importancia médica y científica que tiene la búsqueda de motivos. Por último se lleva a cabo un estudio de los diferentes algoritmos que han ido surgiendo y que tratan de resolver el problema de la forma más óptima posible. El objetivo último de esta fase es encontrar un algoritmo representativo, que haya destacado por sus buenos resultados y que haya influido en el desarrollo posterior de otros algoritmos. El algoritmo elegido es MEME, presentado por el profesor Timothy L. Bailey en 1994 [13], que aún hoy sigue obteniendo buenos resultados, y, a partir del cual se han inspirado otros investigadores para crear algoritmos más eficientes como STEME [30] o EXTREME [31].

Esta fase comienza el 22 de noviembre de 2020 y acaba el 20 de marzo de 2021, lo que son aproximadamente 17 semanas, habiendo dedicado aproximadamente una media de 4 horas por semana a esta fase del trabajo, lo que da una duración total de

$$17 \text{ semanas } \cdot 4 \frac{\text{horas}}{\text{semana}} = 68 \text{ horas}.$$

2.2.2. Investigación sobre el algoritmo MEME

La segunda fase ha tratado de una investigación exhaustiva sobre el algoritmo MEME con el fin de entender el funcionamiento de este, desde lo puramente conceptual hasta el código, pasando por toda la justificación matemática que tiene detrás. Para ello se ha utilizado toda la bibliografía relacionada con el algoritmo que se ha considerado oportuna. Algunos de los documentos utilizados son los tres trabajos publicados por el profesor Bailey [13], [14] y [15]; el trabajo donde se presenta el algoritmo EM [16], que es la base matemática de MEME; una serie de trabajos que presentan varios algoritmos inspirados sobre los mismos principios como STEME [30] o EXTREME [31], que han sido mencionados en el apartado anterior; además del trabajo de presentación de la plataforma Suite MEME [12], una plataforma que junta diferentes algoritmos desarrollados en torno a MEME.

El principal problema que se ha encontrado a lo largo de esta etapa, es que el último trabajo sobre el algoritmo [15] fue publicado en el año 2010. Sin embargo, el algoritmo ha seguido recibiendo actualizaciones, lo que ha provocado que fuera necesario acceder al código fuente del algoritmo para entender cómo funciona en su versión más actualizada.

Esta fase comienza el 22 de marzo y dura hasta principios de junio, aproximadamente 12 semanas, habiendo dedicado una media de 8 horas por semana, lo que da una duración total de

$$12 \text{ semanas } \cdot 9\frac{\text{horas}}{\text{semana}} = 108 \text{ horas.}$$

2.2.3. Redacción del trabajo

La última fase del trabajo comprende toda su redacción, siempre con la idea de hacer una exposición lo más clara y llevadera posible, a cualquier persona que no esté familiarizada con el tema. Esta fase incluye la maquetación del trabajo, múltiples revisiones sobre la estructura y el contenido, ejecuciones de prueba para ilustrar el caso de estudio, etc.

Dentro de esta fase podemos distinguir otras dos bien diferenciadas. La primera, en la que se ha redactado la parte esencial del trabajo que corresponde al funcionamiento del algoritmo MEME, plasmada en el Capítulo 5, y los fundamentos matemáticos sobre los que se plantea, Capítulo 4.

La segunda fase dentro de la redacción corresponde al resto de capítulos del trabajo, estos tienen una intención más complementaria por lo que se han dejado para el final, se ha buscado priorizar la parte más teórica.

Esta fase se solapa con la anterior, se distinguen dos periodos de trabajo, el primero empieza en la última semana de abril y acaba en la penúltima semana de mayo, un periodo de 4 semanas, con cerca de 6 horas de trabajo a la semana, en total

$$4 \text{ semanas } \cdot 5\frac{\text{horas}}{\text{semana}} = 20 \text{ horas.}$$

El segundo periodo abarca desde la última semana de mayo hasta la penúltima de junio, aproximadamente 5 semanas, al haber acabado ya el curso lectivo se dedica mucho más tiempo al trabajo, de media cerca de 30 horas semanales, en total

$$5 \text{ semanas } \cdot 30\frac{\text{horas}}{\text{semana}} = 150 \text{ horas,}$$

es decir, esta fase ha durado aproximadamente 170 horas.

En los tiempos de duración registrados de las tres fases se añaden las reuniones mantenidas con el tutor, presenciales y no presenciales.

En total se han dedicado, aproximadamente, 346 horas al desarrollo de este trabajo.

2.3. Comparación del coste estimado *a priori* con el coste estimado tras el seguimiento

Tras haber realizado un seguimiento del trabajo se ha obtenido que el tiempo empleado en la realización de este ha sido de aproximadamente 346 horas, introduciendo esto en las ecuaciones de costes de la Sección 2.1 obtenemos un coste real aproximado de 4635,55€, una desviación del coste originalmente estimado de 120,05€ , es decir, de un 2,52 % .

Capítulo 3

Estado del arte

Como se ha comentado previamente, de la mano del aumento de las capacidades de computación surgen nuevas líneas de investigación en múltiples campos de la ciencia, una de ellas es la búsqueda de motivos. En este capítulo se hará una exposición de la estructura general que siguen los algoritmos de búsqueda de motivos y de los diferentes tipos de algoritmos que han ido apareciendo en la comunidad científica, así como de los principios teóricos sobre los que se sustentan. También se hará una breve exposición de las diferentes bases de datos de motivos que hay y para qué se usan.

3.1. Tipos de algoritmos de búsqueda de motivos

Podemos clasificar los algoritmos en cuatro grandes grupos dependiendo del tipo de enfoque que tenga la estrategia de búsqueda, estos grupos son, enfoque enumerativo, enfoque probabilístico, enfoque naturalista y enfoque híbrido.

3.1.1. Enfoque enumerativo

Los algoritmos que caen en esta categoría se caracterizan por buscar en todo el espacio de búsqueda, aquellas secuencias que aparecen repetidamente, con mínimos cambios en las bases nitrogenadas que las conforman. Este tipo de algoritmo tiende a encontrar un máximo global, sin embargo, el coste computacional es exponencial, por lo que sólo es rentable usarlos para buscar motivos cortos en conjuntos de datos no muy grandes.

A su vez se pueden distinguir varias clases.

Enumeración simple

La enumeración simple se basa en realizar una búsqueda exhaustiva de las cadenas que se repiten a lo largo de las secuencias y que presentan algunas bases nitrogenadas degeneradas. Es el método de enumeración más simple, algunos de los algoritmos basados en este método son YMF [17] y DREME [18].

Métodos basados en *clustering*

Este método fue propuesto para el algoritmo CisFinder [19] para detectar motivos cortos en cadenas muy largas (de hasta 50 Mb). La idea detrás de este método es realizar una búsqueda enumerativa de muchos motivos cortos y agruparles por por su similitud con el fin de conformar uno más general.

Métodos basados en árboles

Esté método utiliza árboles de sufijos para acelerar la búsqueda enumerativa. El árbol se utiliza para modelar los patrones que se encuentran en la búsqueda enumerativa y dar pesos a aquellos que aparezcan con más frecuencia. Dos algoritmos que utilizan este método son Weeder [20] y FMotif [21].

Métodos basados en grafos

En este método se crea un grafo en el que cada vértice se corresponde con una subsecuencia de una longitud dada que se encuentra en las secuencias de entrada. Aquellos vértices cuyas cadenas sean lo suficientemente parecidas –difieran en un número máximo de bases nitrogenadas– están unidos en el grafo mediante aristas. Después de crear este grafo, el algoritmo busca subgrafos completos con los que crea el modelo del motivo. Dos algoritmos basados en este método son WINNOWER [22] y cWINNOWER [23].

Métodos basados en *hashing*

En este tipo de algoritmos se toman las subsecuencias de una longitud determinada encontradas de forma enumerativa, y se hashean a un mismo contenedor aquellas que compartan una serie de posiciones fijas. Aquí la función *hash* es aquella que toma como hash de cada subsecuencia la cadena formada por los caracteres de las posiciones fijadas. Aquellos contenedores que contengan un número mínimo de subsecuencias son considerados como posibles motivos.

Un ejemplo de este tipo de algoritmos es PROJECTION [24].

Métodos basados en candidatos fijos

Este tipo de algoritmos toma un candidato fijo a partir de las secuencias de entrada, y, posteriormente escanea estas mismas en búsqueda de ocurrencias.

Algunos algoritmos que implementan este método son PMSP, PMSi [25] y PMS3 [26].

Métodos basados en candidatos modificados

Este tipo de algoritmos, al igual que el anterior, toman un candidato de las secuencias de entrada, con la diferencia de que durante el escaneo de las secuencias de entrada se puede modificar el candidato tomado para mejorar la búsqueda de ocurrencias.

Algunos algoritmos que implementan este método son Stemming [27], Provable [29] y Pampa [28].

3.1.2. Enfoque probabilístico

El enfoque probabilístico se basa en construir, a partir de los datos, un modelo probabilístico del motivo que ajuste la distribución de las bases nitrogenadas para cada una de las posiciones de este. Después, este modelo se usa para detectar las secuencias generadas por el motivo.

Este enfoque es más rápido que el enumerativo, lo que permite manejar grandes cantidades de datos, pero es más complejo y no está asegurado encontrar un máximo global.

Dentro de este grupo podemos distinguir varias clases.

Métodos deterministas

La mayoría de algoritmos deterministas se basan en el algoritmo *Expectation Maximization* (EM), este algortimo es utilizado en estadística para encontrar estimadores de máxima verosimilitud en modelos probabilísticos que dependen de variables no observadas, y que se explicará en detalle más adelante. El primero de ellos fue MEME [13] y, a partir de este, surgen muchas derivaciones que incorporan ciertos cambios, como es el caso de de STEME [30] y EXTREME [31].

Métodos estocásticos

El grupo más importante de este tipo de algoritmos, es el de aquellos basados en el algoritmo *Gibbs Sampling* [32]. La esencia de estos algoritmos es (*i*) inicializar un modelo al azar, (*ii*) entrenar el modelo con todas las secuencias, menos una, elegida también al azar, y (*iii*), usar esta secuencia, que ha sido apartada, para evaluar el modelo entrenado. Los dos últimos pasos se realizan iterativamente, hasta que el modelo no mejore o hasta que se llevan a cabo el máximo número de iteraciones. Dos de estos algoritmos son Align ACE [33] y BioProspector [34]

Métodos bayesianos

Estos métodos se basan en la probabilidad bayesiana. Dos de los más destacados son LOGOS [35] que utiliza modelos de Markov locales para modelar cada motivo y, además, usa un modelo de Markov global que modela la aparición general de ocurrencias en el conjunto de secuencias de entrada. Y BaMM [36], que utiliza modelos de Markov para representar los motivos, e introduce relaciones de dependencia entre posiciones del motivo.

Otros

Por último, hay una serie de algoritmos que no encajan dentro de las otras clases, pero que conservan un enfoque probabilístico, por ejemplo, el algoritmo EPP [37] que se basa en la entropía de las posiciones del motivo.

3.1.3. Enfoque naturalista

Este grupo está basado en algoritmos de búsqueda que se inspiran en la naturaleza, por ejemplo, el comportamiento de los insectos, procesos químicos y físicos, etc. Se usan para resolver problemas muy complejos con un coste computacional adecuado. Este tipo de algoritmos logran combinar las principales características de las dos primeras categorías, búsqueda global a un coste computacional asumible.

Dentro de este grupo podemos encontrar varias clases dependiendo del tipo de metaheurísitca en que se basen.

Genetic Algorithm **(GA)**

El algoritmo genético es una metaheurística que se inspira en imitar el proceso de selección natural.

El algoritmo crea un grupo de candidatos a ser solución de forma aleatoria y los evalúa según una función de aptitud. A partir de estos, se crea una nueva generación de candidatos mediante cruces y mutaciones de la anterior generación. Así hasta que se llegue a un máximo o se hayan pasado un número máximo de generaciones.

Hay múltiples algoritmos que han aplicado esto a la búsqueda de motivos, algunos de ellos son GAMI [38] , GAEM [39] y GADEM [40].

Particle Swarm Optimization **(PSO)**

El algoritmo de enjambre de partículas es una metaheurística que simula el movimiento de bandadas de pájaros o bancos de peces, donde cada individuo usa el movimiento del resto para orientarse.

El algoritmo usa partículas que se mueven por el espacio de soluciones, cada una con una cierta dirección y velocidad. Las partículas evalúan cómo de buena es la posición en la que están y cambian de dirección en busca de un máximo, teniendo en cuenta los movimientos que están realizando el resto de partículas.

Algunos algoritmos que han aplicado esto a la búsqueda de motivos son PMbPSO [41], LPBS [42] y DSAPSO [43].

Artificial Bee Colony **(ABC)**

El algoritmo de colonia de abejas es una metaheurística que simula el comportamiento de las abejas a la hora de buscar comida.

Hay tres tipos de abejas trabajadoras, espectadoras y exploradoras. Las trabajadoras tienen asignado un candidato a solución, cada una evalúa su candidato, memoriza sus características y se lo muestra a las espectadoras. Las espectadoras eligen cuál es el mejor candidato. La abeja, cuyo candidato ha sido elegido, busca más candidatos en el entorno de este. Aquellas que han perdido sus candidatos, se convierten a exploradoras y empiezan a buscar mejores candidatos aleatoriamente.

Algunos algoritmos que aplican esto a la búsqueda de motivos son Multiobjective ABC [44], MO-ABC/DE [45] y Consensus ABC [46].

Ant Colony Optimization **(ACO)**

El algoritmo de la colonia de hormigas es una metaheurística que reproduce el comportamiento de las hormigas intentando encontrar el camino más corto desde el hormiguero hasta una fuente de comida.

Al principio las hormigas empiezan a buscar un candidato de forma aleatoria, durante la búsqueda cada hormiga deja un rastro de feromonas, el conjunto de rastros que han dejado las hormigas guiará a las siguientes hormigas que salgan a buscar otro candidato. La intensidad del rastro de feromonas dejado por una hormiga es proporcional a cómo de bueno es el candidato que haya encontrado, esto condiciona el comportamiento de las sucesivas hormigas. Además las feromonas se van evaporando con el tiempo con lo que cada cierto número de iteraciones se pierden los rastros anteriores.

Un algoritmo que implementa esto aplicandolo a la búsqueda de motivos es MFACO [47].

Cuckoo Search **(CS)**

El algoritmo CS es una metaheurística que busca simular el comportamiento parasitario que tienen los cucos a la hora de procrear.

En este caso, se supone que hay un conjunto de cucos y un conjunto de nidos, con candidatos dentro. Cada nido está vigilado por un pájaro anfitrión, y cada cuco pone un candidato en uno de los nidos. El objetivo es remplazar malos candidatos por otros mejores. Además hay una probabilidad de que el pájaro anfitrión descubra el candidato puesto por el cuco y deseche el candidato o incluso el nido entero.

Sólo los nidos con los mejores candidatos pasarán a las siguientes generaciones y, aquellos anfitriones que hayan desechado el nido o le hayan perdido, por tener malos candidatos, construirán uno nuevo.

Un algoritmo que utiliza esto en la búsqueda de motivos es MACS [48].

3.1.4. Enfoque híbrido

El último grupo es el del llamado enfoque híbrido en el que se combinan dos o más algoritmos con la esperanza de que se complementen entre sí y obtener así mejores resultados de los que se obtendrían con ellos por separado. Algunos algoritmos que implementan este enfoque son STGEMS [49], MDScan [50] que combinan enfoques enumerativos y probabilístico o EMD [51] que utiliza múltiples algoritmos.

3.2. Estructura general de un algoritmo de búsqueda de motivos

La mayoría de algoritmos resuelven el problema aplicando una estrategia de filtrado en la que se distinguen tres etapas que se ejecutan de manera secuencial a modo de *pipeline* o sucesión de filtros.

3.2.1. Preprocesado

En la primera etapa, o etapa de preprocesado, se preparan los datos, en este caso, secuencias de ADN, para facilitar la búsqueda.

Dentro de esta etapa se llevan a cabo procesos de diferente índole, previos a la búsqueda en si misma, se seleccionan aquellas secuencias que pueden contener motivos, se recortan las secuencias para hacerlas tan cortas como sea posible, también se desechan secuencias enteras, en concreto aquellas donde es bastante improbable que se puedan encontrar motivos y se lleva a cabo la selección de puntos de partida, que serán utilizados en la siguiente etapa.

3.2.2. Búsqueda

En esta etapa se lleva a cabo el ajuste del modelo que busca representar al motivo. Es en esta etapa donde más difieren los algoritmos entre sí, ya que hay una gran variedad de estrategias de búsqueda, como se ha visto anteriormente.

3.2.3. Postprocesado

En esta etapa se evalúa el motivo resultante. En caso de haber ajustado varios modelos en la etapa de búsqueda se elige aquél de ellos que obtenga una mayor puntuación.

3.3. Bases de datos de motivos

Las bases de datos de motivos son repositorios en los que se almacenan y documentan diferentes conjuntos de datos utilizados en la búsqueda de motivos y que se suele utilizar a modo de banco de pruebas (*testbed*) por los investigadores y poder comparar el rendimiento de los diferentes algoritmos diseñados. Se distinguen dos tipos de bases de datos:

3.3.1. Bases de datos sintéticas

Las bases de datos sintéticas son creadas, por lo general, por los mismos grupos de investigación que crean los algoritmos, para poder poner el algoritmo desarrollado a prueba, de esta forma se pueden moldear los experimentos con unas condiciones específicas.

3.3.2. Bases de datos reales

Estas son las bases de datos fundamentales para la investigación. Las bases de datos reales están pobladas de motivos que han sido encontrados en secuencias de ADN reales. Estas bases incluyen las secuencias donde se han encontrado los motivos, y el organismo al que pertenece. Algunas son públicas y otras son de pago.

Ejemplos de estas bases de datos reales son JASPAR [9], footprintDB [10] y MyHits [11].

Capítulo 4

Fundamentos y herramientas matemáticas

4.1. El problema de la subcadena más próxima

Dadas una serie de cadenas $Y = \{Y_1, Y_2, \ldots, Y_n\}$, una longitud w y un valor de distancia r para una una función de distancia dada $d(a, b)$, el problema de la subcadena más próxima pide encontrar una cadena S, de longitud w, de manera que en cada cadena de entrada Y_i haya una subcadena $S_i \subset Y_i$ que esté a una distancia menor o igual que r, es decir, $d(S, S_i) \leq r$. A la cadena S, solución del problema, se la conoce como centro.

Este problema ha sido estudiado extensamente para conjuntos de cadenas finitos, y se sabe que es un problema NP-completo. Esto quiere decir que el problema es, NP y NP-duro a la vez. Que un problema sea NP quiere decir que puede ser resuelto por una máquina de Turing no determinista en tiempo polinomial, y, que sea NP-duro quiere decir que cualquier otro problema del tipo NP puede ser reducido a este en tiempo polinomial [4].

4.1.1. Adaptación

El problema de la búsqueda de motivos puede ser entendido como un caso particular del problema de la subcadena más próxima relajando la condición que exige que haya exactamente una subcadena S_i en cada una de las cadenas de entrada Y_i a que haya varias o ninguna, y, tomando como medida la distancia de Hamming, que se define como el número de caracteres en los que difieren dos cadenas.

Otra particularidad es que, por lo general, los algoritmos buscan crear el modelo probabilístico del motivo, por así decirlo, se busca crear el modelo que represente el conjunto de esas subcadenas encontradas. Aunque esto difiere del del problema de la subcadena más próxima, podemos interpretarlo como un paso más que se da después de resolverlo.

4.2. Métodos y herramientas matemáticas

En esta sección se hará una introducción a algunos métodos y herramientas utilizados, a su significado, y a su fundamentación matemática, pero, sin contemplar los usos particulares que se harán de estos en el algoritmo MEME, lo que se hará en secciones del trabajo posteriores.

4.2.1. Verosimilitud (Likelihood) y función de verosimilitud

La verosimilitud mide cómo de bien un modelo estadístico describe un conjunto de datos observado en función de uno o varios parámetros de los que depende el modelo.

Dada una variable aleatoria X continua con función de densidad f, que depende del parámetro θ, y unos datos observados x, se define la función de verosimilitud como

$$L(\theta|x) = f_\theta(x),$$

en caso de que X fuera una variable discreta, de nuevo, con parámetro θ, se define la función de verosimilitud como

$$L(\theta|x) = P(X = x|\theta).$$

Es decir, es la probabilidad de que la variable aleatoria tome un determinado valor, condicionado a que los parámetros de la distribución de probabilidad sean unos conocidos, en este caso θ.

4.2.2. Test LLR (Log-Likelihood Ratio)

El test LLR compara la verosimilitud de dos modelos que compiten por ajustar unos datos.

Dada una variable X que depende del parámetro θ, un par de parámetros θ_0 y θ_1 que describen a los modelos enfrentados, y unos datos observados x, se define el LLR como

$$LLR(x) = \log \frac{L(\theta_1|x)}{L(\theta_0|x)}.$$

Es fácil observar que un valor positivo significa que el modelo determinado por θ_1 ajusta mejor los datos observados y un valor negativo lo contrario.

4.2.3. p-valor y significación estadística

Dados unos datos observados x y una variable aleatoria X que se distribuye siguiendo cierto modelo. Se define el p-valor de x como la probabilidad de obtener de X un valor tan extremo o más que x.

Por así decirlo el p-valor es una medida de la rareza de que x sea una ocurrencia de la variable X.

Se dice que los datos observados, x, tienen significación estadística cuando el p-valor de estos para el modelo de X es lo suficientemente pequeño. Por lo general se coge $0,05$ como umbral para asegurar la significación estadística.

4.2.4. Algoritmo EM

El algoritmo EM [16] es una herramienta para la búsqueda de estimadores de máxima verosimilitud para parámetros de una serie de modelos de los cuales se sabe que comparten la misma distribución.

Se tienen

$$\hat{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n]^T$$

una conjunto de datos observados, donde $\mathbf{X}_i$ es un vector de observaciones. Estos surgen de un conjunto de modelos que comparten una misma distribución, con función de densidad f, pero están determinadas por unos parámetros que no se conocen

$$\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$$

y que se quieren estimar.

Se definen unas variables auxiliares

$$Z = [\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_n]^T$$

con

$$\mathbf{Z}_i = (Z_{i,1}, Z_{i,2}, \ldots, Z_{i,m}) \text{ tal que } Z_{i,j} = \begin{cases} 1 \text{ si } \mathbf{X}_i \text{ se distribuye según } \theta_j, \\ 0 \text{ si no,} \end{cases}$$

las variables $Z_{i,j}$ asignan la pertenencia del vector de muestras $\mathbf{X}_i$ al grupo formado por aquellas que surgen del modelo determinado por θ_j, se deduce fácilmente que $\sum_{j=1}^{m} Z_{i,j} = 1$ para $1 \leq i \leq n$.

También se definen los parámetros

$$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m) \text{ con } \sum_{i=1}^{m} \lambda_i = 1 \text{ y } \lambda_i \geq 0 \text{ para todo } i,$$

un variable que controla el grado de mezcla de los diferentes modelos.

A partir de esto se crea un modelo de mezcla que especifica:

Para $1 \leq i \leq n$ se tiene que $(\mathbf{X}_i|\mathbf{Z}_i, \theta, \lambda)$ son independientes con densidades

$$\sum_{j=1}^{m} Z_{i,j} f(\mathbf{X}_i|\theta_j) = \prod_{j=1}^{m} f(\mathbf{X}_i|\theta_j)^{Z_{i,j}} \tag{4.1}$$

y $(Z_i|\theta, \lambda)$ son variables independientes idénticamente distribuidas con distribución

$$\prod_{j=1}^{m} \lambda_j^{Z_{i,j}}. \tag{4.2}$$

De (4.1) y (4.2), se tiene que

$$(\mathbf{X}_i|\theta, \lambda) \overset{\text{i.i.d}}{\sim} \sum_{j=1}^{m} \lambda_j f(\mathbf{X}_i|\theta_j).$$

De aquí, si θ y λ son parámetros que se quieren estimar a partir de $\hat{X}$, se llega a la función de verosimilitud

$$L(\theta, \lambda|\hat{X}) = \prod_{i=1}^{n} \left[\sum_{j=1}^{m} \lambda_j f(\mathbf{X}_i|\theta_j) \right].$$

Si Z fueran datos observados la función de log-verosimilitud puede ser escrita como

$$\log L(\theta, \lambda|\hat{X}, Z) = \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j} \log f(\mathbf{X}_i|\theta_j) + \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{ij} \log \lambda_j. \tag{4.3}$$

Al inicio de la ejecución se dan valores iniciales a $\theta = \theta^{(0)}$ y a $\lambda = \lambda^{(0)}$, el llamado punto de partida. El proceso de optimización es iterativo, cada iteración consta de dos pasos fundamentales:

E-paso

En este paso se calcula el valor esperado de la función de log-verosimilitud sobre la distribución condicionada de las variables ocultas Z dados (a) los datos observados $\hat{X}$ y (b) las estimaciones actuales de los parámetros θ y λ.

El valor esperado de la log-verosimilitud dados $\hat{X}$, $\theta = \theta^{(0)}$ y $\lambda = \lambda^{(0)}$ es

$$\begin{aligned} E(\log L(\theta, \lambda|\hat{X}, Z)) = &\sum_{i=1}^{n}\sum_{j=1}^{n} \log f(\mathbf{X}_i|\theta_j) E(Z_{i,j}|\hat{X}, \theta^{(0)}, \lambda^{(0)}) \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n} \log \lambda_i E(Z_{i,j}|\hat{X}, \theta^{(0)}.\lambda^{(0)}) \end{aligned} \tag{4.4}$$

De (4.1), (4.2) y el teorema de Bayes se deduce que

$$(\mathbf{Z}_i|\mathbf{X}_i, \theta, \lambda) \overset{\text{i.i.d}}{\sim} \prod_{j=1}^{m} \left[\lambda_j f(\mathbf{X}_i|\theta_j)\right],$$

así el valor esperado, que tomamos como $Z_{i,j}^{(0)}$ es

$$Z_{i,j}^{(0)} = E(Z_{i,j}|\hat{X}, \theta^{(0)}, \lambda^{(0)}) = \frac{\lambda_j^{(0)} f(\mathbf{X}_i|\theta_j^{(0)})}{\sum_{j=1}^{m} \lambda_j^{(0)} f(\mathbf{X}_i|\theta_j^{(0)})},$$

y sustituyéndolo en (4.3) se obtiene que

$$\begin{aligned} E(\log L(\theta, \lambda|\hat{X}, Z)) = &\sum_{i=1}^{n}\sum_{j=1}^{n} \log f(\mathbf{X}_i|\theta_j) Z_{i,j}^{(0)} \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n} \log \lambda_i Z_{i,j}^{(0)}. \end{aligned} \tag{4.5}$$

Vale la pena darse cuenta de que a partir de ahora Z_{ij} ya no vale 0 o 1, sino que está entre 0 y 1.

M-paso

Una vez completado el E-paso se maximiza el valor esperado del paso anterior en función de θ y λ para obtener así una nueva estimación de los parámetros $\theta^{(1)}$ y $\lambda^{(1)}$.

La maximización en función de θ es equivalente a maximizar el primer término de (4.5)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \log f(\mathbf{X}_i|\theta_j) Z_{i,j}^{(0)}.$$

La maximización en función de λ es equivalente a maximizar el segundo término de (4.5)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \log \lambda_i Z_{i,j}^{(0)},$$

lo que da

$$\lambda_j^{(1)} = \sum_{i=1}^{n} \frac{Z_{ij}^{(0)}}{n} \quad 1 \leq j \leq m.$$

Este proceso se repite hasta la convergencia de los parámetros, la cual no está asegurada en todas las ocasiones, o hasta que se hayan realizado un número máximo de iteraciones.

El resultado, en caso de convergencia, son estimadores de máxima verosimilitud locales de θ y λ.

Capítulo 5

El algoritmo MEME

5.1. Conceptos básicos

5.1.1. Motivo y ocurrencia de un motivo

En el capítulo de introducción se definió un motivo como una descripción probabilística de una patrón encontrado en el ADN. El motivo, en sí, no se encuentra en las secuencias de ADN, ya que, es una representación probabilística de una serie de posibles secuencias que pueden ocurrir, a estas secuencias, que se materializan dentro de otra secuencia de ADN más larga son llamadas ocurrencias.

Los motivos suelen ser descritos, mediante una matriz con probabilidades. Como representación visual se suele usar el "logo de la secuencia", que es una cadena en la que en cada posición puede haber varias bases nitrogenadas apiladas verticalmente, y donde el tamaño de cada base es proporcional a la frecuencia relativa de dicha base en la posición y a la cantidad de información contenida en esa posición del motivo. Por ejemplo:

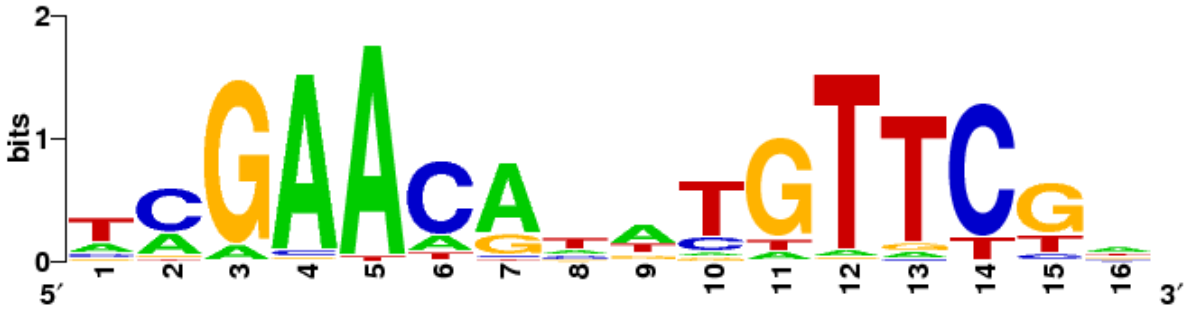


Figura 5.1: Modelo visual de un motivo

En la Figura 5.1 hay representado un motivo de longitud 16, se puede apreciar fácilmente que en la tercera posición de las ocurrencias será muy frecuente encontrar la base G, pero no obligatorio, ya que el modelo contempla que pueda aparecer A, también en dicha posición. De esta manera

$$TCGAACATATGTTCGA$$

es una ocurrencia del motivo descrito por la figura 5.2, pero también lo es la cadena

$$TCAAACATATGTTCGA.$$

Lo que realmente se busca son estas ocurrencias, que son las que se pueden usar para investigar y así descubrir las funciones biológicas que llevan asociadas. Sin embargo, el hecho de que dos ocurrencias del mismo motivo puedan ser diferentes hace que el espacio de búsqueda sea demasiado grande e inabarcable para realizar una búsqueda directa.

Es por esto que los algoritmos no buscan las ocurrencias dentro de las secuencias, a pesar de ser lo que realmente interesa, sino que buscan el motivo, es decir, el modelo probabilístico que mejor describe a las ocurrencias, y que, de hecho, se puede decir las lleva a todas codificadas en sí mismo.

Una vez encontrado el modelo se pueden buscar fácilmente cuáles son las cadenas dentro de nuestras secuencias con más probabilidad de ser ocurrencias, y se puede usar el modelo para buscar dentro de otras secuencias diferentes a las originales.

5.2. Definiciones previas y notación del algoritmo MEME

5.2.1. Datos básicos del algoritmo

Se designará por w la longitud de interés de los motivos a considerar por el algoritmo MEME.

El conjunto de datos es $Y = \{Y_1, Y_2, \ldots, Y_N\}$, siendo N el número de secuencias de ADN de las que se parte. La longitud de cada secuencia de entrada Y_i es arbitraria y se denotará por l_i. Teniendo esto en cuenta, se define el número de posibles posiciones de la secuencia en las que puede comenzar una ocurrencia como $m_i = l_i - w + 1$.

Para facilitar las notaciones y la explicación que sigue se asumirá que, a partir de aquí, todas las secuencias de entrada tienen la misma longitud l y consecuentemente $m = l - w + 1$.

Se supone además que cada secuencia Y_i está escrita sobre un alfabeto fijo $A = \langle a_1, a_2, \ldots, a_L \rangle$, que es parte de los datos de entrada.

El algoritmo MEME no trabaja directamente con el conjunto de secuencias de entrada Y, sino que parte cada una de ellas, en subsecuencias de longitud w. Este conjunto es $X = \{X_{i,j} | 1 \leq i \leq N, 1 \leq j \leq m\}$ siendo Y_i la secuencia el que se encuentra $X_{i,j}$ y j la posición de Y_i en la que empieza esta subsecuencia.

5.2.2. Modelos del algoritmo

MEME utiliza dos modelos para intentar representar los datos:

1. **El modelo del motivo**: Denotado por

$$\theta_1 = \left[\begin{array}{cccc} \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_W \end{array} \right] = \left[\begin{array}{cccc} P_{a_1,1} & P_{a_1,2} & \cdots & P_{a_1,W} \\ P_{a_2,1} & P_{a_2,2} & \cdots & P_{a_2,W} \\ \vdots & \vdots & \vdots & \vdots \\ P_{a_L,1} & P_{a_L,2} & \cdots & P_{a_L,W} \end{array} \right],$$

   donde $P_{a_k j}$ representa la probabilidad de que el k-ésimo elemento del alfabeto A, a_k, aparezca en la posición j de una ocurrencia del motivo.

2. **El modelo** *background*: Denotado por

$$\theta_0 = \left[\begin{array}{c} \mathbf{P}_0 \end{array} \right] = \left[\begin{array}{c} P_{a_1,0} \\ P_{a_2,0} \\ \vdots \\ P_{a_L,0} \end{array} \right],$$

este es un modelo que consideramos cierto, y representa la distribución de las bases nitrogenadas en las secuencias de entrada. De esta manera $P_{a_k 0}$ representa la probabilidad de que la letra a_k aparezca en una posición cualquiera tomada al azar.

Aunque el modelo *background* lo calcula automáticamente el algoritmo, el usuario puede introducir un modelo diferente y obligar a MEME a usarlo.

Juntando ambos se tiene la matriz PSPM (*Position Specific Probability Matrix*), a la que se identifica con la variable θ

$$\theta = \left[\begin{array}{cccc} \theta_0 & \theta_1 \end{array} \right] = \left[\begin{array}{cccc} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_W \end{array} \right]$$

$$= \begin{bmatrix} P_{a_1,0} & P_{a_1,1} & P_{a_1,2} & \cdots & P_{a_1,W} \\ P_{a_2,0} & P_{a_2,1} & P_{a_2,2} & \cdots & P_{a_2,W} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{a_L,0} & P_{a_L,1} & P_{a_L,2} & \cdots & P_{a_L,W} \end{bmatrix}.$$

5.2.3. PSP (*Position-specific prior*)

En las versiones más modernas de MEME se introdujo una forma de crear un sesgo en la búsqueda para priorizar unas posiciones en la cadena por encima de otras. Esto se conoce como PSP *Position-specific prior* y es un conjunto de probabilidades, dadas *a priori*, que se asignan a las posiciones de cada secuencia. De esta manera $PSP_{i,j}$ es el sesgo dado *a priori* a que una ocurrencia empiece en la posición j de la secuencia i.

5.2.4. Modos de ejecución

El algoritmo MEME tiene tres modos de ejecución en los que varía el número de ocurrencias del motivo que se supone hay en cada cadena.

OOPS (*One Ocurrence Per Sequence*) El modo OOPS considera que en cada una de las secuencias Y_i hay una ocurrencia del motivo.

Este modo tiene como hiperparámetros $\phi = \{\theta\}$.

ZOOPS (*Zero or One Ocurrences Per Sequence*) Es decir, una cadena Y_i puede no contener una ocurrencia del motivo.

Este modo tiene como hiperparámetros $\phi = \{\theta, \gamma\}$, donde γ representa la probabilidad, dada *a priori*, de que una secuencia contenga una ocurrencia del motivo.

Notesé que el modelo OOPS puede modelarse igual que ZOOPS sin más que considerar $\gamma = 1$.

ANR (*Any Number of Repetitions*) Por último, ANR es el modelo más general, en este caso se parte de la hipótesis que hay cero o varias ocurrencias por cada secuencia.

Este modo tiene como hiperparámetros $\phi = \{\theta, \lambda\}$, donde λ juega un papel similar al de γ en el modelo anterior. En este caso λ representa la probabilidad, dada *a priori*, de que una posición de la secuencia sea el inicio de una ocurrencia del motivo.

5.2.5. Palíndromos

En el ADN, con alfabeto $A = \{A, G, C, T\}$, existe la noción de complementariedad, es decir, cada secuencia de ADN tiene una complementaria, a la que van unida. Esta complementariedad se da a nivel de las bases nitrogenadas, A y T son complementarios entre si, lo que quiere decir que cada base A de una secuencia va unida a una base T de su secuencia complementaria, de la misma manera ocurre con G y C.

De esta manera la secuencia complementaria de

$$AGTCCAGTA$$

es

$$TCAGGTCAT.$$

Una secuencia es palindrómica si coincide con la inversa de la secuencia complementaria.

Por ejemplo, la secuencia

$$ACCTAGGT$$

es palindrómica porque su complemento base por base es

$$TGGATCCA$$

y en sentido inverso coincide con la primera.

Se puede exigir a MEME que busque motivos palindrómicos. Este modela las columnas exigiendo que las columnas de θ_1 sean las inversas de su complementaria, en el caso del ADN,

$$\theta_1 = \begin{bmatrix} P_{A,1} & P_{A,2} & \cdots & P_{T,2} & P_{T,1} \\ P_{C,1} & P_{C,2} & \cdots & P_{G,2} & P_{G,1} \\ P_{G,1} & P_{G,2} & \cdots & P_{C,2} & P_{C,1} \\ P_{T,1} & P_{T,2} & \cdots & P_{A,2} & P_{A,1} \end{bmatrix},$$

es decir,

$$\begin{aligned} P_{A,i} &= P_{T,W+1-i}, \\ P_{C,i} &= P_{G,W+1-i}, \\ P_{G,i} &= P_{C,W+1-i}, \\ P_{T,i} &= P_{A,W+1-i}. \end{aligned} \text{ para } i = 1, \ldots, \lfloor \frac{W}{2} \rfloor.$$

5.2.6. Test LLR (*Log-Likelihood Ratio*)

El test LLR es una forma de calcular cómo de bien se ajusta una cadena de ADN a un modelo, dado por un motivo, con respecto al modelo *background*. Por así decirlo, muestra cómo de buena es la cadena si fuese candidata a ser ocurrencia del motivo.

Dada la secuencia S, de longitud w_S, y el modelo $\theta = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix}$ considerado por MEME, se define el valor de LLR como

$$LLR = \log \frac{P(S|\theta_1)}{P(S|\theta_0)} = \log \frac{\prod_{j=1}^{w_S} P_{S_j,j}}{\prod_{j=1}^{w_S} P_{S_j,0}} = \sum_{j=1}^{w_S} \log \frac{P_{S_j,j}}{P_{S_j,0}}.$$

Si el valor de LLR es positivo, se supone que la ocurrencia se ajusta mejor al motivo que al modelo *background*, mientras que si es negativo el ajuste del motivo es peor incluso que el modelo *background*.

Por ejemplo, dada la cadena

$$AGCTA,$$

y el modelo

$$\theta = \begin{bmatrix} P_{A,0} & P_{A,1} & P_{A,2} & P_{A,3} & P_{A,4} & P_{A,5} \\ P_{G,0} & P_{G,1} & P_{G,2} & P_{G,3} & P_{G,4} & P_{G,5} \\ P_{C,0} & P_{C,1} & P_{C,2} & P_{C,3} & P_{C,4} & P_{C,5} \\ P_{T,0} & P_{T,1} & P_{T,2} & P_{T,3} & P_{T,4} & P_{T,5} \end{bmatrix}$$
$$= \begin{bmatrix} 0{,}21 & 0{,}72 & 0 & 0{,}26 & 0{,}10 & 0{,}64 \\ 0{,}28 & 0 & 0{,}56 & 0 & 0{,}10 & 0 \\ 0{,}26 & 0{,}15 & 0{,}44 & 0{,}62 & 0 & 0 \\ 0{,}25 & 0{,}13 & 0 & 0{,}12 & 0{,}80 & 0{,}36 \end{bmatrix},$$

el LLR calculado sería

$$\log \frac{0{,}72}{0{,}21} + \log \frac{0{,}56}{0{,}28} + \log \frac{0{,}62}{0{,}26} + \log \frac{0{,}80}{0{,}25} + \log \frac{0{,}64}{0{,}21} = 5{,}07184$$

Se define también el LLR para el motivo como la suma de todos los LLR de sus ocurrencias.

5.2.7. P-valor de una ocurrencia

Se define el P-valor de una ocurrencia como el p-valor de su LLR para el modelo del motivo θ_1 y es una medida de su significación estadística.

5.2.8. E-valor de un motivo

El E-valor de un motivo es la medida de la significación estadística del propio motivo dentro del conjunto de datos. La forma de calcularlo varía según la función objetivo escogida y el test estadístico que esta aplique.

5.2.9. Funciones objetivo

MEME usa una función objetivo para elegir el mejor motivo de los encontrados por el algoritmo EM a partir de los diferentes puntos de partida encontrados en la primera fase.

MEME tiene implementadas varias funciones objetivo, todas ellas se basan en la significación estadística del resultado de aplicar un test estadístico al motivo.

- **Clásica.**

  Esta es la función original implementada por MEME y la que usa por defecto. MEME calcula la significación estadística del LLR del motivo, utiliza una estimación del número de motivos que tendrían un LLR igual o superior a nuestro motivo, si las secuencias de entrada estuvieran generadas por el modelo *background* θ_0. Una vez calculado se multiplica por el mejor P-valor de las ocurrencias del motivo, lo queda el E-valor.

- **Enriquecimiento diferencial selectivo.**

  En caso de elegir esta función, el algoritmo pide dos conjuntos de secuencias de entrada, uno con el que entrena al modelo y otro con el que hacer pruebas, el conjunto de control.

  Una vez entrenado el modelo, se hace una búsqueda de ocurrencias en parte del conjunto primario y en todo el conjunto de control.

  Las ocurrencias se ordenan según su LLR y se toma el P-valor más alto entre todas ellas, luego se multiplica por el número total de motivos que se estén evaluando para obtener así el E-valor.

- **Enriquecimiento diferencial.**

  De nuevo el algoritmo toma dos conjuntos de secuencias como entrada, pero esta vez el segundo, el de control, lo divide en dos, de manera que acaba teniendo tres conjuntos diferentes, el de entrenamiento, el de control y el de evaluación.

  El mejor motivo se elige de la misma forma pero el E-valor final de este se calcula con el grupo de evaluación.

- **Distancia central.**

  Se toman las ocurrencias encontradas por el motivo en las secuencias de entrada y se ordenan según su LLR.

  Para cada ocurrencia se calcula la distancia de la ocurrencia con mejor LLR al centro de la secuencia, el E-valor se calcula usando una distribución Bates [7].

- **Enriquecimiento central.**

  De nuevo, se toman las ocurrencias encontradas por el motivo en las secuencias de entrada y se ordenan según su LLR.

  El E-valor se calcula aplicando un test binomial al número de veces que las ocurrencias con mejor puntuación se encuentran en la región central de la secuencia vs. el número de veces que se encuentra en los flancos.

- *Numerically correct***.**

  Esta función tiene un funcionamiento idéntico a la Clásica con la diferencia de que calcula la puntuación de los motivos en base al algoritmo *NC* de *Hertz* y *Stromo* [8].

5.3. Funcionamiento del algoritmo MEME

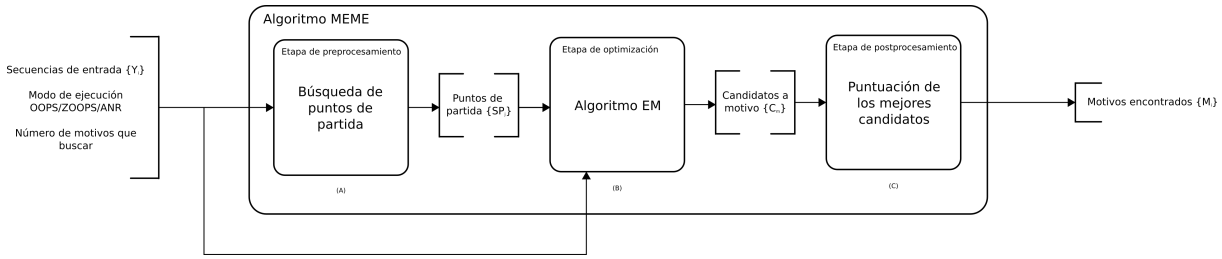


Figura 5.2: Esquema básico del algoritmo MEME

En la Figura 5.2 se distinguen tres etapas claramente diferenciadas en el algoritmo MEME, estas etapas han sido introducidas en el Capítulo 3.

La primera etapa es la que se encarga de la búsqueda de puntos de partida, a la que también se puede identificar como una etapa de preprocesamiento. Inicialmente, el espacio de posibles puntos de partida desde los que realizar la optimización es muy extenso, por lo que en esta etapa se reduce el espacio realizando una búsqueda dentro de este, para quedarnos con los mejores candidatos.

Tras haber seleccionado los mejores puntos de partida del espacio de búsqueda, estos se introducen en la segunda etapa, que consiste en varias ejecuciones del algoritmo EM, una por cada punto de partida seleccionado. También se puede referir a esta etapa como la etapa de optimización. En ella se toma un modelo del motivo buscado, creado a partir de un punto de partida, y se optimizan sus parámetros, en base a una función de verosimilitud, buscando que describan, lo mejor posible, al conjunto de datos y a las posibles ocurrencias de dicho motivo.

Una vez optimizados los modelos, se llega a la última etapa, la etapa de selección. En esta se puntúan todos los modelos ajustos a partir de los puntos de partida, y, aquél con la mejor puntuación es el que devuelve el algoritmo como motivo descubierto.

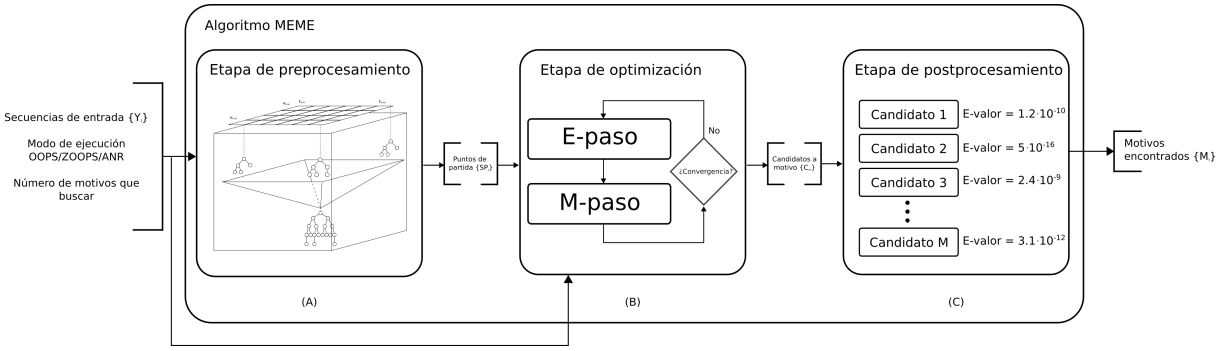


Figura 5.3: Esquema detallado del algoritmo MEME

5.4. Búsqueda de puntos de partida

La primera fase del algoritmo consiste en la exploración del espacio de puntos de partida para seleccionar aquellos que sean más prometedores. El tamaño del espacio de todos los posibles puntos de partida es tan extenso que hace que una búsqueda completa, probar con cada punto de partida, sea inabarcable, por el alto coste computacional que supondría.

La razón por la que tomar varios puntos de partida se debe a que el algoritmo EM es un algoritmo de optimización local, es decir, puede quedar atrapado en un máximo local. Partiendo de varios puntos se espera que uno de los máximos encontrado, sea el máximo global, es decir, haya más probabilidades de escapar de un máximo local.

Para evaluar cada posible punto de partida se calcula el p-valor de un test LLR que se realiza sobre el propio punto de partida, este p-valor se utiliza según la función objetivo elegida por el usuario.

5.4.1. Puntos de partida

Conceptualmente un punto de partida es, como su propio nombre indica, aquel punto del espacio de búsqueda de motivos desde el que el algoritmo EM va a comenzar la búsqueda de un máximo –en concreto de un máximo local. Los puntos de partida son cadenas de ADN que van a ser optimizadas con la esperanza de que al final del proceso de optimización, se obtenga una cadena de consenso representativa de los patrones presentes en nuestras secuencias de entrada.

A su vez, durante la búsqueda el algoritmo va a considerar inicialmente diferentes candidatos para cada punto de partida. A estos candidatos se les conoce como semillas. Una vez finalizada la búsqueda, el punto de partida tomará los datos de la semilla con mayor puntuación.

Un punto de partida es representado por el algoritmo MEME mediante una estructura de datos compleja que contiene toda una serie de atributos requeridos para el funcionamiento del algoritmo.

Estos atributos son:

- **Cadena de consenso como cadena de caracteres del alfabeto**

  Esta es la cadena que el algoritmo EM debe optimizar, por lo general es una subsecuencia que se extrae de las secuencias que se pasan como datos de entrada, representada en caracteres del

alfabeto de interés, y que deben ser imprimibles (por ejemplo, porque están codificados en ASCII). Un ejemplo sería

$$ATTGCTTGAACC.$$

- **Cadena de consenso como lista de enteros**

  Para facilitar las operaciones MEME transforma la cadena a una representación en enteros para lo cual basta sustituir el símbolo utilizado por su código ASCII correspondiente (suponiendo el uso de una codificación ASCII). Por ejemplo, la cadena anterior se convertiría en

  $$[065, 084, 084, 071, 067, 084, 084, 071, 065, 065, 067, 067]$$

- **Número de secuencia**

  Número de secuencia en la que se ha encontrado la subsecuencia.

- **Posición en la secuencia**

  Posición en la secuencia dónde empieza la subsecuencia.

- **Longitud de la cadena**

  Longitud inicial del motivo buscado.

- **Número de ocurrencias requeridas**

  Número inicial de ocurrencias requerido del motivo buscado, el algoritmo puede ajustar este valor por uno más óptimo de forma dinámica.

- **Sesgo al número de ocurrencias**

  Este sesgo se calcula en función del PSP (*Position Specific Prior*) dado por el usuario y de la probabilidad de que una ocurrencia en el punto de partida se superponga a una encontrada anteriormente. Así pues, este valor va cambiando según la iteración del algoritmo y los motivos encontrados anteriormente.

  Se utiliza en la optimización del número de ocurrencias llevado a cabo en el cálculo de las puntuaciones.

- **Puntuación**

  Puntuación que ha recibido la cadena de consenso como punto de partida. El cálculo de este valor depende de su LLR –donde entran en juego las posibles ocurrencias encontradas– y de la función objetiva escogida.

- **Significación**

  La significación estadística del punto de partida candidato como motivo representativo de las posibles ocurrencias encontradas. El cálculo de esta depende de la puntuación obtenida por el punto de partida y de la función objetivo escogida. Las semillas no tienen calculada su significación estadística, es exclusivo del punto de partida.

- *Heap* **de las semillas**

  El *heap* es una estructura de datos en forma de árbol, que almacena las semillas con las puntuaciones más altas. La particularidad de esta estructura de datos es que dado un nodo, este siempre tiene menor puntuación que la de sus nodos hijos, lo que mejora la eficiencia de la búsqueda de las mejores semillas encontradas hasta el momento, eso sí, a costa de complicar la gestión de la estructura cuando se insertan (o se eliminan) elementos del *heap*.

  El tamaño del *heap* está delimitado, y es inversamente proporcional a la distancia de Manhanttan del punto de partida con el punto de partida "central"más próximo.

Es importante dejar claro que, durante la búsqueda, la mayoría de estos valores, exceptuando la longitud de la cadena y el número de ocurrencias, no están determinados *a priori*. Donde sí lo están es en las semillas. Y al final de la búsqueda, cada punto de partida toma los datos de su mejor semilla, exceptuando la significación.

La significación es un dato exclusivo del punto de partida, que se calcula una vez se han tomado los datos de la mejor semilla.

5.4.2. Espacio de búsqueda y matriz de puntos de partida

Inicialmente el conjunto de candidatos por los que comenzar a buscar se restringe al conjunto de secuencias de entrada y X. Más adelante, este se expande y se llevan a cabo mutaciones sobre las semillas más prometedoras, en un intento por considerar otros puntos de partida que no se pueden extraer directamente de las secuencias y que puedan llegar a mejorar la puntuación. Téngase en cuenta que el espacio de búsqueda de motivos, Π, *a priori* coincide con el conjunto de datos de todas las posibles secuencias de longitud w tomadas a partir de símbolos del alfabeto de interés A, y $|\Pi| = |A|^w$, por lo que, por lo general $|\Pi| >> |X|$. Así pues, las mutaciones sobre las semillas permiten aumentar la capacidad de explotación del espacio de búsqueda de motivos entorno a los datos de partida.

La búsqueda está gestionada por una matriz de dos dimensiones, donde se considera que cada posición –en esta matriz– es un punto de partida.

Cada fila se corresponde con una de las longitudes que el algoritmo va a considerar. La lista de las posibles longitudes se define en términos de una longitud mínima w_{min} –por defecto $w_{min} = 6$ aunque el usuario puede elegir– y de una máxima w_{max} –por defecto $w_{max} = 50$, aunque, de nuevo, el usuario puede elegir otra.

Además, la matriz se divide en particiones, para lo cual se crea una progresión geométrica de razón $\sqrt{2}$ –con los valores redondeados para que sean enteros– que comienza en w_{min} y a la que se añade al final w_{max}.

Por ejemplo, con los valores de w_{min} y w_{max}, por defecto, esta lista sería

$$[6, 8, 12, 16, 24, 33, 48, 50]\,.$$

De igual manera ocurre con las columnas, de modo que cada columna representa un número de ocurrencias que se requiere que el motivo tenga en todo el conjunto de datos para ser considerado un motivo. De ahora en adelante, este número de ocurrencias se representará con la variable k. De nuevo, la lista de los números de ocurrencias se define en términos de un valor mínimo k_{min} y un valor máxima k_{max}, con la particularidad de que estos valores dependen del modo de ejecución del algoritmo.

En los modos ZOOPS y ANR, los valores por defecto son $k_{min} = 2$ y $k_{max} = 600$ y el usuario puede elegir otros. En el modo OOPS el usuario no puede elegir los valores que él quiera, sino que los valores por defecto son $k_{min} = k_{max} = N$, siendo N el número total de secuencias de entrada.

De nuevo, con el objetivo de particionar la matriz, y en caso de que se ejecute el algoritmo en modo ZOOPS o en modo ANR, se crea una lista de números de ocurrencias "centrales", con el mismo formato que la de las longitudes "centrales", aunque, en esta ocasión, la razón de la progresión geométrica es 2.

Por ejemplo, con los valores de k_{min} y k_{max} –2 y 600, respectivamente– por defecto, esta lista sería

$$[2, 4, 8, 16, 32, 64, 128, 256, 512, 600]\,.$$

Una vez se tienen las listas de longitudes, longitudes "centrales", números de ocurrencias y números de ocurrencias "centrales" se crea la matriz y sus particiones. Las particiones se crean exigiendo que en cada una de ellas haya un único punto "central", es decir, un punto de partida que esté representados en la matriz por un par (w, k), dónde w y k son ambos "centrales".

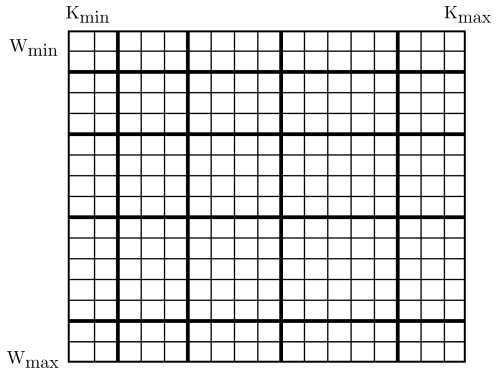


Figura 5.4: Matriz de puntos de partida

De momento, la matriz está vacía y cada posición, o punto de partida, está representado por un par de la forma (w, k).

El siguiente paso es asignar una cadena de consenso a cada punto de partida. Esto se puede abordar de dos maneras:

- La manera más sencilla, y menos costosa (computacionalmente), es usar una cadena de consenso proporcionada por el usuario. Esta cadena se introduce en todos los puntos de partida y es la que optimiza el algoritmo EM. Esta opción se suele usar cuando ya se conoce una ocurrencia del motivo y así puede utilizarse para ajustar el modelo asociado.
- La segunda opción es realizar una búsqueda global en todas las secuencias, en la que la cadena de consenso de cada punto de partida será las subsecuencia que obtenga la puntuación más alta dada por cierta función objetivo.

5.4.3. Cadena de consenso proporcionada por el usuario

En este caso, sólo se introduce la cadena de consenso en los puntos de partida "centrales".

Como se ha comentado anteriormente, sólo hay un punto de partida "central" en cada partición, por lo que al ser el único con cadena de consenso, este será el elegido en cada partición como entrada del algoritmo EM.

5.4.4. Búsqueda global

Como se ha comentado anteriormente, en la búsqueda global se examinan las secuencias del conjunto de datos dado por el usuario, en busca de la mejor cadena de consenso para cada punto de partida. Para ello se sigue un proceso iterativo que recorre toda la matriz.

Para empezar se recorre la matriz de puntos de partida por filas. Esto quiere decir que se recorre la lista de longitudes en orden ascendente, de w_{min} hasta w_{max}.

Tomar fila y candidato

Una vez fijada una longitud w, se toma la fila de la matriz asociada a dicha longitud. Inicialmente, todas las posiciones de la fila están vacías, se hace una búsqueda por todas las secuencias de entrada para evaluar todas las subsecuencias de longitud w, como posible cadena de consenso se cada punto de partida candidato.

Esto se representará en el Algoritmo 1 con la función `tomarFila(`w, MPP`)`, véase la línea 11 del pseudocódigo.

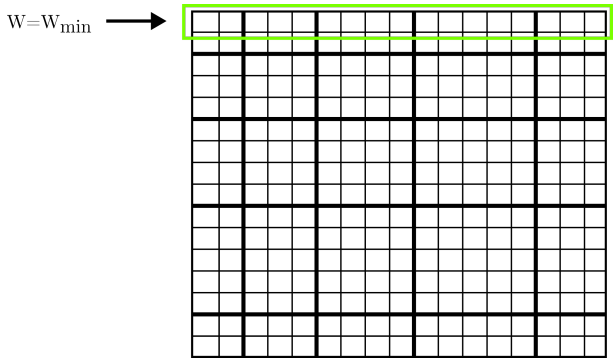


Figura 5.5: Se toma una fila de la matriz

Para buscar la cadena de consenso adecuada se recorren todas las secuencias de entrada tomando cada una de las subsecuencias de longitud w. Se representan cada una de las subsecuencias por $X_{i,j}$ siendo $i, 1 \leq i \leq N$, el índice de la secuencia de entrada de donde se extrae la subsecuencia, y, $j, 1 \leq j \leq m$, la posición de la secuencia Y_i donde empieza la subsecuencia de longitud w.

Esto se representará en el en el Algoritmo 1 con las funciónes `partirSecuenciasEntrada(`w, Y`);` y `tomarCandidato(`w, i, j`)`, véanse las línea del pseudocódigo 10 y 13 respectivamente.

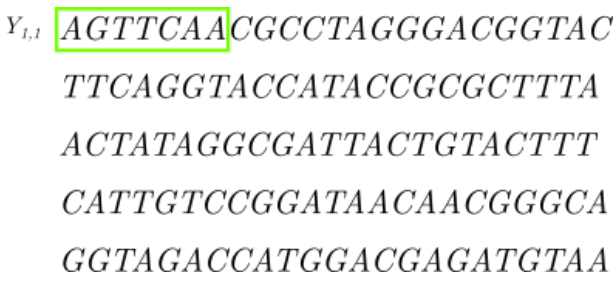


Figura 5.6: Se elige una subsecuencia de longitud w

Calcular modelo

Una vez escogida la subsecuencia, se calcula la matriz θ_1 que representa el modelo del motivo. Para ello, se parte de otra matriz llamada matriz de las frecuencias del alfabeto que nosotros denotaremos por M.

Esta matriz es de dimensión $L \times L$ –siendo L la longitud del alfabeto. Cada columna es un vector de probabilidades que representa cada símbolo del alfabeto,

$$M = \left[\begin{array}{cccc} \mathbf{P}_{a_1} & \mathbf{P}_{a_2} & \cdots & \mathbf{P}_{a_L} \end{array} \right].$$

Por ejemplo, en el caso de secuencias de ADN,

$$M = \left[\begin{array}{cccc} \mathbf{P}_a & \mathbf{P}_g & \mathbf{P}_c & \mathbf{P}_t \end{array} \right]$$

Ahora se rellenan las probabilidades de las columnas, se toman $\mathbf{P}_{a_i}$ y una constante α que puede ser definida por el usuario y que por defecto se le da el valor de $\alpha = 0{,}5$, a la letra representada por la columna a_i se le da la probabilidad de $\frac{1+\alpha}{1+L\alpha}$ y al resto se le da la probabilidad de $\frac{\alpha}{1+L\alpha}$.

El uso de α tiene como objetivo evitar que aparezcan probabilidades nulas que dificulten los cálculos más adelante.

$$\mathbf{P}_{a_i} = \begin{bmatrix} P_{a_1} = \frac{\alpha}{1+L\alpha} \\ P_{a_2} = \frac{\alpha}{1+L\alpha} \\ \vdots \\ P_{a_i} = \frac{1+\alpha}{1+L\alpha} \\ \vdots \\ P_{a_L} = \frac{\alpha}{1+L\alpha} \end{bmatrix}$$

De nuevo, en caso de secuencias de ADN y con $\alpha = 0{,}5$ la matriz que se obtiene es

$$M = \begin{bmatrix} \mathbf{P}_a & \mathbf{P}_g & \mathbf{P}_c & \mathbf{P}_t \end{bmatrix} = \begin{bmatrix} \frac{3}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{3}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{3}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{3}{6} \end{bmatrix}$$

Para calcular el modelo θ_1 de la subsecuencia a partir de la matriz de frecuencias del alfabeto, se crea una matriz con w columnas y se sustituye la columna i por la columna de frecuencias de la letra que se encuentre en la posición i de la matriz.

Por ejemplo, para la cadena

$$AGTTCAA$$

se obtiene el modelo

$$\theta_1 = \begin{bmatrix} \frac{3}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{3}{6} & \frac{3}{6} \\ \frac{1}{6} & \frac{3}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{3}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{3}{6} & \frac{3}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}.$$

Esto se representará en el en el Algoritmo 1 con la función `calcularModelo(`C, α`)`, véase la línea 14 del pseudocódigo.

Calcular la matriz de probabilidades PX

Una vez se ha calculado θ_1, se calcula $P(X_{i,j}|\theta_1)$, para todas las subsecuencias del conjunto de datos de entrada. Por así decirlo, una vez calculado el modelo de la subsecuencia elegida, se calculan las probabilidades de que dicho modelo hubiese generado cada una de las subsecuencias.

Esto da una matriz PX de dimensión $N \times m$ donde la posición (i, j) contiene $P(X_{i,j}|\theta_1)$.

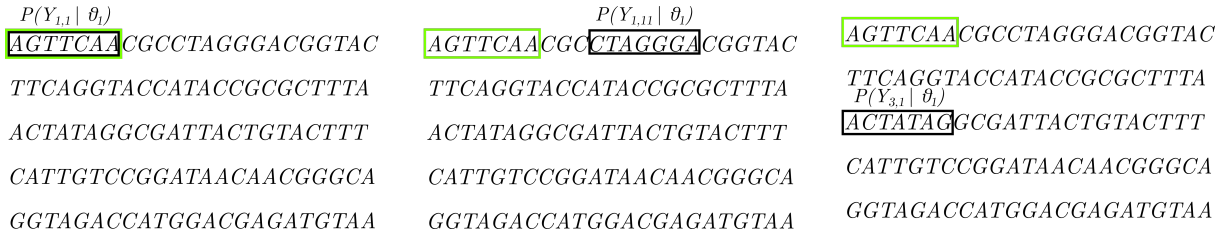


Figura 5.7: Cálculo de PX

En el cálculo de $P(X_{i,j}|\theta_1)$ se supone independencia condicional, es decir, la ocurrencia de un símbolo es independiente del resto, y la probabilidad se puede expresar como el producto de las probabilidades marginales asociadas a cada símbolo y posición, especificadas según el modelo θ_1.

Por ejemplo, si se toma la matriz θ_1 calculada anteriormente a partir de la cadena $AGTTCAA$ y la probabilidad para la cadena $ACTTCGA$ es

$$P(ACTTCGA|\theta_1) = \frac{3}{6} \cdot \frac{1}{6} \cdot \frac{3}{6} \cdot \frac{3}{6} \cdot \frac{3}{6} \cdot \frac{1}{6} \cdot \frac{3}{6} = 0{,}00086805$$

Esto se puede interpretar como la búsqueda de ocurrencias, tomando la subsecuencia como cadena de consenso de un posible motivo. Al calcular estas probabilidades, en esencia, se está calculando, aunque no exactamente, la probabilidad de que cada una de las subsecuencias sean ocurrencias de dicho motivo.

Es aquí donde, en caso de que el usuario hubiese proporcionado un PSP, se suman los sesgos a las posiciones que les correspondan. Recuerdese que $PSP_{i,j}$ introduce un sesgo a la probabilidad de que $X_{i,j}$ sea una ocurrencia, es decir,

$$PX_{i,j} = P(X_{i,j}|\theta_1) \cdot PSP_{i,j}.$$

Esto se representará en el en el Algoritmo 1 con la función `calcularPX(`$X_{i,j}, \theta_1, PSP$`)`, véase la línea 16 del pseudocódigo.

Calcular puntuación de las semillas

Ahora se debe calcular la puntuación de nuestro motivo teórico, se calcula para cada uno de los puntos de partida –pares (w, k)– de la fila de la matriz correspondiente a la longitud w. Para ello se recorre la lista de números de ocurrencias en sentido ascendente, de k_{min} a k_{max}, y, una vez fijado k, se calcula la puntuación en base a dicho k.

Se deben seleccionar las k subsecuencias que presentan una mayor probabilidad de ser ocurrencias de nuestro motivo teórico, y se calcula la puntuación en base a estas. Se usa una función objetivo para dar una puntuación a dicha cadena, que se interpreta como un valor de cómo de buena sería esa cadena si fuese un motivo.

Independientemente de considerar el sesgo o no en el cálculo de $PX_{i,j}$, sí hay que tener en cuenta el modo de ejecución elegido a la hora de calcular la subsecuencia de mayor probabilidad condicionada al modelo actual. En los modos OOPS y ZOOPS, al considerarse solo una ocurrencia por secuencia, esto no supone ningún problema, pues basta con tomar el máximo de cada una de las secuencias.

En ANR, al poder haber más de una ocurrencia por secuencia, el algoritmo debe tener algo más de cuidado al tomar estos máximos, se tienen que tomar máximos locales en entornos de longitud w. Esto quiere decir que no se puede tomar un máximo que esté a una longitud menor o igual que w de otro máximo que se haya tomado anteriormente. De otra manera se estaría permitiendo que las ocurrencias estuvieran superpuestas.

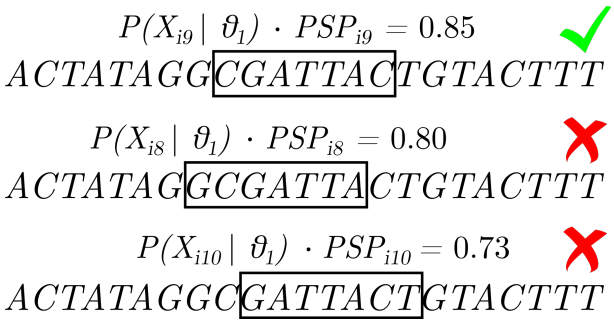


Figura 5.8: Máximos locales

Esto se representará en el en el Algoritmo 1 con la función `calcularPuntuaciónSemilla` (C, k, PX), véase la línea 18 del pseudocódigo.

En caso de haber elegido, como función objetivo, el Enriquecimiento Diferencial o el Enriquecimiento Diferencial Selectivo, durante el cálculo de la puntuación se intenta optimizar el valor de k a aquel que obtenga una mayor puntuación. Es aquí donde interviene el sesgo al número de ocurrencias que tiene almacenado el punto de partida.

Añadir semilla al *heap*

La cadena de consenso, su posición en el conjunto de secuencias de entrada, su puntuación y su k, optimizado o no, se consideran una semilla candidata para el punto de partida (w, k) y, en caso de tener una puntuación lo suficientemente alta se guarda en el *heap* de este.

Esto se representará en el en el Algoritmo 1 con la función `actualizarHeap` (F, k, PC), véase la línea 19 del pseudocódigo.

Siguiente iteración

Acto seguido se pasa a la siguiente subsecuencia de longitud w, que se toma como una hipotética cadena de consenso, y se repite el proceso de búsqueda de ocurrencias y de cálculo de la puntuación para cada punto de partida de la fila.

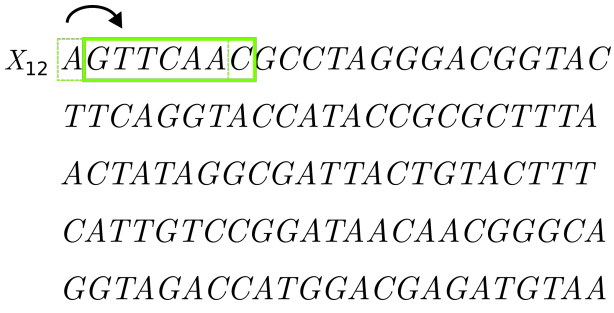


Figura 5.9: Se pasa a la siguiente subsecuencia de longitud w

Una vez recorridas todas las subsecuencias de longitud w de las secuencias de inicio, se pasa a la siguiente fila de la matriz –se elige un nuevo w– con la que se repetirá el proceso hasta haber recorrido toda matriz.

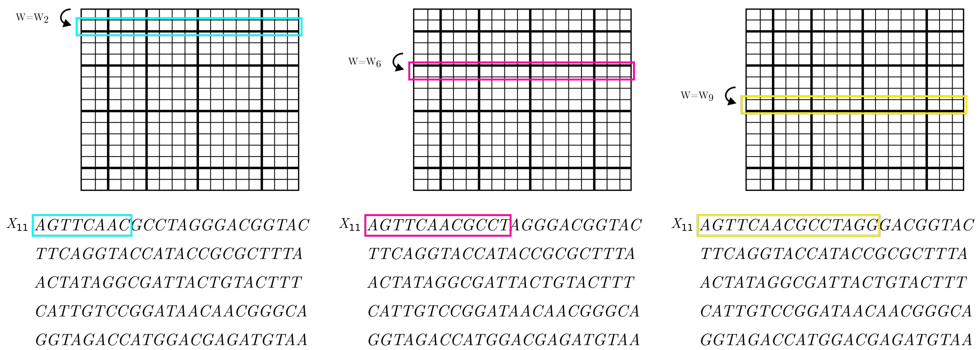


Figura 5.10: Se toma una nueva longitud, con ello se pasa de fila y se amplia la ventana de búsqueda

Tras recorrer toda la matriz hay un *heap* en cada punto de partida con las mejores semillas candidatas a ser la cadena de consenso de dicho punto.

Pseudocódigo de la búsqueda global

Algorithm 1: Búsqueda global

```
input  : Y = conjunto de subsecuencias de entrada;
         PSP = Position Specific Prior;
         α = parámetro del modelo;
output: {{Heap_{w,k}}}
W = crearListaLongitudes();
K = crearListaNúmerosOcurrencias();
MPP = crearMatrizPuntosPartida();
foreach w en W do
    foreach k en K do
        Heap_{w,k} = crearHeap();
    end
end
foreach w en W do
    X = partirSecuenciasEntrada(w, Y);
    F = tomarFila(w, MPP);
    foreach subsecuencia X_{i,j} de longitud w en X do
        C = tomarCandidato(X_{i,j});
        θ_1 = calcularModelo(C, α);
        foreach subsecuencia X_{k,l} de longitud w en X do
            PX = calcularPX(X_{k,l}, θ_1, PSP);
            foreach k en K do
                PC = calcularPuntuaciónSemilla(C, k, PX);
                actualizarHeap(Heap_{w,k}, F, k, PC);
            end
        end
    end
end
return {{Heap_{w,k}}}
```

5.4.5. Mutaciones

Este paso lo realiza MEME por defecto, pero es opcional y se puede obligar al algoritmo a no hacerlo. Aquí MEME realiza mutaciones a las cadenas de consenso de las semillas almacenadas en cada uno de los *heaps*. Esta estrategia se hace para buscar una que obtenga una puntuación mayor, pero que no se haya encontrado por no ser subsecuencia en el conjunto de secuencias iniciales.

En este proceso se llevan a cabo varias iteraciones sobre un *heap* global que se crea agrupando los *heaps* de la matriz del espacio de búsqueda.

Al comenzar cada iteración se crea el *heap* global, ordenado según la puntuación de cada semilla, y se recorren todas las semillas que están almacenadas haciendo mutaciones de la cadena de consenso de cada una y evaluando cada mutación. Por lo general se realizan todos los tipos de mutaciones de forma sistemática, a no ser que se le pida al algoritmo que no realice las de un tipo o las de otro.

Hay varios tipos de mutaciones que se llevan a cabo a una cadena de consenso, unas que varían las letras de la misma y otras que varían la longitud, dada la cadena

$$AGTTCAACG$$

las mutaciones que se realizan sobre ella son

- **Mutación de letra individual**

  Se recorren todas las posiciones de la cadena y se sustituye la letra de dicha posición con las restantes letras del alfabeto. Así, mutando la primera posición se obtiene

  $$GGTTCAACG, CGTTCAACG, TGTTCAACG,$$

  mutando la segunda

  $$AATTCAACG, ACTTCAACG, ATTTCAACG,$$

  y así hasta la última posición de la cadena.

- **Eliminar la primera letra**

  $$GTTCAACG$$

- **Eliminar la última letra**

  $$AGTTCAAC$$

- **Añadir letra al principio**

  $$AAGTTCAACG, GAGTTCAACG,$$
  $$CAGTTCAACG, TAGTTCAACG$$

- **Añadir letra al final**

  $$AGTTCAACGA, AGTTCAACGG,$$
  $$AGTTCAACGC, AGTTCAACGT$$

En caso de que alguna de estas mutaciones no se haya encontrado en la primera búsqueda a través de las secuencias de entrada se evalúa y se añade a los puntos de partida de la fila de la matriz que corresponda a su longitud.

5.4.6. Selección de los mejores puntos de partida

Por último, cada punto de partida toma los datos de la mejor semilla de su *heap*, y la puntuación de esta pasa a ser la puntuación del punto de partida.

Se escoge el mejor punto de partida de cada partición de la matriz. El cálculo del mejor punto de partida atiende al cálculo del E-valor del punto de partida como motivo teórico. Este cálculo del E-valor depende de la función objetivo que se haya escogido. Con la función clásica, se toma el producto de los P-valores de las ocurrencias. Para la función *Numerically Correct* se toma el p-valor del producto de los LLR de sus ocurrencias sesgado por un coeficiente que depende del número de ocurrencias, esto se calcula según [8]. Para el resto de funciones se toma la puntuación del punto de partida en negativo.

Como el E-valor es la medida de la significación estadística de un motivo, de cada partición se toma aquel punto de partida con el E-valor más pequeño. Estos pocos elegidos serán desde los que el algoritmo EM empieza a buscar el motivo, optimizando su modelo de probabilidad conocido.

5.5. Algoritmo EM

El algoritmo *Expectation Maximization* (EM) (o maximización de la esperanza) trata de encontrar para: un modelo de mezcla, que combina varios modelos con una misma distribución, un conjunto de datos dado y las variables que asignan la pertenencia a un grupo o a otro, una serie de parámetro de máxima verosimilitud para el modelo, es decir, parámetros que permitan describir los datos de la mejor manera posible.

En nuestro caso los datos conocidos son $X = \{X_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$, todas las subsecuencias de longitud w en las que se ha partido en conjunto de secuencias de entrada, los modelos son dos, a saber, el modelo del motivo que busca las ocurrencias y el modelo *background*, y juntos forman el modelo θ. θ se crea al inicio del algoritmo, a partir de la cadena de consenso del punto de partida, de la misma manera a la que se explica en la Sección 5.4.4.

Como se ha visto en la Sección 4.2.4 se definen las variables auxiliares

$$Z = \{Z_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$$

con

$$Z_{i,j} = \begin{cases} 1 \text{ si hay una ocurrencia en la posición } j \text{ de } X_i, \\ 0 \text{ si no la hay.} \end{cases}$$

reflejándolo según los términos de pertenencia a un modelo o a otro,

$$Z_{i,j} = \begin{cases} 1 \text{ si se distribuye según } \theta_1, \\ 0 \text{ si se distribuye según } \theta_0. \end{cases}$$

También se define el parámetro de mezcla λ, según el modo de ejecución

$$\lambda = \begin{cases} \lambda = \dfrac{1}{m} \text{ si se ejecuta en modo OOPS,} \\ \lambda = \dfrac{\gamma}{m} \text{ si se ejecuta en modo ZOOPS,} \\ \lambda \text{ es el propio hiperparámetro que se le da a MEME ,} \end{cases}$$

que mezcla los modelos θ_0 y θ_1, y determina la probabilidad de que una subsecuencia se distribuya según θ_0 o según θ_1.

La verosimilitud de los parámetros θ, λ del modelo dada la distribución de los datos X y los datos ocultos Z se define como

$$L(\theta, \lambda | X, Z) = P(X, Z | \theta, \lambda).$$

Téngase en cuenta que, en caso de haber facilitado el usuario un PSP, el sesgo de este se multiplica a la probabilidad dada por

$$P(X_i|Z_{i,j}=1,\theta),$$

por ello se define,

$$\hat{P}(X_i|Z_{i,j}=1,\theta)=\begin{cases}P(X_i|Z_{i,j}=1,\theta)\cdot PSP_{i,j} & \text{si el usuario ha introducido un } PSP,\\ P(X_i|Z_{i,j}=1,\theta) & \text{si no lo ha hecho.}\end{cases}$$

En la práctica se trabaja con la log-verosimilitud porque facilita los cálculos, y se trabaja con una función diferente para cada uno de los tres modos de ejecución:

1. OOPS

$$\log P(X,Z|\theta)=\sum_{i=1}^{n}\sum_{j=1}^{m}Z_{i,j}\log\hat{P}(X_i|Z_{i,j}=1,\theta)+n\log\frac{1}{m}.$$

   Aquí la probabilidad condicionada para las secuencias que contienen una ocurrencia, se define como

$$\log P(X_i|Z_{i,j}=1,\theta)=\sum_{k=0}^{W-1}\log P_{X_{i,j+k},k+1}+\sum_{k\in\Delta_{i,j}}\log P_{X_{i,k}0},$$

   con $\Delta_{i,j}=\{1,2,\ldots,j-1,j+W,\ldots,l\}$, las posiciones de la secuencia X_i que en caso de que una ocurrencia empezase en la posición j estarían fuera de esta.

   Para facilitar la interpretación recordemos que $P_{X_{i,j+k},k+1}$ es la probabilidad de que la letra que se encuentra en la posición $j+k$ de la secuencia i este en la posición $k+1$ de una ocurrencia del motivo.

2. ZOOPS

$$\begin{aligned}\log P(X,Z|\theta,\gamma)=&\sum_{i=1}^{n}\sum_{j=1}^{m}Z_{i,j}\log\hat{P}(X_i|Z_{i,j}=1,\theta)\\&+\sum_{i=1}^{n}(1-Q_i)\log P(X_i|Q_i=0,\theta)\\&+\sum_{i=1}^{n}(1-Q_i)\log(1-\gamma)+\sum_{i=1}^{n}Q_i\log\frac{\gamma}{m}.\end{aligned}$$

   Aquí la variable Q_i se define como $Q_i=\sum_{j=1}^{m}Z_{i,j}$. Es decir

$$Q_i=\begin{cases}1 \text{ si hay una ocurrencia en } X_i,\\ 0 \text{ si no la hay,}\end{cases}$$

   y la probabilidad condicionada de que una secuencia no contenga ninguna ocurrencia se define como

$$P(X_i|Q_i=0,\theta)=\prod_{k=1}^{l}P_{X_{i,k}0}.$$

   También se usa en esta fórmula $\log P(X_i|Z_{i,j}=1,\theta)$ con el mismo significado y la misma definición que en OOPS.

3. ANR

$$\log P(X, Z|\theta, \lambda) = \sum_{i=1}^{n}\sum_{j=1}^{m}(1 - Z_{i,j}) \log P(X_{i,j}|\theta_0) + Z_{i,j} \log \hat{P}(X_{i,j}|\theta_1) + (1 - Z_{i,j}) \log(1 - \lambda) + (Z_{i,j}) \log \lambda.$$

La probabilidad condicionada de que una secuencia haya sido generada por el modelo *background* o por el modelo del motivo se definen como

$$\log P(X_{i,j}|\theta_0) = \sum_{k=0}^{W-1} P_{X_{i,j+k},0},$$

y,

$$\log \hat{P}(X_{i,j}|\theta_1) = \begin{cases} \sum_{k=0}^{W-1} P_{X_{i,j+k},k+1} + PSP_{ij}, & \text{si el usuario ha introducido un } PSP, \\ \sum_{k=0}^{W-1} P_{X_{i,j+k},k+1}, & \text{si no lo ha hecho.} \end{cases}$$

respectivamente.

El proceso de optimización es iterativo y, en cada iteración, se llevan a cabo dos pasos. Se acaba cuando los cambios no sean apreciables –sean menores que un umbral dado por el usuario– o cuando se hayan hecho un número máximo de iteraciones.

5.5.1. E-paso

El E-paso calcula el valor esperado de las variables auxiliares Z.

De nuevo, dependiendo del modo de ejecución del algoritmo se utilizará una fórmula u otra para actualizar los datos ocultos.

Suponiendo que se está realizando la iteración t-ésima:

1. OOPS

$$Z_{i,j}^{(t)} = E(Z_{i,j}|\hat{X}, \theta^{(t)}) = \frac{\hat{P}(X_i|Z_{i,j} = 1, \theta^{(t)})}{\sum_{k=1}^{m} \hat{P}(X_i|Z_{i,k} = 1, \theta^{(t)})}$$

2. ZOOPS

$$Z_{i,j}^{(t)} = E(Z_{i,j}|\hat{X}, \theta^{(t)}, \gamma^{(t)}) = \frac{f_i}{f_0 + \sum_{k=1}^{m} f_k}, \text{ donde}$$

$$f_0 = P(X_i|Q_i = 0, \theta^{(t)})(1 - \gamma^{(t)}), \text{ y}$$

$$f_j = \hat{P}(X_i|Z_{i,j} = 1, \theta^{(t)})(\frac{\gamma^{(t)}}{m}), \quad 1 \leq j \leq m.$$

3. ANR

$$Z_{i,j}^{(t)} = E(Z_{i,j}|\hat{X}, \theta^{(t)}, \lambda^{(t)}) = \frac{\hat{P}(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}{P(X_{i,j}|\theta_0^{(t)})(1 - \lambda^{(t)}) + \hat{P}(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}$$

5.5.2. M-paso

Por último, en el M-paso se reestiman los hiperparámetros del modelo usando $Z_{i,j}^{(t)}$ calculado en el E-paso, con el objetivo de maximizar la función de log-verosimilitud.

Para reestimar θ –y obtener así $\theta^{(t+1)}$– se usa la misma fórmula para los tres modos

$$P_{a_j,k}^{(t+1)} = \frac{c_{a_j,k} + d_{a_j,k}}{\sum_{p=1}^{L} c_{a_p,k} + d_{a_p,k}}, \text{ donde}$$

$$c_{a_j,k} = \begin{cases} t_{a_j} - \sum_{j=1}^{W} c_{a_j,k} & \text{si } k = 0, \\ \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} I(a_j, X_{i,j+k-1}) & \text{en otro caso.} \end{cases}$$

Con $d_{a_j,k}$ un pseudo contador que se usa para incorporar información sobre el *background* –todas las posiciones fuera de la ocurrencia del motivo–, t_{a_j} es un contador de todas las ocurrencias de a_j en el conjunto de datos e $I(a_j, X_{i,j+k-1})$ una función que es 1 si $a_j = X_{i,j+k+1}$ y es 0 en otro caso.

En los modelos ZOOPS y ANR los parámetros λ, γ se reestiman mediante la fórmula

$$\lambda^{(t+1)} = \frac{\gamma^{(t+1)}}{m} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)}.$$

5.6. Puntuación de los mejores candidatos y selección del mejor modelo

Una vez acabado el algoritmo EM se evalúan cada uno de los modelos optimizados calculando el E-valor de cada uno, tal y como se explica en la Sección 5.2.8. Se toma el modelo con el menor E-valor, ya que este es el mejor de todos y se toma como el motivo definitivo devuelto por el algoritmo MEME.

5.7. Búsqueda de múltiples motivos

Para buscar múltiples motivos en las secuencias, MEME lleva a cabo una búsqueda voraz –en cada paso de la búsqueda, se elige una solución óptima local con la esperanza de que al final de esta, el conjunto de las soluciones escogidas forme una solución óptima global. En nuestro caso, cada paso se corresponde con la búsqueda de uno de los motivos.

Es importante notar que no se devuelven, como resultado de la búsqueda de varios motivos, los mejores modelos de la primera búsqueda, sino que se realizan varias búsquedas sucesivas y se toma el mejor modelo de cada una.

La búsqueda de cada motivo, lleva incorporada información sobre las búsquedas anteriores para evitar descubrir motivos ya encontrados.

Inicialmente MEME usa la información contenida en el PSP, en caso de que se haya proporcionado uno, para evaluar la probabilidad de que haya una ocurrencia en una posición de la secuencia o en otra. En caso de no haber proporcionado un PSP, MEME considera que todas las posiciones de la secuencia son igualmente probables para ser el inicio de una ocurrencia.

En las siguientes búsquedas a partir de la primera, MEME incorpora información de lo que ya ha encontrado. Para ello, al calcular el valor esperado de los datos ocultos, $Z_{i,j}$, en el E-paso se tiene en cuenta la probabilidad de que una ocurrencia que empiece en la posición $X_{i,j}$, pueda superponerse con la ocurrencia de un motivo encontrado previamente.

Para calcular esta probabilidad se introduce una nueva variable

$$V_{i,j} = \begin{cases} 1, \text{ si no hay una ocurrencia en } [X_{i,j}, \ldots, X_{i,j+W-1}], \\ 0, \text{ en otro caso.} \end{cases}$$

para $i = 1, \ldots, n$ y $j = 1, \ldots, l$.

Para calcular $V_{i,j}$ se usa otro conjunto de variables binarias

$$U_{i,j} = \begin{cases} 1, \text{ si } X_{i,j} \text{ no pertenece a una ocurrencia encontrada previamente,} \\ 0, \text{ en otro caso.} \end{cases}$$

para $i = 1, \ldots, n$ y $j = 1, \ldots, m$.

Antes de la búsqueda del primer motivo, es evidente que ninguna posición de ninguna secuencia está contenida en una ocurrencia, es decir, $U_{i,j}^{(0)} = 1$ para $i = 1, \ldots, n$ y $j = 1, \ldots, l$.

Como con $Z_{i,j}$, MEME calcula el valor esperado de $U_{i,j}$. y por cada búsqueda en el conjunto de secuencias se van actualizando de acuerdo a la fórmula

$$U_{i,j}^{(p)} = U_{i,j}^{(p-1)}(1 - \max_{k=j-W+1,\ldots,j} Z_{i,k}^{(t)}) \tag{5.1}$$

donde $Z_{i,k}^{(t)}$ es la estimación final de $Z_{i,j}$ al final de la búsqueda actual, p.

Este máximo se debe a que varias ocurrencias de un mismo motivo no pueden superponerse entre sí, por tanto los valores de $Z_{i,j}^{(t)}$ no son independientes. El valor de $U_{i,j}^{(p)}$ es el que luego se usa para calcular $P(U_{i,j} = 1)$ en (5.2) para la siguiente búsqueda, $p + 1$.

Así, MEME calcula la probabilidad de que una ocurrencia, de longitud w, no se superponga a ninguna de las ocurrencias encontradas para motivos en las anteriores búsquedas, como la mínima de las probabilidades de que cada posición pertenezca a una ocurrencia encontrada en la búsqueda anterior. Es decir, MEME define $P(V_{i,j} = 1)$ como

$$P(V_{i,j} = 1) = \min_{k=j,\ldots,j+W-1} P(U_{i,k} = 1). \tag{5.2}$$

De nuevo los valores de $U_{i,j}$ no son independientes y por eso se toma el mínimo.

Ahora, a partir de la primera búsqueda, MEME utiliza información de lo que ha encontrado en las anteriores. Así la fórmula para reestimar $Z_{i,j}$ en el E-paso se cambia a

$$\hat{Z}_{i,j}^{(t)} = \underset{(Z|X,\phi)}{E}[Z_{i.j}]P(V_{i,j} = 1).$$

MEME usa $\hat{Z}_{i,j}^{(t)}$ en el M-paso del algoritmo EM y en la ecuación (5.1), que figura más arriba.

Capítulo 6

Caso de estudio

Es este capítulo se hará una breve guía de instalación y uso, y, se mostrarán los resultados de ejecutar el algoritmo MEME en dos ejemplos.

6.1. Requisitos previos de instalación

El único requisito previo para la instalación de la Suite MEME, si se sigue la guía de instalación del siguiente apartado, es la instalación de Docker.

Docker es una plataforma gratuita que permite el desarrollo, la distribución y la ejecución de aplicaciones[1]. Docker funciona en base a *contenedores Docker* que permiten almacenar todo lo que necesita una aplicación para ejecutarse. Una vez se ha creado el contenedor este se puede distribuir con facilidad a cualquier sistema sin más requisito que este tenga instalado Docker[2].

6.2. Breve guía de instalación y uso

Aunque se puede usar el algoritmo, directamente, desde la página oficial de Suite MEME [12]. Se ofrecen alternativas para la instalación, se dan tres opciones para la instalación de la Suite MEME.

1. Instalar un contenedor de Docker, si se usa macOS o Windows.
2. Instalar en macOS a través de MacPorts.
3. Descargar e instalar directamente el código fuente. En este caso, para instalar en Windows es necesario tener una máquina virtual de Linux.

En esta memoria de TFG se docimenta la instalación a través de Docker. Si se escoge esta opción, el usuario es redirigido a una página, con una breve guía de instalación y arranque, constituida por dos sencillos pasos.

1. Se ejecuta en la consola el comando

   ```
   docker pull memesuite/memesuite:latest
   ```

   esto hará que Docker descargue la imagen más reciente de Suite MEME del repositorio.

[1] https://docs.docker.com/get-started/overview/
[2] https://docs.docker.com/get-docker/

2. Una vez descargada la imagen, se ejecuta en la consola el comando

   ```
   docker run -d -p 8080:8080 memesuite/memesuite
   ```

   lo que hará que la imagen se ejecute en el servidor web que haya instalado la misma imagen, cuando haya acabado la ejecución de esta, Suite MEME estará disponible a través del navegador en https://localhost:8080/meme_*versión*, la versión usada en este trabajo es la 5.3.3 por lo que la dirección en el navegador es https://localhost:8080/meme_5.3.3.

Una vez abierta la dirección en el navegador aparece la página principal de la Suite MEME.

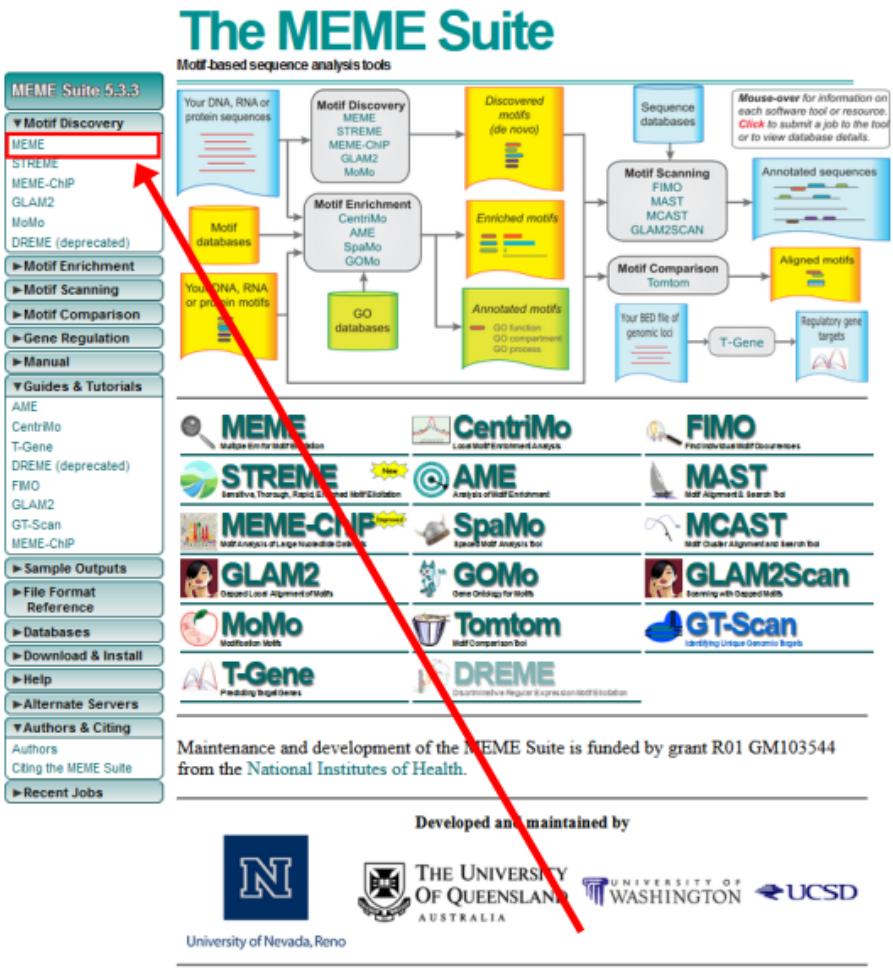


Figura 6.1: Página principal de la Suite MEME

En la interfaz se están presentes todas las funciones que ofrece la Suite MEME, y los diferentes algoritmos disponibles para llevar a cabo estas tareas. Se distinguen:

- *Motif Discovery*: Búsqueda de motivos no conocidos, a partir de secuencias introducidas por el usuario.
- *Motif Enrichment*: Mejora de un modelo conocido de un motivo usando nuevas secuencias dadas por el usuario.

- *Motif Scanning*: Búsqueda, en secuencias dadas por el usuario, de ocurrencias de motivos conocidos.
- *Motif Comparison*: Comparar motivos encontrados con otros almacenados en bases de datos como JASPAR [9] o MyHits [11].
- *Gene Regulation*: Predicción de qué genes tienen regulada su expresión por los motivos dados por el usuario .

Además, en el menú de la izquierda se distinguen algunos apartados auxiliares, algunos de manuales, guías de uso y tutoriales, ejemplos de formato de salida, bases de datos, documentación ,etc.

Para acceder al algoritmo MEME hace falta dirigirse a la sección *Motif Discovery* del menú de la izquierda. Una vez en la página del algoritmo MEME aparece la interfaz de uso del mismo.

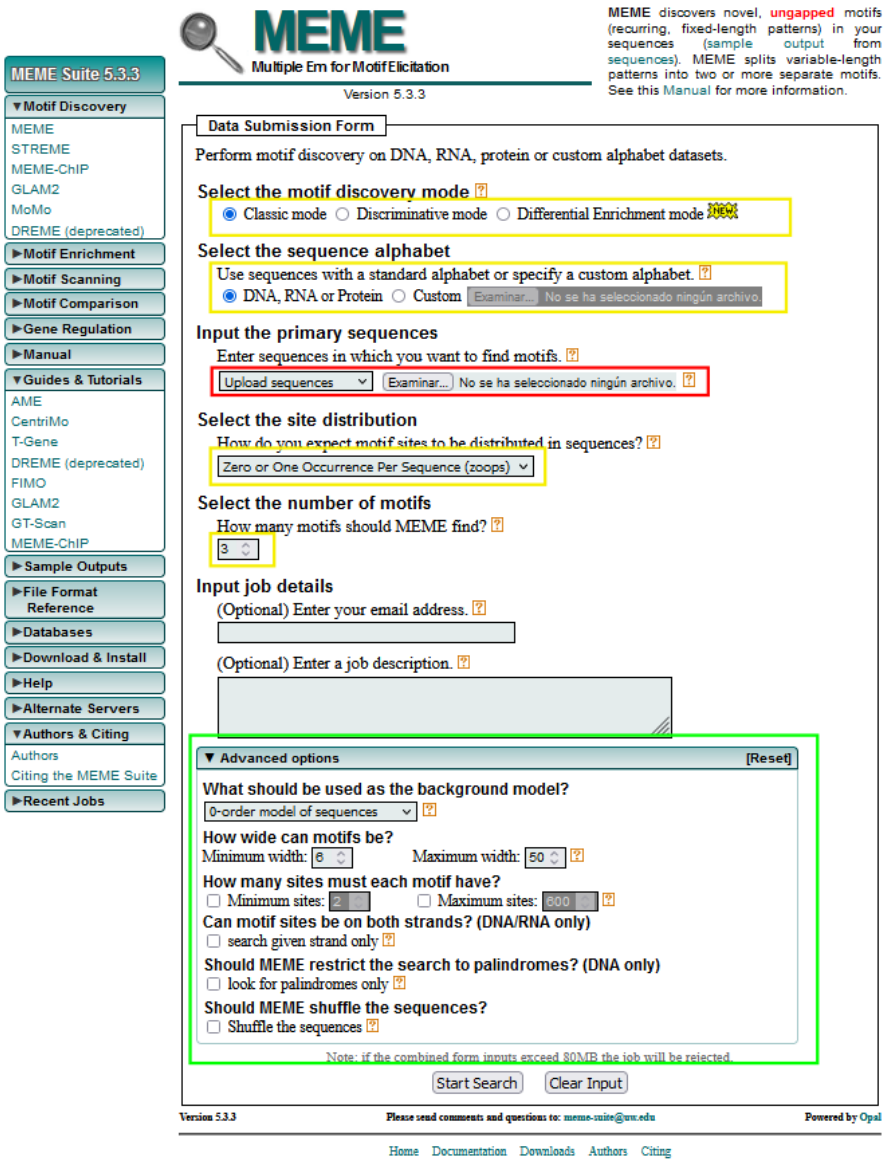


Figura 6.2: Interfaz de uso del algoritmo

En la Figura 6.2 se distinguen, en rojo, una ventana para introducir las secuencias de entrada, en amarillo, las opciones básicas que se han ido comentando a lo largo del trabajo, como el alfabeto en el que están escritas las secuencias, la función objetivo que utilizar en la sección **Select the motif discovery mode**, el modo de ejecución, OOPS, ZOOPS o ANR, y, cuántos motivos buscar. Por último, en verde, aparecen las opciones avanzadas.

Hay que tener en cuenta que esta interfaz es muy básica para poder acceder al conjunto total de opciones hay que ejecutar el algoritmo desde la consola, en la documentación hay una guía para ello con explicaciones de todas las opciones disponibles, a la que también se puede acceder desde la documentación de la página oficial [12].

6.3. Ejemplos de ejecución

A continuación se muestran dos ejemplos de búsqueda sobre conjuntos de secuencias reales que contienen motivos conocidos, ambos ejemplos se han tomado de la base de datos JASPAR [9].

JASPAR es una base de datos de lugares de unión de factores de transcripción encontrados en secuencias de diferentes seres vivos. Se puede descargar información del motivo, el modelo, el logo de secuencia y una serie de secuencias donde se encuentran ocurrencias del motivo.

Las secuencias de entrada se encuentran en un archivo tipo FASTA, que tiene un formato

```
>Nombre_de_la_secuencia
Secuencia
```

por ejemplo, algunas de las secuencias descargadas para la búsqueda del motivo Atoh1 son

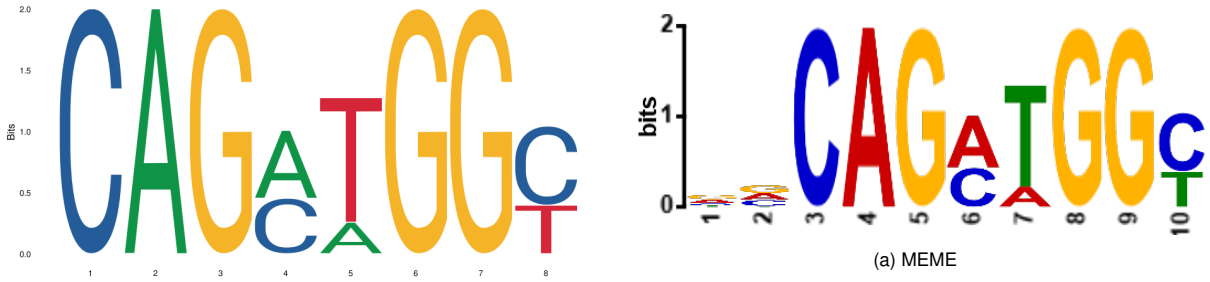
```
>mm9_chr1:6473584-6473591(-)
gagaagagtgctgagccaggcgctcctctgcaagtgcatcctccaagatccagatggcatggcctttctttgcta
ttgccacctggcaccactgcctcttcagcgccc
>mm9_chr1:6729457-6729464(-)
cttgaaaggttatgagagttagagaattttaaccatgcctgccaaatcagcagatggcttgtgattataattata
tgcatcaataaaacgaataatctttacaaaatt
>mm9_chr1:6845801-6845808(+)
ctcctcctatccctagctggcatagaagccccttgatgagcgaatagagcagatggcCcatgtagactccggta
gagcgtgtggaactttatagagaacttgtcaaaa
```

En ambos búsquedas se ha utilizado la función objetivo Clásica, con modo de ejecución ZOOPS, y, se ha encargado la búsqueda de un solo motivo.

6.3.1. Motivo Atoh1

El motivo Atoh1 interviene en la neurogénesis, la creación de nuevas células neuronales.

En este caso se ha pedido buscar un motivo de longitud mínima 6 y longitud máxima 10, se han introducido 11205 secuencias, con una longitud media de 105 bases nitrogenadas. El logo de secuencia obtenido se muestra a continuación.



(a) MEME

Figura 6.3: Logo de secuencia del motivo Atoh1 y logo de secuencia calculado por MEME para el mismo motivo

El E-valor calculado por MEME para el motivo es $2,5 \cdot 10^{-1567}$.

6.3.2. Motivo AR

El motivo AR (*Androgen Receptor*) está ligado a la regulación de la expresión de los genes que afectan a la diferenciación de los tejidos.

En este caso se ha pedido buscar un motivo de longitud mínima 6 y longitud máxima 15, se han introducido 7714 secuencias, con una longitud media de 108 bases nitrogenadas. El logo de secuencia obtenido se muestra a continuación.

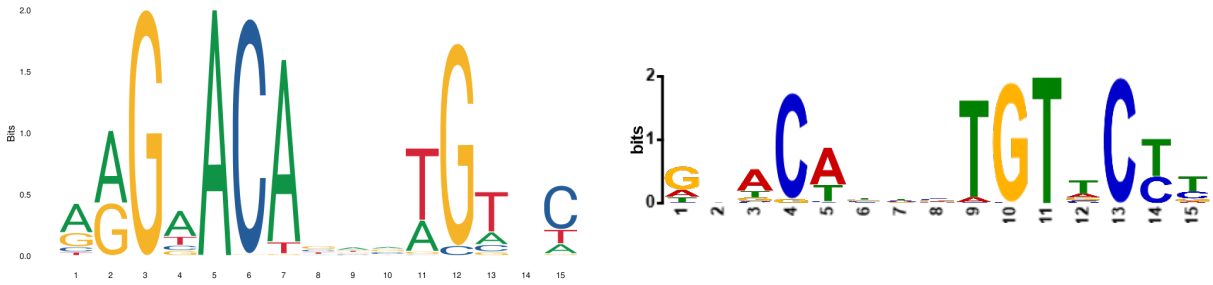


Figura 6.4: Logo de secuencia del motivo AR y logo de secuencia calculado por MEME para el mismo motivo

El E-valor calculado por MEME para el motivo es $5{,}6 \cdot 10^{-1404}$. Se puede apreciar que, aunque MEME ha encontrado un modelo que crear un logo de secuencia que, aunque se aproxima al original en algunas posiciones básicas, difiere en muchas otras.

Capítulo 7

Conclusiones y posibles ampliaciones

A continuación se expondrán una serie de conclusiones técnicas sobre el trabajo, junto con posibles ampliaciones que se pueden hacer al algoritmo MEME para mejorar el rendimiento.

7.1. Conclusiones técnicas

7.1.1. Importancia de los algoritmos de búsqueda de motivos

La búsqueda de motivos es una parte fundamental del esfuerzo por entender la expresión genética, esfuerzo que está en el centro de las investigaciones dirigidas a mejorar las tecnologías médicas de las que disponemos. Sin embargo, el espacio donde se buscan los motivos, en la mayoría de los casos, está formado por cadenas de ADN con una longitud de miles de bases nitrogenadas, es decir, la búsqueda se vuelve una tarea muy costosa. Es aquí donde entran en juego los diferentes tipos de algoritmos que han ido surgiendo, cada uno con su propia estrategia, los algoritmos ayudan a reducir enormemente los tiempos de búsqueda por lo que, en el fondo, aceleran la investigación médica.

7.1.2. Qué aporta MEME a la búsqueda de motivos

Como se ha visto la idea particular en la que se basa el algoritmo *Multiple EM for Motif Elicitation* (MEME) es el ajuste del modelo estadístico del motivo mediante el uso del algoritmo *Expectation Maximization* (EM), un algortimo que permite calcular estimadores de los parámetros del modelo con máxima verosimilitud. Lo que MEME aporta realmente a la búsqueda es una buena selección de los puntos de partida. Al ser EM un algoritmo de búsqueda local los modelos ajustados no son siempre los mejores modelos posibles, mediante la selección de un conjunto de buenos puntos de partida MEME se asegura un mayor rango de exploración evitando quedarse atrapado en el primer máximo local que encuentre.

7.1.3. Dificultades encontradas a buscar documentación para el trabajo

Por último, destacar las grandes dificultades encontradas al buscar documentación sobre el algoritmo, los principales trabajos que tratan el tema, a saber [13], [14] y [15], son publicados por el profesor Timothy L. Bailey, quien propuso por primera vez el algoritmo. Sin embargo a partir de [15], publicado en 2010, no hay documentación oficial sobre el algoritmo y este ha seguido recibiendo actualizaciones por lo que ha sido necesario consultar directamente el código fuente para comprender a fondo el funcionamiento de MEME. Además, durante la fase de investigación se encontraron varios algoritmos mal documentados o cuyas referencias al material complementario no estaban disponibles.

7.2. Posibles ampliaciones

En este apartado se hacen propuestas de posibles ampliaciones que mejoren el funcionamiento del algoritmo.

7.2.1. Cambios en la búsqueda de puntos de partida

El proceso de selección de los mejores puntos de partida que lleva a cabo MEME es un proceso exhaustivo donde se consideran todas las posibles subsecuencias, del conjunto de secuencias de entrada, que tengan unas determinadas longitudes. Una posible mejora sería la utilización de una metaheurística que permita seleccionar los mejores puntos de partida de una manera más eficiente, usando los mismos criterios para evaluarles.

Un ejemplo sería aplicar una versión reducida de los métodos naturalistas que hemos visto en la Sección 3.1.3, *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), *Cuckoo Search* (CS), etc. La idea sería implementar el mismo método, que busque sobre el conjunto de datos, realizando un número de iteraciones bajo en comparación con las iteraciones que se realizan al buscar los modelos, de donde se obtendrá un modelo mínimamente optimizado que será considerado como un punto de partida desde el que ejecutar el algoritmo EM.

7.2.2. Ampliar la búsqueda fuera del conjunto de datos

La búsqueda de motivos tal y como está planteada se limita al conjunto de datos introducido, puede que falte información de otras cadenas que comparten motivo con las que han sido introducidas por lo que el motivo que se encuentre es parcial. Una posible mejora es la conexión del algoritmo a una base de datos de motivos con la que se pueda hacer una búsqueda de modelos de motivos ya conocidos a través de las secuencia y, en caso de haber modelos que se ajusten mínimamente bien, optimizar estos para refinar el motivo extraído de la base de datos.

7.3. Conclusiones personales

Para acabar me gustaría añadir algunas conclusiones personales para ilustrar lo que ha supuesto la realización de este trabajo. Lo más importante ha sido es que me ha permitido darme cuenta de todo lo que he ido aprendiendo a lo largo del grado. Sin la formación que he recibido en estos años de universidad me hubiese sido mucho más difícil abordar este trabajo. También he tenido que aprender a encontrar documentación y a buscar en las diferentes bases de datos bibliográficas los trabajos académicos en los que apoyar mi investigación. Por último, realizar este trabajo me ha hecho conocer el campo de la bioinformática, los últimos avances y sus aplicaciones prácticas, por ejemplo, las mejoras en la medicina, que pueden ser una salida profesional para mí en el futuro.

Siglas

ABC *Artificial Bee Colony. 12*

ACO *Ant Colony Optimization. 12*

ADN *Ácido desoxirribonucleico. 1–3, 13, 14, 19, 20, 22, 25, 29, 30, 47, 51*

ARN *Ácido ribonucleico. 3, 51*

CS *Cuckoo Search. 12, 48*

EM *Expectation Maximization. 6, 11, 16, 23–25, 28, 35, 39, 47, 48*

GA *Genetic Algorithm. 11, 48*

MEME *Multiple EM for Motif Elicitation. 1, 3, 4, 6, 7, 11, 15, 20–26, 34, 35, 38, 39, 41, 43, 45, 47, 48, 51*

PSO *Particle Swarm Optimization. 12, 48*

Glosario

aminoácidos Pequeñas moléculas orgánicas que forman parte de la estructura de las proteínas y están compuestas por 4 elementos químicos básicos: carbono, hidrógeno, oxígeno y nitrógeno. 2

bases nitrogenadas Las bases nitrogenadas son los compuestos biológicos que forman las cadenas de ADN y ARN. Estas son la Adenina A, la Citosina C, la Guanina G, la Tiamina T y el Uracilo U. Las cadenas de ADN se construyen con A, G, C y T y las de ARN con A, G, C y U. 9, 10, 19, 21, 22, 44, 45, 47

bioinformática La bioinformática es un área de de investigación interdisciplinar entre la biología y las ciencias computacionales que se basa en el uso de ordenadores para el almacenamiento, adquisición, manipulación y distribución de información relacionada con macromoléculas biológicas como el ADN, el ARN y las proteínas. 2

motivo Patrón probabilístico que se repite a lo largo de una o varias secuencias de ADN. Este lleva codificado en forma de probabilidades una serie de posibles secuencias, a cada una de estas secuencias que se materializa en forma de ADN la llamamos ocurrencia. 1, 3, 4, 6, 9–15, 19–29, 31, 35–39, 42–45, 47, 48, 51

ocurrencia Cada una de las posibles subsecuencias que puede generar un motivo dentro de una secuencia de ADN. 10, 11, 16, 19–24, 26–28, 31, 32, 35, 36, 38, 39, 43, 44

Suite MEME La Suite MEME es una plataforma que junta una serie de algoritmos, incluidos entre ellos el algoritmo *Multiple EM for Motif Elicitation* (MEME), con el objetivo de facilitar el acceso a herramientas de búsqueda y análisis de motivos. 3, 4, 6, 41, 42

Bibliografía

[1] Sale A.E. (2000) Colossus and the German Lorenz Cipher — Code Breaking in WW II. In: Preneel B. (eds) Advances in Cryptology — EUROCRYPT 2000. EUROCRYPT 2000

[2] Xiong, J. (2006). Essential Bioinformatics. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511806087

[3] Sean Luke, (2013). Essentials of Metaheuristics, Lulu, second edition, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[4] Lewis, H. R. and Papadimitriou, C. H. (1997). Elements of the Theory of Computation, 2nd ed. Prentice-Hall, Englewood Cliffs, New Jersey.

[5] Hashim, F. A., Mabrouk, M. S., & Al-Atabany, W. (2019). Review of Different Sequence Motif Finding Algorithms. Avicenna journal of medical biotechnology, 11(2), 130–148.

[6] Han, Y. S., Ko, S. K., Ng, T., Salomaa, K. (2021). Closest substring problems for regular languages. Theoretical Computer Science, 862, 144-154.

[7] Jonhson, N. L.; Kotz, S.; Balakrishnan (1995). Continuous Univariate Distributions, Volume 2, 2nd Edition, Wiley, 297.

[8] Hertz, G. Z., Stormo, G. D. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics (Oxford, England), 15(7-8), 563–577.

[9] Fornes O, Castro-Mondragon JA, Khan A, et al. (2019). JASPAR 2020: update of the open-access database of transcription factor binding profiles. Nucleic Acids Res.

[10] Sebastian A, Contreras-Moreira B. (2014). footprintDB: a database of transcription factors with annotated cis elements and binding interfaces. Bioinformatics 30, 258–65.

[11] Pagni M, Ioannidis V, Cerutti L, Zahn-Zabal M, Jongeneel CV, Hau J, Martin O, Kuznetsov D, Falquet L. (2007). MyHits: improvements to an interactive resource for analyzing protein sequences. Nucleic Acids Res.

[12] Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., Ren, J., Li, W. W., Noble, W. S. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic acids research, 37(Web Server issue), W202–W208.

[13] Bailey, T. L., and Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, 28–36. AAAI Press.

[14] Bailey, T. L., Elkan, C. (1995). The value of prior knowledge in discovering motifs with MEME. Proceedings. International Conference on Intelligent Systems for Molecular Biology, 3, 21–29.

[15] Bailey, T. L., Bodén, M., Whitington, T., Machanick, P. (2010). The value of position-specific priors in motif discovery using MEME. BMC bioinformatics, 11, 179.

[16] Aitkin, M., Rubin, D. (1985). Estimation and Hypothesis Testing in Finite Mixture Models. Journal of the royal statistical society series b-methodological, 47, 67-75.

[17] Sinha, S., Tompa, M. (2003). YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucleic acids research, 31(13), 3586–3588. https://doi.org/10.1093/nar/gkg618

[18] Bailey T. L. (2011). DREME: motif discovery in transcription factor ChIP-seq data. Bioinformatics (Oxford, England), 27(12), 1653–1659.

[19] Sharov, A. A., Ko, M. S. (2009). Exhaustive search for overrepresented DNA sequence motifs with CisFinder. DNA research : an international journal for rapid publication of reports on genes and genomes, 16(5), 261–273.

[20] Pavesi, G., Mereghetti, P., Mauri, G., Pesole, G. (2004). Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucleic acids research, 32(Web Server issue), W199–W203.

[21] Jia, C., Carson, M. B., Wang, Y., Lin, Y., Lu, H. (2014). A new exhaustive method and strategy for finding motifs in ChIP-enriched regions. PloS one, 9(1), e86044.

[22] Pevzner, P. A., Sze, S. H. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. Proceedings. International Conference on Intelligent Systems for Molecular Biology, 8, 269–278.

[23] Liang, S., Samanta, M. P., Biegel, B. A. (2004). cWINNOWER algorithm for finding fuzzy dna motifs. Journal of bioinformatics and computational biology, 2(1), 47–60.

[24] Buhler, J., Tompa, M. (2002). Finding motifs using random projections. Journal of computational biology : a journal of computational molecular cell biology, 9(2), 225–242.

[25] Davila J., Balla S., Rajasekaran S. (2006) Space and Time Efficient Algorithms for Planted Motif Search. In: Alexandrov V.N., van Albada G.D., Sloot P.M.A., Dongarra J. (eds) Computational Science – ICCS 2006. ICCS 2006. Lecture Notes in Computer Science, vol 3992. Springer, Berlin, Heidelberg.

[26] Rajasekaran, S., Balla, S., amp; Huang, C. H. (2005). Journal of Computational Biology, 12(8), 1117–1128.

[27] Kuksa, P. P., Pavlovic, V. (2010). Efficient motif finding algorithms for large-alphabet inputs. BMC bioinformatics, 11 Suppl 8(Suppl 8), S1.

[28] Dávila, J., Balla, S., Rajasekaran, S. (2007). Pampa : An Improved Branch and Bound Algorithm for Planted (l , d) Motif Search.

[29] Chen, Z. Z., Wang, L. (2011). Fast exact algorithms for the closest string and substring problems with application to the planted (L, d)-motif model. IEEE/ACM transactions on computational biology and bioinformatics, 8(5), 1400–1410.

[30] Reid, J. E., Wernisch, L. (2011). STEME: efficient EM to find motifs in large data sets. Nucleic acids research, 39(18), e126.

[31] Quang, D., Xie, X. (2014). EXTREME: an online EM algorithm for motif discovery. Bioinformatics (Oxford, England), 30(12), 1667–1673.

[32] George Casella Edward I. George (1992) Explaining the Gibbs Sampler, The American Statistician, 46:3, 167-174.

[33] Hughes, J. D., Estep, P. W., Tavazoie, S., Church, G. M. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. Journal of molecular biology, 296(5), 1205–1214.

[34] Liu, X., Brutlag, D. L., Liu, J. S. (2001). BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 127–138.

[35] Xing, E. P., Wu, W., Jordan, M. I., Karp, R. M. (2003). LOGOS: a modular Bayesian model for de novo motif detection. Proceedings. IEEE Computer Society Bioinformatics Conference, 2, 266–276.

[36] Siebert, M., Söding, J. (2016). Bayesian Markov models consistently outperform PWMs at predicting motifs in nucleotide sequences. Nucleic acids research, 44(13), 6055–6069.

[37] Zhang, Y., Wang, P., Yan, M. (2016). An Entropy-Based Position Projection Algorithm for Motif Discovery. BioMed research international, 2016, 9127474.

[38] Congdon, C., Fizer, C., Smith, N.W., Gaskins, H., Aman, J.C., Nava, G., Mattingly, C. (2005). Preliminary Results for GAMI: A Genetic Algorithms Approach to Motif Inference. 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 1-8.

[39] Miao, X. (2014). GAEM: A Hybrid Algorithm Incorporating GA with EM for Planted Edited Motif Finding Problem. Current Bioinformatics, 9, 463-469.

[40] Li L. (2009). GADEM: a genetic algorithm guided formation of spaced dyads coupled with an EM algorithm for motif discovery. Journal of computational biology : a journal of computational molecular cell biology, 16(2), 317–329.

[41] Reddy, U.S., Arock, M., Reddy, A.V. (2010). Planted (l, d) - Motif Finding using Particle Swarm Optimization. International Journal of Computer Applications, 51-56.

[42] Syed-Abdullah, S., Harun, H. (2013). Species motif extraction using LPBS.

[43] Ravi, S.V., Vikas, S., Sanjay, K. (2011). DNA Sequence Assembly using Particle Swarm Optimization. International Journal of Computer Applications, 28, 33-38.

[44] González-Álvarez, D., Vega-Rodríguez, M.A., Pulido, J., Sánchez-Pérez, J.M. (2012). Comparing Multiobjective Artificial Bee Colony Adaptations for Discovering DNA Motifs. EvoBIO.

[45] González-Álvarez, D., Vega-Rodríguez, M.A. (2013). Hybrid Multiobjective Artificial Bee Colony with Differential Evolution Applied to Motif Finding. EvoBIO.

[46] Karaboga, D., Aslan, S. (2016). A discrete artificial bee colony algorithm for detecting transcription factor binding sites in DNA sequences. Genetics and molecular research : GMR, 15(2), 10.4238/gmr.15028645.

[47] Bouamama, S., Boukerram, A., Al-Badarneh, A. (2010). Motif Finding Using Ant Colony Optimization. ANTS Conference.

[48] Gálvez, A., Iglesias, A., Cabellos, L. (2014). Cuckoo search with Lévy flights for weighted Bayesian energy functional optimization in global-support curve data fitting. TheScientificWorldJournal, 2014, 138760.

[49] Makolo, A.U., Osofisan, A., Adebiyi, E. (2012). Comparative Analysis of Similarity Check Mechanism for Motif Extraction.

[50] Liu, X., Brutlag, D., Liu, J. (2002). An algorithm for finding protein–DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. Nature Biotechnology, 20, 835-839.

[51] Hu, J., Yang, Y.D., Kihara, D. (2005). EMD: an ensemble algorithm for discovering regulatory motifs in DNA sequences. BMC Bioinformatics, 7, 342-342.