UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Modelo para la clasificación de secuencias, en problemas de la bioinformática, usando técnicas de inteligencia artificial

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Autor: MSc. Isis Bonet Cruz

Santa Clara

2008

UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Modelo para la clasificación de secuencias, en problemas de la bioinformática, usando técnicas de inteligencia artificial

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Autor: MSc. Isis Bonet Cruz

Tutores: Dr. Ricardo Grau Ábalo

Dra. María M. García Lorenzo

Santa Clara

2008

SÍNTESIS

La motivación del trabajo es predecir la resistencia del VIH ante siete inhibidores de la proteasa, a partir de bases de casos de genotipo y fenotipo, mediante el desarrollo y aplicación de nuevas técnicas de inteligencia artificial. Se obtiene un modelo integrador que facilita la solución de problemas de clasificación de secuencias dentro de la bioinformática, mediante la combinación de métodos y la selección de rasgos basada en información biológica, para potenciar la eficacia de los pronósticos.

Se proponen dos métodos de clasificación que parten del uso de las energías de contacto asociadas a cada aminoácido y consideran como clase la resistencia o susceptibilidad ante fármacos. El primero está basado en redes neuronales recurrentes bidireccionales con un módulo que combina las salidas de los diferentes tiempos en la red. El segundo combina varios modelos de clasificación a partir de un metaclasificador. La red recurrente tiene la ventaja de poder trabajar con secuencias de longitud variable, con inserciones o eliminaciones. El modelo multiclasificador es útil cuando se está en presencia de secuencias alineadas. Además se proponen dos medidas de distancias basadas en información biológica: la primera para comprobar relación entre posiciones en las secuencias, la segunda para analizar visualmente los datos.

ABSTRACT

The motivation of this thesis is to predict the HIV drugs resistance from the available information of the virus genotype and phenotype by developing new Artificial Intelligence techniques. An integrated model with methods combination and features selection based on biological information was obtained improving the efficacy in Bioinformatics sequences classification.

Two classification methods are proposed: the bidirectional recurrent neural networks with an aggregation module to combine the outputs per times into a single classification, and the combination of diverse classification models with a meta-classifier. Both methods take the contact energies of the amino acids as features and the drugs resistance or susceptibility as objective feature. The recurrent neural network is many helpful in sequences with dynamic size caused by deletion or insertion mutations. The multi-classifier model is useful in aligned sequences. Two distance measures with biological meaning are introduced as well: the first one to support position relations, and the second one to visually analyze the data.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los retrovirus están asociados a un amplio rango de enfermedades que incluyen tumores, síndromes de inmunodeficiencia y desórdenes neurológicos. Ellos afectan un gran número de especies. Uno de los que más afectan a la humanidad es el virus de inmunodeficiencia humana (VIH). Cada año este virus causa más de 3 millones de muertes a pesar de los avances en el desarrollo de terapias para combatirlo [1].

La búsqueda de nuevos fármacos efectivos para el VIH es un gran reto para los científicos. Para poder diseñar un fármaco efectivo contra un virus, el cual manifiesta alta capacidad de mutación como el VIH, es necesario conocer la estructura tridimensional de todas las variantes mutantes de la proteína que se pretende atacar. El inmenso costo de la obtención de esta estructura hace que se cuente con un porcentaje muy pequeño de secuencias con esta información. Además de estos impedimentos, el nuevo fármaco, después de ser diseñado y sintetizado, requiere de un largo y costoso proceso de evaluación con pruebas toxicológicas y biomédicas antes de ser aprobado y salir al mercado. Por tales razones, se hace necesario continuar el estudio sobre cómo usar adecuadamente los fármacos ya existentes. Es por esto que este trabajo se centra en el problema de predecir la resistencia de una cepa del VIH ante los fármacos existentes, para encontrar el más apropiado a utilizar.

Actualmente existe un gran número de fármacos antirretrovirales aprobados para el tratamiento del VIH, la mayoría de ellos diseñados para inhibir dos importantes proteínas (enzimas): la proteasa y la transcriptasa inversa. El fenómeno de resistencia a estos fármacos está asociado a la capacidad de mutación, ya comentada, de dichas proteínas. Las variantes mutantes de estas enzimas contienen cambios de aminoácidos que alteran la estructura tridimensional de las mismas. Como consecuencia de estos cambios los fármacos no tienen acceso a los centros activos de las enzimas y no pueden inhibir sus funciones biológicas. La resistencia a fármacos no es un problema nuevo; otros virus, como herpes simples, varicela, citomegalovirus e influenza A, han reportado también alta resistencia. Sin embargo en el caso del VIH es más importante debido al dramático final de los pacientes infectados.

Existen dos formas experimentales de estimar la susceptibilidad de una cepa del VIH ante un fármaco, una basada en el fenotipo y otra basada en genotipo. Las pruebas basadas en el fenotipo dan una cuantificación de la sensibilidad al fármaco, generalmente expresado como la concentración requerida del fármaco para inhibir el virus. Las pruebas basadas en el genotipo están sustentadas en el análisis de las mutaciones asociadas con la resistencia, o sea, la secuencia del ADN que codifica para la enzima analizada es alineada con la secuencia correspondiente de una cepa viral tomada como referencia y se calcula la lista de posiciones mutadas.

Cuando se está en presencia de un nuevo mutante es necesario medir la resistencia que tendrá éste frente a cada fármaco para realizar una selección apropiada. El cálculo de las pruebas basadas en fenotipo sería ideal debido a que ofrecen, precisamente, el valor de concentración del fármaco que se necesita para inhibir el virus y, una vez conocido este valor, se analiza si es posible aplicar esta concentración a un humano sin que peligre su vida. Sin embargo, las pruebas de fenotipo son muy costosas y se requiere de mucho tiempo para obtener los resultados, a diferencia de las pruebas de genotipo que proporcionan los resultados en muy pocos días y son menos costosas. De hecho en los últimos tiempos el costo de la secuenciación genética viral ha bajado rápidamente, haciendo la selección de fármacos basada en las pruebas de genotipo una opción más atractiva que las medidas de fenotipo [2, 3]. Pero la interpretación de la resistencia partiendo de la información del genotipo es muy difícil, hoy en día todavía es un reto y a menudo requiere análisis de expertos. Esto ha llevado a tratar de relacionar la información de las mutaciones con la resistencia, de forma que sirva de punto de referencia para el análisis en una nueva secuencia. La unión de estas pruebas de resistencia ha arrojado un gran cúmulo de información sobre este virus, y algunas de ellas se encuentran disponibles en bases de datos internacionales como "Los Álamos" [4] y "*Stanford HIV Resistance Database*" [5].

Sin embargo, el problema de relacionar el resultado de ensayos genotípicos y fenotípicos ha conducido a otros retos pues el fenotipo debe ser considerado como una consecuencia de un gran número de posibles mutaciones. Lo difícil de analizar la resistencia a los fármacos, a partir de la secuencia genómica del VIH, ha generado la aplicación de una

gran variedad de métodos estadísticos y de inteligencia artificial para predecir la resistencia del fenotipo a partir de la información del genotipo.

Entre las técnicas utilizadas se encuentra el análisis discriminante lineal, el cual es usado para determinar cuáles mutaciones predicen mejor la sensibilidad a dos de los inhibidores de la proteasa: IDV y SQV [6]. En [7] se usaron las redes neuronales para predecir resistencia a Lopinavir, otro de los inhibidores de la proteasa, partiendo de la información del genotipo. Se desarrollaron dos modelos de redes neuronales basados en posiciones de aminoácidos de la proteasa que resultaron de un análisis de preponderancia de categoría. El primero parte de la información de mutaciones de 11 posiciones mientras que el segundo parte de 28. Draghici y Potter también usan redes neuronales, pero para predecir la resistencia a otros dos inhibidores de la proteasa: *Indinavir* y *Saquinavir* [8].

Se han usado también máquinas de soporte vectorial (*Support Vector Machine*, SVM) con función núcleo lineal [9], así como árboles de decisión generados por medio de divisiones recursivas [10]. Murray utiliza otra variante de árbol de decisión y el método *k*-vecinos más cercanos para predecir la resistencia ante fármacos de la proteasa y la transcriptasa inversa [11]. Varios análisis de las mutaciones que establecen resistencia han hecho posible también el éxito de sistemas basados en reglas para este problema [5, 12].

Algunos autores han hecho una comparación de las diferentes técnicas aplicadas [13-15]. Rabinowitz et al. [14] tratan de explicar la superioridad de métodos que generan funciones de costo convexo y dispersan los rasgos, como SVM y LASSO (*least absolute shrinkage and selection operator*). Rhee et al. [15] comparan varias técnicas de clasificación y de regresión para este problema, obteniendo los mejores resultados cuando se realiza una selección de rasgos previa.

Estos modelos computacionales usados para predecir la resistencia de fenotipo desde la información del genotipo, han ofrecido buenos resultados para algunos fármacos, pero para otros, el porcentaje de predicción ha sido bajo. Visto desde la óptica de predecir el fenotipo a partir de la información del genotipo, la predicción de resistencia de virus es un problema de clasificación de secuencias, que se encuentra dentro de uno de los campos de la bioinformática: la farmacoterapia. La bioinformática tiene otros muchos campos: genómica, proteómica y microarreglos, en los que se pueden encontrar muchos

problemas de clasificación de secuencias. Todos tienen en común que se parte de una secuencia de ADN (ácido desoxirribonucleico), ARN (ácido ribonucleico) o proteína para predecir una de sus características o función.

En estos problemas de análisis de secuencias una de las mayores dificultades es la diversidad de tamaño en las secuencias que se analizan, debido a las mutaciones que causan inserciones o eliminaciones de bases del ADN. Otras de las grandes dificultades son la dimensionalidad de los datos y la complejidad de la característica a predecir sólo a partir de la información de la secuencia.

SITUACION PROBLÉMICA

Las secuencias biológicas en general pueden analizarse a partir de la estructura tridimensional o de la estructura primaria de las mismas. Como se ha dicho anteriormente, la obtención de la estructura tridimensional es muy costosa y en la mayoría de estos problemas no se cuenta con gran información de ésta. Por eso, generalmente se usa la estructura primaria como información para describir las secuencias. Las secuencias de ADN codificadas para proteínas pueden ser representadas por los símbolos utilizados para las bases de nucleótidos o por los símbolos de una letra de los aminoácidos codificados. Las secuencias basadas en nucleótidos presentan más información que las basadas en aminoácidos, ya que tenemos 64 codones (cada codón está compuesto por 3 nucleótidos, ver Anexo 2) que codifican para 20 aminoácidos (Ver Anexo 1). No siempre se cuenta con la información de los nucleótidos, por ejemplo la base de Los Álamos representa las secuencias usando los nucleótidos y la de Stanford usando los aminoácidos, pero esta última presenta más casos que la primera. Por esta razón, se seleccionó la base de Stanford como fuente de información para resolver el problema planteado. Esta base cuenta con datos sobre varias de las proteínas del VIH y específicamente se seleccionó la proteasa como la proteína a analizar, la cual tiene información, de la relación entre el genotipo y el fenotipo, de siete fármacos aprobados.

La búsqueda de una solución del problema a partir de la secuencia de aminoácidos no es simple. En el caso de esta proteína se está en presencia de una secuencia de 99 aminoácidos (posiciones), donde cada aminoácido puede ser cambiado por los 19 restantes. Por tanto, teóricamente podrían tenerse hasta 20^{99} posibles secuencias, si se tienen en cuenta sólo las mutaciones por sustitución (realmente el número es bastante

menor porque todas estas mutaciones no son biológicamente factibles, pero de cualquier manera el número es bastante grande). Realmente en la proteasa no son comunes las inserciones y eliminaciones, aunque ocasionalmente ocurren [16]. Todo ello hace necesario, como un primer paso en la solución de este problema de clasificación, buscar una representación adecuada de los datos, que tenga relación con la resistencia o susceptibilidad a un fármaco dado.

Algunos autores han usado varias representaciones para la secuencia como: mutaciones más frecuentes [14, 17], vectores de 20 elementos (binarios) que representan los aminoácidos, perfiles de información mutua [18]. Otros utilizaron información del alineamiento de las secuencias [19], por ejemplo los resultados obtenidos por la familia de las matrices PAM [20] y BLOSUM [21]. Todas estas variantes tratan, de alguna forma, de describir la secuencia en cuanto a sus mutaciones.

En este trabajo se intenta considerar descriptores que tengan relación más fuerte con la estructura espacial de las secuencias, pues a partir de ésta es que se puede conocer, en la práctica, cómo debe ser el fármaco y si inhibirá a la mutación o no. Miyasawa y Jernigan mostraron que la variación en las energías de contacto, en determinadas posiciones, modifica la estructura de la proteína [22]. El reemplazo de un simple aminoácido es suficiente para que cambien los valores observados de las variaciones de energías libres entre el estado completamente desplegado y el estado plegado más estable [23]. Las energías de interacción de los aminoácidos se seleccionaron para representar las secuencias del VIH, pues ellas tienen relación directa con la estructura tridimensional de la secuencia, ya que la atracción que ejerce cada uno de los aminoácidos sobre el resto influye en el plegamiento de la enzima en el espacio.

Después de seleccionada la representación de las secuencias, será necesario seleccionar un modelo de clasificación que, a partir de esta información del genotipo, logre predecir la resistencia a un fármaco determinado. Son muchos los trabajos que se han desarrollado para tratar de solucionar este problema, y aquellos donde se ha realizado una selección previa de rasgos, o el método de clasificación la realiza internamente, son los que han obtenido mejores resultados. Estos métodos sólo trabajan para secuencias con mutaciones en las posiciones seleccionadas, lo cual restringe nuestro problema. Una de las formas de reducir la dimensionalidad es la conocida como técnica de extracción de rasgos, que

puede realizarse previamente o como un proceso interno en algunos de los métodos de clasificación, como las redes neuronales.

Para la búsqueda de una técnica general se debe tener en cuenta la variabilidad del tamaño de las secuencias. Pero, por las características de la proteasa se puede buscar una técnica que trabaje con secuencias fijas, analizando sólo secuencias que presenten sustituciones y no inserciones o eliminaciones.

Entre las técnicas computacionales que típicamente trabajan con secuencias de tamaño variable tenemos las redes neuronales recurrentes, sólo que, hasta ahora, éstas no han sido usadas para clasificación. Existen varias técnicas de inteligencia artificial para problemas supervisados; dentro de las últimas usadas con buenos resultados, están aquellas que combinan varios clasificadores, que son llamadas multiclasificadores. En este trabajo se pretende hacer una comparación de varias técnicas supervisadas, para buscar aquella que produzca mejores resultados en el problema planteado.

Se resume el planteamiento del problema a través de las siguientes:

PREGUNTAS DE INVESTIGACIÓN

- ¿Serán las energías de contacto una buena representación para las secuencias de proteínas virales, cuando se trata de establecer relación entre genotipo y fenotipo?
- ¿Podrán definirse funciones que midan la semejanza entre secuencias proteicas, a partir de la información de las energías de contacto?
- ¿Podrán ser extendidas las redes recurrentes para problemas de clasificación de secuencias?
- ¿Podrá la combinación de diferentes paradigmas de modelos de clasificación, dar mejores resultados en la predicción de resistencia viral del VIH?

Se formula así el siguiente:

OBJETIVO GENERAL

Definir un modelo integrador que facilite la solución de problemas de clasificación de secuencias biológicas, mediante la combinación de métodos y la selección de rasgos basados en información biológica, para potenciar la eficiencia de los pronósticos y, en particular, predecir la resistencia o susceptibilidad de nuevas mutaciones de la proteasa del VIH ante un conjunto de fármacos.

Este objetivo general se desglosa en los siguientes:

OBJETIVOS ESPECÍFICOS

- Definir técnicas de aprendizaje computarizado que permitan seleccionar los rasgos más adecuados para representar los problemas de clasificación de secuencias de proteínas así como analizar los datos a partir de funciones de distancia que tengan en cuenta información biológica.
- Seleccionar entre las técnicas de inteligencia artificial, las que más se adaptan a los problemas de clasificación de secuencias en la bioinformática. Elaborar nuevas técnicas con este fin, en particular:
    - Concebir un modelo basado en redes neuronales recurrentes para problemas de clasificación de secuencias, que tenga en cuenta información biológica, e implementarlo para la solución de problemas de clasificación de secuencias.
    - Concebir un modelo para solucionar problemas de clasificación de secuencias, combinando varios modelos de clasificación e implementarlo combinando ventajosamente varios de ellos.
- Evaluar los dos nuevos modelos concebidos, a través de la solución del problema de la predicción de resistencia de proteínas del VIH ante drogas específicas.

Es deseable que los modelos sean implementados como extensiones a la plataforma inteligente para aprendizaje WEKA (*Waikato Environment for Knowledge Analysis*), que es libre y de código abierto y por tanto permitirá el uso de estas herramientas por la comunidad científica internacional, pero además porque tiene incorporadas muchas técnicas estadísticas o de inteligencia artificial y brinda la posibilidad de experimentar con el conjunto de ellas para investigar con cuáles se obtienen mejores resultados.

Para alcanzar estos objetivos se trazan las siguientes:

TAREAS DE INVESTIGACION

- Ilustrar cómo se pueden usar las energías de contacto de los aminoácidos en la representación de las secuencias para su análisis computacional.
- Definir una medida de semejanza entre las secuencias virales que tenga en cuenta las energías de contacto y compararla con otras.
- Definir una medida de semejanza entre los rasgos para probar que la extracción de rasgos es una técnica útil en este problema.

- Determinar las técnicas de inteligencia artificial más apropiadas para problemas de clasificación de secuencias en la bioinformática.
- Diseñar e implementar un modelo basado en redes neuronales recurrentes adaptado para clasificación y otro que combine ventajosamente varios clasificadores.
- Analizar los resultados de la aplicación del modelo al problema de la predicción de resistencia de la proteasa del VIH ante varios fármacos específicos.

RELEVANCIA DE LA INVESTIGACION

Valor Teórico

Desde el punto de vista teórico se proponen las energías de contacto de los aminoácidos como atributos descriptivos, que representen apropiadamente a los aminoácidos en problemas de clasificación de secuencias. Se proponen dos medidas de distancias basadas en las energías de contacto para analizar casos y reducir rasgos.

Se proponen además dos modelos de clasificación: uno para problemas de clasificación de secuencias de longitud variable y otro para secuencias de longitud fija. El primero es una extensión de las redes recurrentes y el segundo es un método multiclasificador que combina varios paradigmas de clasificación, basándose en el conjunto de instancias que cada uno aprende.

Valor Práctico

El valor práctico está dado por:

- Un software para el diseño de redes neuronales recurrentes.
- Extensión de WEKA con un nuevo modelo de clasificación basado en redes neuronales recurrentes.
- Extensión de WEKA con nuevas variantes de multiclasificadores.
- Contribución a la solución del problema de predicción de resistencia de proteínas del VIH ante un fármaco específico.

Encontrar cuál o cuáles fármacos pueden inhibir la cepa del virus del VIH que porta un paciente, es realmente muy importante desde el punto de vista económico y social.

Después de haber elaborado el marco teórico se plantearon las siguientes:

HIPÓTESIS DE INVESTIGACIÓN

1. Las energías de contacto de los aminoácidos pueden ser una buena representación para la proteasa en aras de predecir la resistencia de la misma ante determinados fármacos.
2. Las energías de contacto pueden ser usadas para establecer medidas de semejanza entre las secuencias así como en los procesos de extracción de rasgos con un incremento del porcentaje de buena clasificación en la predicción de resistencia de las secuencias del VIH
3. La extensión de las redes recurrentes bidireccionales a problemas de clasificación permite, en particular, predecir la resistencia de secuencias del VIH ante determinados fármacos.
4. La combinación de varios paradigmas de clasificación puede mejorar los resultados de predicción de resistencia en secuencias mutantes del VIH.

NOVEDAD CIENTÍFICA

Se defienden los tres aspectos siguientes:

- La representación de las secuencias de proteínas a partir de la determinación de rasgos apropiados usando técnicas computacionales.
    - Uso de energías de contacto para representar los aminoácidos.
    - Medida de distancia entre los rasgos, basada en información biológica, para la extracción de rasgos.
    - Medida que establece semejanza entre las secuencias permitiendo un mejor análisis visual de las mismas.
- El modelo basado en redes neuronales recurrentes para problemas de clasificación de secuencias.
- Un modelo que combina ventajosamente varios modelos de clasificadores.

La tesis se estructura en la presente introducción, cuatro capítulos, conclusiones y recomendaciones.

En el Capítulo I se hace un estudio de los problemas de clasificación más conocidos que ha planteado la bioinformática. Además se analizan algunas de las técnicas de inteligencia artificial más usadas, en sus diferentes fases durante la solución de problemas de clasificación: desde el preprocesamiento y análisis de los datos hasta la evaluación de

los resultados. Se hace un estudio del uso de dichas técnicas en los problemas de bioinformática.

El Capítulo II está dedicado al análisis de los datos a partir de métodos de extracción de rasgos usando dos medidas de distancias propuestas, una basada en los rasgos y otra basada en los casos. Además se describe el modelo extendido de redes neuronales recurrentes para problemas de clasificación y se hace una validación de éste usando bases de datos artificiales.

En el Capítulo III se describe un modelo de combinación de varios clasificadores y se realiza una validación del mismo con bases de datos internacionales.

En el Capítulo IV se diseña y se evalúa el problema de predicción de resistencia del VIH ante fármacos antivirales usando los métodos de análisis de datos propuestos, así como su solución con el uso de los modelos de redes recurrentes y combinación de clasificadores.

Finalmente se lista la bibliografía utilizada y se incluyen algunos anexos con información que sólo es necesaria para análisis más profundo de los resultados. El último de ellos es un glosario de los términos usados en este trabajo (Anexo 17).

Capítulo I. LAS TÉCNICAS DE INTELIGENCIA ARTIFICIAL EN LOS PROBLEMAS DE BIOINFORMÁTICA

En este capítulo se muestra cómo la bioinformática posee gran cantidad de problemas de clasificación, no todos cabalmente resueltos. Se hace un estudio de algunos de los problemas más conocidos, sobre todo en el campo de análisis de secuencias. Seguidamente se analizan las técnicas necesarias para enfrentar estos problemas, desde el preprocesamiento y análisis, hasta la evaluación del modelo de clasificación usado. Se describen algunos de los métodos de clasificación más usados. Además se realiza un estudio del uso de estas técnicas en el campo de la bioinformática.

I.1. Bioinformática y los problemas de aprendizaje supervisado para análisis de secuencias

El advenimiento de nuevas y eficientes técnicas experimentales han revolucionado las ciencias de la vida. Principalmente la secuenciación del ADN, produjo un crecimiento exponencial de las descripciones lineales de proteínas y moléculas de ADN y ARN. Este incremento de la cantidad de datos biológicos ha generado dos problemas: el almacenamiento y manejo eficiente de la información y la extracción de información útil [24]. Son componentes esenciales del proceso de investigación las herramientas computacionales para: clasificación de secuencias, detección de semejanzas, separación de las regiones codificantes de las no codificantes en secuencias de ADN, predicción de estructura molecular y la reconstrucción de la historia evolutiva. Esto, al igual que el descubrimiento de nuevos fármacos y terapias, es fundamental para el entendimiento de la vida y la evolución [25]. Se abre paso a un nuevo campo de investigación, la *bioinformática* que une la biología, la matemática y la ciencia de la computación, con el objetivo de investigar la esencia de la vida.

Para la definición de bioinformática no existe un consenso general en la literatura, si se realizara una búsqueda, se podría ver que los diversos glosarios y diccionarios la definen de manera diferente, de hecho existe cierta discrepancia entre la diferencia entre bioinformática y biología computacional. Sin querer filosofar en el tema, en este trabajo se definirá bioinformática como un área interdisciplinaria, en la que se mezclan las

ciencias exactas y de la vida con las ciencias de la información, para estudiar y desarrollar herramientas útiles que ayuden a comprender el flujo de información desde los genes a las estructuras moleculares, a sus funciones bioquímicas, a su comportamiento biológico, y finalmente, a su influencia sobre las enfermedades y la salud, para en definitiva, mejorar la vida, e incluso, crearla.

El surgimiento de esta área del conocimiento y la primera publicación de una secuencia genómica, provocó una gran avalancha de datos y la creación de bases de datos biológicas, dentro de las más importantes internacionalmente están: *Genebank*[1], EMBL[2], PIR-*International Protein Sequence Database* (PSD)[3], *SwissProt*[4] y DNA *DataBank of Japan* (DDBJ)[5].

Estas grandes bases de datos crean dos desafíos: problemas de minería de datos y el uso de ellas para la solución de problemas biológicos. En este sentido, constituye un reto importante el desarrollo de nuevos algoritmos para el tratamiento innovador de problemas de análisis de secuencias. Los algoritmos de aprendizaje automático son ideales para dominios caracterizados por la presencia de gran cantidad de datos, patrones ruidosos y la ausencia de teorías generales. La idea fundamental de estos algoritmos es aprender automáticamente la teoría de los datos, a través de un proceso de inferencia, modelación o aprendizaje de ejemplos.

Un rasgo fundamental de las cadenas de moléculas, las cuales son responsables del funcionamiento y evolución de los organismos vivos, es que ellas pueden ser emitidas en secuencias de símbolos digitales. Los nucleótidos y monómeros de aminoácidos en ADN, ARN y proteínas son distintos, de forma que las cadenas que los constituyen pueden ser representados por un conjunto de símbolos desde un pequeño alfabeto (Ver Anexo 2). El análisis de estas secuencias de ADN, ARN y proteínas es uno de los campos más amplios en la bioinformática y puede verse desde diferentes puntos dentro de la ciencia de la computación.

---

[1] www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html

[2] www. ebi.ac.uk/embl/index.html

[3] http://pir.georgetown.edu/

[4] http://www.expasy.ch/sprot/

[5] www.nig.ac.jp/home.html

Los problemas pueden dividirse, teniendo en cuenta si tenemos conocimiento de la función o hipótesis que se desea predecir, en problemas supervisados y no supervisados. Los problemas supervisados son aquellos donde se tiene información de la hipótesis y los no supervisados donde no se tiene información de la misma. Cuando la función o hipótesis a predecir es discreta, los algoritmos relacionados con los problemas supervisados son conocidos como clasificadores, y los que tienen que ver con los problemas no supervisados se conocen como métodos de agrupamiento. Este trabajo se va a centrar fundamentalmente en problemas de clasificación de análisis de secuencias, aunque requeriremos en determinado momento de técnicas de análisis no supervisado para el análisis de los datos.

En un problema de clasificación tenemos un conjunto de elementos divididos en clases. Dado un elemento del conjunto se le asigna una clase de acuerdo a los rasgos o características que lo describen y a un conjunto de reglas de clasificación. La mayoría de las veces este conjunto de reglas no es conocido y la única información conocida es el conjunto de ejemplos etiquetados, de forma tal que las etiquetas representan las clases. Los algoritmos de aprendizaje supervisado inducen las reglas de clasificación a partir de los datos. En cualquiera de los dominios de la bioinformática se puede encontrar problemas de clasificación. Los ejemplos siguientes son seleccionados entre una amplia gama, para destacar importancia tanto teórica como práctica.

La localización de genes en un genoma completo, o en una larga secuencia genómica, ha sido considerado durante varios años como el problema principal de la bioinformática. Contribuye de manera importante a su solución, la identificación de sitios de *splicing*, que separan zonas codificantes y no codificantes. Éste es un buen ejemplo de un problema abierto en bioinformática que no será tratado aquí; pero se puede ver un estudio bastante profundo en otros trabajos como los de Saeys [26] y Grau et al. [27].

Un segundo ejemplo importante puede ser el siguiente. Así como los genes son los que contienen la información, las proteínas son las que transforman esta información hacia la vida. Ellas juegan un papel muy importante en el proceso de la vida y su estructura tridimensional es la clave de su funcionalidad, por lo que ésta también es una tarea muy importante dentro del dominio de la proteómica. Por la complejidad de la estructura 3D, se ha definido una estructura conocida como estructura secundaria que es un proceso

"intermedio" en el conocimiento estructural de las proteínas [28, 29]. Este problema puede ser formulado o no como clasificación.

Otro de los dominios muy tratados en bioinformática son los microarreglos (*micro-arrays*) o expresión de genes, entre cuyas aplicaciones se destaca por ejemplo, el problema de detección de tumores con el uso de expresión de genes. En el dominio de los fármacos, un problema muy popular es la predicción de la resistencia a fármacos de secuencias de proteínas virales. Estos y otros muchos problemas de clasificación de secuencias biológicas pueden encontrarse en la bioinformática.

La clasificación puede dividirse en tres procesos fundamentales: preprocesamiento de los datos, selección del modelo de clasificación y, entrenamiento y prueba del clasificador, procesos en los que profundizaremos en los epígrafes que siguen.

I.2. Técnicas de selección y extracción de rasgos en bioinformática

Cada base de datos está compuesta por un conjunto de objetos o casos representativos de un problema en cuestión y a su vez, cada objeto, se describe mediante un conjunto de atributos o rasgos (también conocidos como variables de entrada o características), los cuales pueden ser cualitativos (categóricos) o cuantitativos (numéricos).

Pero lo más importante antes de comenzar a tratar un problema es la representación de los datos. Para esto, en el caso de clasificación, se deben definir los rasgos y las clases que se tienen. La definición de los rasgos no es elemental, un mismo objeto puede ser descrito con rasgos diferentes para distintos problemas. Las secuencias biológicas típicamente son definidas usando propiedades biológicas de las mismas como hidrofobicidad, polaridad y otras. También pueden ser representadas usando su representación primaria, o sea, en el caso de secuencias de aminoácidos cada aminoácido es un rasgo y en el caso de secuencias nucleotídicas cada nucleótido es un rasgo. Después de representados los datos entonces se pueden usar técnicas de preprocesamiento de los mismos.

El preprocesamiento de los datos es muy amplio, pues abarca desde técnicas de estandarización, normalización, posible discretización de los datos, hasta técnicas que permiten seleccionar o escoger cuáles serán las características finales que representarán

los objetos del problema, conocidas estas últimas como técnicas de selección y extracción de rasgos.

En ocasiones algunos rasgos pueden crear ruido en los datos, tanto aquellos que son redundantes como los que son irrelevantes, confundiendo al modelo de clasificación y degradando su rendimiento. La eliminación de estos rasgos resulta en un conjunto restringido de rasgos con igual o mejor rendimiento de clasificación que el conjunto completo. Otras veces la gran cantidad de rasgos genera un proceso de entrenamiento lento y la reducción de los rasgos que describen los objetos facilita una rápida clasificación y una mejor comprensión de los datos. De ahí la funcionalidad de procurar la representación idónea de los casos que conforman el conjunto de entrenamiento del problema de clasificación.

Existen dos técnicas para reducir la dimensionalidad: selección y extracción de rasgos. La primera se basa en seleccionar los rasgos más revelantes para el problema. La segunda construye un número reducido de nuevos rasgos a partir de transformaciones de los iniciales.

I.2.1. Técnicas de selección de rasgos

Dada una base de datos donde los objetos son descritos por N atributos, la selección de rasgos es la búsqueda de un subconjunto de M atributos, donde $M< N$. En el contexto de clasificación, las técnicas de selección de rasgos pueden ser organizadas en tres categorías, dependiendo de cómo ellas combinen la búsqueda del subconjunto de rasgos con la construcción del modelo de clasificación: métodos de filtrado, envoltura y sistemas integrados [30].

Las técnicas de filtrado (*filter*) calculan la relevancia de los rasgos sólo teniendo en cuenta las propiedades de los datos. En la mayoría de los casos se calcula el peso o relevancia para los rasgos, y los rasgos con más bajo peso son eliminados. Además, el subconjunto de rasgos filtrado es el que se presenta como entrada al algoritmo de clasificación. Estas técnicas reducen dimensionalidad fácilmente, son simples y rápidas computacionalmente, y son independientes del algoritmo de clasificación. La selección de rasgos necesita ser ejecutada una sola vez, y como resultado se obtiene una nueva base de datos, con menos rasgos. Una desventaja común es que ignoran la interacción con el

clasificador (la búsqueda en el espacio del subconjunto de rasgos está separada de la búsqueda en el espacio de la hipótesis).

Las técnicas de envoltura (*wrapper*) generan varios subconjuntos de rasgos y se evalúan mediante el entrenamiento y prueba del modelo de clasificación. Como consecuencia la selección de rasgos está directamente relacionada con un modelo de clasificación específico. Sin embargo, como el espacio de subconjuntos de rasgos crece exponencialmente con el número de rasgos, se usan métodos heurísticos para guiar la búsqueda por un subconjunto óptimo. Estos métodos de búsqueda pueden ser divididos en dos clases: determinísticos y aleatorios. Las ventajas de estas técnicas incluyen la interacción entre la búsqueda de un subconjunto de rasgos y la selección del modelo, y la habilidad de tener en cuenta las dependencias entre rasgos. Sus desventajas son: el alto riesgo de sobreentrenamiento (*overfitting*) y la complejidad computacional, especialmente si construir el clasificador tiene un alto costo computacional.

Por último analizaremos los sistemas integrados (*embedded*), donde la selección de rasgos es parte del proceso de modelación, o usa parámetros del algoritmo de clasificación para seleccionar un conjunto de rasgos relevantes. Ejemplos de este tipo de selección de rasgos son los algoritmos de árboles de decisión (ID3, C4.5, CHAID, etc.), donde la selección de rasgos se realiza implícitamente durante la construcción del árbol. Tanto el mecanismo de parada, como la poda del árbol son dos formas de selección de rasgos.

En general las técnicas de selección de rasgos ayudan, no sólo a reducir el costo de los modelos dado por la reducción de los rasgos, sino que en algunos casos puede mejorar el rendimiento, o sea, la predicción en caso de métodos supervisados y la detección de grupos en casos no supervisados. No siempre reducir rasgos es una ventaja para el problema, porque en realidad se elimina información, que no necesariamente tiene que ser irrelevante. Esta es una gran desventaja de estos métodos, cuando las bases de datos no tienen los suficientes casos para ejemplificar la influencia de todos los rasgos, la aplicación de estos métodos puede eliminar rasgos importantes.

En el caso de problemas de análisis de secuencias, si se parte de secuencias primarias, la eliminación de rasgos significa eliminar posiciones de la secuencias, o sea, ignorar la

influencia de aminoácidos o nucleótidos. Por ello, en dependencia del problema, hay que tener mucho cuidado al aplicar estas técnicas.

I.2.2. Técnicas de extracción de rasgos

A diferencia de las técnicas de selección de rasgos, que conservan la naturaleza de los datos, pues se basan en la selección de un subconjunto de rasgos, las técnicas de extracción se basan en la proyección o combinación de los rasgos originales para crear nuevos rasgos. Aunque no siempre estos nuevos atributos que describen los objetos tienen un significado claro, ayudan a reducir la dimensionalidad, y además, muchas veces se pueden obtener representaciones en dos o tres dimensiones, que posibilitan la visualización de los datos y con esto facilitan un mejor análisis de los mismos.

La mayoría de las técnicas de extracción de rasgos se basan en una medida de distancia entre los objetos, tratando de encontrar una transformación que tenga en cuenta la semejanza de los mismos. Según Yang [31] estas técnicas se dividen en dos: extracción de rasgos supervisada y extracción no supervisada; esta caracterización se basa en si tienen o no en cuenta el valor de la clase asociada a cada objeto, en la definición de la distancia o, en general, en el algoritmo de extracción.

Entre las técnicas no supervisadas más conocidas tenemos el análisis de componentes principales (*Principal Component Analysis*, PCA), la cual se basa en encontrar el subespacio que preserve mejor la varianza de los datos [32]. Otra de las más usadas es el escalado multidimensional (*multidimensional scaling*, MDS), la cual encuentra la proyección de bajo rango que preserva mejor la distancia entre puntos, dada por la matriz de distancias [33]. Por otro lado entre las supervisadas podemos encontrar el análisis discriminante lineal, también llamado discriminante lineal de Fisher, que está dado por maximizar el criterio de Fisher de una matriz de transformación lineal [34].

Pero se han desarrollado varios trabajos que hacen extensiones a éstas, combinan algunas o usan otros métodos para transformar los datos. La red neuronal de Kohonen, o también conocida como mapa autoorganizado (*self-organized map*, SOM), es otra de las técnicas más usadas [35]. SOM permite el surgimiento de rasgos estructurales intrínsecos del espacio de los datos, que reduce dimensionalidad en su procesamiento interno hacia un mapa de dos dimensiones, logrando un método conocido como ESOM (*Emergent SOM*)

[36]. También se han usado algoritmos de agrupamiento para reducción de dimensionalidad [37-39]. En general, existen diversas técnicas que permiten reducir la dimensionalidad [40].

Como se ha dicho anteriormente, estas técnicas tienen como desventaja que no conservan la naturaleza de los datos pues hacen una transformación de los mismos, convirtiendo los rasgos originales en otros que se obtienen por alguna combinación de los originales. Esta reducción de dimensionalidad al igual que la selección de rasgos puede mejorar el costo de los métodos de clasificación o agrupamiento que se empleen o elevar el rendimiento de los mismos. Otra ventaja es que algunas de estas técnicas, como ya se explicó, posibilitan reducir la dimensionalidad a un plano bidimensional de forma que los datos puedan analizarse visualmente. Además, en el caso del análisis de secuencias biológicas o temporales, puede realizarse extracción de rasgos sin eliminar información, simplemente combinándola.

La elección entre técnicas de selección y extracción de rasgos depende del dominio de aplicación, del conjunto de datos específico, así como de las intenciones que se tengan con esta reducción de dimensionalidad.

I.2.3. Aplicaciones de selección y extracción de rasgos en bioinformática

Los problemas de la bioinformática se caracterizan por un gran número de rasgos, de ahí la necesidad del amplio uso de las técnicas de reducción de dimensionalidad [30, 41]. Entre estos problemas se encuentran la predicción de propiedades o características de subsecuencias de secuencias codificantes para proteínas.

Debido a que muchos rasgos pueden extraerse de la secuencia y ocurren muchas dependencias entre las posiciones adyacentes, se han desarrollado modelos de Markov de diferentes órdenes. Para tratar con la gran dimensionalidad de los rasgos posibles y la limitada cantidad de ejemplos en ocasiones, Salzberg et al. [42] introducen un modelo de Markov "interpolado", el cual usa interpolación entre diferentes órdenes de los modelos de Markov para tratar con casos de menor dimensión, y un método de filtrado para seleccionar los rasgos relevantes. Por su parte Delcher et al. [43] cruzan un árbol de decisión bayesiano con un método de filtrado para seleccionar los rasgos relevantes.

Otra línea de investigación es la predicción de genes, donde los elementos estructurales tales como los sitios iniciales de traslación y los sitios de *splicing* son modelados como problemas de clasificación. El problema de selección de rasgos para el reconocimiento de elementos estructurales se inicia por Degroeve et al. [44] para el problema de predicción de sitios de *splicing*, combinando un método secuencial con búsqueda hacia atrás junto con un criterio de evaluación de una máquina de soporte vectorial integrada, para calcular la relevancia de los rasgos. En [45] se usa un algoritmo de estimación de la distribución (*Estimation of Distribution Algorithm*, EDA) para distinguir los rasgos relevantes en la predicción de sitios de *splicing*.

Similarmente en el problema de sitios de inicialización de traslación, Liu et al. demuestran las ventajas del uso de técnicas de selección de rasgos para este problema, usando la entropía entre rasgo y clase como medida de filtro para eliminar rasgos irrelevantes [46]. En el problema de microarreglos se usaron también técnicas de selección de rasgos [47]. Recientemente, Saeys et al. combinan diferentes medidas de predicción potencial de código y usan un algoritmo de filtrado [48].

En la predicción de funciones de proteínas a partir de secuencias se aplican varias técnicas de selección de rasgos, por ejemplo, [49] usa una técnica interesante a partir de funciones núcleo selectivas para máquinas de soporte vectorial como una vía de calcular pesos para los rasgos, y subsecuentemente eliminar los rasgos con pesos bajos. También se han usado en el reconocimiento de regiones de promotores [50] y en la predicción de microARN [51].

La tecnología de espectrometría de masas (*mass-spectrometry*) surge en el proceso de predicción de enfermedades. Los algoritmos de envoltura demostraron su utilidad en estos problemas, empleando heurísticas aleatorias basadas en poblaciones como método de búsqueda en la selección de rasgos tales como: algoritmos genéticos [52, 53], optimización basada en enjambre de partículas (*Particle Swarm Optimization*, PSO) [54] y colonias de hormigas [55].

Otra tendencia es la combinación de técnicas de selección de rasgos. Novedosas combinaciones se han aplicado en el dominio de espectrometría de masas y microarreglos, por ejemplo, métodos basados en colecciones de árboles de decisión [56].

Por otro lado, las técnicas de extracción de rasgos han sido también ampliamente usadas en los problemas de la bioinformática: el uso de PCA en la expresión de genes [57]. *Sammon´s mapping* para la expresión de clusters [58], así como el análisis discriminante lineal para microarreglos [59] y una variante de éste usando funciones núcleo [60].
Algunos trabajos conjugan ambas técnicas, primero reducción de dimensionalidad con extracción de rasgos y luego selección de rasgos en la nueva dimensión generada [55, 61, 62].

I.3. Técnicas para aprendizaje no supervisado

En el caso del aprendizaje no supervisado, también conocido como agrupamiento, se tiene una colección de datos no etiquetados y el problema se basa en formar grupos con los datos que tengan algún sentido, fundamentándose en la información de los casos, y asociar entonces a cada grupo una etiqueta o clase. El proceso de agrupamiento puede dividirse según Jain et al. en los siguientes pasos: representación y preprocesamiento de los datos, definición de una medida de proximidad, usualmente caracterizada por una función de distancia definida para pares de patrones; el agrupamiento, abstracción de los rasgos (si es necesario) y la evaluación de los resultados [63].
Estos métodos se dividen, usualmente, en jerárquicos y no jerárquicos. Los jerárquicos crean una descomposición de los objetos en grupos jerárquicos, al estilo de "taxonomías" (superfamilias, familias, especies…). Ellos requieren además un criterio de distancia entre grupos para decidir su unión, los más populares son los algoritmos de enlace simple y enlace completo [63]. Los no jerárquicos construyen una partición de los datos con *k* grupos, donde cada grupo optimiza un criterio determinado. El más conocido de estos algoritmos es el *k*-medias (*k-means*) [64].
K-medias es un algoritmo muy popular que ha sido muy utilizado en diferentes aplicaciones. Dado un conjunto arbitrario de puntos en R^p y un entero *k*, el algoritmo *k*-medias encuentra *k* puntos en R^p, llamados centroides, y asocia cada punto de los datos al centroide más cercano. Para este algoritmo es muy importante seleccionar una función de distancia apropiada para los datos. A diferencia de los métodos jerárquicos, *k*-medias tiene la desventaja de que es necesario definirle *a priori* el número *k* de grupos que se realizarán.

Por otra parte han ido surgiendo muchos otros métodos de agrupamiento, como los basados en densidad, los basados en celdas, los basados en modelos, métodos para datos categóricos, basados en técnicas de inteligencia artificial y métodos para datos de alta dimensionalidad.

Los métodos basados en densidad agrupan objetos de acuerdo a una función objetivo de densidad específica. Entre los más conocidos se tienen a DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), OPTICS (*Ordering Points To Identify the Clustering Structure*), DBCLASD (*Distribution Based Clustering of Large Spatial Databases*) [65]. Los métodos basados en celdas están orientados a datos espaciales. Entre ellos podemos encontrar STING (*STatistical INformation Grid*) [66] y WaveCluster [67]. Los algoritmos basados en modelos buscan una buena aproximación de los parámetros que mejor ajustan los datos, pueden ser jerárquicos o no jerárquicos. EM (*Expectation-Maximitation*) es uno de los métodos considerados en esta categoría. El algoritmo *k-modes* [68] y el ROCK (*RObust Clustering using linKs*) [69] son ejemplos de métodos para datos categóricos, o sea, bases con atributos categóricos.

Existen varios métodos basados en técnicas de inteligencia artificial, por ejemplo basados en algoritmos genéticos [70]. Otro algoritmo de agrupamiento son las redes neuronales SOM, ya mencionadas anteriormente. Se han creado además, algoritmos para tratar bases de datos muy grandes como el *CobWeb* [71].

Jain et al. [63] y Berkhin [65] hacen un resumen acertado de estos métodos.

WEKA (*Waikato Environment for Knowledge Analysis*) es un paquete de software libre que constituye una plataforma para aprendizaje computarizado [72]. Tiene implementados ya varios de los algoritmos mencionados anteriormente que tienen una amplia utilización en diversas áreas. Como además es de código abierto, permite su extensión con otros algoritmos de aprendizaje automatizado o en particular la adaptación de alguno de los existentes, especificando, por ejemplo, las funciones o medidas de distancia acorde a un problema específico.

Ante este maremágnum de métodos o algoritmos de agrupamiento, pueden seguirse diferentes estrategias. La más simple es exhaustiva: probar varios, a ciegas, y comparar resultados, pero el costo computacional puede ser alto y el por qué del resultado óptimo no tiene siempre una interpretación clara. La estrategia inversa puede ser más científica:

seleccionar un método y concentrarse en la definición óptima de los parámetros relacionados con él; pero ello exige más conocimiento del dominio de aplicación específica [73]. Hoy en día WEKA facilita la prueba y comparación de muchas técnicas a través de su "experimentador"; pero sigue siendo más parsimonioso, seleccionar un método bastante general, por ejemplo *k*-medias, priorizar la definición esencial acorde al problema, digamos la función de distancia y después experimentar otros parámetros, por ejemplo el valor de *k*. Ésta es la estrategia que se seguirá en el presente trabajo y conllevará novedosas medidas de distancia, por ejemplo, para la selección de posiciones importantes de una proteína que interactúan en su plegamiento tridimensional.

I.4. Técnicas para aprendizaje supervisado

El aprendizaje supervisado, como se dijo anteriormente, es aquel donde se tiene información sobre la hipótesis. Puede dividirse en clasificación, si la hipótesis es discreta, y regresión, en caso de que sea continua. Este trabajo se va a centrar en la clasificación, donde cada objeto del conjunto de datos tiene una clase asignada. En este epígrafe se describen algunos de los modelos de clasificación más usados; pero primero es importante definir algunos conceptos claves para la comprensión de los mismos.

Cuando se habla de *objeto*, *caso* o *instancia*, se refiere a cada uno de los elementos de la base de casos. Estos objetos pueden ser representados como vectores $X=(x_1,x_2,...,x_N)$. Cada uno de estos objetos va a describirse por *rasgos*, *características*, o *atributos* que lo caracterizan, cada x_i es un atributo y N es la cantidad de atributos. Una *base de casos* (o más específicamente, en inteligencia artificial, *base de conocimientos*) es un conjunto de objetos etiquetados, donde cada uno tiene la clase correspondiente. Es decir, a cada vector $(x_1, x_2,..., x_N)$ se le hace corresponder una clase $c_j \in \Omega$, donde $\Omega=\{1,2,...,C\}$; C es la cantidad de clases del problema.

Partiendo entonces de estos objetos etiquetados, las técnicas de clasificación construyen un modelo de clasificador para aprender la relación entre estos objetos y la clase. Un *clasificador* o *hipótesis* puede definirse como una función objetivo: $F : \Re^n \rightarrow \Omega$, que a cada N-uplo de atributos X asocia la clase a la cual debe pertenecer. La diferencia de los diferentes modelos de clasificación está justamente en cómo representan F y cómo

buscan el espacio de todas las posibles hipótesis. A esta etapa de construcción del modelo de clasificador se le llama *entrenamiento*.

Después de representado el problema, seleccionado el modelo de clasificador y entrenado el mismo, sólo resta comprobar que el clasificador tiene potencialidad de generalización, o sea, se utilizan casos nuevos, cuya clase es conocida, pero no usados en el entrenamiento, para verificar la efectividad del clasificador. A este proceso se denomina fase de *prueba* o *validación*, también conocida como *evaluación del clasificador*. A continuación se describen algunos de los métodos de clasificación más usados: *k*-vecinos más cercanos, árboles de decisión, redes bayesianas, redes neuronales artificiales y máquinas de soporte vectorial.

I.4.1. Redes bayesianas

Una red bayesiana es un modelo gráfico probabilístico que representa un conjunto de variables y sus dependencias probabilísticas. Son grafos acíclicos dirigidos, donde cada nodo representa un atributo y de cada variable en el espacio se especifican dos informaciones: la estructura de dependencias condicionales y las distribuciones de probabilidad correspondientes. Estas redes pueden ser usadas para inferir un valor objetivo dado los valores observados de otras variables, y recíprocamente, inferir el valor probable de una variable, a partir de la evidencia de otras y/o del valor objetivo. Ello constituye el principal mérito de las redes bayesianas. Existen algoritmos de "propagación de evidencias" que facilitan calcular la probabilidad de una conclusión, sobre cualquier variable, "independiente" o "dependiente" a partir de ciertas evidencias. Cuando no se conocen todos los valores de las variables en el conjunto de entrenamiento, el aprendizaje con una red bayesiana puede ser más difícil. Este problema se hace análogo a la búsqueda de pesos en una red neuronal. Se han propuesto varios algoritmos de entrenamiento para estos casos como el llamado K2, que usa un algoritmo de ascenso de colinas (*hill climbing*) restringido por un orden sobre las variables [74]. Se han desarrollado otros como *tree augmented Naive Bayes* y muchos más [72, 75].

I.4.2. Árboles de decisión

El aprendizaje usando árboles de decisión es un método de aproximación de funciones objetivo de valores discretos, en el cual la función aprendida se representa por un árbol en

el cual intervienen las posibles variables predictivas y sus interacciones. Los árboles obtenidos pueden ser representados como conjuntos de reglas "si-entonces" (*if-then*).

Un árbol de decisión es un grafo acíclico donde cada nodo especifica una prueba de algún rasgo y cada arco que sale del nodo corresponde a alguno de los valores posibles del rasgo que representa ese nodo. Estos árboles clasifican casos no conocidos comenzando por la raíz del árbol, probando el rasgo especificado en el nodo y en dependencia del valor de ese rasgo se mueve al próximo nodo. Este proceso es repetido entonces hasta alcanzar un nodo hoja o terminal que define la clasificación [75].

Probablemente el algoritmo más clásico para construir árboles de decisión es el ID3 [76]; pero este algoritmo no trabajaba para atributos continuos y el propio autor sugiere una variante, el C4.5 [77], que usa puntos de corte e introduce varias medidas para evitar el sobreentrenamiento, en particular los criterios de parada de la división y de poda del árbol. El criterio básico de parada de la división se basa en detener el desglose del árbol cuando el número de objetos asignados al nodo hijo están por debajo de determinado umbral. El segundo criterio, poda, simplifica el árbol eliminando subárboles que conducen a una misma conclusión sobre la clasificación, técnica también conocida como *reduced error pruning*.

Hoy en día existen muchos otros algoritmos e implementaciones, de construcción de árboles de decisión; CHAID (*Chi-square Automatic Interaction Detector*), *Exhaustive* CHAID, QUEST, CRT; y también existen muchos otros criterios de parada o poda. Ellos han sido utilizados con bastante eficiencia en la solución de problemas bioinformáticos [27].

I.4.3. Algoritmos basados en casos

Todos los métodos supervisados se "basan en casos", ya que usan una base de casos para aprender. A diferencia del resto de estos algoritmos, la esencia de los llamados algoritmos basados en casos está en que su entrenamiento es simplemente el almacenamiento de los casos, no necesitan crear reglas, ni árboles, ni ajustar parámetros. Este tipo de aprendizaje es conocido como perezoso. Necesita de la definición de una medida de distancia para comparar cada nueva instancia con las de la base de conocimientos, de forma que para cada nueva instancia se use la instancia de la base que

esté más cercana a ella para asignar la clase, éste se conoce como método del vecino más cercano. A menudo se usa más de una instancia cercana y la clase mayoritaria es la asignada al nuevo caso, éste se denomina *k*-vecinos más cercanos (*k* *Nearest Neighbors*, *k*NN) [78]. Existen otros algoritmos basados en casos, como *k*Start que usa una función de distancia basada en entropía [79]. Como el desempeño computacional de estos algoritmos está en correspondencia con la cantidad de casos del conjunto de entrenamiento, se han desarrollado muchos otros algoritmos con la idea de hacerlos más rápidos [72].

Lo más complejo de estos algoritmos es definir la función de distancia. Aunque existen muchas medidas de distancias definidas para trabajar con estos algoritmos (Ver Anexo 4) generalmente se debe intentar buscar una distancia que defina una "distancia real" de acuerdo con el problema que se quiera resolver.

I.4.4. Máquinas de soporte vectorial

Máquina de soporte vectorial o máquina de vectores de soporte (*Support Vector Machine*, SVM) es una técnica de aprendizaje supervisado que se desarrolló en los últimos años pero muy rápidamente, partiendo de la teoría de aprendizaje estadístico y basada en el principio de minimización de riesgo estructural. Concretamente, fundamenta las decisiones de clasificación, no basadas en todo el conjunto de datos, sino en un número finito y reducido de casos, que constituyen los "vectores soporte". Se ha usado tanto para clasificación (aprendizaje supervisado con función objetivo discreta), como para regresión (aprendizaje supervisado con función objetivo continua). Puede dividirse en SVM lineal y no lineal, basado este último en diferentes funciones núcleo (*kernel*).

En el caso del SVM lineal, éste construye un hiperplano *n*-dimensional de separación en el espacio y selecciona el hiperplano, de tal forma que la distancia desde los ejemplos más cercanos al hiperplano sea máxima [80]. En el caso de la clasificación no lineal la idea es similar, excepto que se realiza una transformación no lineal del conjunto de entrenamiento, o sea, el conjunto de puntos originales es remplazado por los obtenidos con una función núcleo, de forma que se fije el hiperplano en el espacio de rasgos transformados. Hay que tener en cuenta que para que una función pueda ser considerada función núcleo es necesario ante todo que sea simétrica y semidefinida positiva.

Si la función núcleo usada es una de las denominadas gaussiana de base radial (*gaussian radial basis*), el espacio de rasgos correspondiente es un espacio de *Hilbert* de dimensión finita. Los clasificadores de margen máximo son bien regulados, así la dimensión infinita no arruina los resultados. Algunas de las funciones núcleo más comúnmente usadas son:

Polinomial: $k(x, x') = \langle x \cdot x' \rangle^d$ **I.1**

Gaussiana de base radial: $k(x, x') = \exp\left(\frac{\|x - x'\|}{2\sigma^2} \right)$ **I.2**

I.4.5. Redes neuronales artificiales

Las redes neuronales son herramientas matemáticas para la modelación de problemas, que permiten obtener las relaciones funcionales subyacentes entre los datos involucrados en problemas de clasificación, reconocimiento de patrones, regresiones, etc. Son consideradas excelentes aproximadores de funciones esencialmente no lineales, siendo capaces de aprender las características relevantes de un conjunto de datos, para luego reproducirlas en entornos ruidosos o incompletos [81].

Una red neuronal puede ser caracterizada por el modelo de la neurona, el esquema de conexión que presentan sus neuronas, conocido como topología, y el algoritmo de aprendizaje empleado para adaptar su función de cómputo a las necesidades del problema particular.

Existe una amplia variedad de modelos de neuronas, cada uno se corresponde con un tipo determinado de función de activación de la neurona. La topología es la forma específica de conexión (arquitectura) y la cantidad de neuronas conectadas (el número de parámetros libres) que describen una red. Se ha producido una amplia variedad y clasificación de topologías de redes neuronales, pero a los fines de orientación del presente trabajo, la mayoría de ellas se encuentran ubicadas en uno de dos grandes grupos: las redes multicapa de alimentación hacia delante (*Feed-Forward Neuronal Networks*, FFN) y las redes neuronales recurrentes (*Recurrent Neuronal Networks*, RNN).

Los algoritmos de entrenamiento constituyen métodos que se aplican sobre los modelos de red para ajustar sus pesos y obtener un comportamiento determinado. Con frecuencia los algoritmos de entrenamiento son caracterizados por la clase de topologías sobre las

que se aplica, los tipos de parámetros libres que afecta (pesos de las conexiones entre neuronas, parámetros del algoritmo de entrenamiento, la topología misma de la red, etc.) y la regla de modificación de los mismos. Existe una amplia variedad de algoritmos de entrenamiento disponibles, y generalmente se clasifican en supervisados o no supervisados.

I.4.5.1. Redes con conexiones hacia delante

La característica fundamental de las redes FFN reside en que sus neuronas están conectadas a manera de grafo acíclico dirigido (con todos sus arcos en una sola dirección). Este tipo de red define una relación de orden parcial entre sus neuronas y con frecuencia éstas pueden agruparse en forma de capas siguiendo dicha relación. El Perceptrón multicapa (*Multilayer Perceptron*, MLP) constituye un ejemplo genérico de las redes FFN. Cuando la red no tiene capas ocultas ésta corresponde al Perceptrón de Rosemblatt, que constituyó el modelo precursor de las redes MLP.

El poder de representación de las redes MLP está relacionado con la existencia de al menos una capa de neuronas ocultas, las cuales transforman la entrada, de manera no lineal, en una representación interna. Esta representación interna puede servir luego a las capas subsecuentes para resolver los problemas tratados. En este sentido Lippman [82] describe el poder de representación arbitrariamente complejo que puede llegar a presentar un MLP de 4 capas.

La red MLP usa un algoritmo de entrenamiento de gran potencialidad, conocido como *propagación del error hacia atrás* o *retropropagación* (*Backpropagation*, BP). Este algoritmo aplica la técnica gradiente descendente para la minimización del error de funcionamiento de la red.

I.4.5.2. Redes neuronales recurrentes

Las redes neuronales recurrentes poseen una diferencia notable con la clase anterior: admiten conexiones hacia atrás, o sea, pueden formar ciclos en el grafo que describe sus conexiones. Estas conexiones hacia atrás, llamadas de retroalimentación, son las que permiten que la red sea capaz de guardar una memoria de los estados anteriores para su uso en el cálculo de las salidas del estado actual, o sea, mantener una especie de memoria

de los procesamientos pasados; ésta es la característica esencial que convierte a este tipo de redes en una herramienta de amplio uso en tareas de reproducción de señales y análisis de secuencias, donde se reflejan relaciones causales en el tiempo y el espacio respectivamente. De manera general son aplicables a problemas reales que reflejan relaciones dinámicas y estructurales de orden complejo [29, 83].

Las RNN se consideran sistemas dinámicos cuyo estado evoluciona siguiendo un marcado comportamiento no lineal. Exhiben dos tipos básicos de funcionamiento: sistema autónomo convergente y sistema no autónomo no convergente. El primer tipo de comportamiento permite realizar tareas de asociación (como las resueltas por los modelos de memorias asociativas [84]), la generación de señales con condiciones iniciales fijas, etc. El segundo permite el análisis y la reproducción de señales bajo condiciones variables en el tiempo, así como realizar tareas de clasificación y predicción en estructuras de información complejas.

De manera general, una red recurrente puede expresarse como un conjunto de ecuaciones de la forma siguiente [85].

$$y_i(t) = \Phi_i\left(x_i(t), I_i(t)\right) \qquad \textbf{I.3}$$

$$x_i(t) = \sum_j y_j(t)\, w_{ji}(t) \qquad \textbf{I.4}$$

donde y_i denota el estado de activación de la neurona i en el tiempo t, $I_i(t)$ denota la entrada exterior de la neurona i y Φ_i la función que gobierna el comportamiento del estado de activación de la neurona i en el tiempo t, constituye una función influida por la topología de la red (expresado en los valores netos de entrada x_i) y por la función de entrada $I_i(t)$ que aporta las señales provenientes del exterior. Los valores w_{ji} son pesos, que deben ser optimizados.

El objetivo principal de las investigaciones sobre redes recurrentes estriba en el conocimiento de la relación entre la estructura y el comportamiento dinámico de las redes, y la producción de algoritmos de entrenamiento que permitan dominar su aplicación, es decir, encontrar las estructuras de redes adecuadas (topologías y pesos) para establecer un comportamiento dinámico correspondiente a los problemas que se pretendan resolver.

Muchos algoritmos exactos y aproximados han sido desarrollados para el entrenamiento de las redes recurrentes. La mayoría de ellos basan su trabajo en la técnica del gradiente descendente y pueden ser agrupados, según Atiya y Parlos [86], en 5 clases generales: propagación del error hacia atrás en el tiempo (*Backpropagation Through Time*, BPTT), propagación hacia delante (*Forward Propagation* o *Real Time*), aprendizaje recurrente (*Recurrent Learning*, RTRL), propagación rápida hacia delante (*Fast Forward Propagation*), Funciones de Green y Actualizaciones en bloque (*Block Update*).

Entre ellas, BPTT es una de las técnicas más empleadas y es una variante extendida del BP original. BP posee dos fases: la primera, el cálculo de las salidas de las neuronas de la red (proceso hacia delante), la segunda, la propagación del error hacia atrás (proceso hacia atrás). En el proceso de retropropagación del error, cada neurona *j* es caracterizada por una magnitud de error. BPTT puede verse dividido en tres procesos, el primero, denominado despliegue (*unfolding*), convierte la red recurrente original en una red que sólo tiene conexiones hacia delante. El segundo proceso consiste en la aplicación de la primera fase del algoritmo BP a la red obtenida. Seguidamente, como tercer proceso, se vuelve a plegar la red, obteniendo la red original, y por último se aplica la segunda fase del algoritmo BP. En el Anexo 5 se muestran los pasos del algoritmo BPTT.

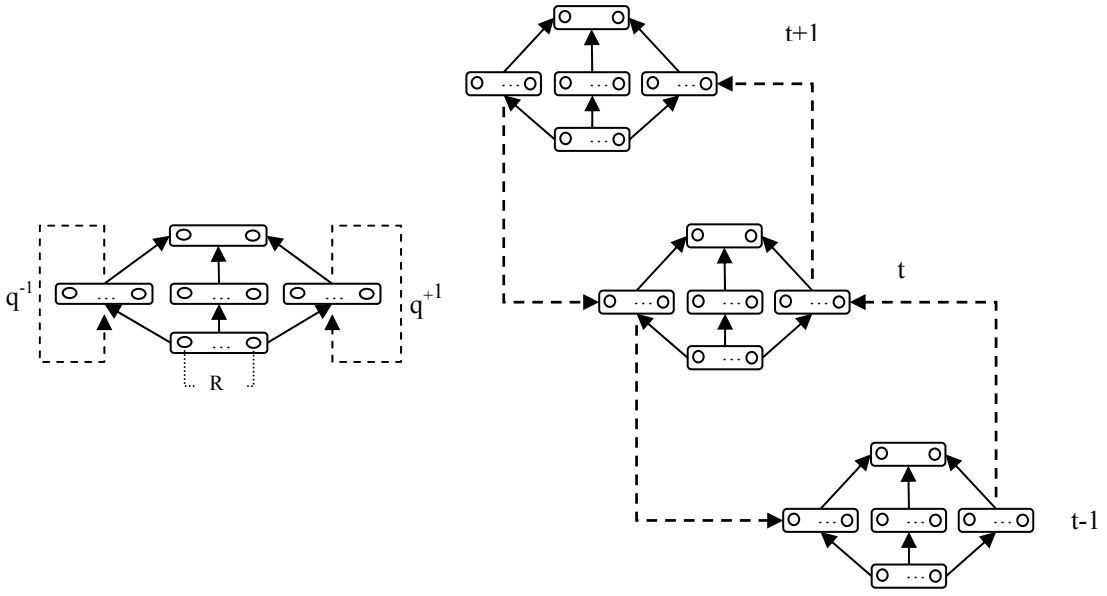


Figura I.1 A la izquierda se muestra la red original con los operadores de desplazamiento q^{-1} y q^{+1} que representa conexiones hacia el tiempo t-1 y tiempo t+1 respectivamente. A la derecha se muestra el proceso de despliegue de la red en tres tiempos.

En la Figura I.1 puede verse cómo la red original se replica (desprovista de las conexiones recurrentes) tantas veces como tiempos sean considerados. Las conexiones recurrentes son identificadas con operadores de desplazamiento q^{l+} y q^{l-}, donde l es la cantidad de tiempos hacia delante o hacia atrás en dependencia de si l es positivo o negativo respectivamente. Teniendo en cuenta esto, después de replicada la red las conexiones recurrentes se conectan al tiempo correspondiente l hacia delante o hacia atrás según el operador de desplazamiento asociado. Cada conexión replicada de la red original comparte los mismos pesos en las diferentes etapas.

Estas redes son muy útiles en análisis de secuencias y señales, en problemas de asociación o reconstrucción de secuencias.

I.5. Técnicas para combinar clasificadores

No existe todavía un clasificador por excelencia; para un problema determinado es difícil seleccionar cual será el clasificador que logre encontrar una mejor frontera de decisión para separar las clases. En la búsqueda de mejores métodos de clasificación aparece una tendencia a combinar varios clasificadores en el mismo problema. Esta tendencia tiene varias explicaciones matemáticas pero sobre todo un pretexto psicológico por su conexión con nuestras experiencias diarias. Es muy común cuando se va a tomar una decisión importante para la vida consultar varias personas, en aras de oír diferentes criterios. Precisamente ésta constituye la base de estos algoritmos que también son conocidos como multiclasificadores: utilizar varios expertos (clasificadores) y combinar sus diferentes salidas [87] en aras de lograr un mejor rendimiento.

Dietterich [88] ofrece tres justificaciones de por qué un sistema multiclasificador puede ser mejor que un clasificador simple. La primera es estadística, pues si efectivamente por cada clasificador tenemos una hipótesis, la idea de combinar estas hipótesis, da como resultado una hipótesis que puede no ser la mejor, pero al menos evita seleccionar la peor de ellas. La segunda justificación es computacional, ya que algunos algoritmos ejecutan búsquedas que pueden llevar a diferentes óptimos locales: cada clasificador comienza la búsqueda desde un punto diferente y termina cercano al óptimo. Existe la expectativa de que alguna vía de combinación puede llevar a un clasificador con una mejor aproximación. Esta última justificación es figurativa ya que es posible que el espacio de

hipótesis considerado no contenga la hipótesis óptima; pero la aproximación de varias fronteras de decisión puede dar como consecuencia una nueva hipótesis fuera del espacio inicial y que se aproxime más a la óptima.

Los multiclasificadores pueden construirse de diversas formas. Existen varios algoritmos desarrollados, algunos para problemas generales como *bagging* y *boosting* y otros para problemas específicos. En esencia estos métodos tienen dos partes importantes: selección de los clasificadores de base y elección de la forma de combinar las salidas.

I.5.1. Selección de clasificadores de base y diversidad

La selección de los clasificadores de base es el primer paso a la hora de construir un multiclasificador. Algunos paradigmas de combinar clasificadores usan el mismo modelo de clasificación, pero no existe evidencia de si esa estrategia es mejor que el uso de diferentes modelos [89].

Una de las condiciones para obtener buenos clasificadores de base es lograr la diversidad de los mismos y la diversidad mide cuán correlacionados son los resultados de los diferentes clasificadores. Esta diversidad puede lograrse de diversas maneras, una de ellas es con la variabilidad en los parámetros del modelo de clasificación, lo que permite poder usar un único modelo, donde se cambie alguno de sus parámetros. Por ejemplo, las redes neuronales son algoritmos en que la simple diferencia en la inicialización de los pesos puede producir diferentes clasificadores resultantes. El conjunto de entrenamiento es uno de los parámetros que puede ser cambiado, como lo hacen *bagging* y *boosting*, que serán discutidos en el epígrafe I.5.3.

No sólo se pueden manipular los conjuntos de entrenamiento, otra técnica es la manipulación del conjunto de rasgos. Esto es posible cuando se cuenta con una base con muchos rasgos: se puede dividir la base según subconjuntos de rasgos y cada clasificador se entrenará con un conjunto de rasgos diferente. Otra vía es la manipulación de las clases, y se aplica cuando el problema tiene muchas clases. Se puede dividir la base según las instancias correspondientes a subconjuntos de clases [88]. También puede lograrse diversidad con el uso de diferentes modelos de clasificación. Existen varias medidas que pueden ser usadas para definir cuál es la combinación de clasificadores más diversa [90, 91].

La diversidad entre los clasificadores de base es muy importante, ya que de esto dependerá en gran medida el resultado final del multiclasificador. Cuán diversos sean los errores de los clasificadores puede dar una medida del mayor valor posible que se puede aspirar con la combinación de esos modelos. Aunque existen varias medidas, todavía hay divergencia de criterios sobre el concepto de diversidad y qué debe medir realmente.

I.5.2. Combinación de salidas

La combinación de las salidas es otro paso importante a la hora de construir un sistema multiclasificador. Puede dividirse en dos tipos: selección o fusión. La selección es la simple elección del "mejor" clasificador para una instancia determinada. La fusión, por otro lado, se basa en combinar, mediante alguna función, las salidas de los diferentes clasificadores. Esta segunda puede basarse en combinar las salidas de las clases ya etiquetadas o las salidas de valores continuos, correspondientes a la distribución de probabilidad dada por el clasificador.

Una de las formas más conocidas de combinar las clases es el voto mayoritario. Aquí cada clasificador le da un voto a las clases acorde con su resultado y finalmente se selecciona la clase con mayor cantidad de votos. Una versión de éste es el voto mayoritario pesado donde cada clasificador tiene además un peso, el cual puede estar dado, por ejemplo, por el error de clasificación obtenido por él en el conjunto de entrenamiento.

Otra de las combinaciones es la de *Naive Bayes* que asume que los clasificadores son mutuamente independientes, de forma que la probabilidad asociada a cada clase en el sistema completo estará dada por la sumatoria de todas las probabilidades condicionales de las diferentes hipótesis. En [89] se comentan otros métodos para combinar las diferentes clases resultantes de los clasificadores de base.

Para entender un poco mejor estos métodos que combinan las salidas continuas, es mejor basarse en las matrices llamadas perfiles de decisión (*decision profile*, DP), como se muestra en la Figura I.2. Sea $H=\{H_1,H_2,...,H_M\}$ el conjunto de clasificadores y $\Omega=\{C_1,C_2,...C_N\}$ el conjunto de clases. Para una instancia x, se denota por $d_{ij}(x)$ el valor de distribución de probabilidad que el clasificador H_i da a la hipótesis que x sea de la clase C_j.

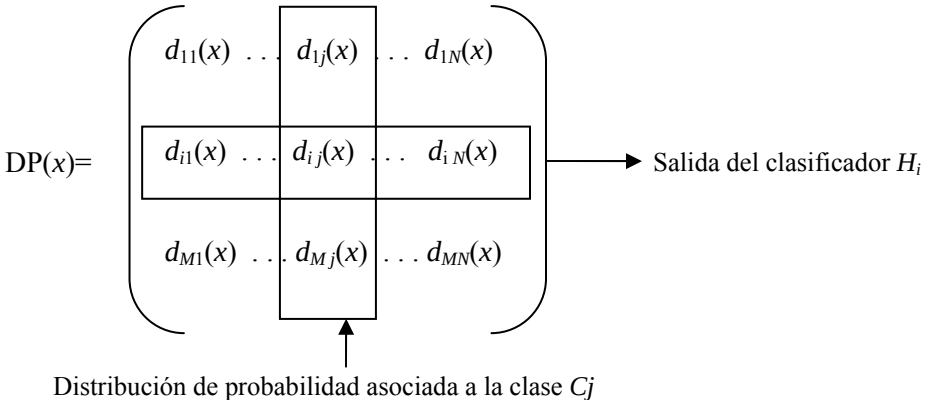


Figura I.2 Perfiles de decisión de un conjunto de clasificadores.

Este tipo de combinaciones de salidas continuas a menudo pueden verse en la literatura clasificadas según si necesitan entrenamiento o no [92]. Aquellas que no necesitan entrenamiento, son las que después de entrenados los clasificadores de base, ya el sistema está listo para clasificar (a partir de la matriz DP). Entre las más conocidas están las combinaciones algebraicas, como la regla de la media, que calcula el promedio de las distribuciones de probabilidades dadas por cada clasificador para cada clase, esto es, el promedio de la columna asociada a cada clase en la matriz DP; o cualquier otra función como el máximo, el mínimo o la moda.

Por otro lado, las combinaciones que necesitan entrenamiento son aquellas que después de entrenar a los clasificadores necesitan cierto ajuste de parámetros antes del sistema estar listo para clasificar. Entre ellas podemos citar el promedio pesado, que es una combinación de la regla de la media con el voto mayoritario pesado. El uso de una medida borrosa también es una de las combinaciones de salida que necesita entrenamiento [93].

Además de este tipo de combinaciones existen otras combinaciones complejas como la propuesta por Kuncheva et al. [94] que se basa en una extensión de los DP, denominada plantilla de decisión (*decision template*, DT), que no es más que DPs promediados para cada una de las clases, o sea, para la clase C_j :

$\mathrm{DT_j} = \frac{1}{\left|X_j\right|} \sum_{Xj \in C_j} DP(X_j)$, donde X_j es el conjunto de instancias del entrenamiento que pertenecen a la clase C_j. Dada una nueva instancia x entonces se construye su DP y se

calcula la semejanza entre DP(x) y las plantillas de decisión para cada clase. La clase con mayor semejanza es entonces la seleccionada.

Otras formas de combinar las salidas usan un metaclasificador que aprende la relación entre las salidas de los clasificadores y la clase real, siendo s*tacking* un ejemplo de este método, y que se describe en detalle en el próximo epígrafe.

I.5.3. Modelos de multiclasificadores

Entre los modelos que combinan clasificadores más populares están *bagging*, *boosting*, *stacking* y mezcla de expertos (*mixture of experts*).

Bagging fue introducido por Breiman en 1996, y su nombre es un acrónimo de *Boostrap AGGregatING* [95]. Es uno de los primeros algoritmos multiclasificadores, que se basa en crear diferentes conjuntos de entrenamiento, extraídos del conjunto inicial de manera aleatoria y con reemplazo con lo cual asegura la diversidad Este modelo necesita la selección de un modelo de clasificador inestable, o sea, un modelo que con pequeños cambios obtenga valores diferentes. Además usa un único modelo de clasificador y la combinación de los clasificadores resultantes se realiza con la técnica de voto mayoritario. Variaciones a este algoritmo son el caso de *ramdom forests* [96] que es construido con árboles de decisión como modelo de clasificador, y para crear los conjuntos de entrenamiento, usa el método de muestreo con reemplazo como en *bagging*, o puede usar subconjuntos de rasgos.

Boosting es parecido a *bagging* porque usa el método de crear bases de entrenamiento aleatorias con reemplazo, a partir de la base original y un único modelo de clasificación para los clasificadores de base. Sin embargo, este algoritmo se realiza de manera secuencial, donde los clasificadores se van entrenando uno detrás del otro porque usan información del anterior. El reemplazo es realizado estratégicamente de forma que los casos mal clasificados tienen mayor probabilidad, que los bien clasificados, de pertenecer al conjunto de entrenamiento del siguiente clasificador del sistema. Este método fue creado en 1990 por Schapire [97]. Poco después en 1997 se crea una extensión del mismo, conocida como *AdaBoost*, que es hoy en día la más utilizada [98]. Es una versión más general que se ha dividido en *AdaBoost.M1* y *AdaBoost.R*, que son capaces de

manipular múltiples clases y problemas de regresión respectivamente. *AdaBoost* usa como método de combinación el voto mayoritario pesado.

Stacking [99] es un método diferente a los anteriores pues la diversidad la busca con el empleo de diversos modelos de clasificación. Para combinar las salidas usa un metaclasificador que aprende la relación entre las salidas de los clasificadores de base y la clase original. Este metaclasificador tiene como base de entrenamiento un nuevo conjunto de instancias formadas a partir del conjunto de entrenamiento inicial para los clasificadores de base, donde por cada instancia del conjunto de entrenamiento se tiene ahora un vector de rasgos compuesto por las clases de salida de cada clasificador de base y como clase, la original de la instancia.

El modelo de expertos mixtos [100] es una técnica conceptualmente similar a *stacking*, donde se tiene un conjunto de clasificadores de base de modelos diferentes; en un segundo nivel tiene un metaclasificador, cuya función es encontrar los pesos que asignará a cada clasificador para en un tercer nivel aplicar una técnica de combinación que use estos pesos. Usualmente en el segundo nivel se usa una red neuronal, que a diferencia de *stacking*, tiene como entrada el conjunto de entrenamiento original. Esta red se entrena a medida que se entrenan los clasificadores de base, ya que la salida de la misma puede verse como el vector de probabilidades de que cada clasificador sea seleccionado, eligiendo el mejor de éstos para cada instancia. Esta red neuronal del segundo nivel es comúnmente usada con el método conocido como *expectation maximization* (EM).

Actualmente continúan desarrollándose métodos que combinan clasificadores. Nguyen et al. proponen una combinación de EM con un método co-evolutivo cooperativo (*cooperative coevolutionary*) [101]. Saha et al. usan un multiclasificador que utiliza conjuntos aproximados (*Rough Set*) como metaclasificador para generar reglas que relacionan conjuntos de rasgos seleccionados (reductos) con el atributo que representa la clase [102]. Dimitrakakis y Bengio utilizan una política adaptativa, basada en el algoritmo *Q-learning*, para entrenar y combinar los clasificadores [103]. También Partalas et al. usan el algoritmo *Q-learning* para podar el conjunto de clasificadores, en el proceso de combinación de la salida [104]. Nanni y Lumini muestran un algoritmo llamado *FuzzyBagging* que divide el conjunto de entrenamiento con un algoritmo de

agrupamiento, de forma que cada clasificador se entrena con uno de los conjuntos resultantes del agrupamiento [105].

I.6. Aplicación de las técnicas de inteligencia artificial para aprendizaje supervisado en bioinformática

Como se ha comentado antes, dentro de la bioinformática son muchos los problemas de clasificación que se pueden encontrar. Baldi y Soren [25], y Larrañaga et al. [24] hacen un estudio de varios métodos supervisados aplicados a diversos problemas de bioinformática. Una de las más importantes aplicaciones puede encontrarse en el dominio de la genómica [106]. Salzberg [107] usa árboles de decisión para la búsqueda de regiones codificantes en el ADN humano. Castelo y Guigó [108] utilizan un nuevo clasificador bayesiano para la predicción de sitios de *splicing*. Carter et al. [109] usan SVMs y redes neuronales para identificar la funcionalidad de genes ARN.

En el dominio de la proteómica, para la predicción de estructura secundaria se han aplicado el *k*NN [110], árboles de decisión [111], redes neuronales recurrentes [28], entre otros. Yang et al. [112] desarrollan un método en dos pasos basado en redes bayesianas y SVM para predecir los residuos de superficie de proteína que participa en las interacciones proteína-proteína. El problema de predicción de una localización subcelular de proteína desde sus secuencias se trató con un *k*NN difuso [113].

En el dominio de los microarreglos [114] se usa una generalización bayesiana del SVM para seleccionar el clasificador óptimo y el conjunto óptimo de genes para diagnosis de cáncer basada en microarreglos. En [115] se hace una comparación entre diferentes paradigmas de clasificación como *k*NN, análisis discriminante, árboles de decisión, *bagging* y *boosting* en tres estudios de expresión de genes de cáncer. Statnikov et al. [116] muestran una comparación extensiva entre varios algoritmos de clasificación: SVM, *k*NN, RNA y varias combinaciones de clasificadores; en 11 bases de datos para el diagnóstico de cáncer.

En la literatura consultada no aparecen trabajos que referencien el uso de las redes neuronales recurrentes para clasificación en la bioinformática. Sin embargo, se usan algunas variantes para resolver otros problemas, como por ejemplo, en [28] se describe una topología de red recurrente bidireccional dinámica en la predicción de estructuras

secundarias. El hecho de que en las secuencias genómicas exista interacción entre posiciones relativamente lejanas y no necesariamente adyacentes fue uno de los aspectos que motivaron que en la presente tesis se investigaran las posibilidades de las redes neuronales recurrentes para clasificación.

La predicción de fármacos cancerígenos también ha sido tratada como un problema de clasificación con redes neuronales MLP [117].

En la predicción de resistencia del VIH se pueden encontrar trabajos con el uso de árboles de decisión, *k*NN [118], uso de SVM [9], el empleo de redes neuronales [8, 119]. Por su parte Cao [13], Rabinowitz [14] y Rhee [15] realizan una comparación de varios de estos métodos en la predicción de la resistencia al VIH, resultando para algunos fármacos un clasificador mejor y para otros otro.

I.7. Evaluación en la clasificación

No existe un modelo de clasificador mejor que otro de manera general; para cada problema nuevo es necesario determinar con cuál se pueden obtener mejores resultados, y es por esto que han surgido varias medidas para evaluar la clasificación y comparar los modelos empleados para un problema determinado. Las medidas más conocidas para evaluar la clasificación están basadas en la matriz de confusión que se obtiene cuando se prueba el clasificador en un conjunto de datos que no intervienen en el entrenamiento. A continuación se muestra la matriz de confusión de un problema de dos clases, donde C_1 es la clase negativa y C_2 la clase positiva:

	Clase obtenida	
Clase Real	C_1	C_2
C_1	TN	FP
C_2	FN	TP

TP y TN son la cantidad de elementos bien clasificados de la clase positiva y negativa respectivamente. FP y FN son la cantidad de elementos negativos y positivos mal clasificados respectivamente. Basados en estas medidas, se calcula el error, la exactitud (*accuracy*), la razón de TP (*TP rate*) o sensitividad, la razón de FP (*FP rate*), la precisión y especificidad, que se dan por las expresiones siguientes:

$$Error = \frac{FP + FN}{TP + TN + FP + FN} \quad \textbf{I.5}$$

$$Exactitud = 1 - Error \quad \textbf{I.6}$$

$$Razón\ de\ TP = Sensitividad = \frac{TP}{TP + FN} \quad \textbf{I.7}$$

$$Razón\ de\ FP = \frac{FP}{FP + TN} \quad \textbf{I.8}$$

$$Precisión = \frac{TP}{TP + FP} \quad \textbf{I.9}$$

$$Especificidad = \frac{TN}{TN + FP} \quad \textbf{I.10}$$

Otra forma de evaluar el rendimiento de un clasificador es por el análisis de la llamada *Receiver Operator Curve* (ROC) [120]. En esta curva es representado el valor de razón de TP vs. la razón de FP, mediante la variación del umbral de decisión. Se denomina umbral de decisión a aquel que decide si una instancia *x*, a partir del vector de salida del clasificador, pertenece o no a cada una de las clases. Usualmente, en el caso de dos clases se toma como umbral por defecto 0.5; pero esto no es siempre lo más conveniente. Se usa el área bajo esta curva (*Area Under the Curve*, AUC) como un indicador de la calidad del clasificador. En tanto dicha área esté más cercana a 1, el comportamiento del clasificador está más cercano al clasificador perfecto (aquel que lograría 100% de TP con un 0% de FP).

Pero estas medidas no son suficientes, usándolas así simplemente, para evaluar un clasificador. La forma de dividir los datos en conjunto de entrenamiento y prueba es también muy importante. Existen varias técnicas para esto, la más vieja es el *método R* (*resubstitution*) el cual se basa en entrenar y probar el clasificador con la misma base de datos. Este método puede traer como consecuencia un sobre-aprendizaje del clasificador, o sea, que el clasificador más que generalizar el conocimiento de los datos, aprenda esto "de memoria". Otro método usado es el *método H* (*Hold-out*), que divide la base a la mitad, una mitad para entrenamiento y otra para prueba. Una versión de éste es el *data shuffle* que realiza *n* veces el *método H* y promedia los resultados. El *método de validación cruzada con k subconjuntos* (*k-fold cross-validation*) es uno de los más usados, este método se basa en dividir la base en *k* segmentos y realizar *k* procesos de

entrenamientos y pruebas, de forma que el proceso *i* toma el segmento *i* para prueba y el resto para entrenamiento. Sea *I* la cantidad de instancias de la base de datos, si *k=I* entonces el método se denomina *validación cruzada dejando uno fuera* (*leave-one-out*). El método *boostrap* se basa en la generación de *n* conjuntos de cardinalidad *I* desde el conjunto de datos original, con reemplazo.

Actualmente se usa también en vez de dos conjuntos de datos, tres: uno para entrenamiento, uno para prueba y un tercero para validación. Este último se usa como seudoentrenamiento, de tal manera que el proceso de entrenamiento se detiene cuando comience a decrecer el rendimiento sobre el conjunto de validación, aunque continúe aumentando sobre el conjunto de entrenamiento [89]. Este método es muy útil para evitar el sobreentrenamiento. También se usa para ajustar parámetros y seleccionar un modelo apropiado. Tiene como desventaja que necesita un conjunto de datos muy grande.

La mejor forma de organizar el experimento en entrenamiento y prueba realmente depende de las características de la base de datos. En grandes volúmenes de datos la validación cruzada con 10 subconjuntos, que es una de las más usadas, no es muy útil pues demoraría demasiado la validación, aquí quizás un método *H* sería mejor. Por el contrario, una base de datos pequeña no permitiría el uso del método *H* ya que la base de entrenamiento sería demasiado pequeña y divisiones de la misma pueden dejar una base no completa; quizás aquí el método *boostrap* sea mejor.

I.8. Consideraciones finales del capítulo

El proceso de aprendizaje supervisado en bioinformática se divide en tres partes: análisis y preprocesamiento de los datos, selección del modelo de clasificación y evaluación del modelo, ninguna de las cuales puede ser obviada pues resultan imprescindibles a la hora de evaluar los resultados que se alcanzan.

Los problemas de clasificación de secuencias en bioinformática son generalmente problemas complejos, requieren muchas veces de un largo proceso de análisis y procesamiento de los datos, reducir la dimensionalidad o encontrar alguna información visual que ayude en los siguientes pasos.

Como se está trabajando con secuencias biológicas con su estructura primaria, es preferible no usar técnicas de selección de rasgos para no perder información. Se

procurará entonces una medida de distancia que tenga en cuenta información biológica, y permita agrupar aquellos rasgos que tengan características parecidas. Se selecciona el uso de métodos de agrupamiento que permitan formar grupos de rasgos a combinar en nuevos rasgos. Se puede usar además cualquier método de extracción basado en distancias que corrobore la hipótesis de que existe relación entre los rasgos. Para esto se deben definir funciones de distancia que caractericen el problema y garantizar con ellas la medición de la efectividad de las técnicas de extracción de rasgos.

La selección del clasificador, no es elemental, se necesita muchas veces probar varios, o quizás asumir ciertas hipótesis, basadas en el análisis previo. Teniendo en cuenta que las secuencias de ADN, ARN y proteínas son representaciones lineales de estructuras que en realidad cuentan con esas mismas posiciones pero plegadas en el espacio en una estructura tridimensional, debieran analizarse como secuencias en el tiempo.

Las redes neuronales recurrentes resultan adecuadas en los problemas de predicción de secuencias o propiedades a partir de otras secuencias, debido a ciertas características que parecen presentar estas secuencias biológicas. En el próximo capítulo se propone un modelo que permite usar estas redes para solucionar problemas de clasificación a partir de secuencias y demostrar que, efectivamente, este tipo de redes son buenos modelos de clasificación cuando hay relación entre los rasgos y la separación entre las clases es compleja.

La combinación de clasificadores debe favorecer la obtención de mejores niveles de exactitud y precisión, elementos fundamentales a alcanzar en problemas de análisis de secuencias. No se puede precisar cuáles de los métodos son mejores para un problema determinado, si los basados en un solo modelo de clasificación o los que combinan varios clasificadores. En este trabajo se propone un nuevo modelo de combinación de clasificadores.

Capítulo II. REDES NEURONALES RECURRENTES, SU EFECTIVIDAD EN ANÁLISIS DE SECUENCIAS.

Las redes recurrentes, al igual que otros métodos de aprendizaje como redes neuronales multicapas y SVM con función núcleo no lineal, realizan una extracción de rasgos interna (en sus capas ocultas o en el uso de la función núcleo) que facilita la transformación espacial de los mismos, en la búsqueda de la función objetivo. Por eso, una extracción de rasgos previa, no sólo tiene el objetivo de reducir la dimensionalidad de los datos, sino que sirve de análisis inicial para decidir qué métodos usar. Aunque las extracciones sean diferentes, ayudan a saber si una combinación de rasgos tiene sentido. En el caso particular de las redes recurrentes puede demostrarse que son buenos clasificadores en problemas donde el rasgo objetivo está influenciado por combinaciones complejas (no lineales) de los rasgos.

Como han mostrado Baldi y Soren [25], las redes recurrentes pueden servir para resolver problemas de análisis de secuencias biológicas. Este tipo de redes es apropiado para resolver problemas de clasificación donde el vector de salida es de dimensión N, $N>1$. En este capítulo se hace una propuesta de combinación de salidas para una topología bidireccional que ayude a resolver problemas de clasificación simple, donde existe un único rasgo objetivo, $N=1$. A partir de ahora, cuando se hable de clasificación se está haciendo referencia, precisamente a clasificación simple.

En el primer epígrafe se propone un análisis inicial de los datos, con el uso de extracción de rasgos, que ayuda a decidir si el uso de redes recurrentes puede ser apropiado.

II.1. Análisis de los datos usando extracción de rasgos

Anteriormente se vieron varias formas de análisis de datos, y se hizo énfasis en la extracción de rasgos como una herramienta útil que ayuda a interpretar características de los datos. A la hora de realizar una extracción de rasgos un punto clave es la selección de la medida de distancia. Existen en la literatura varias funciones de distancia definidas. Entre las más populares están: la distancia *Euclidiana*, métrica de *Minkowski*, distancia *Mahalanobis* y otras (Ver Anexo 4).

Cuando se analiza una base de casos existen dos formas de crear una medida de distancia. La primera tiene que ver con comparar los objetos, o sea, una medida de cuán diferentes son dos casos. Ésta es la forma de aplicación más comúnmente usada. La segunda tiene que ver con los rasgos, establecer una medida para comparar los atributos dentro de un mismo caso. Ésta es una de las más complejas, ya que generalmente no se cuenta con atributos de igual origen; pero en este caso, como trabajamos con secuencias de datos del mismo nivel de medición, cada uno de los rasgos está medido en el mismo dominio. Esto posibilita definir esta medida de distancia entre rasgos que, como se ha dicho, no siempre es posible. A continuación se explican dos medidas de distancia, así como el proceso de extracción de rasgos en cada caso. La primera medida de distancia está basada en la comparación de los rasgos, la distancia se establece entre los rasgos, y la segunda establece la distancia entre los casos u objetos de la base.

Pero antes, se hace necesario entrar en el proceso de representación de las secuencias, que es uno de los pasos más importantes cuando se está tratando con un problema de aprendizaje.

En algunos trabajos se usa la simple representación de vectores de 20 elementos para representar cada aminoácido (teniendo en cuenta que son 20 aminoácidos), donde tiene valor uno el elemento que coincide con la posición del aminoácido que representa y cero todos los demás. En otros trabajos se han usado perfiles basados en información mutua para representar cada secuencia de proteínas [118, 121].

En este trabajo se busca una representación que contenga más información sobre la estructura tridimensional. Se analizó la energía de contacto de los aminoácidos como propicia, pues está asociada con el despliegue de la enzima en el espacio (Ver Anexo 6). El cambio de un aminoácido influye en el cambio de su energía para atraer al resto de los aminoácidos, y este cambio influye en la estructura 3D de la enzima. En [22] se mostró que las energías de contacto influyen en la estructura de la proteína. El reemplazo de un simple aminoácido es suficiente para que los valores observados de las energías libres cambien [23]. Por esta razón, en este trabajo se usa la energía para representar los aminoácidos de las secuencias biológicas. Cada aminoácido representará un rasgo, el cual será descrito por su energía de contacto, por lo que se tendrán tantos rasgos como aminoácidos tenga la secuencia.

II.1.1. Medidas de distancia

A continuación se mostrarán dos medidas de distancia para secuencias de aminoácidos. La primera se centra en comparar los aminoácidos de una secuencia dada, con el objetivo de buscar aquellas posiciones que tienen influencias semejantes. La segunda es otra medida de distancia, que se centra en comparar las secuencias entre sí.

Una medida de distancia puede basarse en una medida de semejanza, ya que es justamente lo contrario, mientras mayor es la semejanza menor es la distancia. Teniendo en cuenta que estos valores estarán en el rango [0,1], la distancia entre dos objetos *X* y *Y*, se relaciona con la semejanza, según muestra la ecuación II.1, donde D(*X*, *Y*) es la distancia entre *X* y *Y*, S(*X*, *Y*) es la semejanza entre estos dos objetos.

$$D(X, Y) = 1 - S(X, Y) \qquad \textbf{II.1}$$

La idea es elaborar nuevas distancias que tengan en cuenta información biológica, de manera que se mantenga, lo mejor posible, el significado de los datos, para evitar la desventaja ya comentada de la selección de rasgos.

II.1.1.1. Medida de distancia entre rasgos

Teniendo en cuenta que se van a usar las energías de contacto de los aminoácidos para representar los mismos, entonces ellos pueden verse como rasgos numéricos, lo que facilita la elaboración de la medida de distancia.

Cuando se comparan los aminoácidos en una secuencia no tiene sentido basarse sólo en las energías de contacto, pues no por tener el mismo aminoácido estas posiciones tienen que estar relacionadas, ello depende también de cuán distantes estén en la secuencia lineal. Finalmente en la estructura tridimensional cada uno de los aminoácidos está cerca de los que tiene a su lado en la estructura primaria, de esta manera se tiene la posición como una información adicional. La idea es agrupar las posiciones de la secuencia teniendo en cuenta la repercusión que tienen las mutaciones. Dadas *n* secuencias, la medida de distancia propuesta queda como muestra la ecuación II.2, donde, para una secuencia *i*, los valores de *k* y *l* son las posiciones de los rasgos que se están comparando, $e_{ak}(i)$ y $e_{al}(i)$ son las energías asociadas a las posiciones *k* y *l* respectivamente. La función *Z*(*X*) representa la normalización *Z-score*. La ecuación II.3 define de manera general la normalización *Z-score* de una variable *X*.

$$d(k,l) = \sum_{i=1}^{n} \left(Z\left(k * e_{ak}(i)\right) - Z\left(l * e_{al}(i)\right) \right)^2 \qquad \textbf{II.2}$$

$$Z(X) = \frac{X - \overline{X}}{S_X} \qquad \textbf{II.3}$$

Aquí $\overline{X}$ es la media de X y S_X es la desviación estándar. Basándose en la ecuación anterior, se puede considerar entonces $k*e_{ak}= X_1$, $l*e_{al}=X_2$, y de esta manera se puede ver la ecuación II.2 como el cuadrado de la distancia Euclidiana.

II.1.1.2. Medida de distancia entre casos

Aquí se focaliza el problema a través de la elaboración de una medida de semejanza, la cual tiene que ver con cuán parecidas son dos secuencias, justamente el otro punto de vista.

Como el problema que se trata aquí tiene que ver con secuencias de aminoácidos, esta medida también está basada en ellos, pero con una condición exterior: que se disponga de una secuencia de referencia, a veces denominada secuencia salvaje, a partir de la cual se obtienen las mutaciones de cada una de las secuencias. Esta consideración es válida cuando se está tratando con secuencias que mutan frecuentemente. Dada una secuencia, serán mutaciones aquellas posiciones que no coincidan con las de la secuencia de referencia.

Sea $R = (r_1,r_2,...r_N)$ el patrón de referencia, y $X = (x_1,x_2,...,x_N)$, $Y = (y_1,y_2,...,y_N)$ las secuencias que serán comparadas; la medida de semejanza $S(X,Y)$ está definida según la ecuación II.4. Las posiciones de las secuencias R, X y Y son representadas con las energías de contacto.

$$S(X,Y) = \frac{\sum_{i=1}^{N} similarity(x_i, y_i)}{\left| M(X) \cup M(Y) \right|} \qquad \textbf{II.4}$$

Tomando en consideración que $M(X)$ representa el conjunto de atributos mutados en la secuencia X con respecto a R, el denominador de la ecuación II.4 representa la cardinalidad de la unión de conjuntos de atributos mutados, que es el número total de atributos que han mutado en X o en Y. La función $similarity(x_i,y_i)$ del numerador

representa la semejanza entre rasgos de secuencias diferentes y está definida en la ecuación II.5.

$$similarity(x_i, y_i) = \begin{cases} 0 & si\ ((x_i = r_i) \vee (y_i = r_i)) \wedge (x_i \neq y_i) \\ w_1 & si\ (x_i \neq r_i) \wedge (y_i \neq r_i) \wedge (x_i = y_i) \\ w_2 & si\ (x_i \neq r_i) \wedge (y_i \neq r_i) \wedge (x_i \neq y_i) \\ & \quad \wedge \operatorname{sgn}(r_i - x_i) = \operatorname{sgn}(r_i - y_i) \\ w_3 & si\ (x_i \neq r_i) \wedge (y_i \neq r_i) \wedge (x_i \neq y_i) \\ & \quad \wedge \operatorname{sgn}(r_i - x_i) \neq \operatorname{sgn}(r_i - y_i) \end{cases} \qquad \textbf{II.5}$$

Existen diferentes valores w_i para distintas situaciones relativas a la secuencia de referencia. Se recomienda $w_1 > w_2 > w_3$, de tal manera que el mayor valor sea asignado si los rasgos tienen la misma mutación en ambas secuencias (w_1); un valor intermedio (w_2) cuando los atributos tienen diferentes mutaciones, pero el cambio de energía ocurre en el mismo sentido; y finalmente el menor valor (w_3) será asignado a aquellas posiciones que han mutado produciendo cambios de energía en sentidos contrarios. Hay un valor nulo para aquellas parejas que no tienen mutaciones. Si las secuencias son iguales *similarity* toma valor N, donde N es el tamaño de la secuencia. Es válido aclarar que $sgn(X)$ representa la función signo. A la medida de semejanza $S(X,Y)$ se le denominó *WeightedMutation*.

Después de definidas estas medidas de semejanza se pasa a definir un modo de usarlas para extracción de rasgos y para ello se hace necesario precisar, como tal el método de extracción.

II.1.2. Métodos de extracción de rasgos

En el primer caso, o sea, la medida de semejanza entre rasgos, se usará un algoritmo de agrupamiento, y los pasos de este algoritmo de extracción de rasgos se concretan como sigue:

1. Dada la matriz de datos M, donde las filas representan las secuencias y las columnas los rasgos que las describen, usar k-medias (u otro algoritmo de agrupamiento) con la traspuesta de esta matriz, M^T, para k= 1,2,..p-1. Donde k es el número de grupos y p es el número de rasgos.
2. Chequear la fiabilidad del agrupamiento. Calcular el subconjunto S_K de número de grupos confiables.

3. Construir nuevos rasgos y una nueva matriz de datos.
4. Correr un algoritmo supervisado (por ejemplo SVM) para cada $sk_i \in S_K$.
5. Chequear la validez de los resultados.

En el segundo de estos pasos, la forma de chequear la fiabilidad del agrupamiento depende de la distribución de los elementos en los grupos. En otras palabras, si un elemento está muy lejos del centro de su grupo esto podría significar que no se han usado suficientes grupos para agrupar los datos, o sea, estos elementos pudieran definir otro grupo. Para chequear la fiabilidad entonces se define otra medida que desde ahora será llamada "distancia máxima al centro" (*Maximum Center Distanc*e, MCD), definida según la ecuación II.6.

$$MCD = \max(CD(f_i)) \qquad \textbf{II.6}$$

donde F es el conjunto de rasgos que se desea agrupar. Para cada $f_i \in F$, $CD(f_i)$ representa la distancia de un rasgo f_i al centro del grupo correspondiente. Un valor de MCD muy grande no es conveniente, porque esto puede significar que al menos uno de los grupos contiene algunos puntos que están localizados muy lejos del centro.

Si se ve la extracción de rasgos como la búsqueda de una matriz (C) que transformará la matriz de datos (M) en otra matriz (U), según la ecuación II.7, entonces el paso anterior se basó en encontrar la matriz C y el paso siguiente se basará en el cálculo de la matriz U. Para un k dado se obtiene un agrupamiento con k grupos.

$$U^T = C \times M^T \qquad \textbf{II.7}$$

En general los elementos c_{ij} de la matriz C representan la pertenencia del rasgo j al grupo i (K_i), como muestra la ecuación II.8.

$$c_{ij} = \begin{cases} 1 & si \quad f_j \in K_i \\ 0 & en\ otro\ caso \end{cases} \qquad \textbf{II.8}$$

S_K es el conjunto de todos los agrupamientos seleccionados en el paso anterior. Para cada $sk_i \in S_K$ se construye una nueva base de datos. La base de datos i tiene sk_i rasgos calculados desde la matriz U definida en la ecuación II.7. Para cada sk_i, una matriz C_{pxki} es representada para calcular la matriz U^T, donde U representa la base de datos con los nuevos rasgos. Esta base de datos tendrá como rasgos los grupos pertenecientes a C. Cada rasgo representa un grupo y se calcula como la suma de todos los rasgos agrupados en él.

Después de calculada la matriz *U*, se entrena la base con un algoritmo de clasificación (por ejemplo SVM) y en el último paso, se chequea el rendimiento del clasificador. Para esto se recomienda el uso de alguna técnica de entrenamiento y prueba. Aquí se usará validación cruzada con 10 subconjuntos y como criterio de medida el AUC.

Por otro lado la extracción de rasgos basada en la semejanza entre objetos, tiene como objetivo la búsqueda de una medida que permita definir en estos problemas cuán lejos o cerca están dos secuencias según sus mutaciones. Para esto se seleccionó como método de extracción de rasgos el MDS (descrito en el epígrafe 1.2.2), que permite abstraer los rasgos a menores dimensiones a partir de una medida de distancia, en aras de visualizar los casos y hacer una selección previa.

II.2. Modelo de redes neuronales recurrentes para clasificación de secuencias

Existen varias topologías para redes recurrentes que se usan en la literatura para resolver diferentes problemas. Inspirados en el modelo de red recurrente de Baldi [29], se decidió el uso de redes bidireccionales dinámicas, como otro método de aprendizaje para resolver también problemas de clasificación simple. Esta topología desarrollada por Baldi, consiste de dos bloques de contexto, uno con recurrencia al pasado y otro con recurrencia al futuro. En cada tiempo *t*, entiéndase como pasado aquellas capas que tienen conexiones que dependen de tiempos menores que *t* y como futuro, aquellas que dependen de tiempos mayores que *t*.

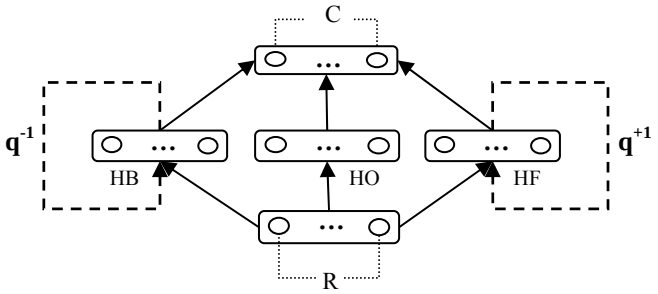


Figura II.1 Topología de red recurrente bidireccional

En la Figura II.1 se muestra un ejemplo de esta topología, donde *R* es la cantidad de rasgos de entrada. Téngase en cuenta, que en la figura, cada una de las flechas que van de capa a capa representan conexión de todas las neuronas de la capa origen con todas las

neuronas de la capa destino. Las flechas discontinuas representan las conexiones en el tiempo; tal y como se describió en el capítulo anterior el operador de desplazamiento q^{+1} significa que la conexión viene desde un tiempo inmediato anterior, y el operador q^{-1} significa que la conexión viene del tiempo inmediato posterior. Justamente la bidirección está dada porque tiene recurrencia en los dos sentidos: en el tiempo t, tiene conexión hacia el tiempo t+1 (por el operador q^{+1}) y hacia el tiempo t-1 (por el operador q^{-1}).

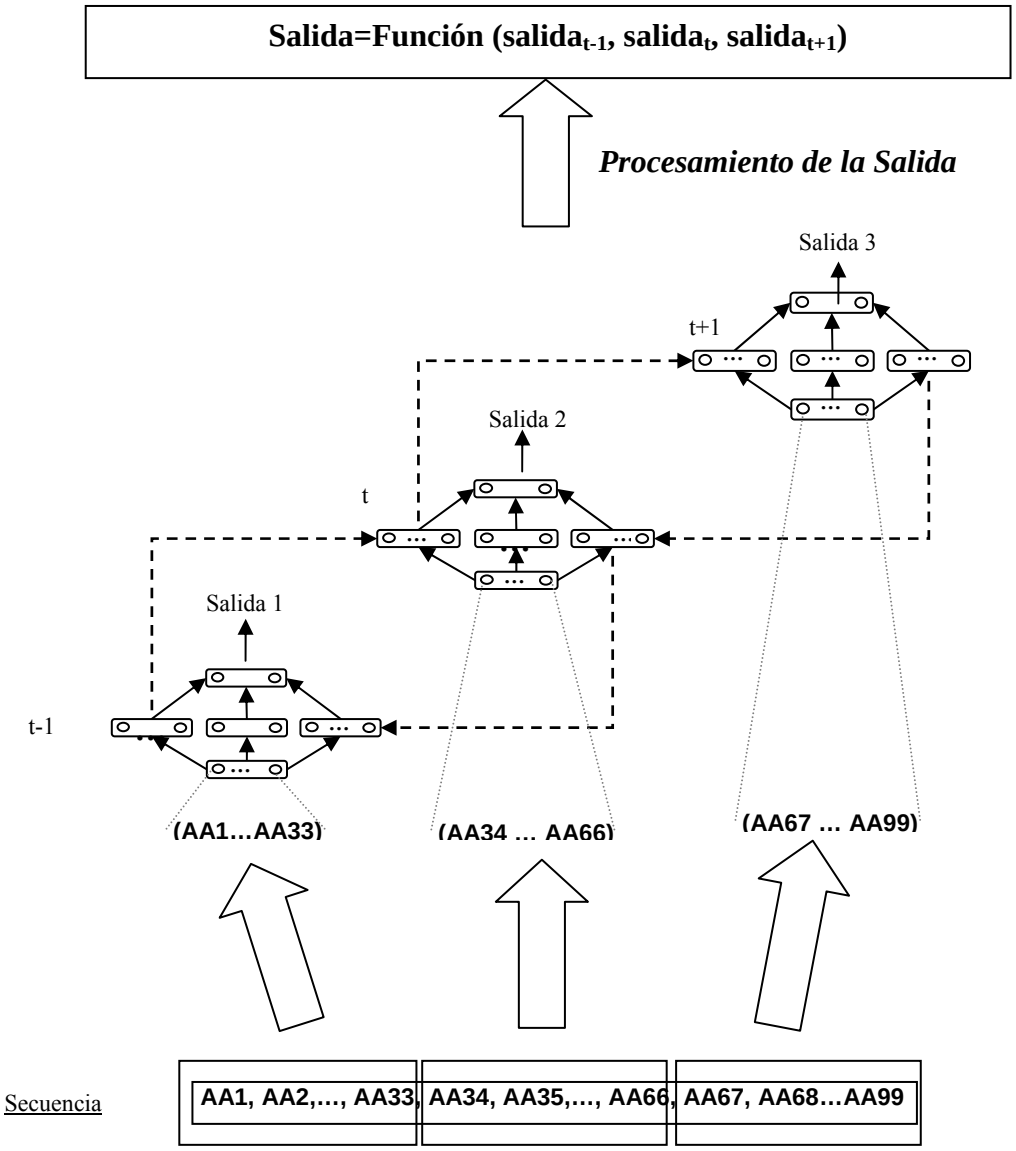


Figura II.2 Despliegue de la red en tres tiempos para un problema de clasificación de secuencia, donde la clase es simple

Esto puede verse más claro cuando la red se despliega en el tiempo, como se muestra en la Figura II.2, en este caso desplegada en tres tiempos. Véase que el tiempo t=2, depende del t=1 y del t=3 a la vez. En el caso t=1, que no tiene tiempo anterior, las neuronas de la parte derecha se inicializan en cero y ocurre igual en el caso de t=3, pero para las neuronas de la capa de la izquierda, pues no tiene tiempo futuro.

En la Figura II.1 se muestra una red que tiene R neuronas de entrada, C neuronas de salida y varias neuronas en las tres capas ocultas: capa de conexión hacia delante (*HF*), capa de conexión hacia atrás (*HB*) y capa de conexión con la salida (*HO*). Suponiendo un problema con dos clases con dos neuronas de salidas, los posibles valores serían (0,1) ó (1,0), significando positivo y negativo respectivamente.

En los problemas de clasificación se tiene una única salida, no contamos con salidas asociadas a tiempos, por eso no es típico resolver estos problemas con redes recurrentes. Sin embargo, sí se puede dividir la secuencia en subsecuencias que representen tiempos y asignar a cada tiempo la misma salida. Si se van a representar T tiempos entonces la secuencia se debe dividir en T partes, cada una asociada a un tiempo. En la Figura II.2 se muestra un ejemplo de una secuencia de entrada dividida en tres partes, una para cada tiempo.

Después de definir la entrada de los rasgos que describen la secuencia, resta definir cuál es la salida final de la red. Como se muestra en el ejemplo, la red tiene tantos valores de salida como tiempos, o sea, la salida es un vector de varios componentes (en el ejemplo, 3). La forma de representar los casos de la base de entrenamiento es la misma, y el algoritmo de entrenamiento no se altera pues la comparación de la salida de cada tiempo se compara con la salida deseada que es única. Pero dado un nuevo caso, en la predicción, se pueden obtener salidas diferentes para cada tiempo, éstas puede ser cualquier combinación de positivos y negativos, y entonces, ¿cuál es la salida final?

Pueden existir diferentes variantes para llegar a una única salida a partir del vector de éstas. En este trabajo se van a tener en cuenta tres de ellas. La más fácil de concebir es seleccionar, de todas las salidas obtenidas, aquella que tenga máxima distribución de probabilidad, analizando las salidas en su variante no procesada, como el vector de probabilidades asociadas a cada valor de clase. Otra variante de salida es el promedio de las salidas y una tercera variante puede ser basada en las salidas procesadas por tiempo,

calculando la moda de ellas. En el caso de la primera variante, se está seleccionando la salida del tiempo que fue más seguro al dar la respuesta de salida. En el segundo caso se da un resultado promedio. En la tercera variante se está ofertando el valor que fue más frecuente en los diferentes tiempos, lo cual le da igual peso a todas las partes de la secuencia. Además de estas tres, se puede analizar cada una de las salidas por separado, si se conoce que para un problema en particular una parte de la secuencia tiene mayor información, se puede seleccionar la salida asociada al tiempo que representa esta parte.

En la Figura II.2 se puede ver un ejemplo de la topología de la red que se propone, para tres tiempos, en su fase de despliegue, donde se puede observar con claridad como el tiempo 2 depende del 1 y del 3 para dar su salida. La tercera variante sólo puede tenerse en cuenta si la cantidad de tiempos es impar.

La topología de este tipo de red es equivalente al procesamiento de las redes MLP, pues como se ve se tienen varias capas de neuronas ocultas, para las que hay que definir la cantidad de neuronas asociadas a cada una.

Finalmente en este trabajo se usa la topología descrita de red recurrente bidireccional como muestran las figuras II.1 y II.2. Como algoritmo de entrenamiento se usó el BPTT descrito en el capítulo anterior.

II.3. Implementación de las redes neuronales recurrentes para clasificación

En esta extensión de redes neuronales recurrentes el cambio esencial es la forma de procesar la salida final para un nuevo caso. Teniendo en cuenta esto, la implementación de la misma es muy similar a las clásicas. En definitiva, las redes recurrentes son siempre algoritmos de entrenamiento y de predicción, como cualquier método de clasificación; pero se ven sutilmente modificados por los criterios de aprendizaje y predicción de la salida final.

En el caso que nos ocupa, se ha seleccionado BPTT como algoritmo de entrenamiento. Para no alterar el algoritmo, se varía el rasgo objetivo en las secuencias de entrada, replicándolo tantas veces como tiempos se analicen, de forma que la comparación de la salida obtenida por la red en cada tiempo se produzca contra la misma salida real. El algoritmo de predicción implementa un módulo adicional de transformación o

combinación de salidas que se basa en una función seleccionada. Como se dijo anteriormente, esta función puede ser la de máxima probabilidad, el promedio, la moda o la salida de un tiempo determinado, esta última puede ser útil cuando se cuenta con información adicional del problema sobre qué parte de la secuencia puede tener mayor peso en la obtención de la salida (por ejemplo el tiempo intermedio).

Una vez discutido el método, sólo resta implementarlo. Para ello se hace uso de la plataforma de aprendizaje WEKA, implementada en la Universidad de Weikato de Nueva Zelanda. Ella brinda un gran número de facilidades para incorporar nuevos modelos de clasificación y facilita la comparación con otros. El WEKA está desarrollado en Java, y está estructurado de forma tal que se hace muy sencillo hacer cambios en el código. Por las características del trabajo que se está desarrollando, sólo se hará referencia a lo concerniente a la adición de un nuevo modelo de clasificación.

En este sistema existe una clase abstracta que implementa los métodos que debe usar cualquier clasificador, y que es denominada *weka.classifiers.Classifier*. Si se desea implementar un nuevo modelo de clasificación se deben redefinir el método *buildClassifier*(), y al menos uno entre los métodos *clasifyInstance*() y *distributionForInstance*().

El diseño de la RNN se dividió en dos paquetes:

Topology: agrupa las clases para la creación de la topología de la red. Se tuvo en cuenta una estructura para las capas de salidas que define si la red se va a utilizar para clasificación (Anexo 7).

Functions: Tiene una interfaz, *NeuralFunction*, que define cómo deben definirse las clases que describan la forma de combinación de salidas. Comprende además algunas clases que pueden ser aplicadas en redes de clasificación para procesamiento de la salida (Anexo 8).

Estos paquetes se añadieron al Weka al igual que la clase principal *RecurrentNeuralNetwork*, que en particular se incluyeron en el paquete *weka.classifiers.functions*.

La forma en que fueron diseñados los paquetes de *Functions* y *Topology* permite que se creen fácilmente nuevas topologías o combinaciones de salida. En el caso de las funciones de combinaciones de las salidas sólo se debe definir una clase que implemente

la interfaz *NeuralFunction* y en el caso de la topología haciendo uso de la clase abstracta *Topology* se puede crear una nueva forma de combinar las salidas. En el Anexo 9 se puede ver un ejemplo de cómo se implementa la función *average*( ) y la implementación de una red recurrente estándar.

Para la creación de la topología de la red se debe definir el fichero del cual se va a tomar la información de la misma. En caso de no definirse en un fichero, se crea una topología por defecto: bidireccional, que tiene tres capas ocultas de 20 neuronas cada una, de forma que cualquiera de ellas tiene una conexión hacia un tiempo posterior *t+1* y otra a un tiempo anterior *t-1*, como se mostró en la Figura II.1.

En el Anexo 10 se puede ver la sintaxis para crear el fichero que describe la topología y un ejemplo del mismo.

II.4. Validación de las redes neuronales recurrentes para clasificación

Después de implementada la red recurrente, se hace necesario comprobar si con este método se pueden obtener buenos resultados en problemas de clasificación de secuencias, donde hay dependencias complejas entre rasgos. ¿Cuándo usar estas redes y cuán superiores pueden ser con respecto a otros métodos? Para dar respuesta a esta incógnita, en este epígrafe se comparan estas redes con otros métodos de clasificación en diferentes problemas.

Relacionado con la hipótesis 2 del trabajo se demostrará primero, en general, que:

1. Las redes neuronales recurrentes son buenos métodos para resolver problemas de clasificación que tienen gran relación entre los rasgos.
2. Las redes neuronales recurrentes son buenos métodos para resolver problemas de clasificación donde la separación de las clases está dada por una región compleja.

Las redes recurrentes que presentamos aquí, al igual que las redes neuronales MLP (tal vez las más universalmente eficientes) tienen como desventaja que la búsqueda de una topología idónea y el algoritmo de entrenamiento son muy costosos en tiempo, porque muchas veces exigen de experimentación, orientada apenas por algunas heurísticas. Ello explica por qué se evita frecuentemente su uso si se pueden tener resultados satisfactorios por otras vías. En las tareas bioinformáticas, como las que motivan la presente tesis, los

problemas de clasificación muestran gran relación entre los rasgos, y simultáneamente la separación de las clases en regiones complejas. La verificación de estas hipótesis avalaría entonces el uso de las redes recurrentes en el análisis de secuencias genómicas.

Esta verificación se realizará mediante un experimento o cuasiexperimento [122]. Para su desarrollo se construye una serie de conjuntos de bases de datos artificiales al azar y donde se manejan 3 factores: la relación entre los rasgos, la dirección de esta relación y la forma de obtener la clase.

La relación entre los rasgos modela la posible dependencia entre los atributos aleatorios a través de funciones lineales, polinomiales o por ramas, según las ecuaciones II.9, II.10 y II.11 respectivamente.

$$f(X) = \frac{\sum_{i=1}^{N} a_i x_i + c}{\sum_{i=1}^{N} a_i + c} \qquad \textbf{II.9}$$

teniendo en cuenta que $X=(x_1, x_2, ..., x_N)$ es el vector de rasgos de un caso dado con dimensión N. En la función lineal, a_i es el coeficiente asociado al rasgo i, y c es el término independiente. Esta función está normalizada con la división por el máximo valor que puede obtenerse, ya que los rasgos fueron generados en el intervalo [0,1].

En la ecuación II.10 se añade un exponente $b_i \in [1,10]$, lo cual convierte esta función en polinomial. Esta función también está normalizada con la división por el máximo valor que puede obtenerse, que convierte esta función en racional, o más generalmente, irracional si los coeficientes o los exponentes no fueran enteros (este último caso no es necesario en la práctica pues los exponentes pueden aproximarse racionalmente).

$$f(X) = \frac{\sum_{i=1}^{N} a_i x_i^{b_i} + c}{\sum_{i=1}^{N} a_i + c} \qquad \textbf{II.10}$$

Por otro lado, la función por ramas (ecuación II.11) se define a partir de las funciones polinomiales: $h(X)$ y $g_i(X)$ $i=1,2,..R$, donde R representa la cantidad de ramas (generada de forma aleatoria en el intervalo [5,15]). Para esto se fija una partición aleatoria del intervalo [0,1]: $U=(u_0,u_1,....,u_R)$, con $u_0=0$, $u_R=1$ y $\forall i \in [1,R]$: $u_{i-1} < u_i$. Así, para cada

intervalo $u_{i-1} < u_i$ se genera una función polinomial $g_i(X)$. El valor de la función $h(X)$ será la que determinará el intervalo, y de esta manera la función $g_i(X)$ a usar.

$$f(X) = g_i(X) \quad si \quad u_{i-1} \le h(X) < u_i, \quad i = 1,2,\ldots,R \qquad \textbf{II.11}$$

Se generó además un primer conjunto de bases creando los rasgos aleatorios de manera independiente, o sea, sin rasgos dependientes, que sirviera también para la comparación a manera de "grupo de control".

Para establecer el sentido de la relación entre los rasgos se tuvo en cuenta la dirección de la dependencia: hacia delante, hacia atrás y en ambas direcciones. Ello determina un segundo factor a tener en cuenta; la dirección, o más precisamente el sentido de la dependencia entre los rasgos. Por otro lado, la obtención de la clase se puede generar, a partir de los rasgos, usando las mismas tres alternativas y funciones mencionadas anteriormente: lineal, polinómica o por ramas.

Se generaron conjuntos con todas las combinaciones de estos tres factores. De esta manera se obtuvieron diez (3*3+1) conjuntos de bases de casos. En total, se generaron 285 bases de casos, ya que, teniendo en cuenta la obtención de la clase, se generaron 95 bases a partir de una función lineal, 95 a partir de una función polinomial y 95 a partir de una función por ramas. Todas estas bases se generaron con nueve rasgos y con clases dicotómicas.

La red recurrente se entrenó con tres topologías diferentes, basadas en el modelo propuesto en el epígrafe anterior. En aras de simplificar la topología se seleccionó la misma cantidad de neuronas para cada capa oculta y tres tiempos. Se crearon 4 topologías para cada base, con 4, 6, 8 y 10 neuronas en las capas ocultas. Se usaron seis funciones de salida: promedio de las salidas, moda, máxima probabilidad y cada una de las salidas asociadas a los tiempos (salida del tiempo 1, salida del tiempo 2 y salida del tiempo 3).

Muchas otras combinaciones de los factores en el experimento pudieran haber sido posibles, pero ellas podrían incrementar la presencia de diferencias "no honestamente significativas" que lejos de arrojar luz, podrían mostrar "oscuridad". Las restricciones de los parámetros anteriores parecen adecuadas a los fines de verificar lo deseado.

Para comparar los resultados se entrenaron: un clasificador J48, una red bayesiana, 10 SVMs con función núcleo polinomial, desde grado 1 (en este caso función núcleo lineal) hasta grado 10, 10 MLPs con 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 neuronas.

Se utilizó validación cruzada con 10 subconjuntos para cada una de las bases, con cada uno de los métodos seleccionados. Se seleccionaron como medidas de comparación la exactitud y el AUC. La primera porque da el porcentaje de clasificación correcta que en este caso, es una buena medida, debido a que las bases están balanceadas según la clase. Se escoge además AUC porque da una medida de balance de los resultados por clases y según el problema, ello puede ser necesario. Se aplicaron pruebas estadísticas a los resultados del experimento. Primero se representaron los datos con ayuda de cubos OLAP para analizar mejor los resultados y compararlos según los diferentes factores (Ver Anexo 11). Se completó este análisis con pruebas estadísticas no paramétricas para datos correlacionados.

Para una mejor comprensión se verán primero los resultados obtenidos con la exactitud, como medida de validación. A simple vista, se evidenció que las redes recurrentes no son el mejor clasificador cuando la obtención de la clase está basada en una función lineal. En este caso otros métodos como el SVM y el MLP obtienen mejores resultados o al menos similares a los de la red recurrente. Esto era esperado, ya que SVM y MLP tienen gran potencialidad para encontrar semiplanos en el espacio y tienen la ventaja de ser computacionalmente menos costosos que la red recurrente. Por tanto, en este caso no es recomendable el uso de redes recurrentes.

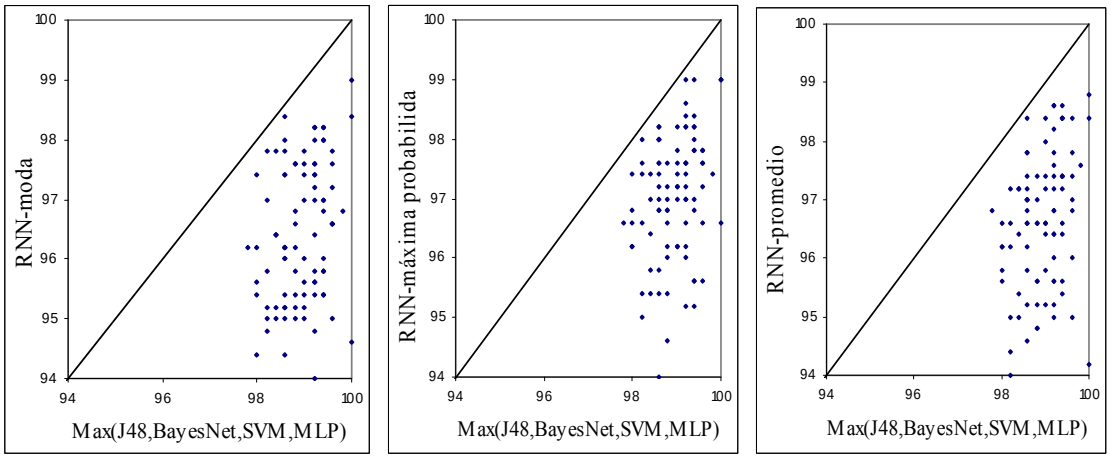


Figura II.3 Comparación de los resultados de exactitud de la mejor variante de red recurrente con las diferentes combinaciones de salidas: moda, máxima probabilidad y promedio, contra el mejor resultado de los obtenidos por J48, la red bayesiana, la mejor variante de SVM y la mejor variante de MLP.

En la Figura II.3 se muestra la comparación de los resultados de exactitud de las redes recurrentes, en cada una de sus variantes de salida, contra el mejor de los resultados, del resto de los clasificadores, en el caso lineal. Cada punto representa una de las bases de datos, teniendo como coordenadas por el eje vertical, la exactitud de la red recurrente, y por el eje de horizontal, el mejor valor del resto de los clasificadores usados. Entonces un punto por encima de la recta significa que el resultado para la red recurrente es mejor. Aunque la comparación no parece justa para la RNN, la comparación contra el MLP tiene resultados similares. Se puede ver que todos los puntos quedan por debajo de la recta, por tanto en ningún caso, donde la separación de la clase es lineal, la red recurrente es superior.

Por otro lado, cuando la obtención de la clase se realiza basada en una función polinomial o por ramas la situación es diferente, las redes recurrentes pueden superar a estos otros métodos, en dependencia del tipo de combinación de salidas que se utilice en la red.

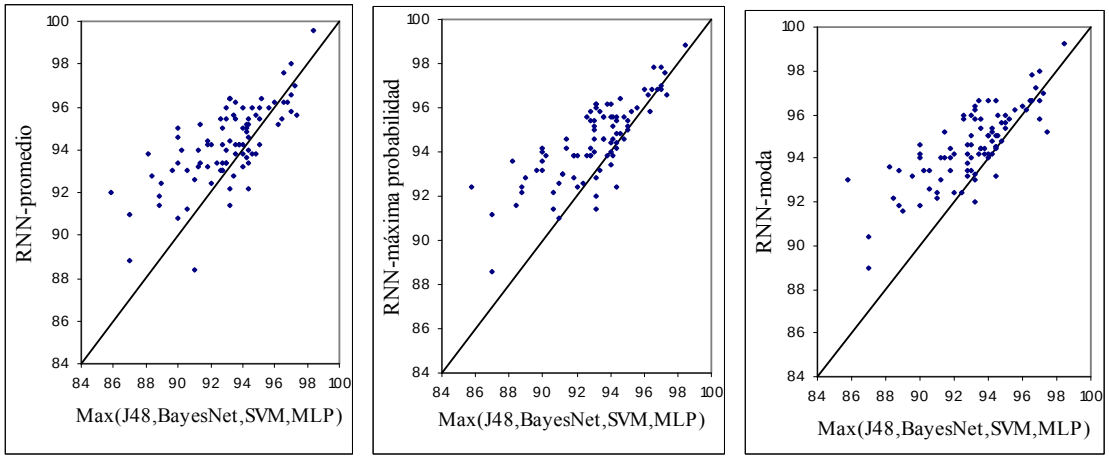


Figura II.4 Comparación de resultados de la exactitud de los clasificadores J48, el mejor de los SVM, la red bayesiana y el mejor de los MLP, para cada una de las bases con relación entre los rasgos y obtención de la clase a partir de una función polinomial.

La Figura II.4 muestra la comparación del mejor de los resultados de los clasificadores J48, SVM, MLP y red bayesiana contra la exactitud obtenida por las tres variantes de redes recurrentes para las bases con relaciones entre los rasgos y función polinomial en la obtención de la clase. Se puede ver a simple vista que incluso, en esta comparación no justa para la red recurrente, ella es superior, resultando además, que la combinación de salidas utilizando la moda es la de mejores resultados.

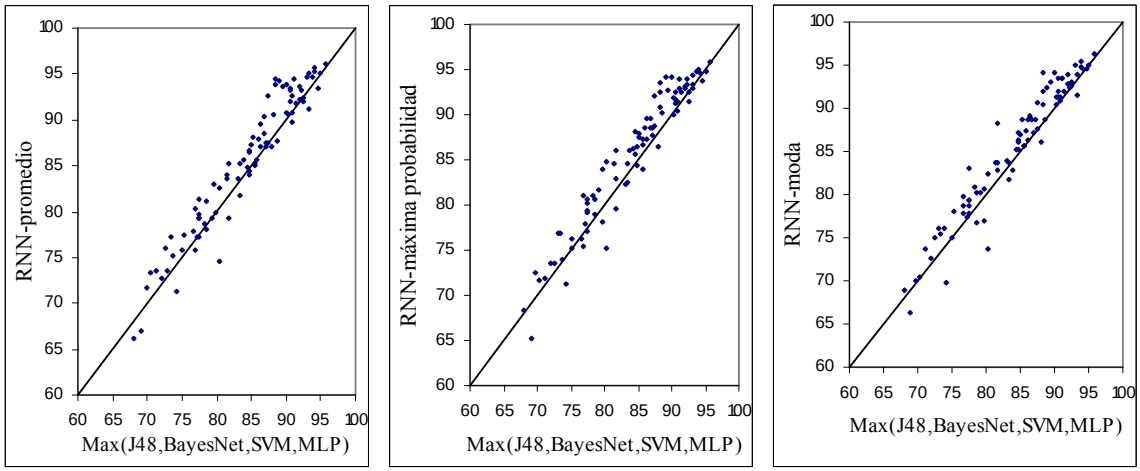


Figura II.5 Comparación de resultados de la exactitud de los clasificadores J48, el mejor de los SVM, la red bayesiana y el mejor de los MLP, para cada una de las bases con relación entre los rasgos y obtención de la clase a partir de una función por ramas.

De igual manera en la Figura II.5 se muestran estos resultados pero para bases donde la obtención de la clase se realiza a partir de una función por ramas y también puede verse a simple vista la superioridad de las redes recurrentes con los tres tipos de combinaciones.

Para analizar detenidamente estos resultados se realizaron pruebas estadísticas no paramétricas que comparan los resultados obtenidos por los diferentes métodos para las mismas bases, o sea, pruebas para *k* muestras relacionadas, específicamente el *two-way Anova* de Friedman. Con este objetivo se agruparon las bases en conjuntos, primero según la relación entre los rasgos y luego según la dirección de ésta (si la relación era hacia delante, hacia atrás o en los dos sentidos).

Cuando no existe relación entre los rasgos los resultados son similares a los obtenidos por MLP, lo cual significa que las RNN no son eficientes en estos casos, pues un MLP puede ser más simple para obtener el resultado esperado. Pero por otro lado, cuando existe relación entre los rasgos, ya sea polinomial o por ramas los resultados son significativamente diferentes en algunas redes recurrentes.

Como puede verse en la Figura II.6 la prueba de Friedman arroja que existen diferencias significativas entre los métodos comparados. Si se organizan los mismos según el Rango Medio (*Mean Rank*) se puede ver que entre los métodos J48, Max-SVM (Valor máximo de los 10 SVMs), red bayesiana y Max-MLP (Valor máximo de los 10 MLPs), este último es el que mayor rango medio tiene. Debido a esto, si existen diferencias

significativas con el MLP entonces podríamos decir que existen diferencias significativas con todos los demás.

Ranks[a]

	Mean Rank
J48	2.75
Max-SVM	1.85
Red Bayesiana	1.55
Max-MLP	4.57
Max-RNN-average	7.78
Max-RNN-maxprob	8.45
Max-RNN-mode	8.38
Max-RNN-left	6.60
Max-RNN-middle	6.87
Max-RNN-right	6.20

a. Relación entre Rasgos = Lineal

Test Statistics[a,b]

N			30
Chi-Square			209.658
df			9
Asymp. Sig.			.000
Monte Carlo Sig.	Sig.		.000
	99% Confidence Interval	Lower Bound	.000
		Upper Bound	.000

a. Friedman Test

b. Relación entre Rasgos = Lineal

Figura II.6 Prueba de Friedman para casos donde la obtención de la clase se basa en una función polinomial y la relación entre algunos rasgos es lineal.

Cuando la relación entre los rasgos es polinomial o por ramas sucede algo similar, las pruebas de Friedman sugieren que existen diferencias significativas entre los métodos y es el MLP el que mayor rango tiene si se compara con los diferentes métodos de las RNN.

En la Tabla II.1 se muestran los resultados de la prueba de Wilcoxon para comparar la red recurrente, con la moda como función para combinar las salidas, contra MLP, donde cada una representa la mejor topología, o sea, el máximo valor de exactitud obtenido. Cuando el valor de significación está por debajo de 0.05 se considera que hay diferencias significativas y por debajo de 0.10 diferencias medianamente significativas, en estos casos se usó un signo “>” para representar que la superioridad está a favor de la red recurrente y un signo “<” para cuando la superioridad está a favor del MLP. Por las columnas se representa la relación entre clases y por las filas la relación entre rasgos. La última columna representa la significación total con respecto a la relación entre rasgos y la última fila es la significación total con respecto a la relación entre clases.

Si se analizan primero los casos por relación entre rasgos, puede verse claramente que no existen diferencias entre el MLP y la RNN. Cuando la relación entre los rasgos es polinomial la RNN es ligeramente superior y cuando la relación es lineal o por ramas la

RNN es significativamente superior. Pero estos resultados están condicionados por la separación entre las clases. Puede verse que cuando la separación entre las clases es lineal MLP es significativamente superior a la RNN. Sin embargo, cuando la separación entre las clases es polinomial o por ramas hay diferencias significativas a favor de la RNN. Si particularizamos en dependencia del tipo de relación entre rasgos, puede verse que siempre que hay relación entre los rasgos y la separación de las clases es polinomial o por ramas las RNN son significativamente superiores.

Tabla II.1 Resultados de la prueba de Wilcoxon para la comparación de RNN, con la función moda, contra MLP según la relación entre rasgos y la separación entre clases.

Relación entre rasgos	***Separación entre clases***			**Total**
	Lineal	Polinomial	Por ramas	
No relación	$0.061^{<}$	0.128	0.811	0.813
Lineal	$0.000^{<}$	$0.000^{>}$	$0.000^{>}$	$0.008^{>}$
Polinomial	$0.000^{<}$	$0.003^{>}$	$0.038^{>}$	$0.094^{>}$
Por ramas	$0.000^{<}$	$0.000^{>}$	$0.000^{>}$	$0.018^{>}$
Total	$0.000^{<}$	$0.000^{>}$	$0.000^{>}$	$0.021^{>}$

El Anexo 12 muestra los resultados para el caso de RNN con las funciones de combinación de salida: máxima probabilidad y promedio. Resulta que las redes recurrentes con combinación de salidas usando la moda, obtienen resultados significativamente superiores al resto de los métodos analizados.

Este mismo proceso se siguió para analizar los resultados del AUC. En esta medida no tiene sentido utilizar la combinación de la moda, ya que necesitamos valores reales como respuesta final de salida. Pero satisfactoriamente se obtuvo que la combinación usando el promedio de las salidas, es significativamente diferente para los mismos casos analizados anteriormente, o sea, cuando existe relación entre los rasgos y la separación de las clases no es lineal.

Por otro lado, teniendo en cuenta la dirección de la relación los resultados son similares si la dirección es hacia delante, hacia atrás o en las dos direcciones. Tanto para el caso en que la clase se obtiene por polinomios o por ramas, la red recurrente (con salida obtenida

por máxima probabilidad o por la moda) es significativamente superior al resto de los métodos. Este resultado no depende de si las relaciones de los rasgos son hacia delante, hacia atrás o en las dos direcciones.

Así podemos concluir que la red recurrente es un método idóneo, para problemas en que existe relación entre los rasgos y la separación de las clases es compleja (polinomial o por ramas). Específicamente se debe usar red recurrente con una función de combinación de salida a partir de la moda si queremos mejorar la exactitud, mientras que debemos utilizar la función promedio si lo que queremos mejorar es el AUC.

Sobre la base de estos resultados es que se refuerza la hipótesis de que el uso de una red recurrente en problemas de análisis de secuencias en bioinformática puede ser un buen método a aplicar.

II.5. Conclusiones parciales

Se proponen dos nuevas medidas de distancia que tienen interpretación biológica: una que compara los rasgos y otra los casos, aplicables a bases de datos de secuencias de aminoácidos para ser usadas con diversos métodos de extracción de rasgos. La representación visual de los datos, con ayuda de técnicas de reducción de dimensionalidad, puede ayudar a la comparación de varias medidas de distancia y la selección de una adecuada para el problema que se analice.

Se propone un método de redes neuronales recurrentes bidireccionales para usar en problemas de clasificación simple, proponiendo varias funciones para combinar las salidas.

La comparación de este modelo de red con otros algoritmos de clasificación evidencia superioridad a J48, red bayesiana, SVM y MLP si existe relación entre los rasgos y la separación de las clases es compleja (polinomial o por ramas), como frecuentemente ocurre en problemas bioinformáticos.

Capítulo III. MODELO DE COMBINACIÓN DE CLASIFICADORES

El modelo de red recurrente para clasificación, propuesto en el capítulo anterior, se asemeja a los modelos de combinación de clasificadores. Si se ve cada despliegue de la red en el tiempo como un posible clasificador, ya que cada uno brinda una salida, entonces la RNN no es más que un conjunto de clasificadores. De hecho es un método con varias salidas para un mismo problema y un módulo de combinación de salidas, al igual que ocurre con los multiclasificadores. ¿El uso de varios clasificadores, que logren diferentes regiones de decisión, podrá mejorar los resultados en problemas complejos? Justamente el surgimiento de los multiclasificadores fue éste, pero saber si, concretamente, en problemas de bioinformática, pueden lograr mejores resultados es uno de los objetivos de este capítulo.

Se propone un nuevo multiclasificador inspirado en los modelos diseñados hasta el momento. Para esto, a continuación se realiza la definición de cada una de sus partes: la topología, los modelos o clasificadores de base y cómo combinar finalmente las salidas.

Como se ha dicho en el capítulo I, existen dos variantes para hacer un multiclasificador: basada en un único modelo para los clasificadores de base o con diferentes modelos para cada clasificador. Se ha seleccionado esta última variante para realizar un multiclasificador que se apoya en los conjuntos de instancias bien clasificadas. Se realiza una validación de este modelo comparando sus resultados con otros de los más populares para demostrar su efectividad, y se usan para ello bases de datos del repositorio de aprendizaje automatizado de la UCI [123].

III.1. Topología del modelo

En las bases de datos bioinformáticas, por ejemplo bases de mutaciones de secuencias, es típico que exista gran cantidad de casos parecidos con clases diferentes, por lo que es difícil encontrar los patrones que caracterizan cada una de las clases del problema. Dos modelos de clasificación diferentes pueden aprender distintos conjuntos de patrones en la base, por ello la unión de la información de varios modelos puede ser beneficiosa en estos casos. Por supuesto, el éxito de este método, está condicionado por la selección adecuada

de los modelos. Mientras menos correlacionados estén los errores de éstos, se tendrán mejores resultados potenciales [124].

Después de la selección de los métodos de base, el problema está en la combinación de los mismos. Si tenemos en cuenta que cada clasificador, una vez entrenado, va a tener asociado un conjunto de casos que ha logrado aprender, y a los que se llamarán casos duros, entonces una idea para combinar los resultados podría ser aprender qué patrones clasifica bien cada clasificador. Para esto se pueden separar los casos en grupos, según el clasificador al que pertenecen. Suponiendo que se tienen C clasificadores de base entonces estaríamos en presencia de C grupos de casos duros, uno por cada clasificador de base.

La idea de este sistema se basa en las topologías que manejan *stacking* y el modelo de expertos mixtos, ya que se agrega un metaclasificador que aprende los resultados de los clasificadores de base, pero basado sólo en los C grupos de casos duros. Cuando se trata de combinar varios clasificadores el problema está en saber qué clasificador seleccionar ante una nueva instancia. Esto puede ser muy complejo, pues cada clasificador define una región de decisión que separa las clases, pero al aparecer una nueva instancia dos clasificadores diferentes pueden clasificarla en clases distintas. ¿Cuál dice la verdad? Lo ideal es saber qué aprendieron cada uno de estos clasificadores. Es por esto que se deben apartar las instancias que no fueron aprendidas por ningún clasificador, porque de cualquier manera serán errores, y concentrarse en las instancias bien clasificadas. Ésta es justamente la idea que se sigue en este modelo.

En la figura III.1, se muestra un ejemplo de este modelo usando tres clasificadores de base. A partir de una base de entrenamiento se entrenan los clasificadores C_1, C_2, y C_3. Cada clasificador logrará un conjunto de instancias bien clasificadas, que para mejor comprensión se ha señalado de manera concentrada en esta figura. Puede verse entonces cómo se solapan estos conjuntos, por lo que existirán casos que serán bien clasificados por los tres clasificadores, otros que sólo serán clasificados por dos de ellos, otros que serán bien clasificados por uno sólo de ellos y por último un conjunto que será mal clasificado por los tres clasificadores. Este último conjunto puede excluirse del modelo o utilizarlo como un grupo aparte.

¿Cómo decidir que una instancia será bien clasificada? Usualmente, cuando los clasificadores dan un vector de salida continuo, por ejemplo una probabilidad de pertenecer a la clase que él decide, se establece un punto de corte o umbral para decidir cuál es la clase, generalmente es aquella que tenga mayor valor. En el caso de dos clases se estila el uso de 0.5 como punto de corte (o al menos es el valor por defecto). De esta manera se crean los conjuntos con las instancias bien clasificadas por cada uno de los clasificadores.

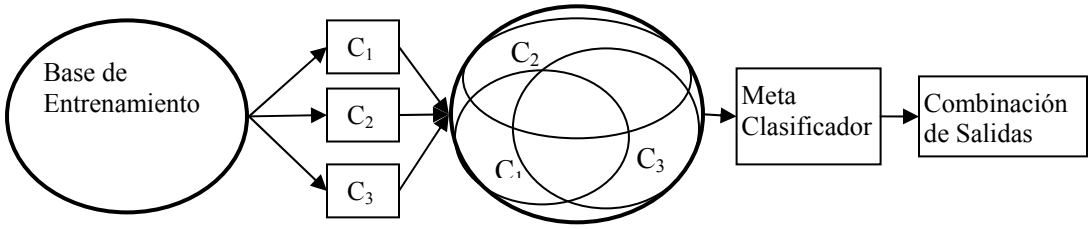


Figura III.1 Modelo multiclasificador basado en conjuntos de instancia bien clasificadas

El siguiente paso, después de entrenados los clasificadores de base, es seleccionar un modelo de clasificación para el metaclasificador, que debe aprender a separar las instancias en estos grupos, o sea, será entrenado con una nueva base que se diseñará a partir de la original. A esta nueva base se adicionarán los casos que conforman cada uno de los grupos de casos duros asociados a los clasificadores de base. Cada caso será descrito por los mismos rasgos descriptivos que tenía en la base original, sólo el rasgo objetivo cambiará, ahora está asociado al conjunto de clasificadores, o sea, cada caso tendrá como clase el grupo al que pertenece.

Como puede verse habrá casos iguales que pertenezcan a conjuntos diferentes, lo que puede ser un problema para que el metaclasificador aprenda. Una posible solución sería hacer conjuntos duros, o sea, a cada caso asociarle el clasificador que "mejor" lo clasifique; pero ello también puede ser un ruido, ya que pudiéramos sobrevalorar un clasificador y subvalorar otro, donde el primero tendrá muchas instancias y el segundo muy pocas, haciendo una base muy desbalanceada y muy difícil de aprender por cualquier modelo de clasificación. Por eso, la decisión es mantener cada caso en el clasificador que lo clasifique bien, aunque se repita en la base con diferentes clases.

En muchas bases de datos en que hay casos bien distinguidos, un alto porciento de ellos pueden ser aprendidos por cualquier clasificador. Pero este conjunto de bien clasificados

por todos, sí puede evitarse, para lo cual creamos un grupo (C+1) que tendrá los ejemplos que son reconocidos por todos los clasificadores usados. También pueden quedar casos que se pueden repetir en los diferentes pares de clasificadores y no en el tercero, pero si las bases y los clasificadores son adecuados, la proporción de ellos debe ser significativamente menor. En cualquier caso, es importante no seleccionar un metaclasificador con clasificación dura, para dar margen a que decida cualquiera de los grupos que la contienen.

Entonces se formarán grupos de casos bien clasificados por cada uno de los clasificadores de base, a los que se les restará el conjunto de instancias que son bien clasificadas por todos. Si esta diferencia entre estos conjuntos produce conjuntos muy pequeños, entonces se tendrán pocas instancias asociadas a los clasificadores que les suceda esto. Por ejemplo, cuando el conjunto de intersección de todos los grupos creados por los clasificadores de base es muy grande es típico que suceda esto y en este caso vamos a dejar para el metaclasificador una base de datos con pocas instancias para las clases, dificultando el aprendizaje. Teniendo en cuenta esto, se estableció un umbral para decidir qué instancias, del conjunto de casos bien clasificados por todos los clasificadores, se pasarán al resto de los conjuntos. Para ello se seleccionan aquellas que son mejor clasificadas por cada uno, o sea, que la probabilidad con que el clasificador da la salida esté por encima del umbral establecido. Este umbral dependerá de la base que se trate, y puede ser un parámetro seleccionado por el usuario en el modelo. Un umbral igual a 0 significa que serán añadidas todas a cada conjunto, o sea, cada clasificador estará representado por el conjunto completo de las instancias que clasifica bien. Un umbral igual a 1 significa que se queda como la idea original, cada clasificador es representado por el conjunto dado por la diferencia entre el conjunto de los casos que él clasifica bien y el conjunto de aquellos que todos clasifican bien.

Formalmente el problema original a resolver tiene como función objetivo FO: $RD \rightarrow O$, donde RD es el conjunto de los rasgos del problema y O={1,2,…,N} es el conjunto de clases. El problema transformado para el metaclasificador quedaría con una función objetivo F: $RD \rightarrow O2$, donde $O2$={1,2,…,C+1} es el conjunto de grupos formados por los C clasificadores y el grupo de las instancias bien clasificadas por todos.

Después de entrenado el metaclasificador sólo resta definir el módulo para calcular la salida general del sistema que se explicará posteriormente. De esta manera queda definida la arquitectura de este multiclasificador. Ahora el problema está en seleccionar los modelos de clasificación para los clasificadores de base y para el metaclasificador.

III.2. Selección de clasificadores de base

La selección de los modelos de clasificadores de base es un gran problema hoy en día. Algunos autores dicen que una forma de hacerlo es seleccionar aquellos, cuyos errores tengan muy baja correlación [124, 125]. Pero ésta es una labor difícil y no comprobada del todo. Muchos autores han propuesto varias formas de medir la diversidad de los clasificadores [90, 126, 127]. No obstante, el presente trabajo no pretende optimizar la búsqueda de los clasificadores, sólo se propone el uso de diferentes paradigmas de clasificación, en términos de distintas formas de buscar las regiones de decisión.

Algunos clasificadores tienen una complejidad computacional muy baja como J48 y SVM, mientras otros, como MLP una complejidad muy alta (por el algoritmo BP). Esto hace que muchas veces el MLP se evite, aunque a decir verdad, la complejidad de este algoritmo radica sólo en el momento de entrenamiento. Cuando se está en presencia de un problema real, puede ser importante no tanto el proceso de entrenamiento, como el proceso de explotación final. En este sentido ya MLP no es problemático, pues después de ajustados los pesos de la red, la salida ante un nuevo patrón de datos es tan rápida como la del J48 y el SVM. Por las características de las investigaciones bioinformáticas, no se prioriza tampoco en este trabajo la minimización del costo computacional, o sea, se decide sacrificar tiempo de entrenamiento en aras de un mejor resultado en explotación. En definitiva, los análisis en muchas tareas de clasificación de secuencias, como la determinación de posible resistencia a un candidato antiviral, la localización de los genes, el pronóstico de cierta propiedad biológica de un compuesto identificado por una secuencia de datos; constituyen a veces sólo el preámbulo *in silico* de una investigación a posteriori *in vitro* infinitamente más costosa en tiempo y dinero.

Se proponen por ejemplo, el uso de un árbol de decisión, una red bayesiana, SVM, y una red neuronal, sólo por mencionar algunos, donde los tres primeros se usen como clasificadores de base y la red neuronal como metaclasificador. Esta decisión puede

cambiar en dependencia del problema. Lo ideal para una base determinada es seleccionar los clasificadores que puedan combinarse mejor, quizás con la primera medida que se menciona, aquellos clasificadores con errores menos correlacionados, o con cualquiera de las medidas propuestas por Kuncheva y Whitaker [90].

III.3. Combinación de salidas

El metaclasificador tendrá como salida un vector de probabilidades en correspondencia con los grupos que se crearon. Para el módulo de combinación de salidas se proponen dos formas de combinarlas: una basada en selección y otra basada en fusión, aunque pudiera usarse cualquiera de las mencionadas en el capítulo I.

La combinación por fusión se basa en pesar las salidas de los clasificadores con la salida del metaclasificador como muestra la ecuación III.1.

$$S = SC * SMC \qquad \textbf{III.1}$$

donde SC es una matriz de dimensiones CxN, siendo C la cantidad de clasificadores y N la cantidad de clases, de manera que SC es la matriz de salida de los clasificadores, o sea, $SC(i,j)$ es la probabilidad de clasificar la clase j por el clasificador i. SMC es el vector columna Nx1 de las probabilidades de salida del metaclasificador, y S es el vector columna Cx1 de probabilidades resultantes del sistema.

La combinación por selección es más simple, consiste en seleccionar el clasificador de mayor probabilidad asignada atendiendo al vector de salida del metaclasificador. Éstas son las dos combinaciones de salidas que se proponen en este trabajo.

Esta propuesta de multiclasificador con varios modelos de clasificadores de base, basado en los casos duros y con funciones de combinación de salida, será denominada en lo sucesivo MEHI (*Muti-Expert by Hard Instances*).

III.4. Comparación con stacking y mezcla de expertos

La variante de combinación de clasificadores que se propone está basada en dos multiclasificadores muy conocidos: *stacking* y mezcla de expertos. Como se explicó en el capítulo I, estos métodos se basan en diferentes modelos de clasificación y usan un metaclasificador diferente para combinar las salidas.

Stacking usa, como base de datos para entrenar al metaclasificador, la salida de los clasificadores de base, o sea, construye una nueva base de casos donde cada caso va a tener como rasgos la salida de los clasificadores y preserva la clase. Por otro lado, MEHI construye también una base de casos para entrenar el metaclasificador, solo que preserva los rasgos de los casos y lo que transforma es el rasgo objetivo, teniendo ahora tantas clases como clasificadores de base se tienen, más una clase que agrupa los casos bien clasificados por todos.

Mezcla de expertos usa un metaclasificador que se entrena con una base de casos parecida a la que se usa en MEHI, solo que va a tener todos los casos de la base original, con el clasificador que dio mayor probabilidad a la clase real del caso, aunque lo haya clasificado mal. MEHI elimina los casos ruidosos que no fueron aprendidos por ningún clasificador, ya que de ninguna forma van a ser bien clasificados y pueden perjudicar el entrenamiento. MEHI puede tener casos repetidos, aquellos que sean bien clasificados por varios clasificadores y no todos de ellos. En dependencia del umbral que se seleccione para decidir si un caso es bien clasificado por un clasificador base o no, este número de casos repetidos puede aumentar o bajar.

III.5. Implementación del modelo de combinación de clasificadores

Este método, al igual que las redes recurrentes, se implementa e incorpora a la herramienta WEKA, la cual tiene varios modelos de multiclasificadores implementados, como *AdaBoost*, *bagging*, *stacking*, entre otros. A pesar de tener varias clases abstractas para la implementación de nuevos multiclasificadores, ninguna tiene utilidad para la forma de combinación de los clasificadores que ahora se propone. Por eso fue necesario la construcción de una clase que heredara directamente de la clase *Classifier*: *MulticlassifierByMetaClassifier*. Así se decidió implementar los dos algoritmos básicos, la construcción del clasificador (*buildClassifier*) y el cálculo de la salida (*distributionForInstance*).

Como parámetros fundamentales de *MulticlassifierByMetaClassifier* se tendrá una lista de clasificadores entre los cuáles el usuario puede escoger, como modelos de base, al inicializar el multiclasificador. Así, el usuario puede seleccionar cualquiera de los

modelos de clasificación que tiene implementado el WEKA además de otros que se propongan y que se incrementen al WEKA en forma consecuente.

Análogamente se trata la definición del metaclasificador; pero el otro parámetro propio de esta clase es el umbral para decidir si una instancia del conjunto de casos bien clasificados por todos los clasificadores será mantenida en el conjunto que representará al clasificador, que por defecto es 1. Con otro parámetro se define la forma de combinación de salidas: selección o fusión.

En el método *buildClassifier*() se construye la lista de clasificadores definidos. Después de entrenados los clasificadores de base, se construye también la nueva base de entrenamiento a partir de los casos duros obtenidos por cada clasificador entrenado. Con esta nueva base se entrena entonces el metaclasificador. Este algoritmo puede definirse por los siguientes pasos, ya explicados anteriormente:

1. Entrenar los clasificadores de base
2. Formación de los conjuntos de casos bien clasificados
3. Detección del conjunto intersección
4. Formación de la nueva base de entrenamiento para el metaclasificador
   - Para cada caso de la base original,
     - Para cada clasificador C_i,
       - Si el caso pertenece a la diferencia del conjunto de bien clasificados y el conjunto intersección, añadir a la base y asociar como clase *i*
       - Si el caso pertenece al conjunto intersección y la probabilidad está por encima del umbral establecido. , añadir a la base y asociar como clase *i*
     - si el caso pertenece al conjunto intersección y no fue añadido con ningún clasificador, añadir a la base y asociar como clase *C*.
5. Entrenar el metaclasificador con la nueva base creada

El método *distributionForInstance*() es el encargado de implementar el módulo para clasificar un caso determinado, el cual puede definirse según los siguientes pasos:

1. Invocar al *distributionForInstance()* del metaclasificador para obtener el vector de probabilidades asociadas a cada clase (clasificador).
2. Según el método de combinación de salida:
   a. Si selección, invocar al *distributionForInstance()* del clasificador asociado a la mayor probabilidad obtenida, devolver el vector obtenido como salida.

```
        i. Si esta probabilidad está asociada a la última
           clase (C+1) proceder igual que en fusión.
    b. Si fusión, invocar al distributionForInstance() de los
       clasificadores de base y pesar los vectores de salida
       con el obtenido por el metaclasificador. Devolver como
       salida el vector obtenido de esta suma pesada,
       normalizada.
```

Téngase en cuenta que este algoritmo devuelve un vector de probabilidades asociadas a las clases del problema. En el caso de combinación de salidas por selección, se devuelve el vector de probabilidades que da como salida el clasificador seleccionado. Vale la pena recordar que el metaclasificador se entrena con un problema donde las clases son los clasificadores, más una clase que representa al conjunto de instancias bien clasificadas por todos. Si el metaclasificador clasifica a una instancia en la última clase, entonces el resultado puede ser cualquiera de los clasificadores, por eso se procede igual que en el caso de fusión. En el caso de fusión se realiza una suma pesada de los vectores resultantes de los clasificadores de base con el vector obtenido por el metaclasificador.

Las facilidades de extensión del WEKA permiten adicionar de manera muy fácil cualquier otra forma de combinación de salidas (Anexo 13).

III.6. Validación del modelo

Relacionado con la hipótesis 4 del presente trabajo, se probará primero, la generalidad del modelo multiclasificador desarrollado y para ello se intentó medir estadísticamente, la posible superioridad respecto a los otros multiclasificadores populares en la literatura como *bagging*, *boosting* y *stacking*. Se escogieron 34 bases de datos que representan problemas de clasificación en general, extraídas del repositorio UCI [123], de las cuales 11 tienen características biomédicas o bioinformáticas. En el Anexo 14 se hace una descripción de estas bases según la cantidad de rasgos discretos y continuos, cantidad de clases, cantidad de casos y la distribución de casos por clases. Las bases de casos son diversas, 10 de ellas tienen rasgos discretos, 11 tienen rasgos continuos y 13 presentan combinación de ellos; 19 de ellas tienen 2 clases y el resto tiene, entre 3 y 24. La cantidad de casos varía también en las bases, desde 23 hasta 3772 casos. De esta manera podemos ver la comparación para bases internacionales de propósito general y en particular el comportamiento de este método ante bases bioinformáticas.

Como se discutió en el primer capítulo de este trabajo, existen varios tipos de modelos de clasificación. Desafortunadamente no se pueden probar todos, y además ello no sería prudente por el riesgo de obtener diferencias no "honestamente significativas". Se deben seleccionar cuáles se usarán como base para los modelos multiclasificadores. Además, se ha argumentado que existen muchos métodos para seleccionar los clasificadores de base adecuados, y ello depende en buena medida de las características de las bases. Debemos tener en cuenta que se emprende una comparación entre bases de datos muy diversas y, en aras de generalidad, se prefiere utilizar clasificadores comunes y hacer selecciones sencillas, aunque no sean las óptimas.

Una primera selección en el presente trabajo incluye un modelo de árbol de decisión, uno neuronal, un modelo probabilístico, uno perezoso y un SVM. Como ejemplo de árbol de decisión se selecciona el J48, MLP como modelo neuronal, y el *k*NN como modelo perezoso. El último detalle a tener en cuenta es la función núcleo que se usará para el SVM, en este caso se usarán tres, uno polinomial de primer grado (lineal), otro polinomial también pero de segundo grado, y el último con función núcleo gaussiana de base radial. De todos estos modelos sólo se escogerán los que mejores resultados aporten en estas bases respecto a la exactitud.

Con el uso de técnicas estadísticas se seleccionó finalmente un J48, una red bayesiana y un SVM lineal. Como metaclasificador, por su potencia en problemas de clasificación, se seleccionó un MLP. Ésta es la misma topología de clasificadores que se empleó para *stacking*. Para *bagging* y *boosting*, que usan un único modelo de clasificador de base, se hicieron pruebas usando estos mismos clasificadores como modelos de base, añadiendo un MLP, ya que es usado también en el modelo propuesto. En el caso de *bagging* y *boosting* se usaron 10 instancias del modelo de clasificador como base, ya que, con esta cantidad fue como se alcanzaron los mejores resultados. En todos los casos, para la comparación se realizaron 10 corridas del algoritmo usando validación cruzada con 10 subconjuntos y se promediaron los resultados.

La figura III.2 muestra la comparación de los resultados entre el modelo que se propone, MEHI con la topología descrita anteriormente, y *bagging* con tres modelos de clasificadores de base diferentes: J48, SVM y MLP. En esta figura cada punto representa una de las 34 bases, con el valor de exactitud de *bagging* por el eje de las *x* y el valor de

MEHI por las *y*. Un punto encima de la recta *y=x* significa que el resultado de MEHI es superior y un punto por debajo de la recta significa que el resultado de *bagging* es superior. Para las tres comparaciones se puede ver que la mayoría de los puntos quedan encima de la recta, lo que está a favor del método MEHI.

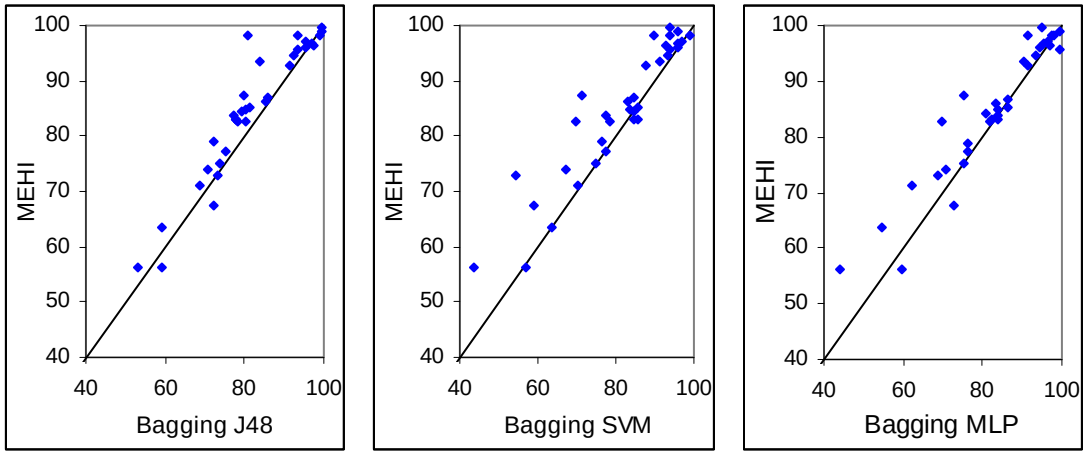


Figura III.2 Comparación del modelo propuesto MEHI con J48, una red bayesiana, SVM como clasificador base y MLP como metaclasificador contra *bagging* usando J48, SVM y MLP como clasificador base.

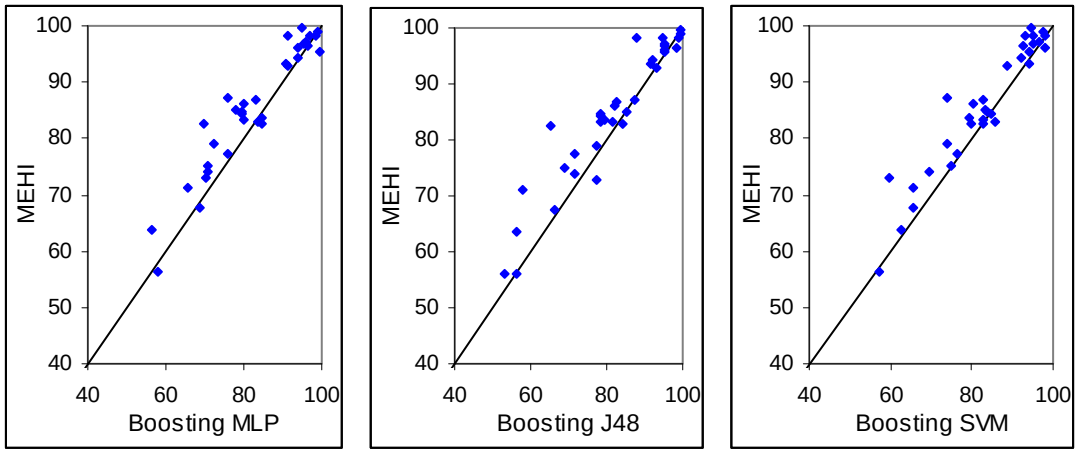


Figura III.3 Comparación del modelo propuesto MEHI con J48, una red bayesiana, SVM como clasificador base y MLP como metaclasificador contra *Boosting* usando J48, SVM y MLP como clasificador base.

De igual manera en la figura III.3 se muestra la comparación de *boosting* usando J48, SVM y MLP como clasificadores de base contra el MEHI y también puede verse que la mayoría de los puntos quedan en la parte superior de la recta, a favor de este último.

De esta misma manera se hace la comparación con *stacking*, con la misma topología usada para MEHI, la cual puede verse en la figura III.4. Esta vez también se puede ver la superioridad de MEHI.

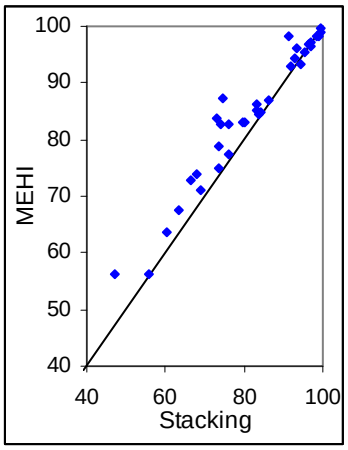


Figura III.4 Comparación del modelo propuesto MEHI contra *Stacking* ambos con J48, una red bayesiana, SVM como clasificador base y MLP como metaclasificador.

Para realizar una comparación más confiable, que esta simple comparación visual se realizaron pruebas estadísticas.

En la figura III.5 se muestran los resultados de Friedman, donde se mostró que había diferencias significativas entre todos los métodos.

En la figura III.6 se muestra la comparación 2 a 2 con la prueba de Wilcoxon donde se puede ver que los resultados obtenidos con MEHI son significativamente superiores con respecto al resto de los métodos: *bagging*, *boosting*, *stacking*. Ya viendo los rangos medios en la prueba de Friedman se podía ver que, organizándolos de mayor a menor, el de mayor rango es el MEHI y el que le sigue es *bagging* con MLP como clasificador de base, por lo que la comparación de estos dos métodos, que en efecto demuestra que los resultados de MEHI son significativamente superiores, hubiera sido suficiente para demostrar que MEHI obtiene resultados superiores al resto.

Aunque las comparaciones no son exactamente con bases de datos de bioinformática, se puede ver que MEHI es un multiclasificador de propósito general con buenos resultados, la mayoría de las veces superiores o similares a los clásicos: *bagging*, *boosting* y *stacking*.

Ranks

	Mean Rank
BAG_J48	4.09
BAG_SVM	4.16
BAG_MLP	4.88
BOO_J48	4.13
BOO_SVM	3.94
BOO_MLP	3.84
Stacking	4.40
MEHI	6.56

Test Statistics[a,b]

N	34
Chi-Square	31.926
df	7
Asymp. Sig.	.000

a. Some or all exact significances cannot be computed because there is insufficient memory.

b. Friedman Test

Figura III.5 Resultados de la prueba de Friedman para comparar *bagging*, *boosting*, *stacking* y MEHI, tal como se obtiene en el paquete estadístico SPSS.

Test Statistics[b]

	MEHI - BAG_J48	MEHI - BAG_SVM	MEHI - BAG_MLP	MEHI - BOO_J48	MEHI - BOO_SVM	MEHI - BOO_MLP	MEHI - Stacking
Z	-3.924[a]	-3.922[a]	-2.949[a]	-3.701[a]	-4.060[a]	-3.804[a]	-4.404[a]
Asymp. Sig. (2-tailed)	.000	.000	.003	.000	.000	.000	.000
Exact Sig. (2-tailed)	.000	.000	.002	.000	.000	.000	.000
Exact Sig. (1-tailed)	.000	.000	.001	.000	.000	.000	.000
Point Probability	.000	.000	.000	.000	.000	.000	.000

a. Based on negative ranks.

b. Wilcoxon Signed Ranks Test

Figura III.6 Resultados de la prueba de Wilcoxon para comparar *bagging*, *boosting*, *stacking* y MEHI, tal como se obtiene en el paquete estadístico SPSS.

III.7. Conclusiones parciales

Se diseña e implementa un modelo de combinación de clasificadores basado en el uso de varios modelos de clasificación que dividen la base de casos en grupos de instancias bien clasificadas, que luego son aprendidos por un metaclasificador.

Se compararon los resultados de MEHI con los multiclasificadores más reconocidos y usados en la literatura, resultando los de MEHI significativamente superiores para una muestra de 34 bases de datos internaciones, de ellas 11 de carácter biomédico o bioinformático.

Este modelo de multiclasificador mejora los resultados del porcentaje de clasificación correcta y el área bajo la curva ROC, y puede ser útil en bases de datos de bioinformática.

Capítulo IV. PREDICCIÓN DE RESISTENCIA DEL VIH

En este capítulo se describe el problema de predicción de resistencia del VIH ante inhibidores de una proteína viral. Se hace un análisis de los datos usando las dos distancias propuestas en el capítulo II. Inicialmente se analiza cómo las energías de contacto ayudan a dividir mejor los casos según la clase. Luego se analiza si los rasgos tienen relación entre sí para buscar cuán significativo sería el uso de extracción de rasgos en este problema.

Se usa el modelo de redes neuronales recurrentes descrito en el capítulo II y se compara con otros clasificadores. Por último se utiliza el multiclasificador propuesto en el capítulo III y se compara con otros multiclasificadores clásicos.

IV.1. Planteamiento del problema de predicción de resistencia del VIH

Los virus son un complejo macromolecular formado por proteínas y ácidos nucleicos, cuya naturaleza puede ser ADN o ARN (estos últimos llamados retrovirus). En este complejo, las proteínas están asociadas formando lo que se conoce como una *cápside* que protege al material genético, teniendo una estructura que puede ser más o menos compleja en dependencia del virus. Existe una enorme diversidad de virus; pero entre los que más afectan a la sociedad en la actualidad se encuentra el Virus de Inmunodeficiencia Humana (VIH).

Hoy en día la búsqueda de fármacos efectivos para el VIH es un reto para los científicos. Se han investigado varias proteínas de este virus, principalmente la proteasa y la transcriptasa inversa, tratando de encontrar un fármaco que las inhiba y de esta manera combatir al virus. Ya existe un gran número de fármacos antivirales, aprobados para tratar la infección del VIH, que son efectivos ante algunas mutaciones de este retrovirus. El tratamiento con combinaciones de estos fármacos puede retener el virus por un tiempo para que se reconstruya el sistema inmunológico. Sin embargo, a menos que la terapia retenga la replicación del virus completamente, el tratamiento antiviral aumenta la aparición de variantes mutadas que son resistentes al fármaco.

Antes de suministrar un fármaco a un paciente es necesario saber qué variantes del virus posee en su organismo y, en base a esto, investigar cuál sería el fármaco, o la combinación de fármacos, más recomendada para su tratamiento. En estos momentos existen dos formas para probar la susceptibilidad de una cepa aislada del VIH ante antirretrovirales: pruebas basadas en fenotipo y genotipo.

Las pruebas genotípicas son simples y se realizan rápidamente, dando como resultado la lista de mutaciones presentes. Sin embargo, la interpretación de la resistencia a inhibidores, partiendo solamente de la información del genotipo, es todavía un reto y a menudo requiere análisis de expertos. Por otro lado, las pruebas fenotípicas proporcionan una cuantificación directa de la resistencia al fármaco, pero requieren más tiempo y un gasto económico muy alto.

La unión de estas pruebas de resistencia han arrojado un gran cúmulo de información sobre este virus, y algunas de ellas se encuentran disponibles en bases de datos internacionales como "Los Álamos" y "*Stanford HIV Resistance Database*". En el presente trabajo se han seleccionado los datos de esta última base para usarlos como información de partida en un método computacional que logre predecir el fenotipo a partir del genotipo. Esta base de datos tiene información de siete de los fármacos más conocidos que se han desarrollado para inhibir la proteasa: *Amprenavir* (APV), *Atazanavir* (ATV), *Indinavir* (IDV), *Lopinavir* (LPV), *Nelfinavir* (NFV), *Ritonavir* (RTV) y *Saquinavir* (SQV).

Lograr encontrar una técnica que ayude a predecir la resistencia a fármacos para combatir el virus del VIH, ahorraría millones de dólares que se necesitan en material de laboratorio para llegar a estimar la resistencia de una cepa del virus. Desde hace varios años esta búsqueda es un problema abierto. Actualmente se han usado algunos métodos de aprendizaje automatizado para relacionar *in silico* el genotipo y el fenotipo del VIH para algunos fármacos específicos. Con algunos de estos métodos se han obtenido resultados satisfactorios. Partiendo de conjuntos de datos compuestos de pares de genotipo y fenotipo, los métodos de aprendizaje automatizado se han utilizado para derivar modelos computacionales que predigan la resistencia del fenotipo desde la información del genotipo. Para algunos fármacos esos modelos ofrecen razones de predicciones razonables, pero para otros, los resultados son menos útiles en el manejo de la infección

del VIH. Encontrar nuevos fármacos para combatir el VIH es muy importante, pero el buen uso de los fármacos ya existentes es también uno de los retos importantes en esta área y es precisamente el problema que se plantea.

Cuando se va a usar un método de aprendizaje se necesita, además de la base de conocimientos con la información a aprender, una representación apropiada de la misma. Éste es uno de los problemas: ¿cómo representar la secuencia mutada de la enzima del VIH a analizar, de forma que tenga relación con la resistencia al fármaco? En el siguiente epígrafe se verá cómo modelar este problema desde el punto de vista de clasificación.

IV.1.1. Modelación del problema y construcción de la base de casos

Según lo explicado en el capítulo I, el diseño de un sistema de aprendizaje, puede verse desglosado en los siguientes pasos: definir la tarea, crear la base de conocimientos, definir la función objetivo, análisis y preprocesamiento de los datos, seleccionar la forma de medir el aprendizaje del sistema y por último seleccionar el mecanismo de aprendizaje. Se puede decir que los tres primeros pasos ayudan a la definición de la problemática. El resto ayuda a la búsqueda del método de solución para ese problema.

En este trabajo la tarea específica se puede definir como: la clasificación de secuencias de la proteasa en resistentes o susceptibles ante el fármaco analizado. Se está en presencia de siete problemas de clasificación, uno para cada fármaco que se analiza.

El segundo paso es la creación de las bases de conocimientos. Existen varias bases de datos con información disponible relacionando los pares de genotipo y fenotipo de la proteasa del VIH. Debido a la variabilidad de esta enzima por su poder de mutación existe gran cantidad de casos. Como se dijo anteriormente, la base de datos de *Stanford* es una de las más conocidas en cuanto a resistencia del VIH se refiere. En esta base de datos la representación de la enzima de la proteasa está basada en la secuencia primaria de la misma descrita por los aminoácidos que la conforman. Esta base puede ser obtenida en: http://hivdb.stanford.edu//cgi-bin/GenoPhenoDS.cgi.

En la parte superior de la Figura IV.1 se puede ver la estructura de la base de datos de *Stanford*, donde cada caso está descrito por el fenotipo y genotipo. El genotipo está representado por el listado de las posiciones mutadas con los aminoácidos que fueron cambiados. El fenotipo está representado por el factor de resistencia basado en la

concentración de cada uno de los siete inhibidores de la proteasa, mencionados anteriormente.

Las mutaciones que describen a cada secuencia están basadas en una secuencia de referencia, HXB2, que puede llamarse secuencia salvaje (*wild type*). Para reconstruir cada secuencia reportada en la base de datos se toma la secuencia de la enzima del HXB2 y se cambian los aminoácidos de las posiciones que se listan por el aminoácido correspondiente, tal y como se muestra en la parte inferior de la Figura IV.1.

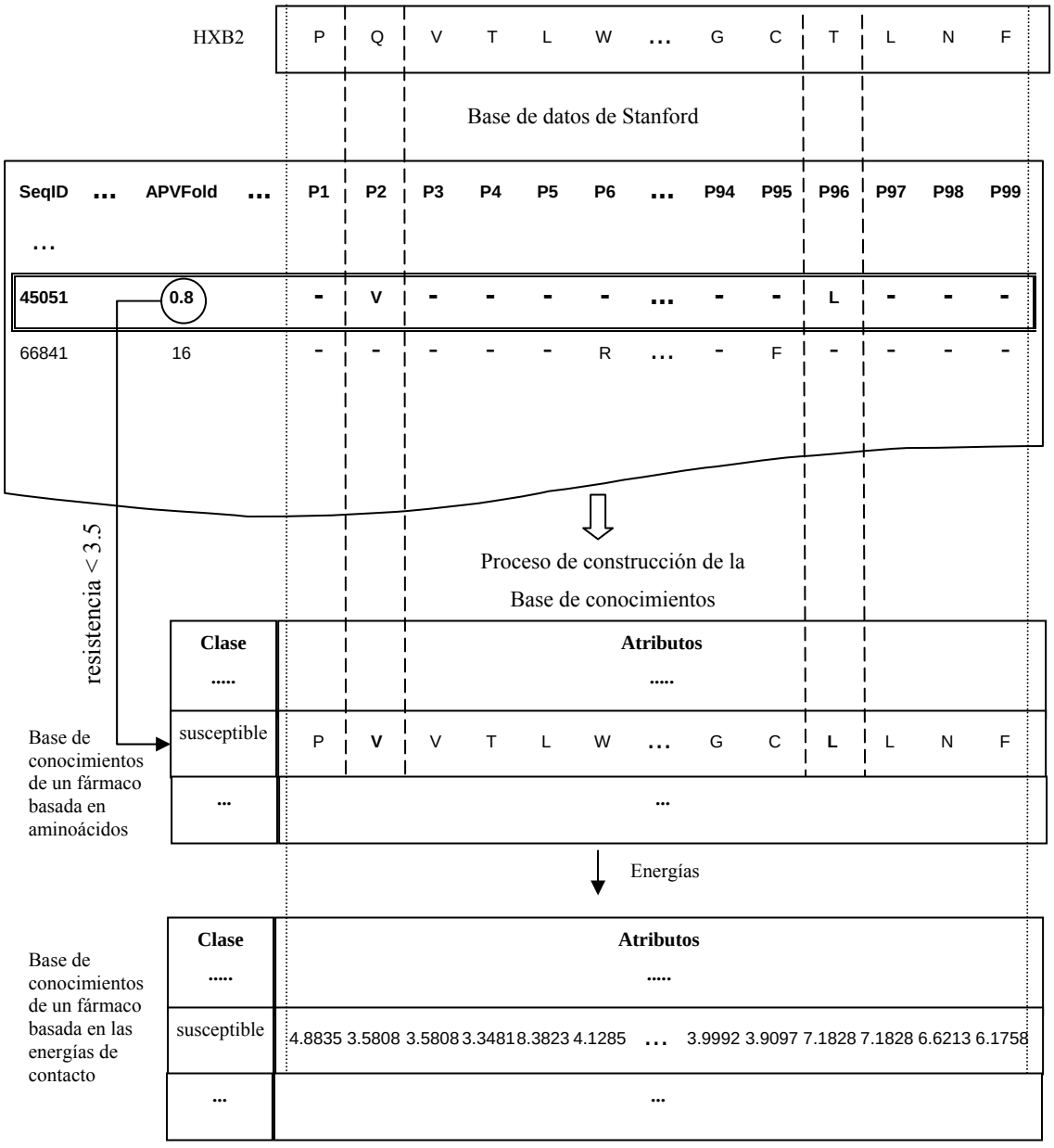


Figura IV.1 Conversión de los casos de la base de Stanford para conformar la base de conocimientos

Una vez que se haya hecho la conversión en términos de aminoácidos se cuenta con la información de experiencia que se necesita para nuestro problema, o sea, una base con un conjunto de secuencias mutadas de la proteasa del VIH a la que se le hace corresponder la resistencia del fármaco que se analice. Así se obtienen las siete bases de casos, una para cada fármaco. El próximo paso en la modelación del sistema de clasificación es definir la función objetivo, es decir, una representación para este conocimiento.

Por todo lo analizado en el epígrafe II.1, se seleccionó la energía de contacto de los aminoácidos para representar la secuencia de la proteasa. En todas las secuencias se sustituyó cada aminoácido por la energía de contacto correspondiente, como muestra la Figura IV.1 y de esta manera cada caso tiene 99 atributos descriptivos, ya que la secuencia de la proteasa tiene una longitud de 99 aminoácidos.

La clase de cada instancia está basada en la razón de resistencia (*fold resistance*) con respecto a la secuencia de referencia, o sea, la razón del valor de IC50 medido para la secuencia aislada y la secuencia de referencia. El IC50 es la concentración del fármaco que se necesita para inhibir la actividad viral al 50%. Según la bibliografía [118], se define un corte (*cut-off*) para cada fármaco, que casualmente en el caso de estos siete fármacos es el mismo: 3.5. Este corte significa que si se usa un valor de concentración del fármaco superior a 3.5 veces la concentración usada con HXB2 para inhibir el virus, puede estar en riesgo la salud del paciente, por lo que sólo es posible suministrar un valor inferior a este corte.

Si la resistencia es superior al corte establecido se clasifica en resistente (clase=1), o sea, la secuencia del VIH es resistente ante ese fármaco; y en caso de ser menor que 3.5 se clasifica como susceptible (clase=0). De esta manera el problema se convierte en un problema de clasificación en dos clases.

La función objetivo queda definida como sigue:

F: $C \rightarrow O$, donde, $C \subseteq R^{99}$. C es un vector de 99 aminoácidos y cada aminoácido está representado por un valor real asociado a su energía de contacto, o sea, los elementos de C van a representar secuencias mutadas de la proteasa. El conjunto O={resistente, susceptible} es el conjunto de clases y puede ser analizado en forma binaria.

De los casos de la base de datos se excluyeron aquellos que tenían cambios por aminoácidos desconocidos, para eliminar la ausencia de información. Las siete bases de

conocimiento se identificarán en lo adelante con las siglas del fármaco y tienen la distribución de casos por clases que se muestra en la Tabla IV.1.

Tabla IV.1 Cantidad de casos en cada base de conocimientos.

	Cantidad de casos		
Fármaco	**Susceptible**	**Resistente**	**Total**
APV	206	167	373
ATV	59	62	121
IDV	172	205	377
LPV	26	149	175
NFV	117	286	403
RTV	146	160	306
SQV	212	174	386

Finalmente ya se tiene el diseño del problema, sólo queda el análisis y preprocesamiento de los datos, y la selección del mecanismo de aprendizaje apropiado.

IV.2. Análisis de los datos

A continuación se describe el análisis de los datos basado en técnicas de extracción de rasgos. La primera técnica tiene como objetivo reducir la dimensionalidad de los datos, de forma que puedan ser analizados de forma visual. La segunda técnica de reducción es con el objetivo de agrupar rasgos que tengan cierta relación.

En este análisis se persigue demostrar los siguientes aspectos:

1. Las energías de contacto son una buena representación para este problema (hipótesis 1 del presente trabajo).
2. Existen relaciones complejas entre los rasgos (relacionado con la hipótesis 3).

IV.2.1. Visualización de los datos usando extracción de rasgos

Para apoyar la primera hipótesis se pasa a hacer una visualización de las siete bases del VIH, que permita analizar cómo se distribuyen los casos por clases. Como método de reducción de dimensionalidad se utilizó MDS, para visualizar los datos en dos dimensiones. Para este método de extracción de rasgos se usó la función de semejanza *WeightedMutation* definida en las ecuaciones II.4 y II.5.

Se seleccionaron como funciones de distancia, para comparar con la distancia propuesta, las distancias *Euclidiana*, *CityBlock*, *Hamming* y *Chebychev* (todas definidas en el Anexo 4). La distancia euclidiana es una métrica para valores continuos y es aplicable a este problema gracias al uso de las energías de contacto como atributos. La distancia *CityBlock* (o de *Manhattan*) también está definida sobre valores continuos. En nuestro problema representa la variación total de energía en las posiciones mutadas. *Chebychev* representa la variación de energía máxima. Por último, la distancia de *Hamming* calcula el total de posiciones diferentes entre dos secuencias, o sea, el número mínimo de sustituciones necesarias para transformar una secuencia en la otra. Esta distancia trata realmente con valores nominales, y esto no es problema, ya que las energías tienen sólo 20 valores posibles, que seguirán representando los 20 aminoácidos.

Por otro lado la distancia *WeightedMutation* calcula la cantidad de mutaciones, haciendo diferencia cuando éstas son iguales y cuando cambian por aminoácidos diferentes, teniendo en cuenta si el cambio provoca aumento o disminución de la energía de contacto, ya que esto hará variar la estructura tridimensional.

Como se explicó en el epígrafe II.1.1.2, los valores de w_i en la distancia *WeightedMutation* deben asignarse en orden descendente a partir de w_1 de forma que se dé mayor valor a aquellas mutaciones que son exactamente iguales, se les dé menor valor a las que son diferentes pero en cuestiones de energía van en el mismo sentido (aumentan o disminuyen las dos) y por último con menor valor aquellas que van en sentido contrario (la energía de una aumenta y la otra disminuye). Por esto proponemos en principio los siguientes valores: w_1=1, w_2=0.5 y w_3=0.25.

La Figura IV.2 muestra los resultados de reducción de dimensionalidad con cada una de las distancias y dos antivirales. A la derecha se ven los resultados de IDV y a la izquierda NFV. Cada punto representa un caso de la base, los puntos negros son los de la clase resistente y los grises los de la clase susceptible.

Estas bases son descritas, como se ha dicho anteriormente, por 99 aminoácidos, así la tarea de reducir la cantidad de rasgos a sólo 2D es muy compleja y se puede perder mucha información. A pesar de esta complejidad y lo difícil de separar las clases en este problema, la visualización usando funciones de distancia puede dar una idea de cuán separados están los casos. En efecto, en estas dos bases se puede ver que el uso de la

distancia propuesta proporciona una división de clases más fácil. Se forma una estructura circular, donde si trazamos una recta vertical, cerca de su centro, se ve que se puede separar la mayoría de los casos según sus dos clases, ya que los objetos de una clase están concentrados hacia la parte derecha y los de la otra hacia la izquierda.

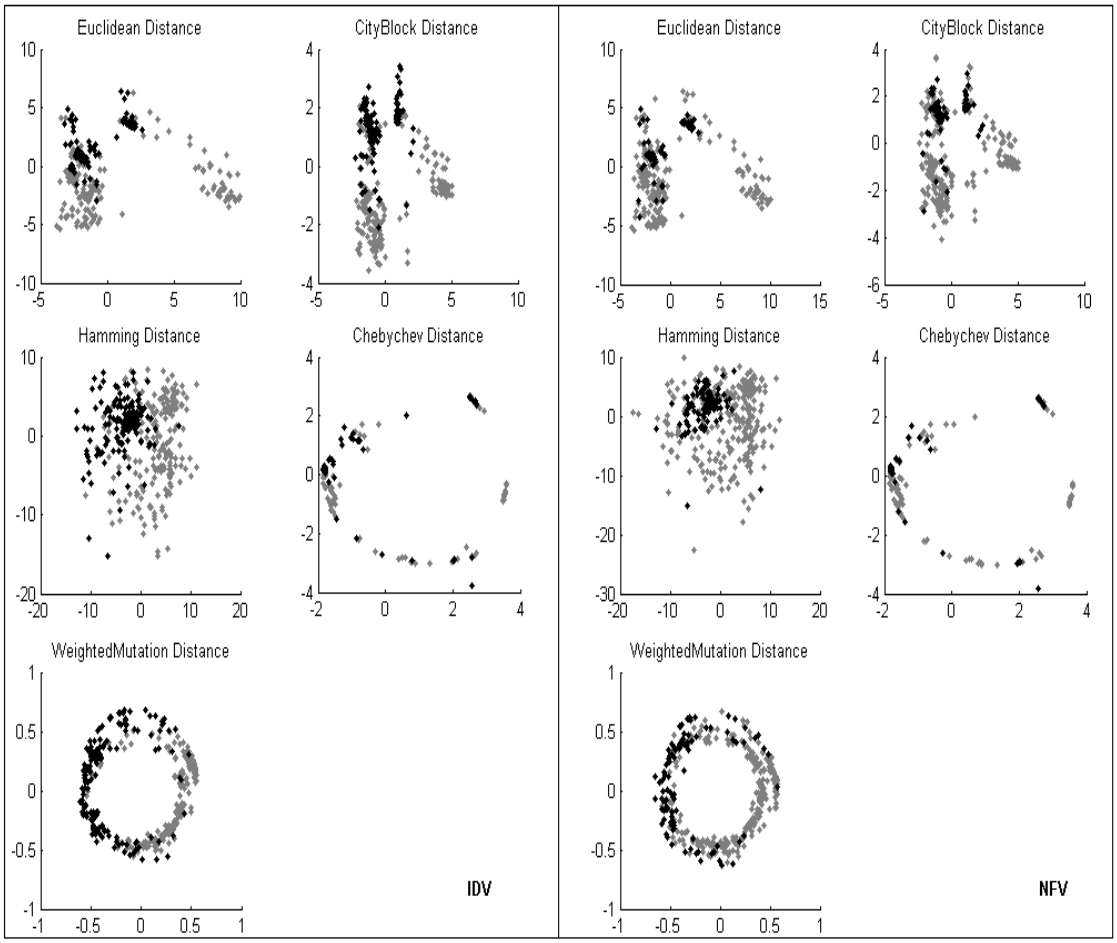


Figura IV.2 Reducción de dimensionalidad de IDV y NFV usando MDS y distancias Euclidiana, City Block, Hamming, Chebychev y WeightedMutation.

Análogamente la Figura IV.3 muestra a la izquierda los resultados de visualización de APV con las diferentes distancias, y a la derecha los resultados para RTV. En estos dos fármacos también puede verse que sucede lo mismo que en los anteriores. Puede notarse que en RTV se pueden separar mejor las clases que en APV, y por tanto la distancia parece ajustarse mejor al problema en el primer antiviral que en este último.

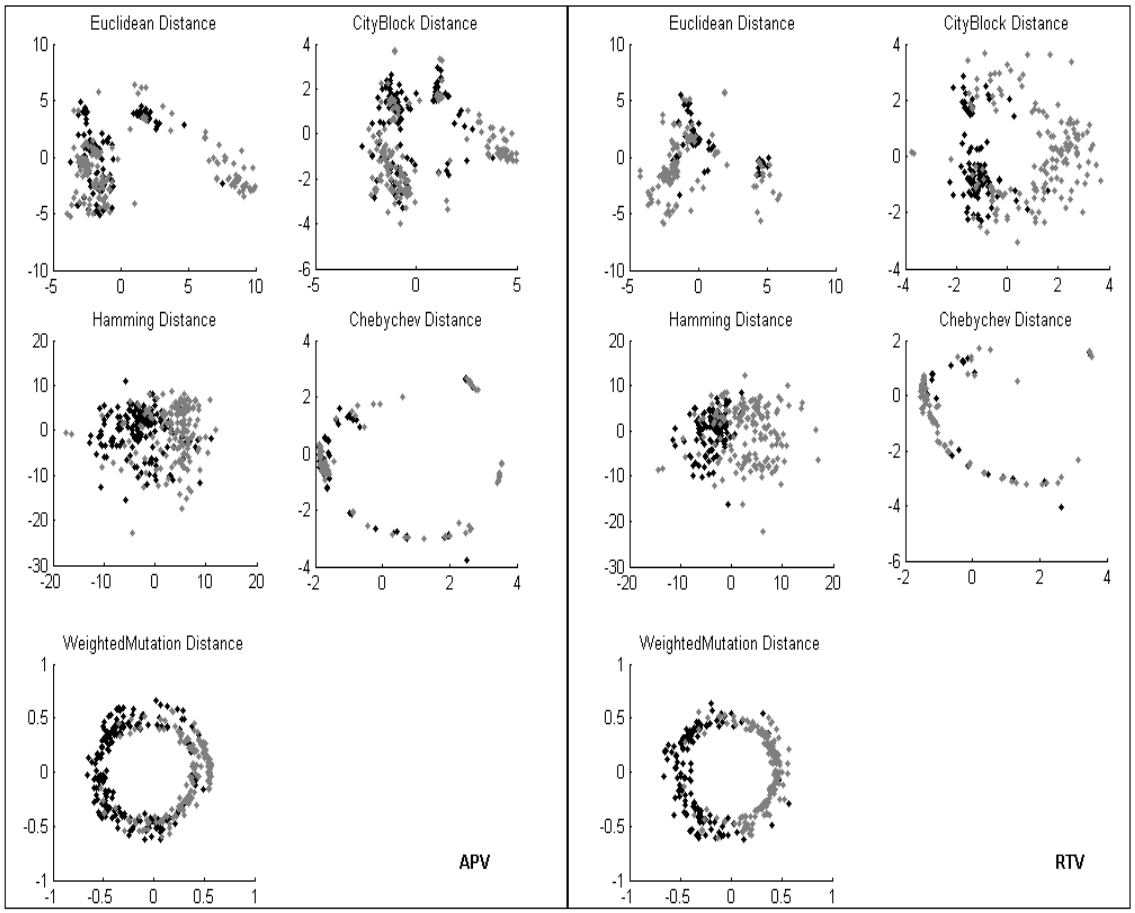


Figura IV.3 Reducción de dimensionalidad de APV y RTV usando MDS y distancias Euclidiana, City Block, Hamming, Chebychev y WeightedMutation.

A diferencia de los fármacos anteriores, en ATV y SQV no pasa lo mismo (Figura IV.4), los casos están más disgregados en la función *WeightedMutation*, al igual que en el resto de ellas, no es fácil decidir cuál sería la mejor de estas distancias en estas dos bases.

La Figura IV.5 muestra los resultados de reducción de dimensionalidad para LPV. En este caso con los resultados visuales no se pueden sacar muchas conclusiones, aunque de las distancias que se analizan es *WeightedMutation* donde la clase representada por puntos negros se muestra menos cercana a la mayoría de los casos de la otra clase. Entonces se espera que los resultados de la distancia propuesta sean ligeramente superiores.

Hay que tener en cuenta que los resultados para LPV y APV pueden estar influenciados por la bases de datos, ya que son las que presentan menos casos. Además esto es una

representación, donde se redujo de 99 rasgos a dos y sólo es para hacer un pequeño análisis que comprobaremos luego con todos los rasgos.

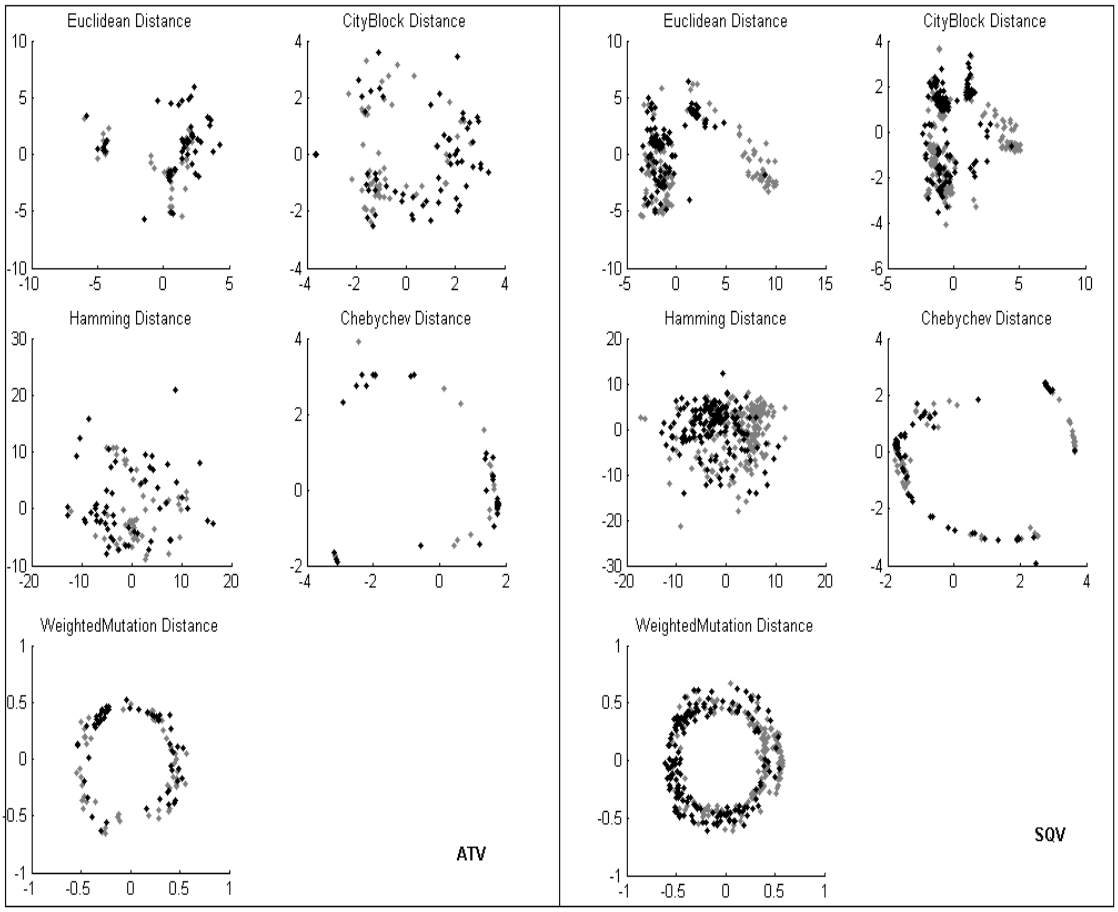


Figura IV.4 Reducción de dimensionalidad de ATV y SQV usando MDS y distancias Euclidiana, City Block, Hamming, Chebychev y WeightedMutation.

Una vez que las distancias han sido analizadas con la ayuda de la visualización de los datos, se pueden verificar estos resultados usando un método basado en medidas de distancia, como el *k*NN, con las bases originales de 99 rasgos. Este método se probó con las mismas distancias, para validar el análisis visual.

Los resultados del *k*NN son mostrados en la Figura IV.6 y son obtenidos con el uso de k=3 y validación cruzada con 10 subconjuntos. Estos resultados están en correspondencia con los obtenidos anteriormente de forma visual. La distancia *WeightedMutation* es la que de manera general obtiene mejores resultados en APV, IDV, NFV y RTV. Por otro lado en el caso de SQV y LPV aunque los resultados visuales no eran tan claros como

para poder decidir entre una de las distancias, los resultados del *k*NN son superiores, lo que muestra que la distancia es un buen indicador para estos dos fármacos también.

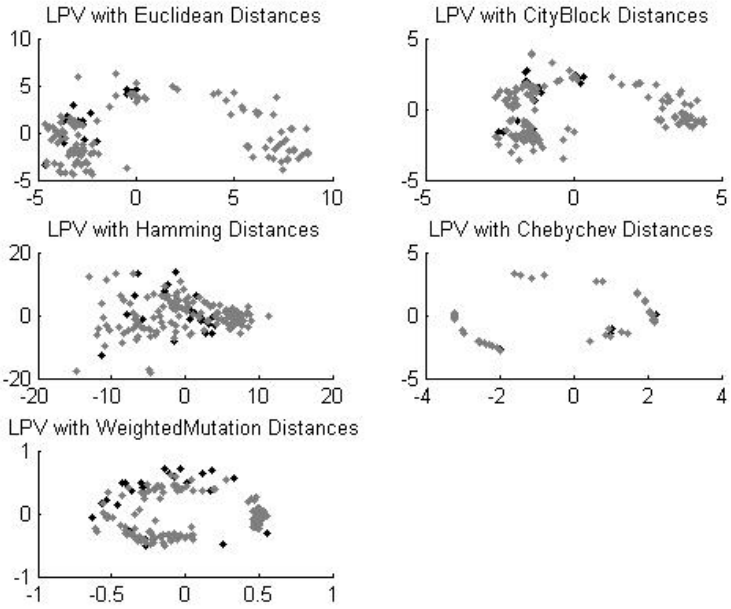


Figura IV.5 Reducción de dimensionalidad de LPV usando MDS y distancias Euclidiana, *CityBlock*, *Hamming*, *Chebychev* y *WeightedMutation*.

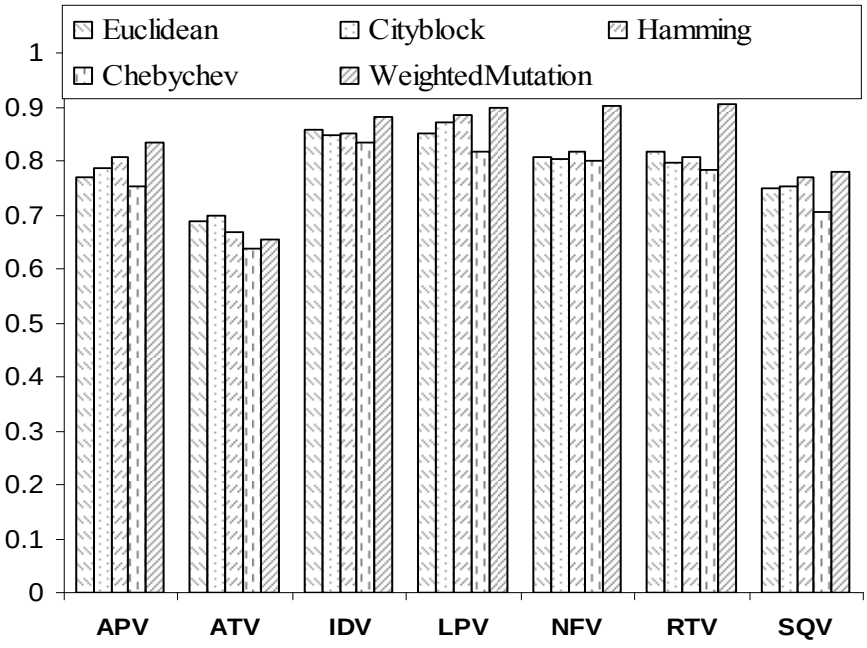


Figura IV.6 Resultados de exactitud del uso de *k*NN usando distancias Euclidiana, *CityBlock*, *Hamming*, *Chebychev* y *WeightedMutation* para APV, ATV, IDV, LPV, NFV, RTV y SQV.

Los resultados obtenidos por ATV son inferiores cuando se usa *WeightedMutation*, en este fármaco se obtienen los mejores resultados con el uso de *CityBlock*.

Los resultados obtenidos en este trabajo muestran que la distancia propuesta basada en las mutaciones pesadas es una medida de distancia prometedora para el trabajo con secuencias del VIH. Se muestra además, que el uso de técnicas de reducción de dimensionalidad como MDS ayuda a la selección de una distancia apropiada para un problema de clasificación determinado.

La factibilidad de esta distancia demuestra la hipótesis 2 indicando que las energías de contacto de los aminoácidos tienen un significado importante para representar el problema, ya que hacen ver a los aminoácidos, no con su carácter nominal sino con el carácter ordinal que éstas les imponen. Por ello se concluye que las energías constituyen una representación apropiada para el problema de la resistencia del VIH, probando así también la hipótesis 1.

En [128] se publicaron los resultados de este epígrafe.

IV.2.2. Análisis de los rasgos usando extracción de rasgos

Teniendo en cuenta la influencia de las mutaciones en el problema, puede pensarse que existe alguna relación entre los rasgos y de hecho, para la conformación de la estructura tridimensional de la proteína los aminoácidos interactúan entre sí. Pero, ¿existirá alguna relación entre los rasgos que influyan sobre la clase? Para dar respuesta a esta interrogante es necesario hacer un análisis de los rasgos.

En la Figura IV.1, donde se mostraba la formación de la base de conocimiento, se podía ver que se tienen 99 rasgos, donde cada uno tiene 20 valores posibles (los 20 aminoácidos). En el ejemplo en la base de conocimientos basada en aminoácidos, puede verse que el aminoácido de la posición 5 coincide con los de la posición 96 y 97. Esto después se traduce, en la base de conocimientos basada en energías de contacto, en que estas posiciones tendrán el mismo valor de energía. Claro que esto no es suficiente para afirmar que esas posiciones estén relacionadas, se necesita más información para comparar los rasgos. Se hace necesario el uso de otra función de distancia, esta vez para comparar los rasgos, no los casos.

En el capítulo II se hace un análisis de dos nuevas medidas de semejanzas, una de ellas basada en la diferencia entre rasgos de un mismo caso que es la que se usará en este epígrafe. La idea se basa en ver si existe asociación entre los aminoácidos que describen la secuencia de una proteína. Se sigue el procedimiento descrito en el epígrafe II.1.2, donde el primer paso es el uso de un algoritmo de agrupamiento sobre la matriz traspuesta de los datos, para crear grupos de rasgos. Para esto se selecciona el método *k*-medias, como efectivamente se propone en el algoritmo. Para chequear la fiabilidad se usa la medida MCD, definida en la ecuación II.6, de forma que se seleccionen los grupos confiables, o sea, aquel agrupamiento que logra grupos más compactos. Luego, en el tercer paso, para cada uno de los agrupamientos seleccionados, se construye una nueva base de casos con tantos rasgos como grupos existan, de forma que los nuevos rasgos van a estar dados por la suma de los elementos de cada grupo.

Para validar la extracción de rasgos se seleccionó SVM con función núcleo lineal, como método de clasificación, de forma que permita comparar los resultados de la base original con los resultados usando cada una de las nuevas bases de casos, producidas por la extracción de rasgos. Como medida del aprendizaje se seleccionó AUC y se utilizó el método de validación cruzada con 10 subconjuntos.

Primero se entrena el SVM usando la base original y después se aplica el algoritmo de agrupamiento *k*-medias para todos los valores posibles de *k*, en nuestro caso *k*=1,2,..,99 Con valores pequeños de *k* se obtuvieron buenos resultados como muestra la Tabla IV.2.

Tabla IV.2 Resultados del AUC del SVM para las siete bases del VIH usando 99, 25 y 28 rasgos.

Número de rasgos	***APV***	***ATV***	***IDV***	***LPV***	***NFV***	***RTV***	***SQV***
99	83.5	75.2	89.3	65	85.7	89.9	85.2
25	**83.6**	**81**	**89.8**	**77.2**	83.8	**92.9**	**87.6**
28	**84**	**80.2**	**90.3**	**78.1**	**85.9**	**92.7**	**86.8**

En la primera fila se muestran los resultados para todos los rasgos. La segunda y la tercera describen los resultados para 25 y 28 grupos respectivamente, para cada fármaco. Como puede verse, los resultados para 25 grupos son mejores, o al menos son tan buenos como los obtenidos usando todos los rasgos, para seis de estos siete fármacos. Usando 28 grupos se obtienen buenos resultados para todas las bases.

Con esta prueba sólo se chequea la validez de la predicción del SVM. En este caso se ha usado una técnica no supervisada (*k*-medias) y una técnica supervisada (SVM). Para la validación de los resultados se necesita chequear la validez de la clasificación, pero también se necesita chequear la confiabilidad del agrupamiento.

Después de seleccionados los agrupamientos con menor valor de MCD, se seleccionan, para cada una de las bases, aquellos con mayores valores de AUC y los resultados son los que se muestran en la Tabla IV.3. Puede verse que los mejores resultados para cada fármaco son obtenidos usando diferente *k* (número de grupos), esto significa que la dependencia de los aminoácidos con respecto a la clase es diferente para cada fármaco. Esto tiene sentido, ya que una secuencia actúa de manera diferente ante distintos fármacos, por ejemplo una cepa del VIH puede ser resistente ante APV y no ante LPV.

En todos los casos mostrados en la Tabla IV.3 se obtienen resultados que son mejores o semejantes a los obtenidos con todos los rasgos.

Tabla IV.3 Resultados de AUC para los mejores agrupamientos por rasgos, basados en el valor de MCD como medida de confiabilidad.

Grupos	***APV***	***ATV***	***IDV***	***NFV***	***RTV***	***SQV***	***LPV***
36					94		
39							82.3
41						89.4	
54		84.3					
62	85						
65			92.2				
66				89.7			

Con el desarrollo de esta medida de semejanza y los resultados obtenidos se afianzan las hipótesis 1 y 2. Finalmente se concluye que existen posiciones dependientes, o sea, existen relaciones complejas entre los rasgos, que influyen en la resistencia a los fármacos. De esta manera surge la hipótesis 3: el uso de un método de clasificación que tenga en cuenta la combinación de resultados de rasgos de diferentes partes de la secuencia, puede producir buenos resultados en este problema. Como uno de estos métodos, se tienen en cuanta las redes recurrentes bidireccionales propuestas en el capítulo II.

En [129] se publicaron estos resultados.

IV.3. Solución del problema usando redes neuronales recurrentes

Existen muchos métodos usados en la literatura para la predicción de resistencia del VIH. Reeh et al. [15] y Cao et al. [13] hacen un buen resumen de los métodos utilizados para este problema, los mejores resultados usan rasgos nominales para representar los aminoácidos, u otras representaciones como la información mutua, pero generalmente usan métodos de selección de rasgos. En este epígrafe se demuestran los siguientes aspectos:

1. Con el uso de las energías de contacto para representar los aminoácidos se obtienen resultados superiores o al menos similares a los obtenidos anteriormente en la literatura para la predicción de resistencia de secuencias del VIH ante inhibidores de la proteasa (reforzando la hipótesis 1).
2. Las redes recurrentes bidireccionales descritas en el capítulo II, con sus características internas de extracción de rasgos, permiten obtener buenos resultados en el problema planteado (hipótesis 3).

Como un método que puede tratar con problemas de clasificación que tengan relación entre los rasgos y compleja separación entre las clases, se seleccionaron las redes neuronales recurrentes descritas en el capítulo II. Estos resultados se comparan con otros de los métodos de clasificación que han sido usados en la literatura para resolver este problema.

Se seleccionó la topología descrita en el epígrafe II.2, o sea, un modelo bidireccional. Para el parámetro T (cantidad de tiempos) se seleccionaron números impares, con el objetivo de facilitar el uso de la función moda para combinar las salidas. A medida que este parámetro aumenta también lo hace la complejidad computacional de la red, ya que para T tiempos la red se despliega T veces. Teniendo en cuenta esto, se probó con tres y cinco tiempos, obteniendo resultados similares, asumiendo finalmente la topología con T=3.

Como las bases de conocimientos tienen 99 rasgos, entonces nuestra red tendrá 33 neuronas de entrada en cada tiempo, una para cada rasgo, y dos neuronas de salida, una

para cada uno de los posibles valores del rasgo objetivo (susceptible, resistente). Como se mostró en la topología descrita en el capítulo II, se cuenta con tres capas ocultas, una para la salida hacia el tiempo futuro, otra para la salida al tiempo pasado y otra para el tiempo actual.

La cantidad de neuronas de las capas ocultas se varió desde 10 hasta 20, estableciendo siempre la misma cantidad de neuronas para las tres capas, resultando APV con 20 neuronas en cada una de las capas ocultas, ATV con 10, IDV con 20, LPV con 10, NFV y RTV con 20, y SQV con 10.

Como función objetivo se usó entropía cruzada (*Cross-Entropy*) y como función de activación de salida se usó *Softmax*. Como algoritmo de entrenamiento se usó el BPTT. Se hicieron pruebas variando los parámetros de la red, como se describe anteriormente, y con las tres funciones de combinación de salidas propuestas en el capítulo II: moda, promedio y máxima probabilidad.

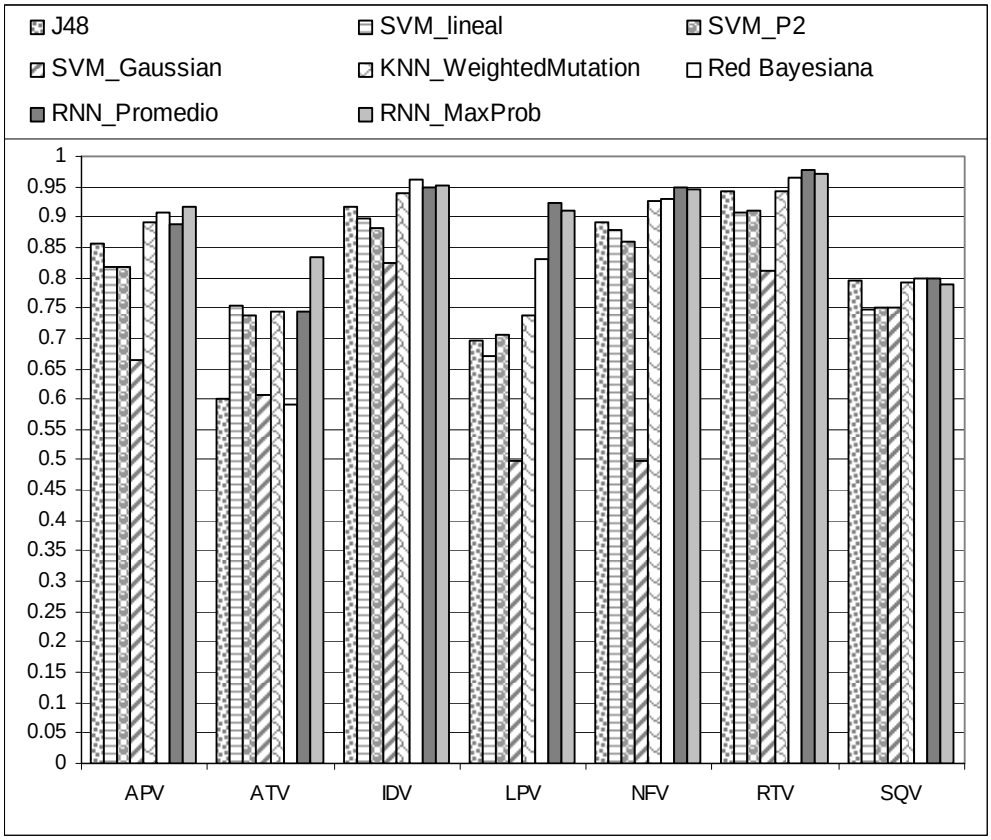


Figura IV.7 Comparación de los resultados de AUC de J48, BN, SVM lineal, SVM polinomial grado 2(SVMP2), SVM con función núcleo gaussiano (SVM *Gaussian*), *k*NN con *k*=3 (*k*NN_WeightedMutation), MLP y RNN.

Para comparar los resultados obtenidos por la red recurrente se seleccionaron clasificadores populares con diferentes paradigmas para formar la región de decisión, que se encuentran implementados en el WEKA. Se seleccionó el J48 como árbol de decisión, una red bayesiana (BN), SVM y una red MLP. Para la BN se seleccionó K2 como algoritmo de aprendizaje. Se utiliza SVM con tres tipos de función núcleo: lineal, polinomial grado 2 y gaussiana. Para MLP, al igual que en la RNN, se probaron diferentes modelos variando el número de neuronas ocultas. Los valores de los pesos de la red se inicializan aleatoriamente con valores entre -0.5 y 0.5 tanto para la red MLP como para la RNN.

Se tuvieron en cuenta dos medidas de eficiencia para comparar los algoritmos: la exactitud y el AUC. En el caso del análisis del AUC se utilizaron las formas de combinación de salidas continuas: promedio y máxima probabilidad, y en el caso de la exactitud éstas dos más la moda. Para todos los métodos seleccionados se realizaron 10 corridas usando validación cruzada con 10 subconjuntos, con cada una de las bases.

En la Figura IV.7 se muestra la comparación de los resultados del AUC, para las siete bases del VIH, usando J48, BN, los tres SVM, MLP y las dos redes recurrentes.

Tabla IV.4 Significación de la prueba estadística de Wilcoxon para la comparación de la RNN con el resto de los clasificadores: J48, BN, SVM lineal, SVM polinomial de grado 2, SVM con función núcleo gaussiano, *k*NN con la distancia *WeightedMutation* y MLP.

RNN_MP vs.	**Significación Asintótica (2 colas)**	**Significación Exacta (2 colas)**	**Significación Exacta (1 cola)**
J48	.018	**.016**	.008
BN	.063	**.078**	.039
SVM_Lineal	.018	**.016**	.008
SVM_P2	.018	**.016**	.008
SVM_*Gaussian*	.018	**.016**	.008
*k*NN_*Weighted Mutation*	.018	**.016**	.008
MLP	.018	**.016**	.008

Se puede ver que los resultados obtenidos por J48, BN, SVM y MLP son muy bajos en ATV y LPV, que son precisamente las bases con menor cantidad de casos, y en estos fármacos el resultado de la red recurrente es significativamente superior al alcanzado por

el resto de los clasificadores. En IDV, NFV, RTV y SQV los resultados máximos alcanzados están por encima del 80%.

Se realizaron pruebas de Friedman y Wilcoxon para el análisis estadístico de los resultados. La Tabla IV.4 muestra los resultados obtenidos por la prueba de Wilcoxon, donde puede verse que los resultados del AUC de la RNN con función de máxima probabilidad son significativamente superiores a los obtenidos por J48, SVM (lineal, polinomial grado 2 y función núcleo gaussiana) y MLP. Y que los resultados comparados con la BN tienen diferencias medianamente significativas (6 vs. 1).

Debe tenerse en cuenta que estas pruebas no paramétricas son bastante fuertes aún cuando se tengan pocas bases para establecer la comparación [130]. No obstante para analizar un poco mejor los resultados véase el comportamiento promedio de estos métodos con todas las bases de la proteasa. En las tablas IV.5 y IV.6 se muestran los resultados obtenidos por la exactitud y el AUC respectivamente. La última fila muestra el comportamiento promedio de estos métodos ante todos los fármacos, o sea, es el promedio de los resultados obtenidos en todas las bases, para cada uno de los métodos de clasificación utilizados.

Tabla IV.5 Resultados de la exactitud obtenidos por los diferentes métodos de clasificación.

	J48	RB	SVM lineal	SVM P2	SVM *Gaussian*	KNN *Weighted Mutation*	MLP	RNN Promedio	RNN Moda	RNN MP
APV	0.823	0.810	0.820	0.823	0.694	**0.829**	0.794	0.828	**0.829**	**0.829**
ATV	0.653	0.678	0.752	0.736	0.612	0.688	0.760	0.744	0.753	**0.769**
IDV	**0.899**	0.894	0.897	0.881	0.822	**0.899**	0.881	0.875	**0.899**	**0.899**
LPV	0.891	0.897	0.874	0.880	0.851	0.867	0.897	0.897	**0.909**	0.886
NFV	0.901	0.859	0.881	0.868	0.710	0.905	0.881	0.916	0.911	**0.921**
RTV	**0.928**	0.905	0.905	0.908	0.804	0.902	0.905	0.915	**0.928**	**0.928**
SQV	**0.764**	0.723	0.749	0.749	0.749	0.739	0.738	0.720	0.715	0.718
Promedio	0.837	0.824	0.840	0.835	0.749	0.833	0.837	0.842	**0.849**	**0.850**

Teniendo en cuenta como medida de validación la exactitud, puede verse que las combinaciones de salida moda y máxima probabilidad son las que obtienen mejores resultados, o al menos semejantes a los obtenidos por el resto de los clasificadores. Si analizamos los resultados del área bajo la curva puede verse que los resultados obtenidos sí son significativamente superiores.

Después de analizados los resultados se llega a la conclusión de que la RNN con función de combinación como máxima probabilidad y moda mejora el indicador de eficiencia AUC del clasificador, y mantiene o supera su exactitud.

Tabla IV.6 Resultados del AUC obtenidos por los diferentes métodos de clasificación.

	J48	RB	SVM lineal	SVM P2	SVM *Gaussian*	KNN *Weighted Mutation*	MLP	RNN Promedio	RNN MP
APV	0.855	0.907	0.816	0.819	0.663	0.891	0.888	0.888	0.918
ATV	0.601	0.592	0.753	0.737	0.607	0.745	0.785	0.746	0.834
IDV	0.916	0.962	0.897	0.882	0.826	0.938	0.943	0.948	0.953
LPV	0.697	0.832	0.672	0.707	0.5	0.74	0.909	0.922	0.911
NFV	0.891	0.93	0.878	0.859	0.5	0.926	0.936	0.948	0.947
RTV	0.944	0.965	0.906	0.91	0.811	0.943	0.966	0.976	0.971
SQV	0.794	0.798	0.749	0.75	0.75	0.792	0.78	0.799	0.799
Promedio	0.814	0.855	0.81	0.809	0.665	0.854	**0.8867**	**0.889**	**0.903**

Estos resultados han demostrado la superioridad de este método para la predicción de resistencia de la proteasa ante un fármaco determinado, incluso en bases de datos con pocos casos. Aunque en las bases seleccionadas las únicas mutaciones que presentan son sustituciones, o sea, no presenta inserciones y eliminaciones en la secuencia (todas las secuencias tienen longitud fija), la potencialidad de estas redes también está en su carácter dinámico, como se explicó en el capítulo II. Esto facilitaría la predicción de nuevas secuencias con estas características, o para otras proteínas del VIH como la transcriptasa inversa que sí es abundante en este tipo de mutaciones.

En [131] se reafirma la hipótesis 1 de que las energías de contacto son una buena representación para las secuencias del VIH, ya esta vez usado varios clasificadores y comparando con resultados obtenidos previamente con otras representaciones. En [132] se argumentan los resultados con la RNN para este problema y se compara con los resultados obtenidos por otros autores.

IV.4. Solución del problema usando el modelo de combinación de clasificadores

Después de usar la red recurrente en la predicción de resistencia del VIH y obtener resultados bastante buenos, se decidió el uso del modelo de combinación de clasificadores descrito en el capítulo III para demostrar entonces la hipótesis 4:

- La combinación de varios clasificadores con un metaclasificador que aprenda los casos duros para cada uno, puede dar buenos resultados en el problema de predicción de resistencia del VIH ante determinados fármacos.

La posibilidad de estar en presencia de una base, como se decía anteriormente, sólo con mutaciones de tipo sustitución, permite el uso del modelo de clasificador propuesto en el capítulo III. Se realiza además una comparación con otros multiclasificadores clásicos, como *bagging*, *boosting* y *stacking*.

IV.4.1. Arquitectura del modelo de combinación de clasificadores

En esta fase del problema se procedió de la misma manera que para la validación realizada en el epígrafe III.6. Se escogieron diferentes paradigmas de clasificadores: un árbol de decisión (J48), BN, SVM (polinomial grado 2), *k*NN y una red neuronal MLP. La BN con K2 como algoritmo de entrenamiento, *k*NN con la distancia *WeightedMutation* y Euclidiana, y k=3. Para el MLP se procedió de la misma manera que en el epígrafe anterior, se usaron diferentes pruebas variando la cantidad de neuronas de la capa oculta.

No se usan medidas para seleccionar los clasificadores de base, sólo se escogieron aquellos que habían dado mejores resultados. Por eso se seleccionaron: BN, J48 y SVM polinomial grado 2, sin dejar fuera el MLP, el cual se usará como metaclasificador. En el modelo MEHI se probaron variantes del umbral para decir cuáles instancias del grupo de casos bien clasificados por todos los clasificadores se van a poner como casos que caractericen un clasificador determinado: 0, 0.5, 0.7, 0.9 y 1, seleccionando para cada base aquel que produzca un mejor resultado.

Se escogieron *bagging*, *boosting* y *stacking* para comparar los resultados obtenidos con MEHI. Para *bagging* y *boosting* se realizaron pruebas usando como clasificadores de base J48, BN, SVM y MLP, siempre entrenando 10 clasificadores. En el caso de *stacking* se seleccionó la misma topología que en MEHI: J48, BN y SVM como clasificadores de base y MLP como metaclasificador. Para todos los experimentos se realizaron 10 corridas usando el método de validación cruzada con 10 subconjuntos y se promediaron los resultados.

IV.4.2. Evaluación del método

En los anexos 15 y 16 se muestran las tablas de los resultados de exactitud y AUC para cada uno de los multiclasificadores viendo que estos oscilan entre 77.98% (SQV) y 94.44% (RTV) para la exactitud y entre 84%(SQV) y 98.52% (RTV) para el AUC. La Tabla IV.7 muestra los resultados de los promedios de estos resultados para todas las bases, dando una idea del comportamiento general de estos algoritmos. Puede verse que de manera general MEHI supera al resto de los multiclasificadores.

Se utilizaron pruebas estadísticas para comparar los resultados de estos multiclasificadores, como se hizo antes en la comparación de la RNN con el resto de los clasificadores. Se seleccionaron también las pruebas de Friedman y Wilcoxon.

La Tabla IV.7 muestra los resultados obtenidos en exactitud y AUC en los diferentes multiclasificadores que fueron comparados con MEHI. Puede verse que en el caso de la exactitud se obtienen valores superiores para MEHI tanto con la combinación de salida por selección como por fusión para la mayoría de las comparaciones.

Tabla IV.7 Resultados promedios para las 7 bases de la proteasa del VIH, de los diferentes multiclasificadores para la exactitud y el AUC.

		Exactitud	**AUC**
Boosting	BN	0.8587	0.9094
	J48	0.8225	0.8134
	MLP	0.83	0.8708
	SVM	0.8268	0.8684
	SVMP2	0.8351	0.8999
Bagging	BN	0.8548	0.9034
	J48	0.8353	0.8782
	MLP	0.8422	0.8661
	SVM	0.8458	0.867
	SVMP2	0.8442	0.895
	Stacking	0.853	0.8854
	MEHI Selección	**0.8782**	**0.9134**
	MEHI Fusión	**0.8751**	**0.9214**

En la Tabla IV.8 se muestran los resultados de la prueba de Wilcoxon, ordenada por los valores de los rangos de la exactitud según Friedman, para comparar el método MEHI contra el resto de los multiclasificadores: *bagging*, *boosting* y *stacking*, cada uno de ellos con los diferentes modelos de clasificación seleccionados. La tercera y cuarta columna de la tabla muestran el resultado de significación de esta prueba para las dos combinaciones de clasificadores propuestas; la tercera usando la estrategia de selección y la última usando fusión. En negritas están advertidos los casos en que los resultados de los multiclasificadores clásicos son similares a los obtenidos con MEHI, estos son *boosting* con J48 y con MLP, y *bagging* con MLP. En todos los restantes casos, la significación es menor que 0.05 a favor de MEHI.

Tabla IV.8 Resultados de la prueba de Wilcoxon, ordenada por los valores de los rangos, de la exactitud de MEHI usando selección y fusión como métodos de combinación de salida contra *boosting* y *bagging* usando J48, SVM polinomial grado 2 y MLP como clasificadores de base.

		Rangos medios	**Significación Exacta (2 colas)**	**Significación Exacta (2 colas)**
	MEHI_Selección **MEHI_Fusión**	14.71 14.86	**vs**	 **vs**
Boosting	BN	7.29	.016	.016
	J48	12.86	**.063**	**.063**
	MLP	9.14	**.297**	**.297**
	SVM	7.71	.016	.016
	SVMP2	7.29	.016	.016
Bagging	BN	8.71	.016	.016
	J48	12.29	**.125**	.031
	MLP	10.14	**.297**	**.297**
	SVM	9.14	.016	.016
	SVMP2	10.64	.016	.016
	Stacking	12.21	.016	.016

De aquí se concluye que la técnica de fusión resulta un poco mejor que la de selección. En cuanto a exactitud los resultados obtenidos por MEHI son similares a los obtenidos con *bagging* y *boosting* usando MLP como clasificador de base, pero superior al resto de los multiclasificadores.

Por otro lado, los resultados de las pruebas estadísticas del AUC se muestran en la Tabla IV.9. Aquí también se advierten en negritas los casos en que los resultados son similares. Estos son: MEHI con combinación por selección contra los resultados de *boosting* y

bagging con J48. Sin embargo, puede verse que en el caso de MEHI usando fusión siempre es significativamente superior al resto de los multiclasificadores.

Tabla IV.9 Resultados de la prueba de Wilcoxon, ordenada por los valores de los rangos, del AUC de MEHI usando selección y fusión como métodos de combinación de salida contra *boosting* y *bagging* usando J48, SVM polinomial grado 2 y MLP como clasificadores de base.

		Rangos Medios	**Significación Exacta (2 colas)**	**Significación Exacta (2 colas)**
	MEHI_Selección	12.86	**vs.**	
	MEHI_Fusión	12.57		**vs.**
Boosting	BN	9.11	.016	.016
	J48	11.86	**.297**	.016
	SVM	7.86	.016	.031
	SVMP2	5.86	.016	.016
Bagging	BN	8.86	.016	.016
	J48	11.29	**.109**	.047
	KNNWM	6.29	.016	.016
	SVM	7.71	.047	.016
	SVMP2	6.71	.016	.016
	Stacking	9.71	.031	.031

Tabla IV.10 Resultados de la prueba de Wilcoxon para comparar el indicador de AUC de RNN, con la función moda como combinación de salida, y *bagging*, *boosting* y MEHI.

RNN vs.		**Rangos Medios** 11.00	**Significación Exacta (2 colas)**
Boosting	J48	**11.86**	**.469**
	BN	3.00	.016
	SVM	8.14	.156
	SVMP2	5.86	.016
Bagging	J48	**11.57**	**1.000**
	BN	9.14	.469
	SVM	7.71	.078
	SVMP2	6.86	.031
MEHI	Fusion	**13.43**	**.016**
	Selection	**14.57**	.297

Así se obtiene un multiclasificador que mejora los resultados obtenidos para las siete bases del VIH con respecto a los demás multiclasificadores clásicos. Ahora sólo resta comparar con los resultados obtenidos con la RNN en el epígrafe anterior. Teniendo en cuenta que los resultados de la función máxima probabilidad fueron los mejores

obtenidos tanto para la exactitud como para el AUC entonces se compara contra esos resultados.

La Tabla IV.10 muestra los resultados de la prueba de Wilcoxon para esta comparación. En negritas se resaltan los resultados que tienen mayores rangos medios que el rango medio de la RNN (11), y se analiza si la significación de la prueba fue superior o no. En el caso de *boosting* y *bagging* con J48 los resultados son similares a los obtenidos con la RNN. Pero en el caso de MEHI con combinación por fusión se puede ver que los resultados son significativamente superiores a los obtenidos por la RNN.

IV.5. Conclusiones parciales

La ventaja de las energías de contacto de aminoácidos para la representación de las secuencias de aminoácidos con información de la estructura tridimensional, deriva en la obtención de buenos resultados en el análisis de resistencia del VIH ante diferentes fármacos.

La medida de distancia propuesta, basada en los rasgos, muestra que la extracción de estos es una alternativa para disminuir la dimensionalidad en este problema y aumentar los porcentajes de buena clasificación.

La medida de distancia basada en casos ratifica que las energías son una buena representación para las secuencias del VIH y ayudan a un mejor análisis de los datos ya que permite disminuir los rasgos de manera que los casos puedan ser visualizados.

Las RNN obtienen resultados superiores o al menos similares a otros modelos de clasificación: J48. BN, SVM, *k*NN, en el problema de la predicción de resistencia del VIH. De manera general son útiles para resolver problemas de clasificación de secuencias dinámicas, cuando se está en presencia de inserciones y eliminaciones de aminoácidos.

MEHI con la topología: J48, BN y SVM como clasificadores de base y MLP como metaclasificador reporta resultados significativamente superiores comparados con los resultados de *bagging*, *boosting* y *stacking*. La comparación de la RNN con el multiclasificador MEHI con fusión arroja resultados favorables a éste. MEHI puede usarse en problemas con secuencias de longitud fija o si contamos con un mecanismo fiable de alineación de secuencias.

CONCLUSIONES

1. Se demostró que las energías de contacto de los aminoácidos constituyen buenos descriptores de la secuencia mutada de la proteasa del VIH para predecir la resistencia de la misma ante sus inhibidores porque tales descriptores han conducido a mejores resultados de clasificación que los que se habían reportado hasta el momento.
2. Se propusieron dos nuevas medidas de distancia basadas en energías de contacto: una entre secuencias que ayuda a visualizar las mismas y separarlas según la resistencia y otra entre rasgos que facilita la extracción de éstos. Se demostró que ambas medidas contribuyen a mejorar el proceso de clasificación.
3. El modelo diseñado de redes neuronales recurrentes bidireccionales con un módulo de combinación de salidas puede ser usado en problemas de clasificación. Se evidenció la superioridad del nuevo modelo ante varios otros clasificadores si la obtención de la clase está relacionada con rasgos en alguna variante no lineal: polinomial o por ramas. En particular para el problema de predicción de resistencia de la proteasa del VIH el modelo de redes neuronales recurrentes muestra resultados similares o superiores a los obtenidos por el resto de los clasificadores frente a los siete fármacos, desde el punto de vista de la exactitud y el área bajo la curva ROC de la clasificación.
4. El nuevo modelo multiclasificador diseñado (MEHI), basado en el uso de varios clasificadores que dividen la base de casos en grupos de instancias bien clasificadas que luego son aprendidos por un metaclasificador, arrojó resultados que fueron similares o significativamente superiores a los obtenidos con algunos de los multiclasificadores más reconocidos y usados en la literatura. En particular, el uso de MEHI con una combinación de salida por fusión, obtuvo siempre resultados significativamente superiores al resto.

Se concluye en general que el uso de RNN es muy útil para resolver problemas de clasificación de secuencias de longitud variable, cuando se está en presencia de inserciones y eliminaciones de aminoácidos. Pero si se tienen secuencias de longitud fija, o se cuenta con un mecanismo fiable de alineación de secuencias, entonces se recomienda usar MEHI.

RECOMENDACIONES

1. Utilizar el modelo de clasificación basado en redes recurrentes en otros problemas de bioinformática, donde la longitud de las secuencias sea variable, como la predicción de resistencia de otras proteínas ante inhibidores.
2. Buscar descriptores que caractericen los fármacos, de forma que se reduzcan las siete bases a una única, donde sea muy fácil añadir nuevos fármacos.
3. Buscar heurísticas que faciliten la selección de la mejor topología de red recurrente.
4. Buscar medidas de diversidad, o heurísticas para seleccionar los clasificadores de base en el modelo multiclasificador.
5. Utilizar el modelo de multiclasificadores, quizás con algunas adaptaciones, en otros problemas complejos de bioinformática, por ejemplo la clasificación de verdaderos y falsos sitios de *splicing*.

REFERENCIAS BIBLIOGRÁFICAS

1. Kontijevskis, A., Prusis, P., Petrovska, R., Yahorava, S., Mutulis, F., Mutule, I., Komorowski, J., Wikberg, J.E.S., *A look inside HIV resistance through retroviral protease interaction maps.* PLOS Computational Biology, 2007. 3(3): p. 424-435.
2. *Macrogen DNA Sequencing Service*. 2005. http://www.nucleics.com/DNA_sequencing_support/sequencing-service/macrogen.html.
3. *Genomex DNA Sequencing Service*. 2008. http://www.genomex.com/.
4. *HIV Databases*. Los Alamos National Laboratories. 2005. http://www.hiv.lanl.gov.
5. *HIV Drug Resistance Database*. 2008. http://hivdb.stanford.edu//cgi-bin/GenoPhenoDS.cgi.
6. Sevin, A.D., DeGruttola, V., Nijhuis, M., *Methods for investigation of the relationship between drug susceptibility phenotype and human immunodeficiency virus type 1 genotype with applications to AIDS.* The Journal of Infectious Diseases, 2000. 182: p. 59-67.
7. Wang, D., Larder, B., *Enhanced Prediction of Lopinavir Resistance from Genotype by Use of Artificial Neural Networks.* The Journal Of Infectious Diseases 2003. 188: p. 653-660.
8. Draghici, S., Potter, R.B., *Predicting HIV drug resistance with neural networks.* Bioinformatics, 2003. 19(1): p. 98-107.
9. Beerenwinkel, N., Daumer, M., Oette, M., Korn, K., Hoffmann, D., Kaiser, R., Lengauer, T., Selbig, J., Walter, H., *Geno2pheno: estimating phenotypic drug resistance from HIV-1 genotypes.* Nucl. Acids Res., 2003. 31(13): p. 3850-3855.
10. Beerenwinkel, N., Schmidt, B., Walter, H., Kaiser, R., Lengauer, T., Hoffmann, D., Korn, K., Selbig, J. *Diversity and complexity of HIV-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype.* in *National Academic of Sciences*. 2002. USA.
11. Murray, R.J., *Human Immunodeficiency Virus Type 1 Drug Resistance From Genotype Using Machine Learning*. 2004, University of Edinburgh.
12. *Agence Nationale de recherches sur le SIDA*. 2004. http://www.hivfrenshresistance.org.
13. Cao, Z.W., Han, L.Y., Zheng, C.J., Ji, Z.L., Chen, X., Lin, H.H., Chen, Y.Z., *Computer prediction of drug resistance mutations in proteins.* Drug Discovery Today, 2005. 10(7): p. 521-529.
14. Rabinowitz, M., Myers, L., Banjevic, M., Chan, A., Sweetkind-Singer, J., Haberer, J., McCann, K., Wolkowicz, R., *Accurate prediction of HIV-1 drug response from the reverse transcriptase and protease amino acid sequences using sparse models created by convex optimization.* Bioinformatics, 2006. 22(5): p. 541-549.
15. Rhee, S.-Y., Taylor, J., Wadhera, G., Ben-Hur, A., Brutlag, D.L., Shafer, R.W., *Genotypic predictors of human immunodeficiency virus type 1 drug resistance.* Proceedings of the National Academy of Sciences, 2006. 103(46): p. 17355-17360.
16. Shafer, R.W., *Genotypic testing for human immunodeficiency virus type 1 drug resistance.* Clinical Microbiology Reviews, 2002. 15(2): p. 247.

17. Beerenwinkel, N., Sing, T., Lengauer, T., Rahnenfuhrer, J., Roomp, K., Savenkov, I., Fischer, R., Hoffmann, D., Selbig, J., Korn, K., Walter, H., Berg, T., Braun, P., Fatkenheuer, G., Oette, M., Rockstroh, J., Kupfer, B., Kaiser, R., Daumer, M., *Computational methods for the design of effective therapies against drug resistant HIV strains.* Bioinformatics, 2005. 21(21): p. 3943-3950.
18. Murray, R.J., *Predicting Human Immunodeficiency Virus Type 1 Drug Resistance from Genotype Using Machine Learning*. 2004, University of Edinburgh.
19. Jensen, M.A., Li, F.S., van 't Wout, A.B., Nickle, D.C., Shriner, D., He, H.X., McLaughlin, S., Shankarappa, R., Margolick, J.B., Mullins, J.I., *Improved coreceptro usage prediction and genotypic monitoring of R5 to X4 transition by motif analysis of human imnunodeficiency virus type 1 env V3 loop secuences.* Journal of Virology, 2003. 77(24): p. 13376-13388.
20. Dayhoff, M.O., Schwartz, R., Orcutt, B.C., *A model of evolutionary change in proteins.* Atlas of Protein Sequence and Structure, 1978. 5: p. 345-352.
21. Henikoff, S., Henikoff, J., *Amino acid substitution matrices from protein blocks.* Proccedings of the National Academy of Sciences of the United States of America. PNAS 1992. 89: p. 10915-10919.
22. Miyazawa, S., Jernigan, R.L., *Protein stability for single substitution mutants and the extent of local compactness in the denatured state.* . Protein Eng. , 1994. 7: p. 1209-1220.
23. Miyazawa, S., Jernigan, R.L., *Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading.* J. Mol. Biol. , 1996. 256: p. 623-644.
24. Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armananzas, R., Santafe, G., Perez, A., Robles, V., *Machine learning in bioinformatics.* Briefings in Bioinformatics, 2006. 7(1): p. 86-112.
25. Baldi, P., Soren, B., *Bioinformatics: The Machine Learning Approach*. 2 ed. 2001: MIT Press. .
26. Saeys, Y., *Feature Selection for Classification of Nucleic Acid Sequences*, in *Faculty of Sciences*. 2004, University of Ghent.
27. Grau, R., Chavez, M.C., Sánchez, R., Morgado, E., Casas, G., Bonet, I., *Bolean Algebraic Structures of the Genetic Code Posibilities of Applications.* Lecture Notes on Bioinformatics, 2006. 4366: p. 10-21.
28. Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., Soda, G., *Bidirectional Dynamics for Protein Secondary Structure Prediction.* In: R. Sun and C.L. Gile (eds.) Sequence Learning, LNAI, 2000. 1828: p. 80-104.
29. Baldi, P., *New Machine Learning Methods for the Prediction of Protein Topologies*. Artificial Intelligence and Heuristic Methods for Bioinformatics. 2002: In: P. Frasconi and R. Shamir (eds.) Artificial Intelligence and Heuristic Methods for Bioinformatics, IOS Press.
30. Saeys, Y., Inza, I., Larrañaga, P., *A review of feature selection techniques in bioinformatics.* Bioinformatics, 2007. 23(19): p. 2507–2517.
31. Yang, L. *An overview of distance metric learning*. 2007. www.cse.msu.edu/~yangliu1/dist_overview.pdf
32. Jolliffe, I.T., *Principal Component Analysis*. 1986, New York: Springer-Verlag.
33. Cox, T., Cox, M., *Multidimensional Scaling. Chapman and Hall.* 1994.

34. Fisher, R.A., *The utilization of multiple measurements in taxonomic problems.* Annu. Eugenics, 1936. 7: p. 179-188.
35. Kohonen, T., *Self-Organization and Associative Memory*. 3ra ed, ed. S.i.s. series. 1989, New York: Springer-Verlag.
36. Ultsch, A., *U-Matrix: a Tool to visualize Clusters in high dimensional Data*. 2003, University of Marburg, Department of Computer Science.
37. Duda, R.O., Hart, P.E., Stork, D.G., *Pattern Classification*. 2nd ed. 1997: Wiley-Interscience.
38. Slonim, N., Tishby, N. *The power of word clusters for text classification*. in *23rd European Colloquium on Information Retrieval Research*. 2001.
39. Steinbach, M., Karypis, G., Kumar, V. *A comparison of document clustering techniques*. in *KDD Workshop on Text Mining*. 2000.
40. Fodor, I.K., *A survey of dimension reduction techniques* in *Technical report*. 2002, LLNL.
41. Bhaskar, H., Hoyle, D.C., Singh, S., *Machine learning in bioinformatics: A brief survey and recommendations for practitioners.* Computers in Biology and Medicine, 2006. 36(10): p. 1104-1125.
42. Salzberg, S.L., Delcher, A.L., Kasif, S., White, O., *Microbial gene identification using interpolated Markov models.* Nucleic acids research, 1998. 26(2): p. 544-548.
43. Delcher, A.L., Harmon, D., Kasif, S., White, O., Salzberg, S.L., *Improved microbial gene identification with GLIMMER.* Nucleic acids research, 1999. 27(23): p. 4636-4641.
44. Degroeve, S., De Baets, B., Van de Peer, Y., Rouzé, P., *Feature subset selection for splice site prediction.* Bioinformatics, 2002. 18(2): p. 75-83.
45. Saeys, Y., Degroeve, S., Aeyels, D., Rouzé, P., Van de Peer, Y., *Feature selection for splice site prediction: A new method using EDA-based feature ranking.* BMC Bioinformatics, 2004. 5(1): p. 64.
46. Liu, H., Han, H., Li, J., Wong, L., *Using amino acid patterns to accurately predict translation initiation sites.* In silico biology, 2004. 4(3): p. 255-269.
47. Jafari, P., Azuaje, F., *An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors.* BMC Med. Inform. Decis. Mak., 2006. 6(27).
48. Saeys, Y., Rouzé, P., Van de Peer, Y., *In search of the small ones: improved prediction of short exons in vertebrates, plants, fungi and protists.* Bioinformatics, 2007. 23(4): p. 414-420.
49. Zavaljevski, N., Stevens, F.J., Reifman, J., *Support vector machines with selective kernel scaling for protein classification and identification of key amino acid positions.* Bioinformatics, 2002. 18(5): p. 689-696.
50. Conilione, P., Wang, D., *A comparative study on feature selection for E.coli promoter recognition.* International Journal of Information Technology, 2005. 11(8): p. 54–66.
51. Kim, S.-K., Nam, J.-W., Rhee, J.-K., Lee, W.-J., Zhang, B.-T., *miTarget: microRNA target gene prediction using a support vector machine.* BMC Bioinformatics, 2006. 7(1): p. 411.

52. Li, L., Umbach, D.M., Terry, P., Taylor, J.A., *Application of the GA/KNN method to SELDI proteomics data.* Bioinformatics, 2004. 20(10): p. 1638-1640.
53. Petricoin, E., Ardekani, A., Hitt, B., Levine, P., Fusaro, V., Steinberg, S., Mills, G., Simone, C., Fishman, D., Kohn, E., *Use of proteomic patterns in serum to identify ovarian cancer.* The Lancet , Volume 359 , Issue 9306 , Pages 572 - 577, 2002. 359(9306): p. 572 - 577.
54. Ressom, H.W., Varghese, R.S., Abdel-Hamid, M., Eissa, S.A.-L., Saha, D., Goldman, L., Petricoin, E.F., Conrads, T.P., Veenstra, T.D., Loffredo, C.A., Goldman, R., *Analysis of mass spectral serum profiles for biomarker selection.* Bioinformatics, 2005. 21(21): p. 4039-4045.
55. Ressom, H.W., Varghese, R.S., Drake, S.K., Hortin, G.L., Abdel-Hamid, M., Loffredo, C.A., Goldman, R., *Peak selection from MALDI-TOF mass spectra using ant colony optimization.* Bioinformatics, 2007. 23(5): p. 619-626.
56. Diaz-Uriarte, R., Alvarez de Andres, S., *Gene selection and classification of microarray data using random forest.* BMC Bioinformatics, 2006. 7(1): p. 3.
57. Hubert, M., Engelen, S., *Robust PCA and classification in biosciences.* Bioinformatics, 2004. 20(11): p. 1728-1736.
58. Ewing, R.M., Cherry, J.M., *Visualization of expression clusters using Sammon's non-linear mapping.* Bioinformatics, 2001. 17(7): p. 658-659.
59. Hwang, D., Schmitt, W.A., Stephanopoulos, G., Stephanopoulos, G., *Determination of minimum sample size and discriminatory expression patterns in microarray data.* Bioinformatics (Oxford, England), 2002. 18(9): p. 1184-1193.
60. Cho, J.H., Lee, D., Park, J.H., Lee, I.B., *Gene selection and classification from microarray data using kernel machine.* FEBS letters, 2004. 571(1-3): p. 93-98.
61. Bhanot, G., Alexe, G., Venkataraghavan, B., Levine, A.J., *A robust meta-classification strategy for cancer detection from MS data.* Proteomics, 2006. 6(2): p. 592-604.
62. Tibshirani, R., Hastie, T., Narasimhan, B., Soltys, S., Shi, G., Koong, A., Le, Q.-T., *Sample classification from protein mass spectrometry, by 'peak probability contrasts'.* Bioinformatics, 2004. 20(17): p. 3034-3044.
63. Jain, A.K., Murty, M.N., Flynn, P.J., *Data clustering: A review.* ACM Computing Surveys, 1999. 31(3): p. 264-323.
64. McQueen, J. *Some methods for classification and analysis of multivariate observations*. in *Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1967.
65. Berkhin, P., *Survey of Clustering Data Mining Techniques*, in *Technical report*. 2002, Accrue Software: San Jose, CA.
66. Wang, W., Yang, J., Muntz, R. *STING: a statistical information grid approach to spatialdata mining.* in *23rd Conference on VLDB*. 1997. Athens, Greece.
67. Sheikholeslami, G., Chatterjee, S., Zhang, A. *WaveCluster: A multiresolution clustering approach for very large spatial databases*. in *24th Conference on VLDB*. 1998. New York, NY.
68. Huang, Z. *Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values*. in *Workshop on Research Issues on Data Miing and Knowledge Discovery*. 1998.

69. Guha, S., Rastogi, R., Shim, K. *ROCK: A robust clustering algorithm for categorical attributes*. in *15th ICDE*. 1999. Sydney, Australia.
70. Hall, L.O., Ozyurt, B., Bezdek, J.C., *Clustering with a genetically optimized approach.* IEEE Transaction on Evolutionary Computation, 1999. 3(2): p. 103-112.
71. Fisher, D., *Knowlege acquisition via incremental coneptual clustering.* Machine Learning, 1987. 2: p. 139-172.
72. Witten, I., Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques*. 2 ed. Morgan Kaufmann, ed. M.R. Jim Gray. 2005, San Francisco: Diane Cerra. 525.
73. Grau, R., *Estadística aplicada con ayuda de paquetes de software*. 1994, Guadalajara, Jalisco, México: Editorial Universitaria.
74. Cooper, G., Herskovits, E., *A Bayesian method for the induction of probabilistic networks from data.* Machine Learning, 1992. 9(4): p. 309-347.
75. Mitchell, T.M., *Machine Learning*. 1997: McGraw-Hill Science/Engineering/Math; (March 1, 1997).
76. Quinlan, J.R., *Induction of decision trees.* Machine Learning, 1986. 1(1): p. 81-106.
77. Quinlan, J.R., *C4.5: Programs for Machine Learning.* 1993, San Mateo,CA: Morgan Kaufmann.
78. Aha, D., Kibler, D., *Instance-based learning algorithms.* Machine Learning, 1991. 6: p. 37-66.
79. Cleary, J.G., Trigg, L.E. *K*: An Instance-based Learner Using an Entropic Distance Measure*. in *12th International Conference on Machine Learning*. 1995.
80. Vapnik, V., *The Nature of Statistical Learning Theory*. 1995, New York: Springer-Verlag.
81. Hammer, B., Villmann, T. *Mathematical Aspects of Neural Networks*. in *European Symposium on Artificial Neural Networks 2003*. 2003.
82. Lippmann, R.P., *An introduction to computing with neural nets*. Vol. 16. 1988: ACM. 7-25.
83. Pearlmutter, B., *Dynamic Recurrent Neural Networks.* DARPA Research, 1990.
84. Hilera, J.R., Martínez, V.J., *Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones*. 1995, Madrid / Mexico: RA-MA / Addison Wesley Iberoamericana.
85. Ronald, J.W., David, Z., *Gradient-based learning algorithms for recurrent networks and their computational complexity*, in *Backpropagation: theory, architectures, and applications*, Y. Chauvin, D.E. Rumelhart, (eds). 1995, Lawrence Erlbaum Associates, Inc. p. 433-486.
86. Atiya, A.F., Parlos, A.G., *New Results on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence.* IEEE Transactions on Neural Networks, 2000. 11(3): p. 697.
87. Polikar, R., *Ensemble based systems in decision making.* IEEE Circuits and Systems Magazine, 2006. 6(3): p. 21-44.
88. Dietterich, T.G., *Ensemble methods in machine learning*, in *Multiple Classifier Systems*. 2000, Springer-Verlag Berlin: Berlin. p. 1-15.

89. Kuncheva, L.I., *Combining Pattern Classifiers, Methods and Algorithms*. 2004, New York, NY: Wiley Interscience.
90. Kuncheva, L.I., Whitaker, C.J., *Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy.* Machine Learning, 2003. 51(2): p. 181-207.
91. Brown, G., Wyatt, J., Harris, R., Xin, Y., *Diversity creation methods: a survey and categorisation.* Information Fusion, 2005. 6(1): p. 5-20.
92. Duin, R.P. *The combining classifier: to train or not to train?* in *16th International Conference on Pattern Recognition, ICPR'02*. 2002. Canada.
93. Kuncheva, L.I., *"Fuzzy" vs "non-fuzzy" in combining classifiers designed by boosting.* IEEE Transactions on Fuzzy Systems, 2003. 11: p. 729-741.
94. Kuncheva, L.I., Bezdek, J.C., Duin, R., *Decision templates for multiple classifier fusion: An experimental comparison.* Pattern Recognition, 2001. 34(2): p. 299-314.
95. Breiman, L., *Bagging predictors.* Machine Learning, 1996. 24: p. 123-140.
96. Breiman, L., *Random Forests.* Machine Learning, 2001. 45: p. 5–32.
97. Schapire, R.E., *The strength of weak learnability.* Machine Learning, 1990. 5(2): p. 197-227.
98. Freund, Y., Schapire, R.E., *Decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, 1997. 55: p. 119–139.
99. Wolpert, D., *Stacked generalization.* Neural Networks, 1992. 5(2): p. 241-259.
100. Jacobs, R.A., Nowlan, S.J., Hinton, G.E., *Adaptative mixtures of local experts.* Neural Computation, 1991. 3: p. 79-87.
101. Nguyen, M.H., Abbass, H.A., McKay, R.I., *A novel mixture of experts model based on cooperative coevolution.* Neurocomputing, 2006. 70(1-3): p. 155-163.
102. Saha, S., Murthy, C.A., Pal, S.K., *Rough set based ensemble classifier for web page classification.* Fundamenta Informaticae, 2007. 76(1-2): p. 171-187.
103. Dimitrakakis, C., Bengio, S., *Online adaptive policies for ensemble classifiers.* Neurocomputing, 2005. 64: p. 211-221.
104. Partalas, I., Tsoumakas, G., Katakis, I., Vlahavas, I., *Ensemble pruning using reinforcement learning*, in *Advances in Artificial Intelligence, Proceedings*. 2006, Springer-Verlag Berlin: Berlin. p. 301-310.
105. Nanni, L., Lumini, A., *FuzzyBagging: A novel ensemble of classifiers.* Pattern Recognition, 2006. 39(3): p. 488-490.
106. Mathé, C., Sagot, M.F., Schlex, T., Rouzé, P., *Current methods of gene prediction, their strengths and weaknesses.* Nucleic Acids Research, 2002. 30(19): p. 4103-4117.
107. Salzberg, S., *Localing protein coding regions in human DNA using a decision tree algorithm.* Journal of Computational Biology, 1995. 2: p. 473-485.
108. Castelo, R., Guigó, R., *Splice site identification by idlBNs.* Bioinformatics, 2004. 20(1): p. 169-176.
109. Carter, R.J., Dubchak, I., Holbrook, S.R., *A computational approach to identify genes for functional RNAs in genomic sequence.* Nucleic Acids Research, 2001. 29(19): p. 3928-3938.

110. Kim, S., *Protein b-turn prediction using nearest-neighbor method.* Bioinformatics, 2004. 20(1): p. 40-44.
111. Selbig, J., Mevissen, T., Lengauer, T., *Decision tree based formation of consensus protein secondary structure prediction.* Bioinformatics, 1999. 15(12): p. 1039-1046.
112. Yang, C., Dobbs, D., Honavar, V., *A two-stage classifier for identification of protein-protein interface residues.* Bioinformatics, 2004. 20: p. 371-378.
113. Huang, Y., Li, Y., *Prediction of protein subcellular locations using fuzzy k-NN mathos.* Bioinformatics, 2004. 20(1): p. 21-28.
114. Krishnapuram, B., Carin, L., Hartemink, A.J., *Joint classifier and feature optimization for comprehensive cancer diagnosis using gene expression data.* Journal of Computational Biology, 2004. 11(2-3): p. 227-242.
115. Dudoit, S., Fridlyand, J., Speed, P., *Comparison of discrimination methods for classification of tumors using gene expression data.* Journal of the American Statistical Association, 2002. 97: p. 77-87.
116. Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., Levy, S., *A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis.* Bioinformatics, 2005. 21(5): p. 631-643.
117. González-Díaz, H., Bonet, I., Terán, C., De Clercq, E., Bello, R., García, M.M., Santana, L., Uriarte, E., *ANN-QSAR model for selection of anticancer leads from structurally heterogeneous series of compounds.* European Journal of Medicinal Chemistry, 2007. 42: p. 580-585.
118. James, R., *Predicting Human Immunodeficiency Virus Type 1 Drug Resistance from Genotype Using Machine Learning*. 2004, University of Edinburgh.
119. Wang, D.C., Larder, B., *Enhanced prediction of lopinavir resistance from genotype by use of artificial neural networks.* Journal Of Infectious Diseases, 2003. 188(5): p. 653-660.
120. Provost, F., Fawcett, T. *Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions*. 1997.
121. Beerenwinkel, N., Schmidt, B., Walter, H., Kaiser, R., Lengauer, T., Hoffmann, D., Korn, K., Selbig, J., *Diversity and complexity of HIV-1 drug resistance: A bioinformatics approach to predicting phenotype from genotype.* PNAS, 2002. 99(12): p. 8271-8276.
122. Grau, R., Correa, C., Rojas, M., *Metodología de la investigación*. 2da ed. 2004, Colombia: EL POIRA S.A.
123. Asuncion, A., Newman, D.J. *UCI Machine Learning Repository*. 2007. http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html.
124. Canuto, A.M.P., Abreu, M.C.C., Oliveira, L.D., Xavier, J.C., Santos, A.D., *Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles.* Pattern Recognition Letters, 2007. 28(4): p. 472-486.
125. Hansen, L.K., Salamon, P., *Neural networks ensembles.* IEEE Transactions on Pattern Analisys and Machine Intelligence, 1990. 12(10): p. 993-1001.
126. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P., *Ensemble diversity measures and their application to thinning.* Information Fusion, 2005. 6(1): p. 49-62.

127. Tang, E.K., Suganthan, P.N., Yao, X., *An analysis of diversity measures.* Machine Learning, 2006. 65(1): p. 247-271.
128. Bonet, I., Rodríguez, A., Grau Ábalo, R., García, M.M., Saeys, Y., Nowé, A., *Comparing distance measures with visual methods*, in *MICAI 2008, LNAI 5317*, A. Gelbukh, E.F. Morales, (eds). 2008, Springer-Verlag Berlin Heidelberg. p. 90-99.
129. Bonet, I., Saeys, Y., Grau, R., García, M.M., Sanchez, R., Van de Peer, Y., *Feature Extraction Using Clustering of Protein*, in *CIARP 2006, LNCS 4225*, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, J. Kittler, (eds). 2006, Springer-Verlag Berlin Heidelberg. p. 614-623.
130. Siegel, S., *Diseño experimental no paramétrico, aplicado a las ciencias de la conducta*. 1970: México, Trillas. ME. .
131. Bonet, I., García, M.M., Grau, R., Sánchez, R. *Prediction of Human Immunodeficiency Virus Drug Resistance Using Contact Energies*. in *International Conference on Neural Networks and Brain*. 2005: IEEE.
132. Bonet, I., García, M.M., Saeys, Y., Van de Peer, Y., Grau, R., *Predicting Human Immunodeficiency Virus Drug Resistance Using Recurrent Neural Networks*, in *IWINAC 2007, LNCS 4527*, J. Mira, J.R. Alvarez, (eds). 2007, Springer-Verlag Berlin Heidelberg. p. 234-243.

PRODUCCIÓN CIENTÍFICA DEL AUTOR SOBRE EL TEMA DE LA TESIS

1. **Bonet, I.**, Salazar, S., García, M.M. Redes neuronales recurrentes para análisis de señales en el tiempo. Memorias de UCIENCIA 2005. I Conferencia Científica de las Ciencias Informáticas. I Taller de Inteligencia Artificial de la UCI. ISBN. 959-16-0318-5.
2. **Bonet, I.**, Salazar, S., García, M.M. Redes neuronales recurrentes para el análisis de secuencias de proteínas. Memorias de Informática'2005. ISBN. 959-7164-87-6.
3. Salazar, S., **Bonet, I.**, García, M.M. Plataforma de software con características extensibles para el manejo de redes neuronales. Memorias de Informática'2005. ISBN. 959-7164-87-6.
4. **Bonet, I.**, Salazar, S., García, M.M., Grau, R., Sánchez, R. Redes neuronales recurrentes bidireccionales como un diseño natural para el tratamiento de secuencias de proteínas. Boletín de la Sociedad Cubana de Matemática y Computación. Vol. 3 No. 1, 2005. ISSN. 17286042.
5. **Bonet, I.**, García, M.M., Grau, R., Sánchez, R., Salazar, S. Prediction of human immunodeficiency virus drug resistance using contact energies. Proceeding of International Conference on Neural Networks and Brain. IEEE Catalog Number: 05EX1177. ISBN. 0-7803-9422-4.
6. **Bonet , I.**, Saeys, Y., Grau, R., García, M M, Sanchez, R., Van de Peer, Y. (2006). Feature extraction using clustering of protein. J.F. Martínez-Trinidad et al. (Eds.): CIARP 2006, Springer-Verlag Berlin Heidelberg LNCS 4225, pp. 614 – 623. ISSN. 0302-9743.
7. **Bonet, I.**, Grau, R, García, M.M. Redes neuronales recurrentes para análisis de secuencias. Memorias de VII Conferencia Científica Internacional UNICA 2006. ISBN. 959-16-0473-4.
8. **Bonet, I.**, Grau, R, Rodríguez, A, García, M.M. Predicción de splice sites usando redes neuronales recurrentes. Memorias de la XII Convención y Expo Internacional de Informática, INFORMÁTICA 2007, ISBN. 978-959-286-002-5.
9. Rodríguez, A, **Bonet, I.** Sistema multiagente para combinar técnicas de aprendizaje automatizado sobre plataforma libre. Memorias de la XII Convención y Expo Internacional de Informática, INFORMÁTICA 2007, ISBN. 978-959-286-002-5.
10. **Bonet, I.**, García, M M., Saeys, Y, Van de Peer, Y, Grau, R. Predicting human immunodeficiency virus drug resistance using recurrent neural networks. J. Mira and J.R. Alvarez (Eds.): IWINAC 2007, Part I, LNCS 4527, pp. 234–243, 2007. Springer-Verlag Berlin Heidelberg 2007. ISSN. 0302-9743.

11. **Bonet, I.**, Rodríguez, A., Grau, R., García, M.M. Combinación de clasificadores para Bioinformática. Memorias del Congreso Nacional de Reconocimiento de Patrones. Ciudad de la Habana, Cuba, 2007. ISBN. 978-959-286-006-3.
12. Rodríguez, A., **Bonet, I.**, Grau, R., García, M.M. Sistema de jueces para la especialización de clasificación. Memorias del Congreso Nacional de Reconocimiento de Patrones. Ciudad de la Habana, Cuba, 2007. ISBN. 978-959-286-006-3.
13. Rodríguez, A., **Bonet, I.**, Grau, R., García, M.M. (2008) Judges system for classifiers specialization, Proceedings in CD of Second International Workshop on Bioinformatics, Cuba- Flanders, 2008. ISBN. 978-959-250-394-6.
14. **Bonet, I.**, Rodríguez, A., Grau, R., García, M.M. (2008). Combining classifiers for Bioinformatics. Proceedings in CD of Second International Workshop on Bioinformatics, Cuba- Flanders, 2008. ISBN. 978-959-250-394-6.
15. **Bonet, I.**, Rodríguez, A, Grau, R, García, M.M., Saeys, Y, Nowé, Ann. Comparing distance measures with visual methods. A.Gelbukh and E.F. Morales (Eds.): MICAI 2008, LNAI 5317, pp. 90-99, 2008. ISSN. 0302-9743. Springer-Verlag Berlin Heidelberg.
16. **Bonet, I.**, Rodríguez, A, Grau, R, García, M.M., Izquierdo, Y. Predicting HIV protease drugs resistance with multiclassifier systems. XVII International AIDS Conference, Mexico, 2008. ISBN. 978-92-95069-05-3.

Aprobados para publicación

17. **Bonet, I.**, Salazar, S, Rodríguez, A., Grau, R., García, M.M. Redes neuronales recurrentes para análisis de secuencias. Revista Cubana de Ciencias Informáticas. Número 4, Volumen 1. ISSN. 1994-1536.
18. **Bonet, I.**, Rodríguez, A., Grau Ábalo, R., García, M.M. Combining classifiers to Bioinformatics. International Journal of Biological and Medical Sciences. World Academy of Science, Engineering and Technology (WASET). ISSN. 1307-7457.
19. Rodríguez, A., **Bonet, I**., Grau, R., García, M.M. Judges system for classifiers specialization. International Journal of Biological and Medical Sciences. World Academy of Science, Engineering and Technology (WASET). ISSN. 1307-7457.

Registro de software

Bonet, I., Salazar, S., García, M. M., Grau, R. NEngine v 1.0: Una Herramienta Software para Redes Neuronales Recurrentes. Reg. CENDA 2528-2005, dic. 2005.

ANEXOS

Anexo 1. Nomenclatura de los aminoácidos

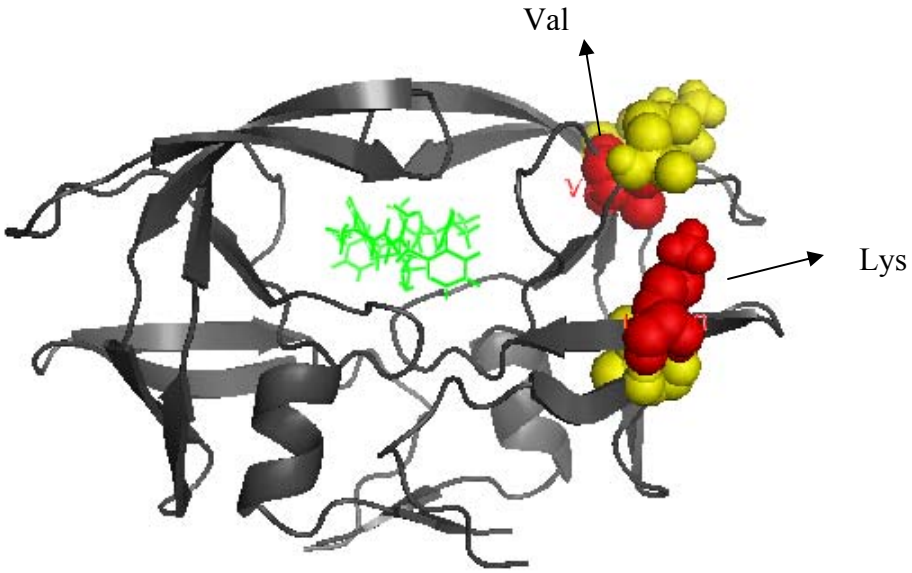
Nombre completo	**Código de tres letras**	**Código de una letra**
Ácido aspártico	Asp	D
Ácido glutámico	Glu	E
Alanina	Ala	A
Arginina	Arg	R
Asparagina	Asn	N
Cisteína	Cys	C
Fenilalanina	Phe	F
Glicina	Gly	G
Glutamina	Gln	Q
Histidina	His	H
Isoleucina	Ile	I
Leucina	Leu	L
Lisina	Lys	K
Metionina	Met	M
Prolina	Pro	P
Serina	Ser	S
Tirosina	Tyr	Y
Treonina	Thr	T
Triptófano	Trp	W
Valina	Val	V

Anexo 2. Tabla del código genético

	G			T			A			C			
	No	I	II	No	I	II	No	I	II	No	I	II	
G	0	GGG	G	16	GTG	V	32	GAG	E	48	GCG	A	G
	1	GGT	G	17	GTT	V	33	GAT	D	49	GCT	A	T
	2	GGA	G	18	GTA	V	34	GAA	E	50	GCA	A	A
	3	GGC	G	19	GTC	V	35	GAC	D	51	GCC	A	C
T	4	TGG	W	20	TTG	L	36	TAG	-	52	TCG	S	G
	5	TGT	C	21	TTT	F	37	TAT	Y	53	TCT	S	T
	6	TGA	-	22	TTA	L	38	TAA	-	54	TCA	S	A
	7	TGC	C	23	TTC	F	39	TAC	Y	55	TCC	S	C
A	8	AGG	R	24	ATG	M	40	AAG	K	56	ACG	T	G
	9	AGT	S	25	ATT	I	41	AAT	N	57	ACT	T	T
	10	AGA	R	26	ATA	I	42	AAA	K	58	ACA	T	A
	11	AGC	S	27	ATC	I	43	AAC	N	59	ACC	T	C
C	12	CGG	R	28	CTG	L	44	CAG	Q	60	CCG	P	G
	13	CGT	R	29	CTT	L	45	CAT	H	61	CCT	P	T
	14	CGA	R	30	CTA	L	46	CAA	Q	62	CCA	P	A
	15	CGC	R	31	CTC	L	47	CAC	H	63	CCC	P	C

Anexo 3. Representación de la energía de contacto

En esta imagen se ven dos ejemplos de aminoácidos representados en color rojo y en color amarillo se muestran los aminoácidos que interactúan con ellos (no se representan todos para permitir su visualización). El caso superior ejemplifica un aminoácido que presenta más contactos con otros aminoácidos que con el solvente (agua), encontrándose hacia el interior hidrofóbico. El segundo caso muestra un comportamiento inverso, y éste se orienta hacia el exterior hidrofílico. Un aminoácido que, como tendencia estadística, se comporte de la manera mostrada por el primer caso en un conjunto representativo de secuencias no relacionadas, tendrá un valor elevado en su energía de contacto, y uno que lo haga como el mostrado en el segundo caso, tendrá un bajo valor de la misma.



Anexo 4. Medidas de distancias

Distancia Euclidiana:	$D(X,Y)=\sqrt{\sum_{i=1}^{N}(x_i-y_i)^2}$
Distancia de Manhattan:	$D(X,Y)=\sum_{i=1}^{N}\lvert x_i-y_i\rvert$
Distancia de Chebychev:	$D(X,Y)=\max_{i=1}^{N}\lvert x_i-y_i\rvert$
Distancia de Hamming:	$D(X,Y)=\sum_{i=1}^{N}d(x_i,y_i)$ $d(x_i,y_i)=\begin{cases}1 & si \quad x_i \neq y_i \\ 0 & si \quad x_i = y_i\end{cases}$

Anexo 5. Algoritmo Backpropagation Trough Time

Sea *N* una red recurrente, *CE* un conjunto de vectores de valores de entrada y salida (conjunto de entrenamiento) y ξ el error medio máximo que se tolerará en el entrenamiento:

BPTT (N, CE, ξ)

1 Para cada c del conjunto *CE* hacer:

2 $E \leftarrow 0$

3 $N' \leftarrow$ *Despliegue* (N, $|c|$)

4 $S \leftarrow$ *Salida* (N', c)

5 $\bar{e} \leftarrow$ *Error*(N', S, c)

6 *Actualización*(N', $\bar{e}$)

7 $N \leftarrow$ *Plegamiento*(N', $|c|$)

8 $E \leftarrow E +$ *Total* ($\bar{e}$)

9 $E \leftarrow E / |CE|$

10 Si $E > \xi$ ir al paso 1

La función *Despliegue*(N, $|c|$) desplegará la red recurrente *N* recibida como parámetro teniendo en cuenta la cardinalidad del caso *c*.

La función *Salida*(N', c) devolverá la salida de la red N' para el caso *c*.

La función *Error* (N', S, c) devolverá el vector de errores cometidos en cada neurona de la red N' para el caso *c* teniendo como salida a *S*.

Actualización(N', $\bar{e}$) ajustará los pesos de las conexiones entre las neuronas de la red N' teniendo en cuenta los errores $\bar{e}$ cometidos en cada neurona.

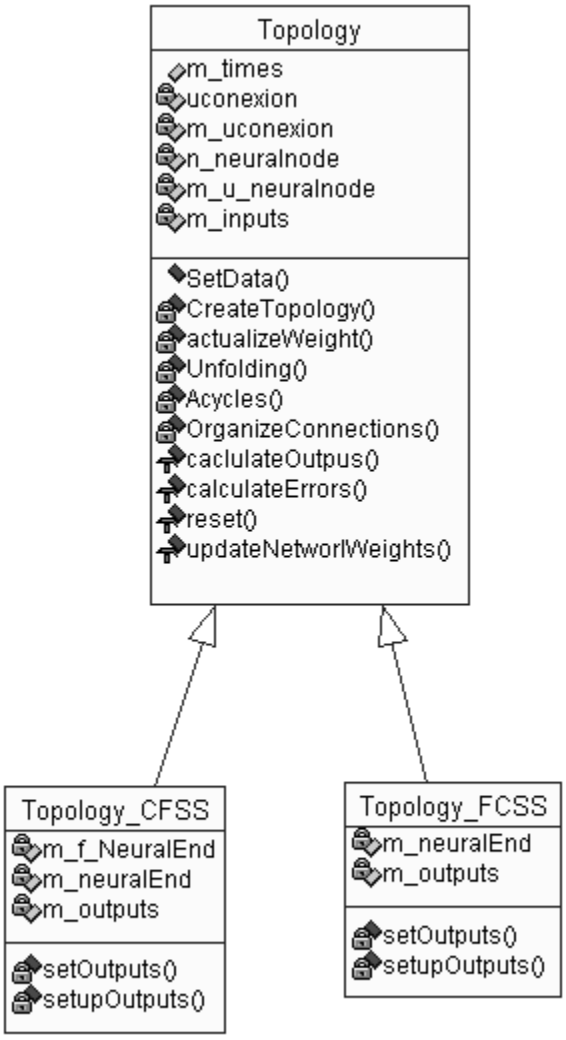
La función *Plegamiento*(N', $|c|$) invertirá el procesamiento de despliegue de la red, agregando los valores de los pesos de conexiones compartidas en único valor.

Total ($\bar{e}$) devolverá el error absoluto cometido por la red.

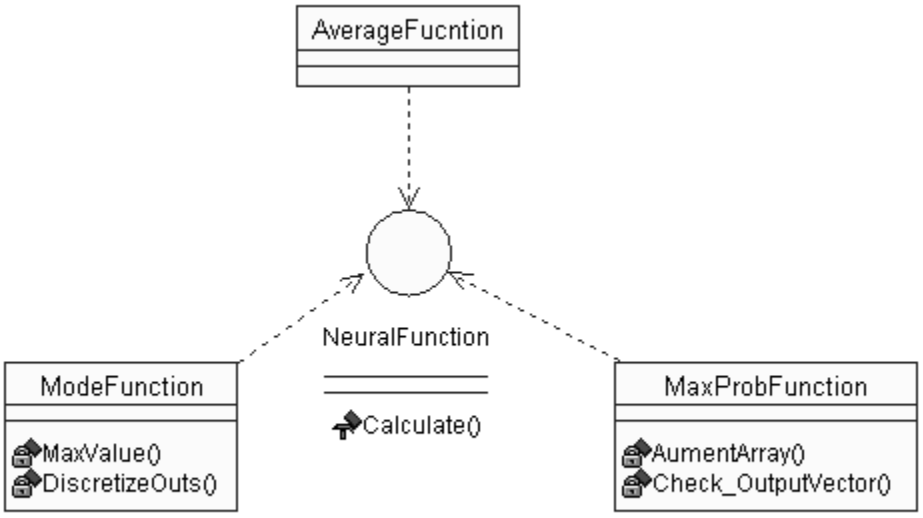
Anexo 6. Energías de contacto de los aminoácidos

			Energías 1996 -0,6*qi*eir/2
Aminoácido	eir 1996	qi, 1996	kcal/mol
A	-2.57	6.334	4.883514
C	-3.57	6.646	7.117866
D	-1.84	6.487	3.580824
E	-1.79	6.235	3.348195
F	-4.76	5.87	8.38236
G	-2.19	6.284	4.128588
H	-2.56	6.241	4.793088
I	-4.42	6.042	8.011692
K	-1.52	6.569	2.995464
L	-4.81	6.087	8.783541
M	-3.92	6.137	7.217112
N	-1.92	6.574	3.786624
P	-2.09	5.858	3.672966
Q	-2	6.469	3.8814
R	-2.11	6.318	3.999294
S	-1.98	6.582	3.909708
T	-2.29	6.486	4.455882
V	-3.89	6.155	7.182885
W	-3.81	5.793	6.621399
Y	-3.41	6.037	6.175851

Anexo 7. Diagrama de clases para el paquete Topology



Anexo 8. Diagrama de clases para el paquete Functions



Anexo 9. Ejemplo de implementación de una función de combinación de salidas

Para añadir una nueva función para mezclar las salidas de los tiempos de la red recurrente bastará con crear una nueva clase que implemente la interfaz *weka.classifiers.functions.rnn.functions.NeuralFunction*. En esta interfaz se declara un método abstracto para calcular un valor de distribución de probabilidades para cada clase a partir de la distribución de probabilidades para cada tiempo. A continuación se muestra la definición de la interfaz *NeuralFunction*:

```
package weka.classifiers.functions.rnn.functions;
import java.io.Serializable;

public interface NeuralFunction extends Serializable{
	public double[] Calculate(double[][] outs) throws Exception;
}
```

Así como un ejemplo de función, en este caso la función media:

```
package weka.classifiers.functions.rnn.functions;

public class AverageFunction implements NeuralFunction{
	private static final long serialVersionUID =
-4716092979156562087L;
	public double[] Calculate(double[][] outs) {
		double[] result = new double[outs[0].length];
		for (int i = 0; i < outs.length; i++) {
			for (int j = 0; j < outs[i].length; j++) {
				result[j]+=outs[i][j];
			}
		}
		for (int i = 0; i < result.length; i++) {
			result[i]/=outs.length;
		}
		return result;
	}
}
```

Anexo 10. Sintaxis para crear el fichero que contiene la información de la topología de la red recurrente

La sintaxis para crear una capa se define de la siguiente forma:

CREATE LAYER *<nombre de la capa>* AS *<tipo de conexión>* WITH *<cantidad de neuronas>*

Donde el *nombre de la capa* es una cadena que identifica a esa capa de las demás.

El *tipo de conexión* va a representar las conexiones que entran y salen de la capa que se está creando. De forma que quede implícito en la capa, cuando se construya, de dónde provienen las conexiones principales que recibe y hacia dónde se dirigen las conexiones que salen de ella. Se toman las notaciones:

- *Lin*, capas de entrada
- *Lmed*, capas ocultas
- *Lout*, capas de salidas

Las posibles variantes son:

- *Lin-Lout*, la capa recibe conexiones desde las entradas y dirige sus salidas hacia capas de salidas.
- *Lin-Lmed* la capa recibe conexiones desde las entradas y dirige sus salidas hacia capas ocultas.
- *Lmed-Lmed* la capa recibe conexiones desde las capas ocultas y dirige sus salidas hacia capas ocultas.
- *Lmed-Lout* la capa recibe conexiones desde las capas ocultas y dirige sus salidas hacia capas de salidas.

Se considera que una capa siempre puede conectarse a capas ocultas aún cuando su tipo de conexión sea *Lin-Lout*.

Con este formato no es necesario preocuparse por crear los enlaces desde capas de entrada ni enlaces hacia las capas de salidas pues al detectar alguna capa que tenga *Lin* o *Lout* se conectan automáticamente a las mismas. La cantidad de neuronas indica cuántas neuronas tiene la capa.

Para especificar las conexiones debe añadirse

CREATE CONNECTION FROM <nombre de la capa origen> TO <nombre de la capa destino> AS DELAY <valor del desplazamiento>

Para realizar las conexiones las capas que se desean conectar deben haberse creado con anterioridad. El desplazamiento puede ser cero, negativo o positivo, o sea, el enlace es hacia una capa del mismo tiempo, hacia una capa de tiempos anteriores o hacia una capa de tiempos posteriores, respectivamente.

A continuación se muestra un ejemplo del archivo que describe una de las topologías utilizadas para el problema del VIH:

```
CREATE LAYER layer1 AS Lin-Lout WITH 20
CREATE LAYER layer2 AS Lin-Lout WITH 20
CREATE LAYER layer3 AS Lin-Lout WITH 20
CREATE CONNECTION FROM layer2 TO layer2 AS DELAY 1
CREATE CONNECTION FROM layer3 TO layer3 AS DELAY -1
```

Anexo 11. Cubo OLAP de los resultados de exactitud obtenidos por los diferentes clasificadores con bases de datos artificiales

Valores máximos por tipo de clasificador

Tipo de Relación: Total
Obtención de la Clase: Total

Relación entre Rasgos		N	Mean	Std. Deviation	Minimum	Maximum
Sin relación	J48	15	78.73	6.865	64	92
	SMO-Max	15	87.41333	9.143137	67.200	97.800
	Bayes-Net	15	80.29333	7.018696	62.400	92.600
	MLPMax	15	90.96000	8.592422	72.800	99.800
	Max-RNN-both(ave)	15	91.12000	7.802490	72.800	97.200
	Max-RNN-both(maxprob)	15	91.57333	7.582260	73.400	97.200
	Max-RNN-both(mode)	15	90.98000	7.852406	72.200	96.400
	Max-RNN-both(left)	15	91.06667	7.548573	73.400	96.800
	Max-RNN-both(middle)	15	90.64000	7.519954	73.200	97.400
	Max-RNN-both(right)	15	90.74667	8.121037	71.400	97.000
	Max-RNN	15	91.93333	7.533418	74.000	97.400
Lineal	J48	90	84.31	5.669	68	94
	SMO-Max	90	87.86889	8.484389	63.000	98.800
	Bayes-Net	90	80.97333	5.993720	64.000	92.200
	MLPMax	90	91.76778	7.405856	66.800	100.000
	Max-RNN-both(ave)	90	92.45778	6.377153	66.200	98.800
	Max-RNN-both(maxprob)	90	92.77556	6.262485	68.400	99.000
	Max-RNN-both(mode)	90	92.54444	6.168060	69.000	99.000
	Max-RNN-both(left)	90	91.92000	6.651505	66.000	98.600
	Max-RNN-both(middle)	90	91.82222	6.700826	66.000	98.600
	Max-RNN-both(right)	90	91.83556	6.698823	66.400	99.400
	Max-RNN	90	93.16444	6.118131	69.000	99.400
Polinomial	J48	90	82.82	6.972	56	94
	SMO-Max	90	88.07556	9.704977	60.600	99.200
	Bayes-Net	90	80.35778	7.120503	56.600	92.200
	MLPMax	90	92.11111	8.027207	69.800	100.000
	Max-RNN-both(ave)	90	91.60222	7.196035	71.200	98.400
	Max-RNN-both(maxprob)	90	91.97333	7.282622	71.200	98.200
	Max-RNN-both(mode)	90	91.67556	7.374528	69.800	98.200
	Max-RNN-both(left)	90	91.14222	7.642062	69.600	98.200
	Max-RNN-both(middle)	90	90.91111	7.884966	69.000	98.000
	Max-RNN-both(right)	90	91.22000	7.528884	69.600	98.600
	Max-RNN	90	92.36333	7.131737	71.200	98.600
Por ramas	J48	90	82.68	6.132	59	96
	SMO-Max	90	88.29333	8.841740	60.600	98.600
	Bayes-Net	90	80.82000	5.953230	60.800	94.800
	MLPMax	90	92.06000	7.370726	69.000	99.600
	Max-RNN-both(ave)	90	92.14667	6.287551	67.000	99.600
	Max-RNN-both(maxprob)	90	92.23778	6.524209	65.200	98.800
	Max-RNN-both(mode)	90	92.08111	6.238191	66.200	99.200
	Max-RNN-both(left)	90	90.84889	6.797707	64.600	98.800
	Max-RNN-both(middle)	90	90.63333	6.809131	65.800	98.800
	Max-RNN-both(right)	90	91.50667	6.815000	64.600	98.400
	Max-RNN	90	92.82556	6.182461	67.000	99.600
Total	J48	285	83.03	6.400	56	96
	SMO-Max	285	88.04421	8.985850	60.600	99.200
	Bayes-Net	285	80.69474	6.384037	56.600	94.800
	MLPMax	285	91.92596	7.623603	66.800	100.000
	Max-RNN-both(ave)	285	92.01895	6.672800	66.200	99.600
	Max-RNN-both(maxprob)	285	92.28912	6.724481	65.200	99.000
	Max-RNN-both(mode)	285	92.04140	6.662035	66.200	99.200
	Max-RNN-both(left)	285	91.29123	7.046512	64.600	98.800
	Max-RNN-both(middle)	285	91.09684	7.150447	65.800	98.800
	Max-RNN-both(right)	285	91.48000	7.051405	64.600	99.400
	Max-RNN	285	92.73965	6.525643	67.000	99.600

Anexo 12. Resultados de la prueba de Wilcoxon en la comparación de la RNN contra MLP

Tabla A.1. Significación obtenida con la prueba de Wilcoxon en la comparación de RNN, con la función de máxima probabilidad, contra MLP según la relación entre rasgos y la separación entre clases.

Relación entre rasgos	***Separación entre clases***			**Total**
	Lineal	Polinomial	Por ramas	
No relación	0.0625$^{<}$	0.0642$^{>}$	0.13	0.4164
Lineal	0.0$^{<}$	0.0$^{>}$	0.0$^{>}$	0.0001$^{>}$
Polinomial	0.0$^{<}$	0.0018$^{>}$	0.0171$^{>}$	0.541
Por ramas	0.0$^{<}$	0.0$^{>}$	0.0012$^{>}$	0.0176$^{>}$
Total	0.0$^{<}$	0.0$^{>}$	0.0$^{>}$	0.021

Tabla A.2. Resultados de la prueba de Wilcoxon para la comparación de RNN, con la función de promedio, contra MLP según la relación entre rasgos y la separación entre clases.

Relación entre rasgos	***Separación entre clases***			**Total**
	Lineal	Polinomial	Por ramas	
No relación	0.0614$^{<}$	0.0646$^{>}$	0.503	0.8784
Lineal	0.0$^{<}$	0.0$^{>}$	0.0$^{>}$	0.0124$^{>}$
Polinomial	0.0$^{<}$	0.2463	0.0208$^{>}$	0.0314$^{<}$
Por ramas	0.0$^{<}$	0.0006$^{>}$	0.0$^{>}$	0.0176$^{>}$
Total	0.0$^{<}$	0.0$^{>}$	0.0$^{>}$	0.021$^{>}$

$^{<}$ A favor de MLP

$^{>}$ A favor de RNN

Anexo 13. Forma de agregar un clasificador en Weka

Para crear un nuevo clasificador se deben seguir los siguientes pasos:

1. Crear la clase con el nombre del clasificador en el paquete referido al tipo de clasificador. Esta clase debe heredar de la clase abstracta *weka.classifiers.Classifier*. Recuerde redefinir el método *toString* para obtener una descripción del objeto.
2. Implementar los métodos necesarios, de forma tal que *buildClassifier()* implemente el entrenamiento. Además se implementará al menos uno de los métodos *distributionForInstance()* y *clasifyInstance()*, teniendo en cuenta que ambos definen la forma de clasificar un nuevo caso después de entrenado el clasificador. *distributionForInstance()* produce un vector de salida con las probabilidades de pertenencia a cada clase, mientras que *clasifyInstance()* debe resultar directamente en una etiqueta de clase.
3. Para agregar parámetros que sean necesarios en el entrenamiento se deben modificar los métodos *listOptions()*, *setOptions()*, *getOptions()*. Para cada uno de estos parámetros que se manipularán por medio de *setOptions()* y *getOptions()*, deben existir los métodos *get<Nombre_del_parámetro>()* y *set<Nombre_del_parámetro>()*. Adicionalmente se deben redefinir los métodos *<Nombre_del_parámetro>TipText* que devolverán una cadena de caracteres con la especificación del parámetro.
4. Adicionalmente se debe redefinir el método *getCapabilities* para definir las características de la base de entrenamiento que son admitidas por el clasificador.

Anexo 14. Bases del repositorio de la UCI usadas para la validación del modelo MEHI

Nombre	*Rasgos discretos*	*Rasgos continuos*	*Clases*	*Casos*	*Distribución por clase*
audiology	69	0	24	226	-
autos	8	17	6	205	3 | 22 | 67 | 54 | 32 | 27
balloons	4	0	2	76	35 | 41
breast-cancer-w	0	4	3	625	288 | 49 | 288
cars	6	0	4	1728	1210 | 384 | 69 | 65
colic	15	7	2	368	232 | 136
credit-a	9	6	2	690	307 | 383
credit-g	13	7	2	1000	700 | 300
diabetes	0	8	2	768	500 | 268
glass	0	9	6	214	70 | 76 | 17 | 13 | 9 | 29
heart-c	7	6	2	303	165 | 138
heart-h	7	6	2	294	188 | 106
heart-statlog	0	13	2	270	150 | 120
hepatitis	13	6	2	155	32 | 123
horse-colic	4	23	2	300	99 | 201
hypothyroid	22	7	4	3772	3481 | 194 | 95 | 2
ionosphere	0	34	2	351	126 | 225
iris	0	4	3	150	50 | 50 | 50
kr-vs-kp	36	0	2	3196	1669 | 1527
labor	8	8	2	57	20 | 37
lenses	4	0	3	23	4 | 5 | 14
liver-disorders	0	6	2	345	145 | 200
lung-cancer	56	0	3	32	9 | 13 | 10
lymph	2	16	4	148	2 | 81 | 61 | 4
monks	6	0	2	415	229 | 186
postoperative_pat	7	1	3	90	2 | 24 | 64
promoters_integer	57	0	2	106	53 | 53

Nombre	***Rasgos discretos***	***Rasgos continuos***	***Clases***	***Casos***	***Distribución por clase***
segment	0	19	7	2310	330 | 330 | 330 | 330 | 330 | 330 | 330
sick	12	7	2	3772	3541 | 231
sonar	0	60	2	208	97 | 111
soybean	35	0	19	683	-
vote	16	0	2	435	267 | 168
wine	0	13	3	178	59 | 71 | 48
yeast	0	8	10	1484	-

Anexo 15. Resultados de exactitud para las bases de datos del VIH con los diferentes multiclasificadores

		APV	*ATV*	*IDV*	*LPV*	*NFV*	*RTV*	*SQV*	**Promedio**
AdaBoost	**J48**	0.839	0.736	0.889	0.914	0.921	0.928	0.785	0.859
	BayesNet	0.823	0.653	0.899	0.897	0.854	0.899	0.733	0.823
	SVM	0.810	0.711	0.881	0.903	0.851	0.912	0.744	0.830
	SVMP2	0.815	0.711	0.849	0.880	0.868	0.905	0.759	0.827
	SVMGaussian	0.788	0.620	0.865	0.874	0.814	0.866	0.736	0.795
	IBk15	0.799	0.661	0.822	0.794	0.814	0.853	0.725	0.781
	MLP	0.804	0.730	0.866	0.885	0.895	0.906	0.761	0.835
Bagging	**J48**	0.834	0.727	0.883	0.914	0.906	0.935	0.785	0.855
	BayesNet	0.815	0.719	0.907	0.897	0.868	0.902	0.738	0.835
	SVM	0.820	0.769	0.891	0.880	0.864	0.902	0.769	0.842
	SVMP2	0.828	0.760	0.894	0.891	0.881	0.912	0.754	0.846
	SVMGaussian	0.689	0.653	0.817	0.851	0.710	0.824	0.756	0.757
	IBk15	0.756	0.661	0.785	0.863	0.801	0.752	0.725	0.763
	MLP	0.811	0.750	0.882	0.890	0.901	0.905	0.771	0.844
Stacking		0.820	0.752	0.894	0.891	0.908	0.922	0.783	0.853
MEHI	Selection	0.861	0.810	0.918	0.914	0.931	0.935	0.780	0.878
	Fusion	0.853	0.793	0.915	0.920	0.921	0.944	0.780	0.875
Promedio		0.810	0.719	0.874	0.886	0.865	0.894	0.758	

Anexo 16. Resultados del AUC para las bases de datos del VIH con los diferentes multiclasificadores

		APV	*ATV*	*IDV*	*LPV*	*NFV*	*RTV*	*SQV*	**Promedio**
AdaBoost	**J48**	0.924	0.83	0.95	0.9	0.95	0.98	0.83	0.91
	BayesNet	0.827	0.59	0.91	0.87	0.85	0.92	0.72	0.81
	SVM	0.878	0.7	0.93	0.92	0.9	0.97	0.8	0.87
	SVMP2	0.875	0.8	0.92	0.83	0.9	0.96	0.79	0.87
	SVMGaussian	0.85	0.67	0.94	0.84	0.87	0.92	0.8	0.84
	IBk15	0.829	0.7	0.88	0.73	0.87	0.92	0.76	0.81
Bagging	**J48**	0.927	0.8	0.95	0.88	0.94	0.98	0.85	0.9
	BayesNet	0.914	0.72	0.96	0.84	0.94	0.96	0.8	0.88
	SVM	0.878	0.83	0.92	0.76	0.91	0.95	0.81	0.87
	SVMP2	0.888	0.81	0.92	0.8	0.9	0.96	0.79	0.87
	SVMGaussian	0.75	0.65	0.87	0.5	0.5	0.93	0.8	0.71
	IBk15	0.848	0.76	0.91	0.85	0.91	0.93	0.8	0.86
Stacking		0.91	0.75	0.95	0.9	0.93	0.93	0.82	0.89
MEHI	Selection	0.931	0.81	0.97	0.91	0.96	0.98	0.84	0.91
	Fusion	0.924	0.84	0.97	0.93	0.96	0.99	0.85	0.92
Promedio		0.877	0.75	0.93	0.83	0.89	0.95	0.8	

Anexo 17. Glosario de términos

Abreviatura	Nombre en español	Nombre en inglés
ADN	Ácido desoxirribonucleico	Deoxyribonucleic Acid
APV	-	Amprenavir*
ARN	Ácido ribonucleico	Ribonucleic Acid
ATV	-	Atazanavir*
AUC	Área bajo la curva ROC	Area Under the Curve
BN	Red bayesiana	Bayesian Network
BP	Propagación del error hacia atrás, Retropropagación	Backpropagation
BPTT	Propagación de errores hacia atrás en el tiempo, Retropropagación en el tiempo	Backpropagation Through Time
DP	Perfiles de decisión	Decision Profile
DT	Plantilla de decisión	Decision Template
EDA	Algoritmo de estimación de la distribución	Estimation of Distribution Algorithm
EM		Expectation-Maximization
ESOM	SOM emergente	Emergent SOM
FFN	Redes Multicapas de Alimentación Hacia Delante	Feed-Forward Neuronal Networks
FN	Elementos mal clasificados de la clase negativa	False Negatives
FP	Elementos mal clasificados de la clase positiva	False Positives
IDV	-	Indinavir*
*k*NN	*k*-vecinos más cercanos	*k* Nearest Neighbors
LPV	-	Lopinavir*
MCD	Distancia máxima al centro	Maximum Center Distance
MDS	Escalado multidimensional	Multidimensional Scaling

MEHI	Multiexperto basado en casos duros	Muti-Expert by Hard Instances
MLP	Perceptrón multicapa	Multilayer Perceptron
NFV	-	Nelfinavir*
PCA	Análisis de componentes principales	Principal Component Analysis
PSO	Optimización basada en enjambre de partículas	Particle Swarm Optimization
RNN	Redes neuronales recurrentes	Recurrent Neuronal Networks
ROC		Receiver Operator Curve
RTRL	Aprendizaje recurrente	Recurrent Learning
RTV	-	Ritonavir*
SOM	Mapa autoorganizado, Red neuronal de Kohonen	Self-Organized Map
SQV	-	Saquinavir*
SVM	Máquina de soporte vectorial, Máquina de vectores de soporte	Support Vector Machine
TN	Elementos bien clasificados de la clase negativa	True Negatives
TP	Elementos bien clasificados de la clase positiva	True Positives
VIH	Virus de Inmunodeficiencia Humano	Human immunodeficiency virus
WEKA	-	Waikato Environment for Knowledge Analysis