UNIVERSIDAD CENTRAL DE LAS VILLAS
FACULTAD DE MATEMATICA, FISICA Y COMPUTACION
CARRERA DE CIENCIA DE LA COMPUTACION

TRABAJO DE DIPLOMA

USeparator: una herramienta para separación de genes en Hongos y Plantas

Autor: Doi Pham Anh

Tutores:
Dr. Ricardo Grau Ábalo
Lic. Aminael Sánchez Rodríguez

Santa Clara
2008
"Año 50 de la Revolución"

Hago constar que el presente trabajo fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Doi Pham Anh
Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____

Aminael Sánchez Rodríguez
Tutor

_____

Ricardo Grau Ábalo
Tutor y Jefe del Lab. de Bioinformática

Agradecimiento

A toda la Facultad de Matemática, Física y Computación, a las facilidades que me han dado en el transcurso de los cinco años de la carrera.

A todo el Grupo de Investigación de Bioinformática que me ha llevado a poder terminar el trabajo con mucho entusiasmo y alegría.

A los tutores Dr. Ricardo Grau Ábalo y Aminael Sánchez Rodríguez, por todo.

A mi novia por acompañar y compartir el tiempo difícil conmigo.

A la vida que me han transmitido todos ustedes a mí.

Tabla de contenidos

Resumen

En este trabajo se describe la implementación de USeparator: un algoritmo basado en un Modelo Oculto de Markov aplicado a la problemática de la separación de secuencias de ADN de plantas y hongos fitopatógenos. USeparator basa su clasificación en la modelación de las diferencias en el uso de codones en ambos organismos. Además introduce un nuevo concepto de penalización de los codones de parada en los segmentos de ADN analizados que permite rechazar los ORF (-del inglés *Open Reading Frame*) incorrectos. Dicho algoritmo presentó un mejor poder de clasificación al compararlo con otros reportados en la literatura con el mismo fin. Entre estos últimos se encuentran modelos probabilísticos simples y otros basados en *Support Vector Machines*. Para la implementación del algoritmo se usó el paquete de programación *BioJava* y otros paquetes propios del lenguaje de programación *Java* como *Hibernate*, *Hsqldb* y otros. El empleo de estas herramientas resulta muy apropiado pues facilita el trabajo con diferentes formatos de secuencias biológicas así como la creación de bases de datos asociadas. Con este trabajo incorporamos una herramienta mejorada que se puede emplear en flujos de trabajo de proyectos de investigación en Biología Molecular, Fitopatología y Bioinformática.

Introducción

El estudio de los mecanismos de defensa en plantas y de patogénesis en hongos fitopatógenos requiere del estudio de la interacción entre ambos organismos. Una metodología clásica en este sentido incluye la identificación y caracterización de genes involucrados en ambos procesos. Ello normalmente involucra el aislamiento de ARN mensajero (ARNm) de los tejidos infectados y la aplicación de técnicas moleculares modernas como la construcción de bibliotecas de ADN complementario (ADNc). Un problema que surge en estas estrategias es que las secuencias de ADN así obtenidas constituyen una mezcla procedente de las especies interactuantes. Por ello el paso subsiguiente de asignación funcional no es posible llevarlo a acabo sin resolver esta mezcla. Determinar el origen de las secuencias obtenidas es entonces una tarea decisiva para el éxito de la investigación. El desarrollo de herramientas bioinformáticas con este fin es una temática en plena actividad científica.

Antecedentes

Una metodología clásica para la asignación funcional de secuencias biológicas hace uso de herramientas de detección de homología con secuencias previamente identificadas y almacenadas en bases de datos. Tal es el caso de las herramientas del tipo BLAST (Joseph Bedell 2003). Sin embargo, cuando estas se aplican al caso específico de aquellas secuencias provenientes de estudios de interacción planta-hongos fitopatógenos en la mayoría de los casos no ofrecen buenos resultados. Ello se debe principalmente a la insuficiencia de secuencias en bases de datos, el pequeño tamaño de las secuencias que se obtienen por métodos de biología molecular y a la alta homología de secuencias entre ambos organismos (Maor 2003; Caroline C. Friedel 2005; Jeppe Emmersen 2007). Son necesarios entonces nuevos métodos de Bioinformática que permitan determinar de forma fiable y rápida el origen de fragmentos de secuencia expresados (ESTs –del inglés *Expressed Sequence Tags*) resultantes de plantas infectadas por hongos.

Un resultado importante en esta tarea es la confirmación de la existencia de diferencias en el uso de codones intra e interespecies (Nakamura 2000). En el primero de los casos se han adelantado varias hipótesis para responder la correlación positiva entre el uso de codones en genes específicos y el nivel de expresión de los mismos. Se piensa que la relación refleja la selección preferencial por el uso de aquellos codones reconocidos por las moléculas de ARN de transferencia (ARNt) más abundantes. Así, la evolución en el uso de codones en genes altamente expresados se supone se deba a la selección natural para lograr una mayor eficacia traslacional. Otra posible interpretación del uso de codones es la que lo relaciona con la estructura secundaria del ARNm (Carlini 2001). Aquellos ARNm más estables tendrán un mayor probabilidad de ser transcriptos y sucesivamente traducidos a proteínas.

En el caso de las diferencias en el uso de codones entre especies es importante ante todo conocer cómo es posible cuantificar estas diferencias. Primeramente se debe destacar que los 64 codones que forman el código genético clásico se pueden dividir en dos grupos fundamentales: el primero comprende aquellos que terminan en G o C y se denotan como codones GC_3. El segundo comprende los que terminan en A o T y su notación es AT_3. La fuente principal de variación que contribuye a las diferencias en este fenómeno está dada por el hecho de que un mismo aminoácido puede ser codificado por codones de ambos tipos. Se conoce que los genes de muchos hongos y plantas monocotiledóneas tienen preferencia por el uso de codones del tipo GC_3. Por su parte, en los genes de las plantas dicotiledóneas los codones del tipo AT_3 están sobre- representados.

Varios grupos de investigación han abordado el problema de la separación de secuencias de ADN de plantas y hongos fitopatógenos, basándose principalmente en algoritmos que modelan las diferencias en el uso de codones entre ambas especies. Uno de los primeros trabajos publicados con este fin es el que describe el algoritmo probabilístico PF-IND (Maor 2003). Este compara el número real de ocurrencias de los diferentes tipos de codones (GC_3 y AT_3) en una secuencia de ADN dada con el número esperado de ocurrencias en planta y hongo respectivamente usando una distribución de Poisson. Las probabilidades resultantes son entonces empleadas para clasificar las secuencias problema como de origen planta u hongo. Si bien este algoritmo logra clasificar un considerable porciento de secuencias, no hace uso de toda la información que es posible extraer del análisis de las diferencias del uso de codones entre dos especies. Los resultados de PF-IND fueron superados con el surgimiento del algoritmo ECLAT (Caroline C. Friedel 2005). En este caso, el algoritmo usa una técnica de inteligencia artificial de clasificación conocida como *Support Vector Machine* y trabaja específicamente con una función *Kernel* no lineal, del tipo RBF (-del inglés *Radial Basic Function*) para identificar el origen probable de secuencias de ADN. El cómputo de las diferencias en el uso de codones entre especies usa como atributo de estas, las frecuencias de ocurrencias de cada uno de los 64 codones existentes.

Ambos algoritmos comparten la necesidad de un conjunto de entrenamiento abundante con más de 1000 secuencias por especie. Además necesitan conocer de antemano el marco de lectura real de dichas secuencias. Dichos requerimientos constituyen una debilidad de estos algoritmos, pues crear conjuntos de entrenamientos tan grandes con información de anotación biológica –necesaria para conocer el marco de lectura real- es imposible para algunos pares de planta-hongo fitopatógeno.

Este inconveniente fue superado en el algoritmo más recientemente reportado de separación de secuencias (Jeppe Emmersen, 2007). En él, la clasificación se basa en las frecuencias de ocurrencia de trinucleótidos entre especies sin importar el marco de lectura en el que aparecen. A diferencia de los trabajos anteriores, este es el único que es posible aplicar a secuencias no codificantes como es el caso del

fragmento de ADN 3' que resulta durante el proceso de aislamiento de ESTs así como las secuencias de intrones (regiones del gen no codificantes, que yacen entre las regiones codificantes, o exones).

Por otra parte, existen en la teoría de la Inteligencia Artificial (específicamente *Machine Learning*) muchos otros modelos de aprendizaje supervisado que pueden adaptarse mejor a las características de este problema específico de clasificación. Entre ellos se encuentran por ejemplo los Modelos Ocultos de Markov, (HMM –del inglés *Hidden Markov Model*)

En los años 1990s solo aproximadamente una tercera parte de las nuevas secuencias de proteínas predichas mostraban similaridad convincente a otras secuencias conocidas (Pierre Baldi, 2001). La situación es incluso más difícil cuando se habla de nuevas secuencias o fragmentos incompletos, y sin embargo las grandes bases de datos disponibles tratan de tales fragmentos como resultado del estudio de varios genomas, de los ADN complementarios y otros proyectos de sub-secuencias, especialmente aquellas produciendo ESTs (*Expressed Sequences Tags*). Ya al comienzo de 1997, aproximadamente la mitad de la conocida base de datos GenBank consistía de datos de fragmentos y cubrían, si no todo, una parte sustancial del genoma humano expresado.

Resulta de gran interés reconocer y clasificar fragmentos en cualquier genoma y recuperar cualquier información adicional que pueda ser útil. Una tendencia prometedora para mejorar la sensibilidad y velocidad de técnicas de búsquedas en bases de datos ha sido usar modelos "consenso" construidos a partir de alineamientos de familias de proteinas (Sánchez, 2006; Bairoch, 1993; Barton, 1990; A. Krogh, 1994) A diferencia de las comparaciones convencionales par a par, los modelos "concenso" permiten en principio explotar información adicional, tal como la posición e identidad de los residuos que son más o menos conservados a través de una familia, así como las probabilidades de inserción y delección. Todas las descripciones de secuencias consenso, tales como perfiles (M. Gribskov, 1987), patrones flexibles y bloques estructurales (conocidos en inglés como *building blocks*) (Henikoff, 1994) pueden ser vistos como casos especiales de los llamados Modelos Ocultos de Markov.

Los Modelos Ocultos de Markov son una clase especial de modelos gráficos probabilísticas, usados en las últimas décadas para modelar procesos estocásticos (series de tiempo), por ejemplo en reconocimiento del habla (S. E. Levinson, 1983; Rabiner, 1989), y reconocimiento de otras señales (Rice., 1992; Pieraccini, 1993); pero más recientemente, han sido aplicados exitosamente a problemas de Biología Computacional, incluyendo la modelación de regiones codificantes/no-codificantes en el ADN(Churchill., 1989), de sitios de retorno en el plegamiento de proteínas en el ADN [352] y de superfamilias de proteínas (J. V. White, 1994). Los Modelos Ocultos de Markov están estrechamente relacionados a casos especiales de redes neuronales, gramáticas estocásticas y redes bayesianas; pero pueden ser teóricamente introducidos de forma independiente.

Se pretende en este trabajo ensayar el uso de los Modelos Ocultos de Markov para resolver el problema de clasificación o de separación de mezclas de secuencias de ADN de plantas y de hongos, utilizando información adicional de carácter biológico. El trabajo supone por supuesto, la implementación de los nuevos algoritmos y ello se abordará en el lenguaje de programación *Java*, por ser de código abierto y por tener además un conjunto de herramientas que facilitan la solución de este problema, en particular *BioJava* para el tratamiento de bases de datos de secuencias biológicas con muy diferentes estructuras. El algoritmo implementado será validado además en el problema concreto de separación de genes de *Blumeria graminis (Planta) y Hordeum vulgare (Hongo).*

Visto como ejercicio docente de culminación de estudios de la carrera de Ciencia de la Computación, el presente trabajo de diploma integra conocimientos de Análisis y Diseño de Algoritmos, Estructura de Datos, Programación, Uso de software libre, Probabilidades, Estadística, Inteligencia Artificial (*Machine Learning*), y Bioinformática, entre otras disciplinas, y de acuerdo a lo anteriormente explicado, afronta el siguiente problema de investigación que fue abordado conjuntamente por el Centro de Estudios de Informática y el Instituto de Biotecnología de las Plantas, ambos de la Universidad Central "Marta Abreu" de Las Villas, Cuba.

Problema

Se necesita elaborar e implementar un nuevo algoritmo para la separación de secuencias de ADN de pares planta-hongo fitopatógeno, basado en Modelos Ocultos de Markov incluyendo nuevos atributos biológicos que permitan lograr un poder de clasificación comparable con la de otros métodos ya reportados y que dependa de un conjunto de entrenamiento menor.

Objetivo *general*

Diseñar e implementar un algoritmo para la separación de genes de plantas y hongos, basado en Modelos Ocultos de Markov, y utilizando nuevos atributos biológicos que permitan maximizar la información con conjuntos de entrenamiento más reducidos y mejorar así el poder de clasificación de algoritmos reportados en la literatura con ese fin.

Objetivos *específicos*

1. Demostrar que con un algoritmo basado en un Modelo Oculto de Markov es posible obtener un poder de clasificación comparable a los obtenidos mediante PF-IND y ECLAT.

2. Incorporar nueva información biológica al algoritmo con el objetivo de extraer el máximo de información contenida en el conjunto de entrenamiento para reducir los requisitos de éste.

Preguntas de investigación

¿Un algoritmo basado en un Modelo Oculto Markov, con un conjunto de entrenamiento relativamente simple puede dar resultados comparables con los obtenidos mediante otros métodos de Inteligencia Artificial?

¿Cuál puede ser la nueva información biológica a adicionar al algoritmo que se creará para mejorar los resultados y reducir los requisitos del conjunto de entrenamiento?

Justificación

Las enfermedades en plantas causadas por hongos fitopatógenos tienen un gran impacto en la economía mundial, tal es el caso de la enfermedad Sigatoka Negra principal problema en el cultivo de plátanos y bananos a nivel mundial (INIBAP, 2003; FAO, 2003). Esta es causada por el patógeno *Mycosphaerella fijiensis*. En el Instituto de Biotecnología de las Plantas (IBP), de la Universidad Central "Martha Abreu" de Las Villas se esta trabajando intensamente en la biología de este patosistema. Como resultado de estas investigaciones se han obtenido una cantidad suficiente de secuencias de ADN proveniente de ambos organismos. Entonces es necesario tener una herramienta que permita resolver la distinción de las mismas con rapidez, de forma confiable y fácil de usar.

Las investigaciones en Bioinformática en general y sobre el problema particular que aborda el presente trabajo, han sido abordadas solo recientemente. Por eso, todavía los algoritmos no han aprovechado, en su totalidad, el uso de las características biológicas de este fenómeno, y pueden ser mejorados. El uso de los Modelos Ocultos de Markov, auxiliados por información biológica adicional, puede producir resultados mejores o al menos alentadores respecto a los disponibles.

Los paquetes hechos en muchos lenguajes de programación pueden ayudarnos en la construcción de un nuevo software, especialmente en Java, un lenguaje de código abierto. Es necesario investigar las herramientas disponibles en Java para facilitar el trabajo a realizar y brindar un software de alta calidad. *Biojava* es la principal de ellas, pues brinda muchas clases para procesar secuencias biológicas y para el trabajo con base de datos (Schreiberm 2005).

Después de haber desarrollado el marco teórico se formula la siguiente

Hipótesis *de investigación*

Es posible obtener resultados en la distinción de genes de planta-hongos, basados en un algoritmo del tipo Modelo Oculto de Markov, comparable al menos en resultados de exactitud con los precedentes de otros métodos de Inteligencia Artificial y con un conjunto de entrenamiento más fácil de conseguir mediante la adición de nuevos atributos biólogos.

Estructura del trabajo de tesis

El trabajo de diploma está estructurado en la presente Introducción, y tres capítulos esenciales:

Capítulo 1. Conceptos básicos.

Capítulo 2. El algoritmo, la implementación y el manual de usuario.

Capítulo 3. Evaluación del algoritmo.

En el Capítulo I se hace la revisión bibliográfica fundamental. Se describe la motivación biológica y las características generales de los Modelos Ocultos de Markov. Se describe con detalle el algoritmo de este modelo y su aplicación al problema de investigación.

En el Capítulo II se describe con detalles el algoritmo desarrollado y la implementación de las clases utilizando rigurosamente el Lenguaje de Modelación Unificada (UML). Se incluye el manual de usuario del software que se ha denominado USeparator

En el Capítulo III se presentan los resultados de la validación del algoritmo con datos de conjuntos de secuencias de Hongos y Plantas disponibles en bases de datos internacionales. Se discute la incorporación de nuevos atributos biológicos de las secuencias de ADN y su impacto en el poder de clasificación del algoritmo propuesto.

Finalmente se presentan las conclusiones, recomendaciones, la bibliografía utilizada y algunos anexos que amplían la información de la validación de los resultados.

Capítulo I: Conceptos básicos

Este capítulo está dedicado a una breve introducción de conceptos biológicos, las características específicas de nuestro problema de investigación, las posibles herramientas de aprendizaje supervisado que pudieran ser utilizadas, enfocando las ventajas y/o desventajas del modelo seleccionado: los procesos ocultos de Markov, y la caracterización teórica de estos últimos.

1.1. Breve introducción biológica

El proceso de transportar la información genética almacenada en las secuencias lineales de ADN a las proteínas es complejo, pero los elementos esenciales son la transcripción de ADN a ARN y posteriormente la traducción de esta última en el proceso de síntesis de las proteínas (Marchal 2004) mediante el código genético (Tabla 1.1).

La estructura del ADN es una doble hélice. Las bases nitrogenadas se aparean en la doble hélice formando enlaces por puentes de hidrógeno de acuerdo a las siguientes reglas: $G \equiv C$, $A = T$, donde cada "—" simboliza un enlace por puente de hidrógeno y cada par contiene una purina (A o G) y una pirimidina (C o T). Las bases que forman el par se denominan bases complementarias. Una hélice simple es una cadena de nucleótidos unidos por enlaces fosfodiester. Cada hélice simple se une con la hélice complementaria a través de los enlaces de puentes de hidrógenos que forman las bases formando la doble hélice (Ver Figura 1.1).

La molécula del ADN es una molécula direccional debido a la estructura asimétrica del azúcar desoxirribosa que forma el esqueleto de las hélices. Cada azúcar es conectada cadena arriba por el quinto carbono y cadena abajo por el tercer carbono, de manera que si una de las cadenas tiene orientada el azúcar en la dirección 5'→3' (se lee de cinco prima a tres prima), la cadena complementaria estará orientada en la dirección 3'→5'. El ARNm está formado por una cadena de nucleótidos unidos también por enlaces fosfodiester. En particular constituye una hélice en la cual la base nitrogenada *T* es sustituida por la base Uracilo (*U*). La función del ARNm es transportar hacia el citoplasma la información genética que se encuentra almacenada en el núcleo de la célula en la secuencia de nucleótidos del ADN. Ya en el citoplasma la secuencia de nucleótidos del ARNm es traducida para formar la secuencia de aminoácidos de la proteína correspondiente.

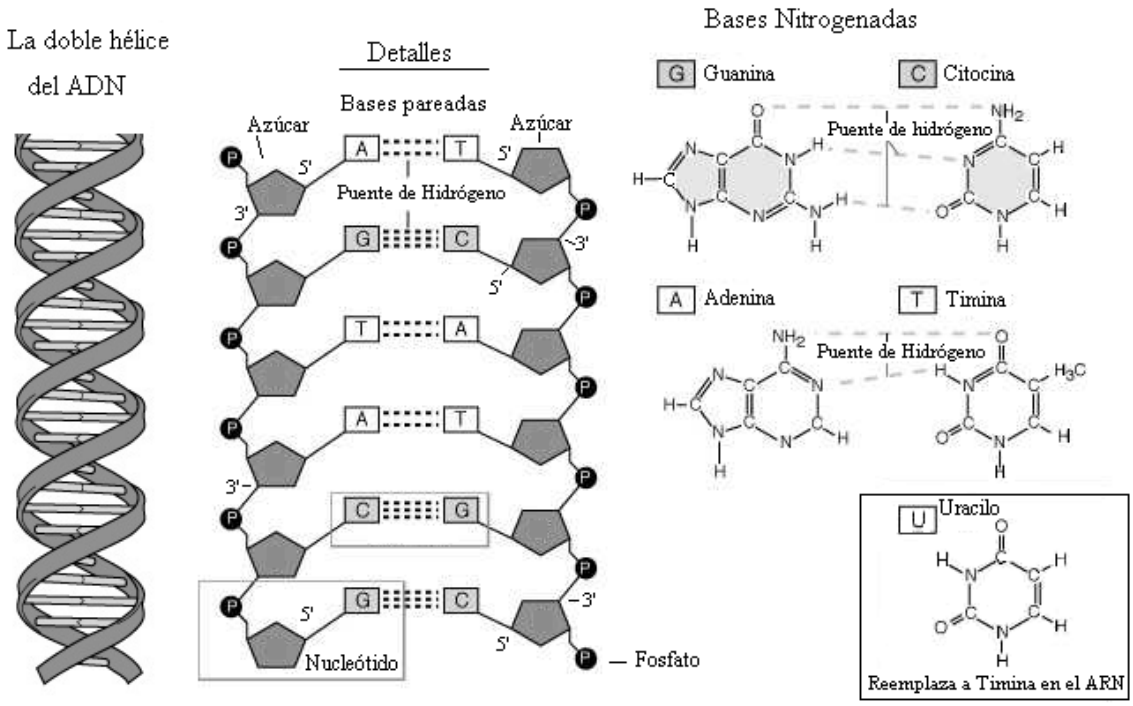


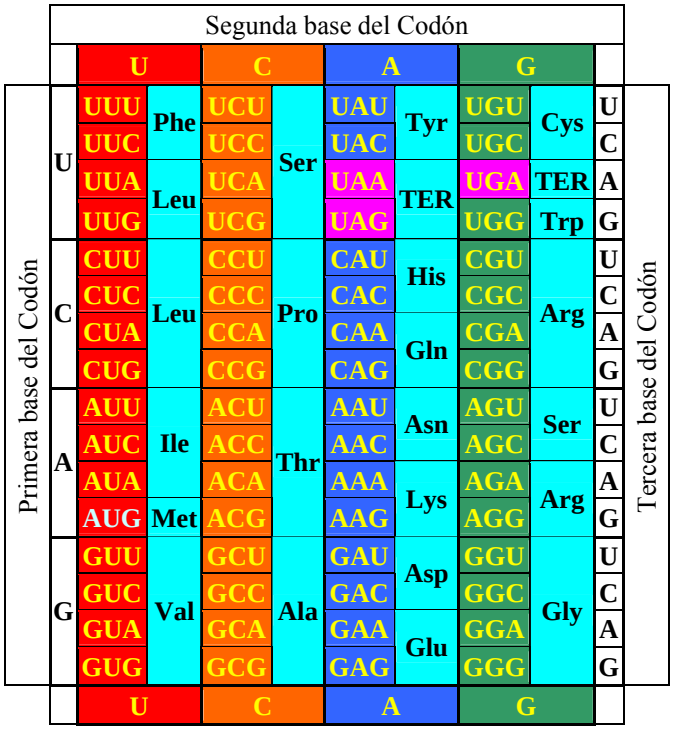
Figura 1.1 La doble hélice del ADN. Detalles de las bases nitrogenadas y su apareamiento en el ADN.

La secuencia de nucleótidos del ARNm es traducida para formar la secuencia de aminoácidos de la proteína correspondiente. Las propiedades biológicas de las proteínas están determinadas por la interacciones físico-químicas de los amino ácidos presentes en la secuencia. Por lo cual, estas propiedades biológicas de las proteínas están determinadas esencialmente por la secuencia de bases o codones de la molécula de ADN.

Tabla 1.1. Tabla del Código genético Standard. Los aminoácidos están escritos con el símbolo de tres letras. El codón utilizado con mayor frecuencia como codón de inicio de la transcripción corresponde al aminoácido Metionina: *AUG* Los codones *UAA*, *UAG* y *UGA* son marcadores del final de los genes [Tomada de (Sánchez 2003)].

La codificación de cada aminoácido se realiza a través de una secuencia de tres bases. Como el número total de secuencias de tres letras formadas a partir de un alfabeto de cuatro letras es 64, existe exactamente esta cantidad de codones. Pero como fue anteriormente dicho, solo son 20 aminoácidos, luego se dice que el código genético está degenerado en el sentido de que hay más de un vocablo de código para la mayoría de los aminoácidos. De tal forma que encontramos aminoácidos como *Leucina*, *Serina* y *Arginina*, cada uno de los cuales posee 6 codones que codifican para el mismo, mientras que otros como el *Triptófano* y la *Metionina* solo poseen un solo codón. El codón del aminoácido *Metionina* es usualmente

utilizado en la mayoría de los seres vivos como codón de inicio de la cadena polipeptídica. Tres de los codones del código son utilizados como señales de terminación de la cadena polipeptídica.



La codificación de cada aminoácido se realiza a través de una secuencia de tres bases. Como el número total de secuencias de tres letras formadas a partir de un alfabeto de cuatro letras es 64, existe exactamente esta cantidad de codones. Pero como fue anteriormente dicho, solo son 20 aminoácidos, luego se dice que el código genético está degenerado en el sentido de que hay más de un vocablo de código para la mayoría de los aminoácidos. De tal forma que encontramos aminoácidos como *Leucina*, *Serina* y *Arginina*, cada uno de los cuales posee 6 codones que codifican para el mismo, mientras que otros como el *Triptófano* y la *Metionina* solo poseen un solo codón. El codón del aminoácido *Metionina* es usualmente utilizado en la mayoría de los seres vivos como codón de inicio de la cadena polipeptídica. Tres de los codones del código son utilizados como señales de terminación de la cadena polipeptídica.

Esta degeneración hace que cada organismo tiene una preferencia de usar algunos codones para un amino ácido Este fenómeno se llama uso de codones (Wikipedia 2007) y alienta la posibilidad de utilizar este hecho para distinguir entre secuencias similares (isofuncionales) cuáles son originarias de la planta, y cuáles del patógeno. En nuestro trabajo se revisó, por ejemplo el uso de codones para las dos especies *H. vulgares*, un hongo y *B. graminis* una planta (Japan 2008) (ver Tabla 1.2).

Tabla 1.2 *El porcentaje de uso de codones con la tercera letra en G, C o A, T*

	GC3	*AT3*
H. vulgare	65.25%	*34.75%*
B. graminis	40.77%	*59.23%*

Los ESTs (*Expressed Sequence Tags*) son los segmentos cortos de ADN que constituyen partes de una molécula del ADNc y pueden actuar como el identificador de un gen (Wikipedia 2006).

1.2. Nuestro problema de aplicación: la separación de genes de la planta y el hongo

Cuando el hongo afecta a la planta contamina su genoma. Así se degeneran las características de planta. Una solución es investigar para buscar una genética que puede defenderse al hongo. Esta solución fue exitosa en los años 60s del siglo pasado cuando encontró una variedad que se llama *Cavendish*, que podía defenderse frente el hongo *Fusarium oxysporum.f. sp cúbense*. Con las herramientas modernas actualmente los científicos ya pueden encontrar bastantes secuencias de este genoma mezclado. Pero las secuencias obtenidas son subgenes o *EST* y no se sabe su origen. Uno de los pasos siguientes es investigar el origen de estas secuencias y determinar el marco abierto de lectura para determinar la secuencia de proteína generada.

Esta separación sería un dilema por la homología alta de estas secuencias. Gracias a la existencia de diferencias de uso de codones entre especies diferentes y especialmente entre hongos y plantas existe la esperanza de separarlos.

1.3. Posibles herramientas de aprendizaje supervisado para resolver el problema de la separación

El problema de la separación de genes de la planta y el patógeno puede ser enfrentado como un problema de clasificación por varias técnicas de Estadística y de la Inteligencia Artificial si se dispone de una base de entrenamiento con secuencias de ambos organismos. La aplicación sin embargo, de técnicas estadísticas, como el análisis discriminante incluye restricciones sobre las variables predictivas y la distribución de las mismas y la aplicación de técnicas alternativas como la Regresión Logística, requiere de

un gran volumen de datos no siempre disponible, restricción que es además aplicable a la técnica más fuerte del Análisis Discriminante.

En la intersección difusa entre las técnicas Estadísticas y de Inteligencia Artificial, los Árboles de Decisión y las Redes Bayesianas, podrían aportar sus propias virtudes. En el campo de la Inteligencia Artificial, las Redes Neuronales, en su más amplia concepción, parecen idóneas por su carácter esencialmente no lineal, pero definitivamente, todas estas técnicas necesitan de bases de datos de aprendizaje grandes y además resulta difícil incorporar a dicho aprendizaje, "información adicional" de carácter biológico.

Los Modelos Ocultos de Markov pudieran ser una alternativa posible sobre la base de la experiencia que se cuenta en el análisis de secuencias biológicas y las ventajas y desventajas de su aplicación.

1.4. *Experiencias* en la aplicación de los Modelos Ocultos de Markov

El Modelo Oculto de Markov ha sido utilizado tradicionalmente en problemas de

- Criptoanálisis
- Reconocimiento del habla, de gestos y de movimientos corporales, reconocimiento óptico de caracteres
- Traducción automática
- *Musical score following*

Pero además, ellos han sido utilizados felizmente en el análisis de secuencias biológicas. Muchos de los estudios bioinformáticos pueden diseñarse, entrenarse y posteriormente analizarse a través de Modelos de Markov. Independientemente del diseño y el método de entrenamiento que será comentado posteriormente una vez que el modelo ha sido entrenado exitosamente a partir de una familia de secuencias, el puede ser usado en diferentes tareas computacionales actuales, tales como:

- Alineamiento múltiple
- Minería de datos y clasificación de secuencias y fragmentos
- Análisis Estructural y Reconocimiento de Patrones

Y estas técnicas generales han sido aplicadas en particular a:

- predicción de regiones proteína-codificables en secuencias de genomas
- modelado de familias de secuencias de proteínas o de DNA relacionados

- predicción de elementos de estructura secundarias de secuencias primarias de proteínas (Galpert D. 2005).

Todas estas aplicaciones están basadas en el cálculo, a partir del modelo, de la sucesión de elementos de la secuencia, tanto como sea probablemente factible. En la mayoría de los casos, los Modelos Ocultos de Markov proporcionan buenas respuestas, competitivas, por ejemplo con aquellas derivadas del trabajo arduo de expertos humanos (Pierre Baldi 2001)

En el presente trabajo se muestra como puede ser utilizado este modelo en problemas de la Separación de los Genes, de Plantas y Fitopatógenos, en particular de *Blumeria graminis* y *Hordeum vulgare*. El problema tiene que ver con las tareas computacionales mencionadas anteriormente, y en particular con el Análisis Estructural y el Reconocimiento de Patrones, estos últimos basados en la frecuencia de uso de codones y de información adicional de carácter biológico.

1.5. Modelo Oculto de Markov

En este epígrafe se describen brevemente los conceptos relacionados con los Modelos de Markov y en particular los Modelos Ocultos de Markov que serán la herramienta utilizada para atacar este tipo de problemas, visto como un problema de clasificación.

1.5.1. Concepto de Modelo Oculto de Markov

Los Modelos de Markov son herramientas para analizar el comportamiento y el gobierno de determinados tipos de procesos estocásticos, esto es, procesos que evolucionan de forma no determinista a lo largo del tiempo en torno a un conjunto de estados (Rabiner 1989)

Un modelo de Markov, por tanto, representa un sistema que varía su estado a lo largo del tiempo, siendo cada cambio una transición del sistema. Dichos cambios no están predeterminados, aunque si lo está la probabilidad del próximo estado en función de los estados anteriores, probabilidad que es constante a lo largo del tiempo (sistema homogéneo en el tiempo). Eventualmente, en una transición, el nuevo estado puede ser el mismo que el anterior y es posible que exista la posibilidad de influir en las probabilidades de transición actuando adecuadamente sobre el sistema (decisión).

En este trabajo nos ocuparemos de los llamados modelos de Markov finitos, caracterizados porque el número de estados del sistema es finito.

Para definir formalmente, un modelo (cadena o proceso) de Markov finito hace falta determinar por tanto, los siguientes elementos:

- Un conjunto de estados del sistema.
- La definición de transición.
- Una ley de probabilidad condicional, que defina la probabilidad del nuevo estado en función de los anteriores.

En los llamados Modelos Ocultos de Markov, aparece además un conjunto finito de símbolos de un cierto alfabeto, cada uno de los cuales es emitido una vez que el sistema pasa a un nuevo estado. En los Modelos Ocultos de Markov no es apreciable el movimiento del sistema a través de los estados, sino la expresión de este movimiento a través de los símbolos emitidos. De allí el apellido de "oculto". Junto con el conjunto de símbolos emisibles, aparece entonces otra ley de probabilidad asociada a cada estado: la probabilidad de emitir cada uno de los símbolos al arribar a dicho estado.

Cuando no se conocen teóricamente las probabilidades de las transiciones (ni de las emisiones, en el caso de los Modelos Oculto de Markov) se puede intentar "aprenderlas" a partir de una base de datos de entrenamiento y el aprendizaje de tales probabilidades se convierte en una técnica de aprendizaje supervisado (*Machine Learning*).

En general, en cualquier modelo de Markov los estados son una caracterización de la situación en que se halla el sistema en un instante dado; dicha caracterización puede ser tanto cuantitativa como cualitativa. Desde un punto de vista práctico, probablemente, la mejor definición de que debe entenderse por estado es la respuesta que se daría a la pregunta "¿Cómo están las cosas?"

Formalmente, estado de un sistema en un instante t es una variable cuyos valores sólo pueden pertenecer a un conjunto dado (conjunto de estados del sistema). El sistema es modelizado por una cadena de estados y es por tanto, una variable que cambia de valor en el tiempo, cambio al que llamamos transición.

Dicho de otro modo, se trata de una colección indexada de variables X_t, donde t denota intervalos temporales significativos para el fenómeno estudiado. Los posibles valores de X_t, se toman de un conjunto de categorías mutuamente excluyentes, denominadas estados del sistema. Para ser el sistema estocástico, no se conocerá con certeza el estado del sistema en un determinado instante, sino tan solo la probabilidad asociada a cada uno de los estados y condicional a los estados anteriores: puede expresarse en términos de probabilidad condicional:

$$\mathrm{P}\left\{X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, ..., X_{t-1} = i_{t-1}, X_t = i\right\}$$

Aquí, i, j, i_t pertenecen al conjunto de estados posibles del sistema. Si existe algún k≥0, tal que para cualquier t se tenga que

$$\mathrm{P}\left\{X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, ..., X_{t-1} = i_{t-1}, X_t = i\right\} =$$

$$\mathrm{P}\left\{X_{t+1} = j \mid X_{t-k} = i_{t-k}, X_{t-k-1} = i_{t-k-1}, ..., X_{t-1} = i_{t-1}, X_t = i\right\}$$

se dice que estamos en presencia de un proceso de Markov de orden *k*

Para el caso particular de un modelo de Markov de orden 1, tenemos

$$\mathrm{P}\left\{X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, ..., X_{t-1} = i_{t-1}, X_t = i\right\} = \mathrm{P}\left\{X_{t+1} = j \mid X_t = i\right\} = P_{ij}$$

donde P_{ij} recibe el nombre de probabilidad de transición del estado *i* al estado *j*. En un modelo de Markov de primer orden, el estado del sistema en el futuro inmediato (*j*) sólo depende del estado presente (*i*) y resulta independiente de los estados anteriores al actual (se dice que el sistema no tiene "memoria").

En este trabajo sólo nos hace falta trabajar con procesos de Markov que sean de primer oden, que tengan un número finito de estados y con probabilidades de transiciones entre ellos constantes a lo largo del tiempo. Para este tipo de cadenas, tendremos que la ley de probabilidad condicional es de la forma:

$$\mathrm{P}\left\{X_{t+s} = j \mid X_{t+s-1} = i\right\} = \mathrm{P}\left\{X_t = j \mid X_{t-1} = i\right\}$$

1.5.2. Matrices de probabilidades de transición

En los procesos que estudiamos aquí la forma más cómoda de expresar la ley de probabilidad condicional de la misma, es mediante la llamada matriz de probabilidades de transición *P*, o más sencillamente, la matriz del proceso. Dicha matriz es cuadrada con tanta filas y columnas como estados tiene el sistema, y los elementos de la matriz representan la probabilidad de que el estado próximo sea el correspondiente a la columna si el estado actual es el correspondiente a la fila.

Como el sistema debe evolucionar a alguno de los *n* estados posibles, las probabilidades de transición cumplirán que, para todo *i*:

$$\sum_{j=i}^{n} P_{ij} = 1$$

Además, por definición de probabilidad, cada una ellas ha de ser no negativa

$$P_{ij} \geq 0$$

Cuando los valores P_{ij} cumplen las propiedades arriba indicadas, la matriz P se denomina matriz estocástica: la suma de valores de la filas de la matriz será siempre igual a 1 (la suma de valores de las columnas no tiene ninguna propiedad especial).

1.5.3. Matrices de emisión

En los Modelos Ocultos de Markov debemos hablar además del conjunto de símbolos que se pueden emitir al arribar a cada estado y la probabilidad de emisión de cada uno. En cada estado puede haber más de un símbolo, pero hablamos en general del conjunto de símbolos que pueden emitirse en todos los estados. Hay entonces otra función que define la probabilidad de que se emita cada símbolo en un estado dado:

$$e_k(b) = P(X_i = b \mid y_i = k)$$

Así, $e_k(b)$ es la probabilidad de emisión del símbolo *b* en el estado *k*. (R. Durbin 1998). Se forma así otra matriz estocástica, no necesariamente cuadrada, que tiene tantas filas como estados tiene el sistema y tantas columnas como símbolos puedan emitirse en los estados

1.5.4. Estimación de los parámetros en un Modelo Oculto de Markov

Cuando tenemos el conjunto de entrenamiento, podemos calcular las frecuencias de transiciones (A_{kl}) y de emisiones ($E_k(b)$) y estimar entonces las probabilidades de transición y emisión como sigue:

$$P_{ij} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

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

Como podemos ver, el tamaño del conjunto de entrenamiento es muy importante pues cuando no hay suficientes datos hay riesgo que algún denominador sea cero. Esto se puede evitar introduciendo un número pequeño en forma aditiva. Así, tenemos la fórmula:

$$A_{kl} = A^*_{kl} + r_{kl}$$

$$E_k(b) = E^*_k(b) + r_k(b)$$

donde $A^*_{kl}, E^*_k(b)$ representan respectivamente la cantidad de transiciones del estado *k* al estado *l* y la emisión en el estado *k* del símbolo *b* (R. Durbin 1998).

1.5.5. Algoritmos de búsqueda

Existen muchos algoritmos para encontrar las probabilidades de que una secuencia de estado se ajusta con un modelo dado. Aquí citamos algunos más clásicos (Pierre Baldi 2001).

1.5.5.1 Algoritmo de búsqueda hacia adelante.

Define $\alpha_i(t) = P(S^t = i, X^1 \ldots X^t \mid w)$, la probabilidad de estado *i* en el tiempo *t*, teniendo los estados observados $X^1 \ldots X^t$ en el modelo *M(w)*.

El objetivo del algoritmo es $P(O \mid w) = \alpha_{end}(T)$ la probabilidad de los estados *O* en el modelo *M*(*w*) o la probabilidad en el estado final (*end*) en el tiempo final(*T*), podemos buscar mediante la inicialización y la recursión.

Inicialización

$$\alpha_{start}(0) = 1;$$
$$\alpha_k(0) = 0, k \neq start$$

Recursión

Para los estados de emisión

$$\alpha_i(t+1) = \sum_{j \in S} \alpha_j(t) t_{ij} e_{iX^{t+1}} = \sum_{j \in S - \{i\}} \alpha_j(t) t_{ij} e_{iX^{t+1}}$$

Para los estados de eliminación

$$\alpha_i(t+1) = \sum_{j \in N - \{i\}} \alpha_i(t+1) t_{ij}$$

1.5.5.2 Algoritmo de búsqueda hacia atrás.

Define $\beta_i(t) = P(X^{t+1} \ldots X^t \mid S = i, w),$ la probabilidad en el estado i en momento t con la observación parcial de secuencia desde X hasta final (end). Obviamente

$$\beta_{end}(T) = 1.$$

La ecuación para calcular β_S:

$$\beta_i(t) = \sum_{j \in N + \{i\}} \beta_j(t+1) t_{ji} e_{jX^{t+1}}$$

para los estado de emisión. Para los estados de eliminación:

$$\beta_i(t) = \sum_{j \in N+\{i\}} \beta_j(t) t_{ji}$$

1.5.5.3 Algoritmo de Viterbi.

Se definen las variables

$$\delta_i(t) = \max_{\pi_i(t)} P(\pi_i(t) \mid w)$$

donde $\pi_i(t)$ representa un camino "prefijo", con las emisiones X1…Xt termina en el estado i. Entonces $\delta_i(t)$ es la probabilidad asociada al camino más probable el cual tiene los t primeros símbolos de O y termina en el estado i. Estas variables se actualizan usando el mecanismo de propagación como los algoritmos hacia delante o atrás:

$$\delta_i(t+1) = [\max_j \delta_i(t) t_{ij}] e_{iX^{t+1}}$$

para los estados de emisión, y para los estados de eliminación tenemos:

$$\delta_i(t+1) = [\max_j \delta_i(t) t_{ij}]$$

1.5.6. Arquitectura de un Modelo Oculto de Markov

La arquitectura de un modelo de Markov es el diagrama que representa las transiciones y emisiones que hay entre los estados de dicha cadena.

El diagrama que se encuentra en la figura 1.2 muestra la arquitectura general de un HMM. Cada óvalo representa una variable aleatoria que puede tomar determinados valores. La variable aleatoria $x(t)$ es el valor de la variable oculta en el instante de tiempo t. La variable aleatoria $y(t)$ es el valor de la variable observada (emitida) en el mismo instante de tiempo t. Las flechas indican dependencias condicionales.

Del diagrama queda claro que el valor de la variable oculta $x(t)$ (en el instante t) *solo* depende del valor de la variable oculta $x(t-1)$ (en el instante $t-1$). Esto es lo que llamamos antes propiedad de Markov de orden 1. De forma similar, el valor de la variable observada $y(t)$ solo depende del valor de la variable oculta $x(t)$ (ambas en el instante t).

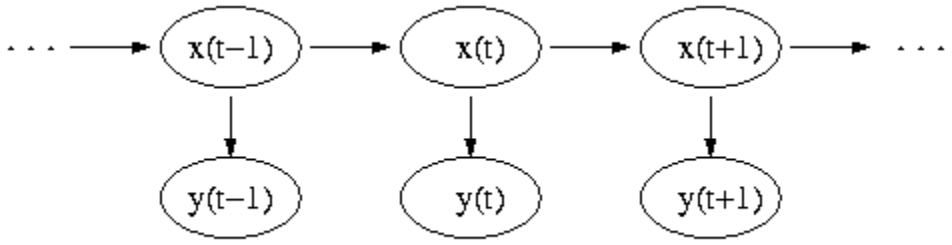


Figura 1.2 Arquitectura de un modelo oculto de Markov

Sin embargo, este diagrama no ilustra mucho el tránsito por los estados y puede dar la idea de que el procedimiento es lineal. En realidad es solo lineal a lo largo del tiempo. Las transiciones pueden conducir al retorno a estados anteriores, e incluso a lazos dentro de un mismo estado. Por ello a veces la arquitectura se representa mejor en forma de un grafo dirigido cuyos nodos son los estados (utilizando además un estado inicial y un estado final) y cuyas saetas son las transiciones. El peso de cada arco representa la probabilidad de la transición y asociado a cada nodo hay una distribución de probabilidad emisión de símbolos, ordenados esos previamente:

Un ejemplo muy simple de un Modelo Oculto de Markov y su arquitectura puede ser el mostrado en la figura 7.2. En este caso hay dos estados (que pueden imaginarse como dos “dados”, cada uno de 4 caras) y que pueden “emitir” los símbolos A, C, G o T. Los estados se incrementan con un estado inicial (Inicio) y un estado final (Fin), que no emiten símbolos

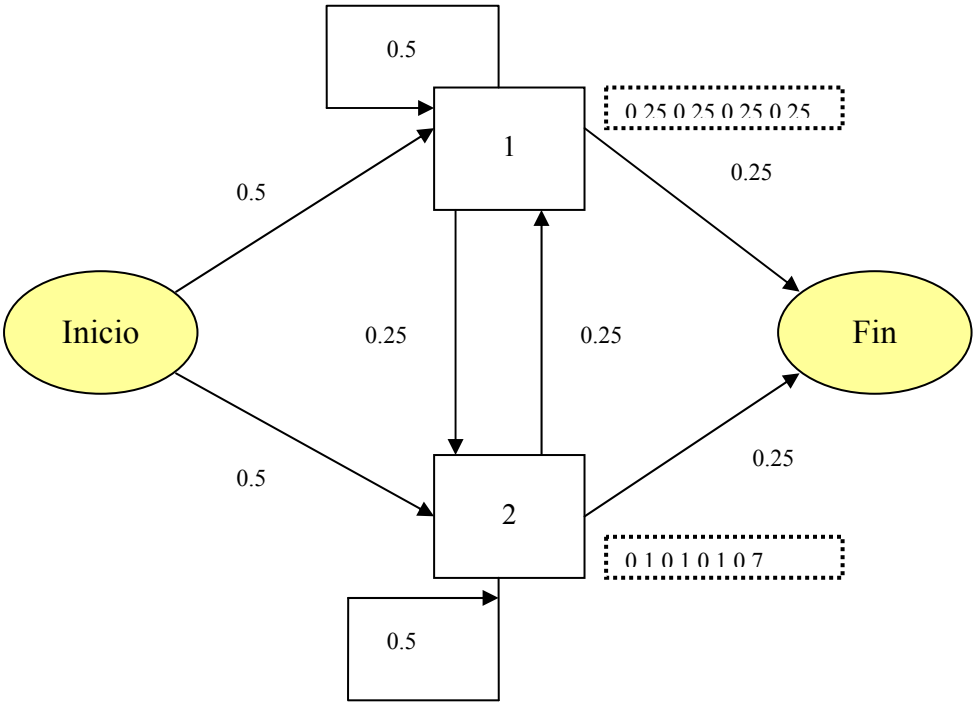


Figura 1.3 Arquitectura de un Modelo Oculto de Markov como grafo dirigido (Tomado de Pierre Baili y Soren Brunak)

En el primer estado intermedio el vector de probabilidades de emisión es:

$$(e_{1A}, e_{1C}, e_{1G}, e_{1T}) = (0.25,0.25,0.25,0.25)$$

En el segundo estado intermedio el vector de probabilidades de emisión es:

$$(e_{2A}, e_{2C}, e_{2G}, e_{2T}) = (0.1,0.1,0.1,0.7)$$

Las probabilidades de transición se indican en los arcos del grafo. Suponga ahora que tenemos una secuencia tal como ATCCTTTTTTTCA. Podríamos preguntarnos

- ¿Cuan probable es esta secuencia para este Modelo Oculto de Markov? (esta es la pregunta sobre la verosimilitud?
- ¿Cuál es la secuencia más probable de transiciones y emisiones a través del modelo para producir esta secuencia? (esta es la pregunta sobre la decodificación)
- Suponiendo que las probabilidades de transición y emisión no sean previamente conocidas con toda certeza, ¿cómo estos parámetros pudieran ser revisados a la luz de la secuencia observada? (este es el problema del aprendizaje)

En particular, la tercera tarea es el tipo de problema fundamental a resolver a partir de una base de secuencias conocidas.

1.5.7. Ventajas y desventajas de los Modelos Ocultos de Markov en general.

Las ventajas de los Modelos Ocultos de Markov en aplicaciones a la Biología Molecular Computacional han sido reconocidos en varios trabajos (ver por ejemplo Baili). Tales modelos surgen con una sólida fundamentación estadística y algoritmos de aprendizaje eficiente. Las aplicaciones hasta ahora logradas muestran que ellos permiten entre otras cosas:

- Un tratamiento consistente de las penalidades por inserciones y eliminación
- Que las probabilidades asociadas al modelo sean aprendidas a partir de datos de secuencias
- Acomodar la longitud de las variables (secuencias) de entrada sin necesidad de un aprendizaje supervisado, a diferencia de lo que ocurre en las redes neuronales convencionales
- Generalizar flexiblemente los llamados perfiles de secuencias

Todo ello es lo que les da una gran utilidad en un espectro amplio de tareas relacionadas con el alineamiento múltiple, la minería de datos, la clasificación, el análisis estructural y el descubrimiento de patrones. Ellos pueden ser fácilmente combinados en diferentes formas modulares y jerárquicas, constituyendo por ejemplo, verdaderos multiclasificadores.

A pesar de esas ventajas, se le atribuyen dos debilidades. Primero, ellos tienen frecuentemente un número bastante grande de parámetros no estructurados, que es necesario determinar. Llegan a ser eficientes cuando se tienen un número suficientemente pequeño de datos de aprendizaje, pero al mismo tiempo tiempo, se requiere un número mínimo de datos para lograr el aprendizaje de los parámetros. Nuestro problema se ajusta precisamente a esas restricciones y por tanto no es una dificultad esencial. En segundo lugar, los Modelos Ocultos de Markov de primer orden están limitado por su propiedad markoviana precisamente de primer orden, porque ellas no pueden expresar dependencia entre estados ocultos y menos a largo “plazo”, como las relaciones que ocurren entre posiciones alejadas de la proteína que son necesarias para el folding, aunque ellos son capaces de captar correlaciones a largo plazo si son expresadas en forma constante a través de una familia de secuencias. De cualquier manera, no es este un

problema esencial en el análisis de clasificación que vamos a emprender y que está basado fundamentalmente en la preferencia del uso de codones por plantas y patógenos.

1.6. Algunas particularidades sobre las plataformas de software a utilizar

Los objetivos a tomar cuando elegimos las herramientas a utilizar para implementar el algoritmo son la licencia libre, la independencia de ambiente en particular el sistema operativo a correr el software, las facilidades de programar como la modificación. Así elegimos el lenguaje de programación de código abierto Java con el ambiente integrado para desarrollar *NetBeans* gratis. El lenguaje Java tiene otra ventaja es muy fácil de programar y es independiente de sistema operativo y tiene una herramienta *hibernate* que nos brinda otra independencia al conectar con tipo de servidor de la base de datos.

Y acompañado con estas ventajas es la existencia de un paquete de código abierto *Biojava* el cual brinda muchas herramientas para procesar los problemas biológicos.

1.7. Consideraciones finales del Capítulo

En este capítulo se han introducido en primer lugar conceptos básicos de Biología y una formulación de nuestro problema de aplicación. Se han comentado diferentes herramientas de aprendizaje supervisado que pudieran ser utilizadas para la solución de nuestro problema y se han dado argumentaciones sobre el por qué intentar soluciones con los Modelos Ocultos de Markov a partir de experiencias anteriores. Se han introducido los conceptos básicos relacionados con este tipo de modelo y las ventajas y desventajas de los mismos. Finalmente se han comentado características fundamentales de las plataformas de software en que implementaremos nuestro algoritmo.

Capítulo II: Precisión del algoritmo e implementación del software

En este capítulo se precisará el algoritmo de entrenamiento, así como los aspectos fundamentales de la implementación de un software que se ha denominado *USeparator* y que sirve tanto para el entrenamiento de dos Modelos Ocultos de Markov, uno para cada especie, así como la clasificación de nuevas secuencias incógnitas. Se caracteriza la estructura de los ficheros estándar que usa *USeparator* y finalmente se describe un breve manual de usuario de este software.

2.1. Precisión del algoritmo

El Modelo Oculto de Markov aplicado en este trabajo es de primer orden. El conjunto de estados tiene 4 elementos *Start,* AT_3, GC_3 y *End* (o abreviadamente: k_0 , k_1, k_2 , k_3) con determinadas probabilidades de las transiciones ($a_{i\to j}$) del estado k_i al k_j y de las emisiones ($e_j(i)$) desde los estados k_j, j=1,2 . Ver figura 2.1.

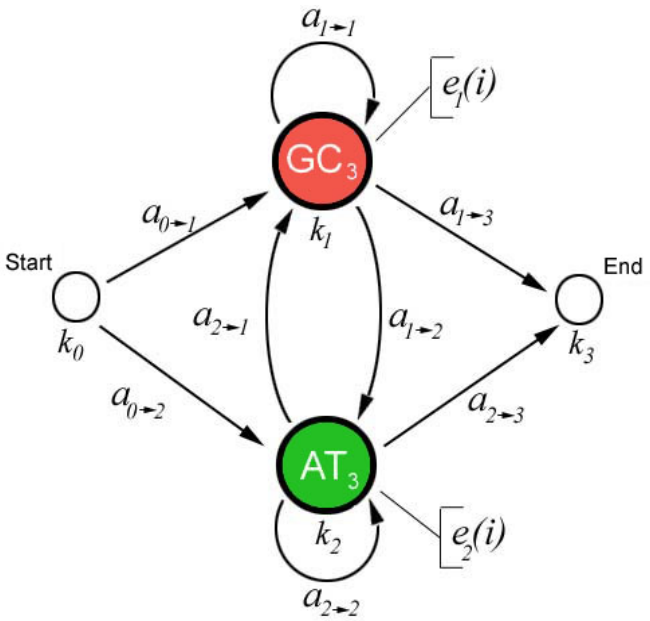


Figura 2.1 Modelo Oculto de Markov de primer orden diseñado para modelar secuencias típicas de Plantas y secuencias típicas de Hongos

Se usarán dos modelos de este tipo (con diferentes conjuntos de probabilidades de transición y emisión) para identificar secuencias típicas de la planta y el hongo respectivamente. Las probabilidades de

transiciones están especificadas en la estructura como etiquetas de los arcos que unen los estados. Para cada estado emisor del modelo: k_1 y k_2 la probabilidad de las emisiones ($e_1(i)$, $e_2(i)$) representan dos grupos de N valores. N es la cantidad de los aminos ácidos que tienen una diferencia significativa de uso de tipo de codones (AT_3 o GC_3) en las dos especies. En nuestro problema *Hordeum vulgare-Blumeria graminis* (hongo y planta) sólo hay dos aminoácidos que no presentan esta diferencia entonces N es 18.

2.1.1. Entrenamiento en términos de la estimación de las probabilidades de transición y emisión

Calculamos estas probabilidades de emisiones de los dos estados AT_3 y GC_3 como se comentó anteriormente:

$$e_1(i) = \frac{GC_3(i)}{GC_3(i) + AT_3(i)}$$

$$e_2(i) = 1 - e_1(i)$$

donde $AT_3(i)$ es la frecuencia con que aparece el aminoácido i en las secuencias de entrenamiento.

Las probabilidades de transiciones desde el estado de inicio k_0, y las arribantes al estado de terminación k_3 son las probabilidades que las secuencias dadas comiencen o terminen con los codones de tipo AT_3 o GC_3. Más precisamente:

Refiriéndonos a las transiciones que parten del estado inicial atendemos a las secuencias que comienzan con codones tipo GC_3 o AT_3

$$a_{0\to 1} = \frac{GC_3}{GC_3 + AT_3}$$
$$a_{0\to 2} = \frac{AT_3}{GC_3 + AT_3}$$
$$a_{0\to 2} = 1 - a_{0\to 1}$$

Refiriéndonos a las transiciones que arriban al estado final atendemos a las secuencias que terminan con codones tipo GC_3 o AT_3

$$a_{1\to 3} = \frac{GC_3}{GC_3 + AT_3}$$
$$a_{2\to 3} = \frac{AT_3}{GC_3 + AT_3}$$

donde GC_3 y AT_3 representan ahora las cantidades de secuencias que finalizan con codones de estos tipos respectivos. Observe que utilizamos las mismas letras para no complicar demasiado las notaciones pero se refieren a una posición diferente en la secuencia que contabilizamos. Además, aunque no son probabilidades asociadas a transiciones del mismo estado, está claro que se cumple que:

$$a_{2\to 3} = 1 - a_{1\to 3}$$

Escogimos las cantidades de secuencias que empiezan o terminan en estos codones y no en codones de inicio y terminación porque las *ESTs* son realmente subsecuencias que se cortan en posiciones aleatorias de las secuencias completas.

Las probabilidades de transiciones entre los estados k_1 y k_2: $a_{i\to j}, i = 1,2; j = 1,2$ son

$$a_{i\to j} = \frac{C_{i\to j}}{C_{i\to j} + C_{i\to i}}$$

$C_{i\to j}$ es la cantidad de veces que cambian de estado i al j (j≠*i*)

$C_{i\to i}$ es la cantidad de veces que se mantienen en el estado i

Con estas definiciones se puede hacer una estimación completa de dos Modelos de Markov, uno para la planta y otro para el patógeno, a partir de sendas bases de datos de entrenamiento.

2.1.2. Clasificación en términos del mejor ajuste a las probabilidades asociadas al Modelo

La probabilidad generada cuando una secuencia dada recorre un camino *S* en el modelo con los parámetros θ (probabilidades de emisiones y transiciones) viene dada por:

$$P(S,\theta) = \prod_{i \in S} a_{i-1,i} * e_i$$

donde *a,y* e_i representan las probabilidades de transición y emisión respectivamente obtenidas para cada modelo.

Para cada secuencia incógnita se calculan doce probabilidades de este tipo con respecto a seis *ORF* (del inglés *Open Reading Frame*, o en español, Marco Abierto de Lectura) en el Modelo del hongo y seis *ORF* en el Modelo de la planta.

Pero además, la secuencia puede ser muy larga, y entonces el producto de las probabilidades muy pequeño. Para evitar esto, calculamos dichas probabilidades relativas a las obtenidas en una secuencia general S_r generada aleatoriamente y con la misma longitud de la secuencia incógnita. Para ganar fiabilidad en los valores de las probabilidades asociadas a esta secuencia aleatoria, y reducir los errores de cálculo, construimos no una sino varias secuencias aleatorias de este tipo y promediamos las probabilidades. En nuestro caso trabajamos con diez secuencias. Finalmente utilizamos el logaritmo de la relación de las probabilidades para evitar desbordamiento u otros errores de cálculo (R. Durbin 1998) .

$$R_{i,j} = \log_2 \frac{P(S_i,\theta_j)}{P(S_r,\theta_j)}$$

donde:

- i = -3,-2,-1,+1,+2,+3 son los *ORF*, de manera que S_i es la secuencia S incógnita considerada con el *ORF i*. Por el carácter aleatorio de S_r no es necesario considerar en ésta diferentes *ORF*
- j = Hongo o Planta

$R_{i,j}$ es el valor obtenido después de recorrer la secuencia en el Modelo Oculto de Markov con los parámetros determinados. Si al recorrer la secuencia con el *ORF j* encontraramos k codones de parada entonces (ver penalidad de ocurrencia de parada (*SCO*) en epígrafe 3.2.2)

$$R'_{i,j} = R_{i,j} - k*SCO$$

$$R_{max} = \max_{i,j} \{R'_{i,j}\}$$

A partir del resultado de R_{max} obtenemos la especie y el *ORF* "biológico" como los argumentos, en términos de j y de i de dicho máximo.

2.2. Implementación de USeparator

Ahora se explica la implementación del software que permite reducir el tiempo de desarrollo el algoritmo y la interfaz cómoda para los usuarios.

El uso de lenguaje unificado de modelado (UML) (Booch 1998) es muy importante para reducir el tiempo de implementación que es de hecho un riesgo en el presente trabajo. La búsqueda de una solución de un problema que realmente no estaba tan clara, requiere en mayor medida las facilidades de mostrar los resultados intermedios del algoritmo a los fines de poder modificarlo. Una vez que se tienen implementados los resultados de este algoritmo se trabaja en programar las facilidades de entrada para el almacenamiento de información, esto es, la interfaz para el usuario. Resulta así imprescindible el desarrollo en forma de cascada que se explica en el libro del autor citado anteriormente.

2.4.1. Los casos de uso y el actor

El software tiene dos casos de uso y un solo actor al sistema (ver Figura 2.2).

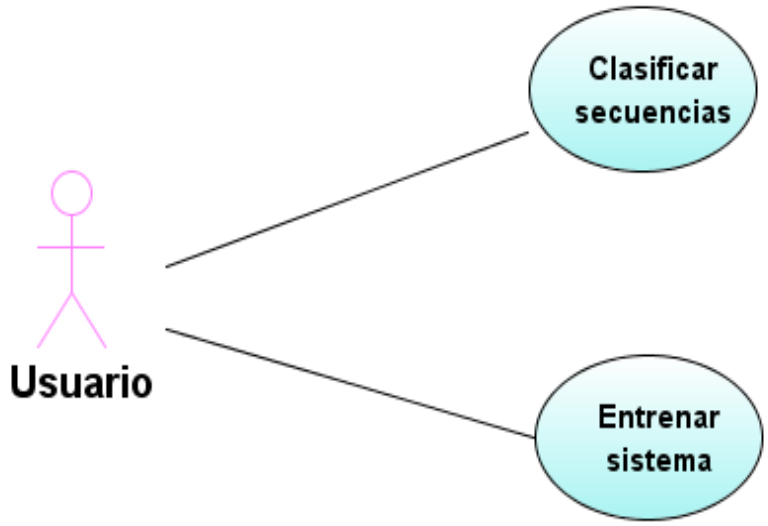


Figura 2.2 Diagrama de casos de uso

- **Entrenar sistema**

Permite al usuario incorporar las secuencias de cada una de las dos especies y con *ORF* biológico conocido (siempre en la posición 1), en ficheros de formatos estándares al sistema y crear los Modelos Ocultos de Markov de las dos especies (calcular sus parámetros) a partir de estos dos grupos de secuencias.

- **Clasificar las secuencias**

Permite al usuario incorporar nuevas secuencias incógnitas almacenadas en los ficheros de formato estándares del sistema.

El usuario puede definir parámetros adicionales del algoritmo (como penalidades por presencia de codones de terminación).

Clasificar las secuencias incógnitas y obtener el *ORF* "biológico" de las mismas.

2.4.1. Las relaciones entre los paquetes y las clases

El presente software se terminó con la ayuda de varios paquetes de código abierto. En la figura 2.3 se presentan las relaciones de estos paquetes y los programados específicamente para *USeparator*.

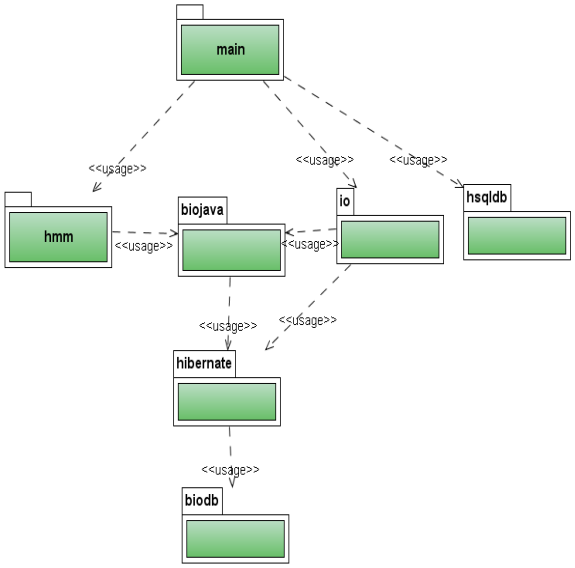


Figura 2.3 Diagrama de relaciones de los paquetes

El paquete *hsqldb* brinda un conjunto de herramientas para activar un servidor de base de datos tipo *hsqldb* en la misma máquina (HSQLDB 2007). Así, con esta herramienta el software puede trabajar con una base de datos profesional.

Biojava es un proyecto muy importante de Bioinformática implementado en lenguaje de programación Java (Schreiberm 2005). El brinda facilidades para trabajar con las secuencias biológicas, base de datos biológicos y formatos estándares de biología.

Una vez que se usa el lenguaje Java se obtiene una independencia de los sistemas operativos. Otra ventaja que este lenguaje brinda a los fines de actuar con las bases de datos, es una herramienta (*Hibernate*) que permite trabajar con tales bases sin que el desarrollador tenga que preocuparse del tipo de servidor donde se encuentra (ver figura 2.4) (Hibernate 2007).

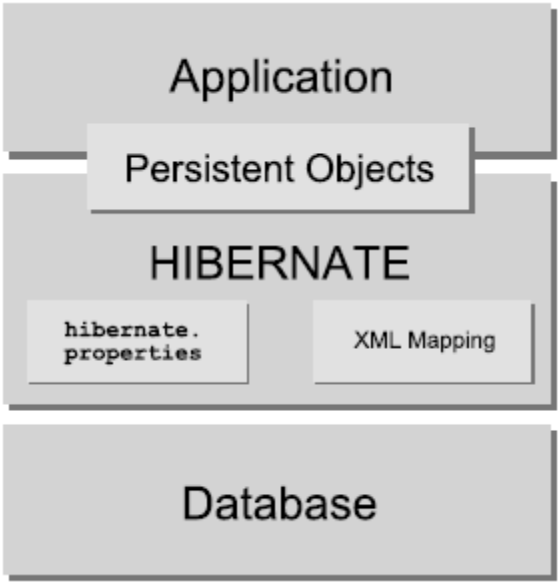


Figura 2.4 La funcionalidad de herramienta *hibernate*

En el diagrama de la Figura 2.3 las clases que están en el paquete *main* son las clases de interfaces que interactúan con las clases de control en los paquetes *io* y *hmm* para que el usuario entre los datos y mostrarle los resultados al mismo.

Las dos clases en el paquete *io* usan las facilidades que brinda *biojava* para validar las secuencias antes de adicionarlas a la base de datos así como darles el formato apropiado para que *main* pueda mostrarlas al usuario y almacenarlas en formatos estándares.

El paquete *hmm* tiene dos clases principales y tienen relaciones con las clases externas (ver Figura 2.5). Las otras clases le ayudan a cumplir los requisitos más fáciles. Un objeto de la clase *Trainer* recibe las secuencias clasificadas con *ORF* biológico (por separado de Hongos y Plantas), las procesa, y almacena los datos procesados en un fichero. Una vez lo se tiene entrenado y se suministra una nueva secuencia incógnita, un objeto de la clase *Classifier,* la clasifica como especie y reporta el *ORF* "biológico".

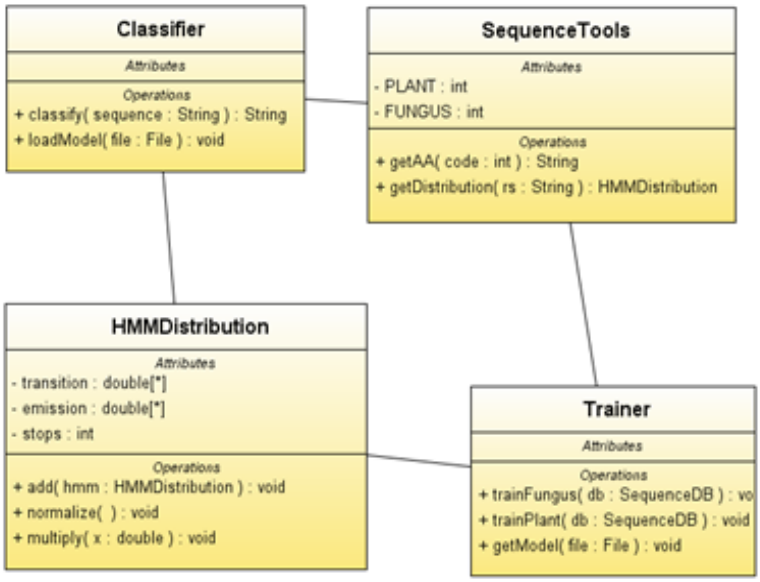


Figura 2.5 Diagrama de clases del paquete hmm

2.3. Sobre la estructura de los ficheros estándares

El software adiciona y exporta las secuencias en los ficheros en formato estándar *fasta* (Genomatix 2005). En este formato cada secuencia tiene una línea de atributos como el identificador, comentarios, longitudes, propiedades, etc. y una línea de la secuencia. Así una secuencia está atrás de otra.

Cada secuencia pertenece a un espacio de nombre. Lo usamos para agrupar las secuencias. Por eso cada vez que incorporamos nuevas secuencias tenemos que entre este espacio de nombre. Una vez tenemos las secuencias clasificadas, este resultado está en el campo propiedades (de inglés *properties*) de la secuencia.

2.4. Manual de usuario

En este epígrafe se describe en resumen como el usuario puede realizar las dos tareas fundamentales del sistema: el entrenamiento y la clasificación

2.4.1. Cómo entrenar el sistema

- Ir a la pestaña *Train* de la ventana de *USeparator* (ver Figura 2.6).

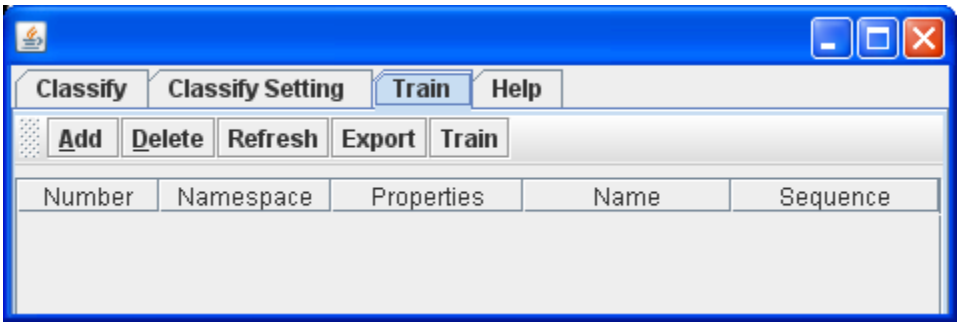


Figura 2.6 Pestaña *Train* de *USeparator*

- Adicionar las secuencias de las dos especies (Hongo o planta) en dos espacios de nombre diferentes (en inglés *namespace)* desde ficheros de formatos estándares. Esta acción se realiza presionando el botón *Add* y aclarando el espacio de nombre de las secuencias que van a ser adicionadas (ver Figura 2.7).

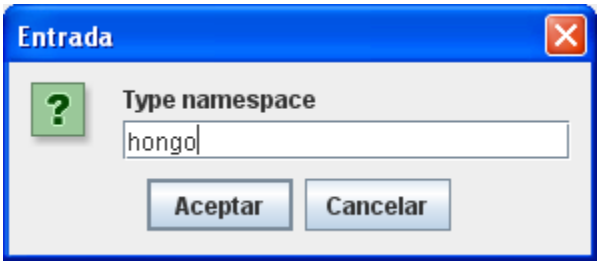


Figura 2.7 Diálogo de entrada de espacio de nombres

Se pueden eliminar las secuencias que no interesen mediante la selección de las mismas y la presión del botón *delete*.

- Se ordena el entrenamiento presionando el botón *Train* y seleccionando los espacios de nombres correspondientes a las dos especies Hongo y Planta (Ver Figura 2.8 y 2.9).

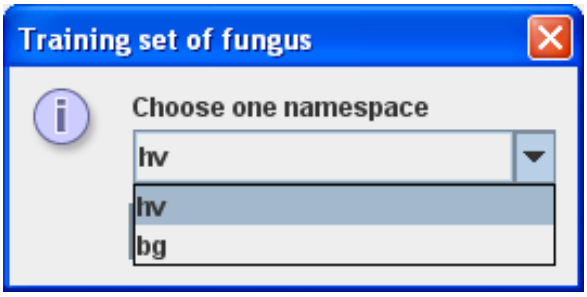


Figura 2.8 Selección de espacio de nombre de secuencias para Hongo *Hordeum vulgare* (hv)

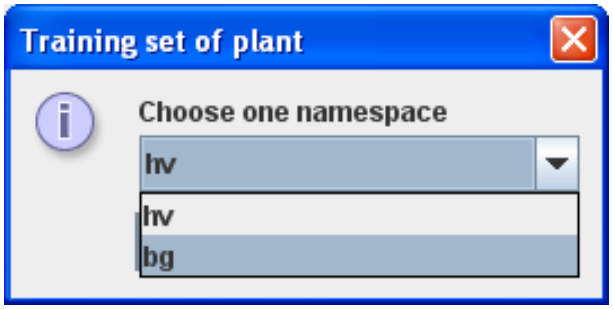


Figura 2.9 Selección de espacio de nombre de secuencias para Planta *Blumeria graminis* (bg)

Al final se debe salvar el fichero de resultados de entrenamiento (ver Figura 2.10).

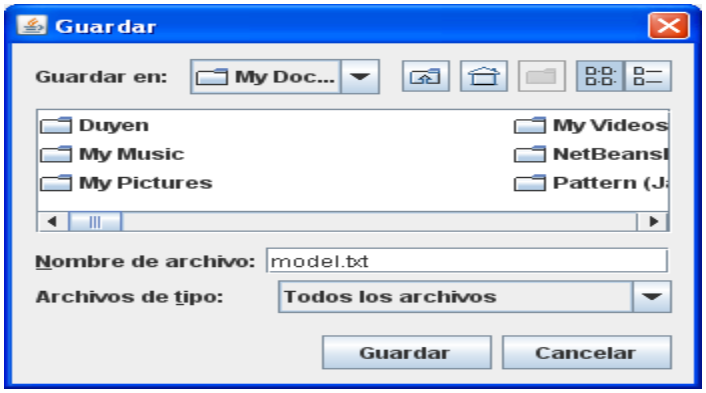


Figura 2.10 Salva del fichero resultado del entrenamiento

2.4.1. Cómo clasificar nuevas secuencias

- Si anteriormente se ha hecho el entrenamiento de sistema, el fichero con los resultados del mismo debe estar salvado con su formato específico. Si no se, ha hecho un nuevo entrenamiento, utilizará como tal, a los fines de la clasificación el fichero por defecto ya entrenado: *training_data.txt* que yace en el directorio que contiene a *USeparator.jar*. Si quiere cambiar este fichero o la penalidad de ocurrencia de codón de parada (*SCO penalty*) se puede ir a la pestaña *Classify Setting* (ver Figura 2.11).

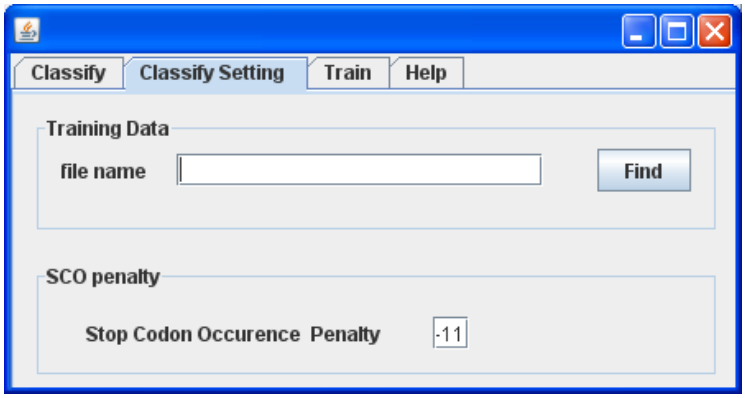


Figura 2.11 Pestaña *Classify Setting* de USeparator

- Ir a la pestaña *Classify* de USeparator(ver Figura 2.4.7).

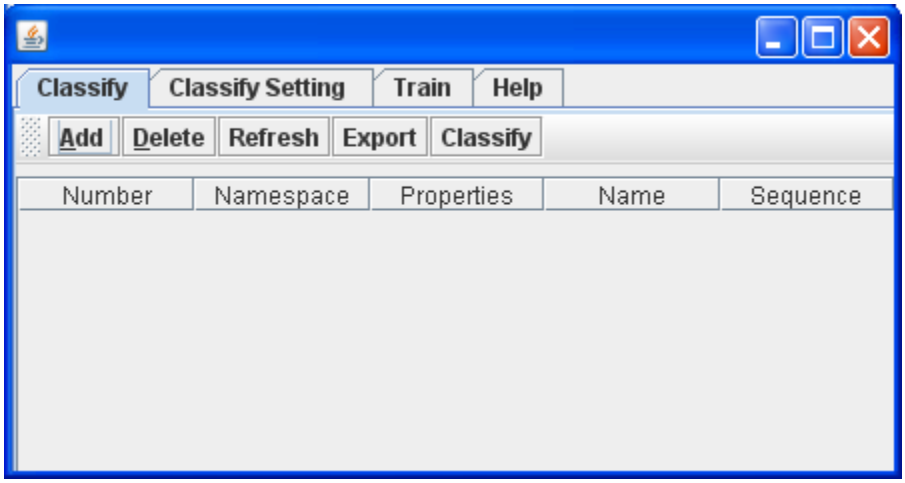


Figura 2.12 Pestaña *Classify* de USeparator

- Adicionar las secuencias o eliminar algunas tal como se trabaja en el proceso de entrenamiento descrito anteriormente.
- Clasificar estas secuencias presionando el botón *Classify*. Los resultados se observan en el campo *Properties* de la tabla.
- Si se quiere exportar estas secuencias a ficheros de algún formato estándar, presionar el botón *Export* y guardar estos ficheros.

2.5. Conclusiones parciales del capítulo

En este capítulo se describe la parte fundamental del trabajo realizado desde el punto de vista computacional, teniendo en cuenta la teoría de algoritmos, la estructura de datos y la programación. Por eso, las conclusiones parciales de este capítulo se refieren, no necesariamente a la validación de los resultados finales (que están en el Capítulo 3) sino más bien, a los procedimientos computacionales

experimentados y la forma en que fueron desarrollados para producir dichos resultados y que pueden ser extendidos al análisis de otros problemas de separación de genes de otras parejas de especies planta-hongo.

- La idea de la separación de secuencias codificantes de genomas de hongos y plátanos se concretó en el diseño de dos Modelos de Markov de primer orden, uno para cada especie, con la misma estructura pero con parámetros (distribuciones de probabilidades) específicas. La estructura de tales modelos concibió los estados en términos de la preferencia del uso de codones en estas especies y los parámetros de los mismos se pueden aprender a partir de bases de entrenamiento de secuencias conocidas de estas dos especies, donde además se conocen los marcos de lectura. Se describieron las formas de calcular los parámetros de los modelos a partir de las respectivas bases de datos de secuencias de hongos y plantas.
- A los fines de la clasificación de una nueva secuencia incógnita, cuyos marcos de lectura no son conocidos previamente, se analiza el problema de su verosimilitud ante los dos modelos de Markov y se decide por la máxima en cualesquiera de los seis posibles marcos de lectura. Se calcula la máxima probabilidad de aceptación de la secuencia en cada uno de los modelos, a partir de las probabilidades determinadas en el entrenamiento y ciertos parámetros adicionales. en particular las penalidades impuestas al encontrar sorpresivamente codones de terminación en la subsecuencia que se clasifica.
- Algunas modificaciones de este algoritmo fueron novedosamente impuestas para que el cálculo de estas probabilidades fuera relativo al de frecuencias generadas aleatoriamente y finalmente fueran influenciadas por una función logaritmo, que evitara desbordamiento y otros errores en los cálculos.
- Se describen los aspectos fundamentales del software implementado, los casos de uso y los actores, en lenguaje UML así como las relaciones entre paquetes profesionales (*Open Source*) y los específicamente elaborados para el sistema. Esto es parte importante de las posibilidades de la generalización del software para otros pares planta-hongo
- Se caracterizan en particular, las estructuras estándares de los ficheros que utiliza el software, procurando sus posibilidades de intercambio a nivel internacional a través de la plataforma Biojava. También esto es parte importante de las posibilidades de la generalización del software para otros pares planta-hongo
- Finalmente se hace una breve descripción del Manual de Usuarios del USeparator que concreta como realizar las tareas en los dos casos de uso: el entrenamiento y la clasificación y que es generalizable también a otros problemas de separación planta-hongo, al menos de características biológicas similares

Capítulo III: Validación del algoritmo y el Software

La validación del algoritmo y el software elaborados requiere la comparación de su poder de clasificación respecto a otros algoritmos existentes, mencionados en la introducción. El principal interés consiste en demostrar que es posible obtener valores de clasificación similares a los obtenidos por la metodología basada en *Support Vector Machine* partiendo de un conjunto de entrenamiento mucho menor que el usado en este caso.

Para ello se construyen dos conjuntos de validación por especie. El primero de estos es denominado conjunto de validación reducido, consistente de solo 30 secuencias por especie de las que se tiene validación experimental del *ORF* biológico. Dichas secuencias fueron obtenidas de la base de datos GenBank (Pubmed 2008). Por el carácter limitado de la información disponible, se conforman conjuntos extendidos de entrenamiento, mediante una cierta metodología que se detalla en el primer epígrafe del presente capítulo.

Por otra parte el propio proceso de validación sugiere optimizar el algoritmo utilizando información sobre el número óptimo de secuencias a tratar e información biológica adicional, específicas sobre el problema en cuestión. Finalmente la comparación de los resultados con otros clasificadores deben valorarse en términos de eficacia (absolutos) de exactitud de la clasificación en sí y de la eficiencia (relativos) en relación a la cantidad y longitud de las secuencias necesarias para el entrenamiento.

3.1. Creación del conjunto de validación extendido

La disponibilidad de secuencias validadas experimentalmente respecto al *ORF* biológico para las especies estudiadas es realmente limitada. Teniendo en cuenta que el algoritmo elaborado basa su poder de clasificación en el uso de codones, es obligatorio conocer el *ORF* biológico de las secuencias que se emplean para su entrenamiento. Por ello, para crear un conjunto de validación extendido fue necesario desarrollar una metodología que permitiera inferir con gran certeza el *ORF* activo en secuencias reportadas en GenBank sin validación experimental.

La principal fuente de este tipo de secuencias resulta de experimentos de obtención de ESTs. Sin embargo, producto del desarrollo experimental mediante el cual estas se obtienen, existen múltiples errores en sus secuencias. Por otro lado, se debe tener en cuenta que las colecciones de ESTs son redundantes pues en muchos casos provienen de variantes de *splicing* de un mismo gen o resultan de la secuenciación de clones sobre- representados en una condición biológica determinada. De no reducir dicha redundancia,

podríamos finalizar en un conjunto de validación que no sea representativo del uso de codones de las especies estudiadas. Esto afectaría grandemente el poder de clasificación del algoritmo.

3.1.1. En el caso de Blumeria graminis (Planta)

Para disminuir el efecto de la baja calidad de las colecciones de ESTs de *B. graminis* (planta) se aplica el flujo de trabajo representado en la Figura 3.1. De la base de datos GenBank se obtuvo una colección consistente de 4025 ESTs. Estos fueron ensamblados mediante el algoritmo CAP3 (Xiaoqiu Huang 1999) generándose 1246 contigs. Posteriormente se analiza la posible redundancia funcional de estas secuencias y para ello estas se agruparon mediante el algoritmo tribe-MCL (Hoon 2007). Como resultado se obtuvieron 512 grupos, para conformar el conjunto de validación final se toman 440 secuencias representativas de estos tratando de cubrir tantas categorías funcionales como fuera posible. De esta forma se puede caracterizar más eficientemente el uso de codones en el contexto genómico de la especie. Cada una de las 440 secuencias fue objeto de un análisis de homología por BlastX para inferir su *ORF* biológico. En principal parámetro de este análisis es el valor de corte y se usa E=1e-15.

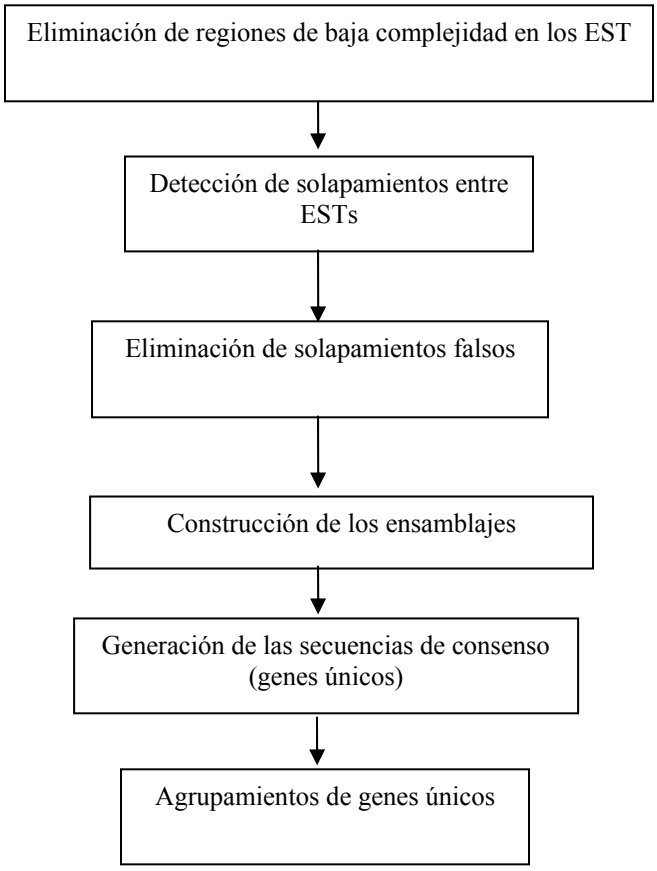


Figura 3.1 Los pasos principales de algoritmo de ensamblaje.

3.1.2. En el caso de Hordeum vulgare (Hongo)

En el caso de *Hordeum Vulgare* (hongo) fue posible obtener del GenBank secuencias anotadas como fragmentos completos codificantes para genes. Se obtiene un total de 225 secuencias entre las cuales se lleva a cabo una búsqueda por homología mediante BlastX y al filtrar por el valor E = 1e-15 se seleccionan finalmente 172 secuencias.

3.2. Optimización del algoritmo a partir de criterios computacionales e informaciones biológicas adicionales

En este proceso se optimizan los parámetros del algoritmo, y se brindan recomendaciones al usuario a la hora de usar el software. Después que se tienen los parámetros mejores se hacen pruebas estadísticas para comparar el poder de clasificación con los productos de software previamente existentes para resolver este mismo tipo de problemas.

3.2.1. Optimización de la cantidad de secuencias en el entrenamiento

En un primer análisis se evalúan los efectos de la cantidad de secuencias usadas en el conjunto de entrenamiento sobre la estabilidad de los parámetros del algoritmo (probabilidades de emisión y transición) para determinar el conjunto mínimo requerido por el mismo. El objetivo es tener estos parámetros estables, que describan realmente el consenso de toda la especie y evitar trabajar con máximos locales. Por eso se debe buscar el tamaño mínimo del conjunto de secuencias de entrenamiento a partir del cual dichos parámetros empiezan a converger.

Con este objetivo, se divide el conjunto de entrenamiento total para cada especie en tantos subconjuntos como fue posible de tamaño 10, 20, 30, 40. Luego se obtuvieron las probabilidades de los modelos ocultos de Markov correspondientes para cada tamaño. En la Tabla 3.1 se muestra como ejemplo los valores de probabilidades para 6 modelos ocultos de Markov correspondiente al tamaño 30 para la especie *H. vulgare*. La estabilidad de los parámetros se evaluó mediante la prueba no paramétrica Wilcoxon de muestras pareadas con la ayuda de paquete estadístico SPSS versión 13 para Windows. Este análisis estadístico mostró que no existen diferencias significativas entre los modelos ocultos de Markov que provienen de subconjuntos con una tamaño igual o mayor a 20 secuencias, para *B. graminis* y de 30 secuencias o más para *H. vulgare*.

Tabla 3.1 Valores de las probabilidades del HMM para seis conjuntos de entrenamiento consistentes en 30 secuencias cada uno de *H. vulgare.* Se utilizan en la primera columna, las abreviaturas clásicas de tres letras de los aminoácidos

	1	*2*	*3*	*4*	*5*	*6*
pro	0,6611	0,5274	0,6405	0,5338	0,5168	0,5249
thr	0,6719	0,5787	0,6290	0,5281	0,6082	0,5726
ser	0,6575	0,5687	0,6548	0,5802	0,6121	0,6018
ala	0,6661	0,6267	0,7173	0,5765	0,6331	0,5776
his	0,6706	0,5667	0,7057	0,5663	0,6289	0,5724
asn	0,7391	0,6349	0,7581	0,5771	0,6476	0,6865
lys	0,7030	0,6092	0,6917	0,5958	0,6554	0,5841
tyr	0,6994	0,6554	0,6045	0,5973	0,5847	0,5735
asp	0,6754	0,6768	0,6904	0,6107	0,5789	0,6398
glu	0,6570	0,5860	0,7074	0,5441	0,5392	0,5550
leu	0,7330	0,6732	0,7500	0,6500	0,6661	0,6257
ile	0,7268	0,6951	0,7508	0,6708	0,6617	0,6960
met	0,5635	0,5182	0,5330	0,4746	0,5608	0,4627
phe	0,6981	0,6567	0,6567	0,6331	0,6150	0,6500
val	0,7273	0,6794	0,7372	0,6517	0,7103	0,6343
arg	0,7043	0,6209	0,7292	0,6193	0,6338	0,6333
cys	0,6655	0,5605	0,6471	0,5920	0,5978	0,6236
gly	0,6598	0,6121	0,7193	0,5696	0,6189	0,5745
AT AT	0,5888	0,5941	0,5728	0,5935	0,5859	0,6001
GC AT	0,7025	0,7080	0,8013	0,7162	0,7739	0,7221
Start AT	0,6443	0,6652	0,6861	0,6421	0,6521	0,6320

También se comprueba el efecto de la calidad de la anotación biológica de las secuencias de entrenamiento sobre la estabilidad de los parámetros del modelo. Para ello se comparan los resultados provenientes del conjunto de entrenamiento reducido y de varios subconjuntos del conjunto extendido conteniendo un mismo número de secuencias. Puede verse en la tabla 3.2. que en todos los casos la diferencia en el uso de codones de las dos especies es mucho más notable en el conjunto de secuencias reducidas que en el conjunto de secuencias extendidas.

Tabla 3.2 Probabilidades de los Modelos Ocultos de Markov resultantes del entrenamiento con el conjunto reducido y conjunto extendido. Se utlizan en la primera columna, las abreviaturas clásicas de tres letras de los amonácidos

	H.vulgare reducido	*B.graminis reducido*	*H. vulgare extendido*	*B. graminis* extendido
pro	0,7254	0,3591	0,5647	0,3429
thr	0,7500	0,3091	0,5918	0,3959
ser	0,7797	0,3621	0,6101	0,4201
ala	0,7162	0,3608	0,6283	0,3888
his	0,8116	0,506	0,6116	0,4514
asn	0,8372	0,4072	0,6676	0,3912
lys	0,8703	0,4274	0,6325	0,4284
tyr	0,8764	0,4741	0,6165	0,3388
asp	0,8922	0,5057	0,6471	0,4671
glu	0,8085	0,347	0,595	0,3927
leu	0,863	0,3806	0,6788	0,3886
ile	0,8441	0,4308	0,6991	0,4612
met	0,7453	0,3036	0,5085	0,2981
phe	0,8439	0,4731	0,6512	0,4309
val	0,8393	0,4418	0,6852	0,4489
arg	0,7633	0,3416	0,6531	0,4039
cys	0,8476	0,4071	0,6114	0,4339
gly	0,7326	0,3372	0,6184	0,3951
AT AT	0,578	0,3993	0,4756	0,4076
GC AT	0,8579	0,4322	0,7372	0,4446
Start AT	0,8055	0,4141	0,6443	0,4233

3.2.2. Concepto de ORF equivalente y Penalidad de ocurrencia de Codón de parada SCO (Stop Codon Ocurrente)

Originalmente USeparator buscaba el *ORF* (*Open Reading Frame*) "biológico" a través de leer las secuencias con los seis *ORF* posibles independientemente del organismo y consideraba que el *ORF* que daba la mejor puntuación (R_{max}) era el correcto. Pero el *ORF* localizado en la dirección complementaria del *ORF* "correcto" según este criterio, tenía una puntuación muy cercana de $R_{max,}$ incluso una puntuación mayor en varias ocasiones. Esto hacía que Useparator fallara a la hora de predecir el *ORF* más correcto. Esto llevó a considerar el concepto de *ORF* equivalente para los *ORF* que ocurren en las direcciones complementarias (ver figura 3.2)

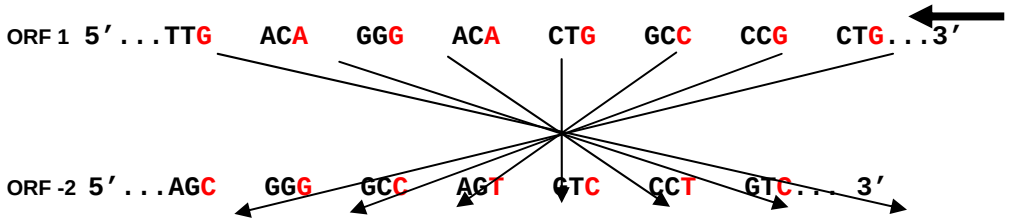


Figura 3.2 Representación esquemática del concepto de ORF equivalente.

En la figura 3.2. se puede ver que la frecuencia de los tipos de codones (GC_3, AT_3) en los *ORF* equivalentes es igual. En este caso, por ejemplo, el *ORF* 1 y *ORF* -2 casi tienen la misma cantidad de los dos tipos. Esta diferencia ya no es significativa cuando la longitud es muy grande. Así se explica para el resto de los casos.

Ahora solo el conocimiento biológico ayuda a mejorar USeparator. Se sabe que los ESTs no tienen codones de parada. El personal que trabaja en las bases de datos accesibles internacionales de nucleótidos como GenBank, EMBL, DDBL da mucha atención a este problema. Pero las secuencias aleatorias o los *ORF* incorrectos introducen mucha probabilidad de que aparezca un "codón de parada" sorpresivo (posiblemente falso). Por eso se da una penalidad por la aparición de codones de parada. Durante los experimentos con USeparator se puede apreciar que fijando la penalización de la ocurrencia de codones de parada a -11 se obtienen los mejores resultados. Ello puede ser una medida de la probabilidad de la aparición de tal error sorpresivo.

3.3. Los resultados de la validación

La validación incluye la comparación de los resultados de la clasificación con otros algoritmos en bases de datos idénticas así como la comparación de los resultados en función de la longitud de las secuencias.

3.3.1. Resultados de la clasificación

Se cuenta con una base datos compuesta por 295 secuencias de *Blumeria graminis* y 172 secuencias de *Hordeum vulgare* obtenida por los métodos previamente descritos. Se usa la técnica de prueba de validación cruzada (-en inglés *Crossvalidation*). Esta prueba consistió en dividir el conjunto de validación en 5 subconjuntos aleatorios. De estos, un subconjunto se usó como el conjunto de entrenamiento y los cuatro restantes se emplearon para la validación. Este mismo procedimiento se repitió por cinco veces.

Se obtiene una clasificación correcta de 91,1% (comparativamente: PF-IND 81,2%; ECLAT 93,1%) en organismos. Con esta clasificación se construye una curva ROC (-del inglés *Receiver Operating Characteristics*) asociada al nuevo clasificador (ver Figura 3.2.3) (Thomas 1999). El área bajo la curva fue de 0,890. Este resultado es mucho mayor al que se obtiene para clasificadores aleatorios donde el área de la curva es aproximadamente de 0,5.

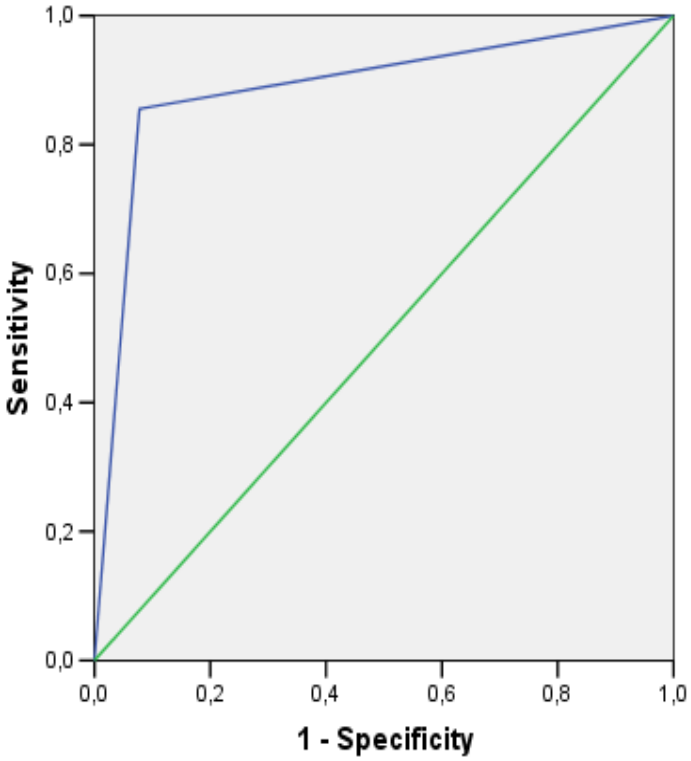


Figura 3.3 Curva ROC.

Otro resultado es el 91% de clasificación correcta del *ORF* biológico. Constituye un avance importante comparado con PF-IND de 71% y se acercó a ECLAT con 97.7%.

3.3.2. Resultados respecto a la longitud de secuencias

A partir de la base de datos por especies, se generan secuencias con longitud variable para determinar como influye este parámetro sobre el poder de clasificación. En todos los casos se procedió con una validación cruzada, como se describió anteriormente. Los resultados se muestran en la Tabla 3.3.

Tabla 3.3 Dependencia del poder de clasificación por algoritmos con la longitud de las secuencias analizadas.

	20-50	*51-100*	*101-200*	*201- 300*	*301-400*	*401-500*	*>500*
USeparator	65,46	77,11	82,5	88, 1	88,6	**91,1**	91.1
PF-IND	71.9	81.6	81.0	83.5	**86.7**	81.1	71.8
ECLAT	78.8	90.0	90.6	92.2	95.6	**95.7**	94.4

De la tabla 3.2.4 y el algoritmo usado se puede concluir que si la secuencia es más larga que 400 nucleótidos el resultado alcanza su máximo poder de clasificación lo cual no se afecta como en otros algoritmos (particularmente el PF-IND) con el aumento de la longitud de la secuencia. Ello se debe al hecho de que el algoritmo incluye la penalización por ocurrencia de supuestos codones de parada.

3.4. Conclusiones parciales del capítulo

En este capítulo se resumen los resultados del proceso de validación del algoritmo y el software, aplicados concretamente al problema de la separación de genes en el par de especies específicamente considerado; pero resulta particularmente interesante como se utilizan determinadas informaciones de carácter biológico para mejorar y validar la eficiencia del mismo en el problema particular que se ataca

- En primer lugar, y producto de la limitación de la disponibilidad para el entrenamiento, de secuencias validadas experimentalmente respecto al *ORF* -especialmente del hongo- se hizo necesario, y se logró, definir una metodología de "extensión" de las bases de datos disponibles internacionalmente, con información que permitiera inferir con gran certeza el *ORF* activo sin previa validación experimental. Esta metodología de extensión puede ser útil para los usuarios que enfrenten problemas similares
- En segundo lugar, se muestra como el algoritmo puede ser optimizado en general, desde el punto de vista de la cantidad de secuencias que en cada problema planta-hongo particular, pueden producir la mayor estabilidad de los parámetros -especialmente la distribución de probabilidades de transición y emisión- además de las posibles estimativas de las penalidades por codones de parada sorpresivos en determinados marcos de lectura.

- Finalmente, se muestran los resultados de la clasificación en el problema concreto del par de especies que se trata. Quedó demostrado que se obtienen resultados de buena clasificación, comparables y/o superiores a los reportados en clasificadores anteriores; pero además, quedó demostrado que tales resultados pueden obtenerse a partir de un número de secuencias de aprendizaje menor que los necesarios tradicionalmente, lo cual brinda la esperanza de ser aplicado en otros casos donde la información experimental disponible no es abundante.

Conclusiones

- Con un algoritmo basado en un Modelo Oculto de Markov fue posible obtener un poder de clasificación del 91,1% el cual es comparable a los obtenidos mediante PF-IND (81,2%) y ECLAT (93,1%).

- Incorporando nuevos atributos biológicos de las secuencias EST tales como la penalización de la ocurrencia de codones de parada, fue posible mejorar el poder de clasificación de USeparator.

- USeparator tiene menores requerimientos de información, referidos al tamaño del conjunto de entrenamiento, comparado con otros algoritmos reportados en la literatura como PF-IND y ECLAT.

Recomendaciones

- Extender el uso de USeparator a otros pares de Plantas-Hongos fitopatógenos como el caso de Musa spp.-Mycosphaerella fijiensis y determinar su poder de clasificación en estos casos.

- Usar las probabilidades obtenidas del Modelo Oculto de Markov, como una nueva característica de secuencias ESTs, en un modelo combinado del tipo *Support Vector Machine* para aumentar el poder de clasificación.

- Integrar el poder de detección de regiones codificantes al algoritmo basado en el uso de codones, en estrategias de detección de sitios de *splicing*.

Bibliografía

A. Krogh, M. B., I. S. Mian, K. Sjölander, and D. Haussler. (1994). "Hidden Markov models in computational biology: Applications to protein modeling." J. Mol. Biol.,.

Bairoch, A. (1993). "The PROSITE dictionary of sites and patterns in proteins, its current status." Nucl. Acids Res.

Barton, G. J. (1990). "Protein multiple sequence alignment and flexible pattern matching." Meth. Enzymol.

BBCNews (2003). "Bananas could split for good."

Blaine Simpon, F. T. (2005, 2005). "HSQLDB User Guide ", 2007, from http://hsqldb.org.

Booch, R., and Jacobson (1998). El Lenguaje Unificado de Modelado.

Carlini, D. B., Chen, Y., Stephan, W (2001). "The relationship between third-codon position nucleotide content, codon bias, mRNA secondary structure and gene expression in the drosophilid Alcohol Dehydrogenase genes adh and adhr." Genetics.

Caroline C. Friedel , K. H. V. J., Selina Sommer , Stephen Rudd , Hans W. Mewes , Igor V. Tetko (2005). Support vector machines for separation of mixed plant-pathogen EST collections based on codon usage. Bioinformatics.

Churchill., G. A. (1989). "Stochastic models for heterogeneous DNA sequences." Bull. Mathem. Biol.

FAO, O. d. l. N. U. p. l. A. y. l. A. (2003). "Bananas may be extinct within 10 years." Retrieved 2007, from http://www.fao.org.

Galpert D. , S. R., Ricardo G. (2005). "CLASIFICACIÓN DE LA ESTRUCTURA SECUNDARIA DE SECUENCIAS DE AMINOÁCIDOS UTILIZANDO LAS ALGEBRAS DE BOOLE DEL CÓDIGO GENÉTICO ".

Genomatix. (2005). "DNA Sequences formats." from http://www.genomatix.de/online_help/sequence_formats.html.

Henikoff, S. H. a. J. G. (1994). "Protein family classification based on searching a database of blocks." Genomics.

Hibernate (2007). Hibernate Reference Documentation.

Hoon, S. (2007). TribeMCL.

INIBAP, I. N. f. t. I. o. B. a. P. (2003). "Just How Far are Bananas from Extinction." 2008, from http://www.biotech-info.net.

J. V. White, C. M. S., and T. F. Smith. (1994). "Protein classification by stochastich modeling and optimal filtering of amino-acid sequences." Mathem. Biosci.,.

Japan, K. D. R. I. (2008). Codon usage database.

Jeppe Emmersen , S. R., Hans-Werner Mewes, Igor V. Tetko (2007). "Separation of sequences from host–pathogen interface using triplet nucleotid frecuencies." Fungal Genetics and Biology.

Joseph Bedell, I. K., Mark Yandell (2003). BLAST, O'Reilly.

Lieberman, G. J. (1997). Introducción a la investigación de operación.

M. Gribskov, A. D. M., and D. Eisenberg (1987). "Profile analysis: Detection of distantly related proteins."

Maor, R., Kosman, E., Golobinski, R., Goodwin, P., Sharon, A. (2003). "PF-IND: probability algorithm and software for separation of plant and fungal sequences." Curr Genet **43**: 296-302.

Marchal, K. (2004). "Introduction to Molecular Biology."

Nakamura, Y., Gojobori, T., Ikemura, T. (2000). "Codon usage tabulated from international DNA sequence databases: status for the year 2000." **28**.

National Library of Madicine, N. I. o. H. (2008). National Center for Biotecnology Information.

Pieraccini, E. L. a. R. (1993). "Planar hidden Markov modeling: From speech to optical character recognition. In S. J. Hanson, J. D. Cowan, and C. Lee Giles, editors." Advances in Neural Information Processing Systems **5**.

Pierre Baldi, S. B. (2001). Bioinformatics The Machine Learning Approach. London, England.

R. Durbin, S. E., A. Krgh, A. Mitchison (1998). Biological sequence analysis: Probablitistic models of proteins and nucleic acids, Cambrige University Tress.

Rabiner, l. R. (1989). A tutorial on Hidden Markov Models and selected application in speech recognition.
Rice., F. G. B. a. J. A. (1992). "Stochastic models for ion channels: Introduction and bibliography."
S. E. Levinson, L. R. R., and M. M. Sondhi. (1983). "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition."
Sánchez, M., Avilés, Sánchez, Grau (2006). "Building Blocks: A novel paradigm in protein structure. Evidences of their existence."
Sánchez, R. (2003). Estudio del orden en el Código Genético mediante la aplicación de métodos algebraicos y estadísticos. Facultad de Matemática Física y Computación. Santa Clara, Universidad Central ŦMarta Abreu╪de Las Villas. Master en Matemática Aplicada.
Schreiberm. (2005). "BioJava Core API." Retrieved 20 de Enero, 2006, from www.biojava.docs.biojava score.
Villamil, L. A. P. (2004). Espacio Vectorial N-dimensional de las secuencias de ADN sobre el Campo de Galois del Código Genético. Santa Clara. Master.
Wikipedia. (2006). "cDNA library."
Wikipedia (2007). Codon usage bias.
Xiaoqiu Huang, A. M. (1999). CAP3: A DNA Sequence Assembly Program.