Universidad Central “Marta Abreu” de Las Villas
Facultad de Matemática, Física y Computación

Trabajo para optar por Título de Master en Ciencia de la Computación

Predicción de la resistencia del VIH a partir del genotipo y fenotipo usando técnicas de inteligencia artificial

Autora
Lic. Isis Bonet Cruz

Tutores
Dra. Maria Matilde García Lorenzo
Dr. Ricardo Grau Ábalo

Consultante
MSc. Robersy Sánchez Rodríguez

2005

A mi mamá.

Agradecimientos

A mis tutores, Marilín y Grau, por todo el tiempo dedicado y por haber confiado hasta el último momento en mí.
A mi mamá, por haber estado siempre conmigo y por su fe en mí.
A mi hermana y mi familia que sé que estuvieron conmigo.
A Robersy, por su apoyo y tiempo dedicado, y porque sin él no hubiera surgido este trabajo.
A Sain, por ser parte de estos resultados.
A Leticia y su familia, por haber sido un eslabón importante en la culminación de este trabajo.
A mis compañeros, por su apoyo e ideas en este trabajo.
A mis amigos, por los momentos alegres y por haberme comprendido siempre.
A todas las personas que han contribuido a la realización de este trabajo.
A todos aquellos que de alguna forma les tocó un poquito de mi estrés.

Resumen

El objetivo de este trabajo es predecir la resistencia del virus del VIH ante inhibidores de la Proteasa, a partir de bases de casos de genotipo y fenotipo. En aras de de dar solución a este problema se diseñó como un modelo de clasificación, para el que se construyeron las bases de casos, a partir de la información disponible en la base de datos de Stanford para 7 inhibidores de esta enzima, quedando conformadas 7 bases, una para cada inhibidor. El conocimiento se representó usando, como descriptores del genotipo, dos variantes: las energías de contacto asociadas a cada aminoácido y el cambio de energía con respecto a al HXB2, como secuencia de referencia. Además se transformó la medida de resistencia en dos clases: resistente y susceptible.

Los tres métodos de clasificación usados en este trabajo son SVM, MLP y redes recurrentes bidireccionales. La implementación del software NEngine permite la creación, entrenamiento y explotación de este último método. Se analizaron dos formas de procesar las salidas de las redes bidireccionales, una partiendo de la moda y otra, seleccionando sólo la salida del medio. Para validar los resultados obtenidos con estas técnicas se usó *cross-validation* y métodos estadísticos de comparación de poblaciones, obteniéndose como conclusión que las dos representaciones de la secuencia mutada son buenos descriptores del genotipo. Se demostró, además, que las redes bidireccionales pueden ser usadas como un método de clasificación para este problema, obteniéndose los resultados de predicción más altos o comparables con resultados anteriores.

Índice

Introducción

Los virus son un complejo macromolecular formado por proteínas y ácidos nucleicos, cuya naturaleza puede ser DNA o RNA. 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, entre los que más afecta 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 Reverso Tranquistasa, tratando de encontrar un fármaco que lo combata. Ya existe un gran número de drogas anti-retrovirales aprobadas para tratar la infección del VIH que tienen efecto contra algunas mutaciones del virus. El tratamiento con combinaciones de estas drogas puede ofrecer la retención del virus dando un tiempo para reconstruir el sistema inmunológico. Sin embargo, a menos que la terapia retenga la replicación del virus completamente, el tratamiento anti-retroviral 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 la droga, o la combinación de drogas, más recomendada para su tratamiento. En estos momentos existen dos formas para probar la susceptibilidad de una cepa aislada del VIH ante fármacos anti-retrovirales: fenotipo y genotipo. Estas pruebas pueden mostrar, para una cepa particular del VIH, cuan probable es que sea inhibida por una droga. Los ensayos genotípicos sólo proveen pistas hacia la resistencia a la droga. O sea, el genotipo intenta establecer la presencia o ausencia de mutaciones genéticas de una proteína del VIH que ha sido previamente asociada con la resistencia a la droga. Éste supone un procedimiento relativamente simple y puede ser hecho rápidamente. Sin embargo, la interpretación de la resistencia a drogas partiendo solamente de la información del genotipo es todavía un reto y a menudo requiere análisis de expertos. Por otro lado las pruebas fenotípicas proveen una cuantificación directa de la resistencia a la droga, pero requieren 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 disponible en bases de datos internacionales como “Los Álamos” y “Stanford HIV Resistance Database”. En este trabajo se han seleccionado los datos de esta última base de datos para usarlos como

información de partida en un método computacional que logre predecir el fenotipo a partir del genotipo.

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 gran reto, actualmente se han usado algunos métodos de aprendizaje automatizado para relacionar automáticamente 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 han sido usados para derivar modelos computacionales que predigan la resistencia de 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.

Cuando se va a usar un método de aprendizaje necesitamos, además de la base de conocimientos con la información a aprender, una representación apropiada de la misma. Este 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?

Otros autores han usado varias representaciones para la secuencia como: mutaciones más frecuentes, vectores de 20 elementos (binarios) que representan los aminoácidos, *profiles* de información mutua y otros. Todas estas variantes tratan, de alguna forma, de describir la secuencia en cuanto a sus mutaciones. En este trabajo tratamos de tener unos descriptores que tuvieran relación más fuerte con la estructura espacial de las secuencias, pues a partir de esta es que se puede saber, en la práctica, cómo debe ser el fármaco y si inhibirá a la mutación o no.

Teniendo en cuenta que las energías de interacción de los aminoácidos, tienen relación directa con la estructura tridimensional de la secuencia, ya que la enzima se pliega en el espacio debido a la atracción que ejercen cada uno de los aminoácidos sobre el resto, se seleccionó esta variante para representar la secuencia del VIH.

Según la problemática planteada, nos hicimos las siguientes preguntas de investigación:

¿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?

¿Es factible usar redes recurrentes bidireccionales para predecir la resistencia de mutaciones de la Proteasa ante determinados fármacos?

Para responder estas preguntas nos trazamos los objetivos siguientes:

Objetivo General

Predecir la sensibilidad de mutaciones de la enzima de la Proteasa del VIH ante 7 de los fármacos más usados como antivirales, a partir de información de genotipo y fenotipo usando técnicas de inteligencia artificial.

Objetivos Específicos

- Estudiar resultados obtenidos hasta el momento en la predicción de resistencia antiviral de las mutaciones de la Proteasa del VIH.
- Seleccionar los descriptores que caracterizarán la Proteasa para predecir su resistencia.
- Usar técnicas de inteligencia artificial para predecir la sensibilidad de las mutaciones de la Proteasa del VIH ante 7 de los fármacos disponibles.
- Evaluar comparativamente los resultados de las técnicas de inteligencia artificial usadas para predecir resistencia de mutaciones ante estos fármacos.

Como aspecto novedoso, relacionado con la tesis, se tiene el uso de las energías y cambios de energías como descriptores de las mutaciones de la enzima de la Proteasa del virus, así como el uso de redes neuronales bidireccionales como método de clasificación para predecir resistencia de esta enzima.

La tesis se estructura por tres capítulos. En el capítulo 1 se aborda el estado del arte para los problemas de clasificación de la resistencia a fármacos del VIH. Allí se introduce la problemática desde el punto de vista biológico, los métodos de clasificación más usados hasta el momento y los resultados alcanzados para predecir esta resistencia. Se analizan las energías como descriptores de las proteínas del virus. Se formulan consecuentemente las hipótesis de investigación.

En el capítulo 2 se describe la modelación como un problema de clasificación. Se detalla la construcción de las bases de casos y los descriptores usados para representar la

secuencia, así como el uso de Support Vector Machine (SVM), redes neuronales de tipo Multilayer Perceptron (MLP) y Redes Recurrentes Bidireccionales (RNN). También se refiere el diseño y la implementación de un software para la creación, entrenamiento y explotación de la última técnica mencionada.

En el capítulo 3 se hace una evaluación comparativa de los métodos de inteligencia artificial descritos anteriormente y los nuevos, propuestos por la autora. En particular quedará demostrado que el uso de las "energías" como variables predictivas y la implementación de redes recurrentes bidireccionales permiten prometedores resultados.

Capítulo I: Análisis del estado del arte para problemas de clasificación de resistencia a fármacos del VIH.

1.1 Introducción

1.1.1 Infección del VIH

El virus de inmunodeficiencia humana (VIH) es un patógeno adaptado. A finales del 2003 había un estimado de 37.8 millones de personas en el mundo (35.7 millones adultos y 2.1 millones de niños menores de 15 años) que viven con VIH/ SIDA. Además, un estimado de 4.8 millones de esos fueron nuevos infectados del virus.

Una vez que el virus entra en el torrente sanguíneo, a través de reproducción sexual, intercambio de sangre o leche materna, éste busca una célula huésped para reproducirse. Aunque el VIH puede infectar a un gran número de células en el organismo, el principal objetivo es una célula inmune llamada linfocito (un tipo de célula T). Una vez que el VIH entra en contacto con una célula T se sujeta él mismo y usa la célula como una maquinaria para reproducir miles de nuevas copias del virus. Las células T son una parte importante del sistema inmune porque ellas ayudan a facilitar la respuesta del organismo ante muchas infecciones comunes pero potencialmente fatales. Sin la cantidad suficiente de este tipo de células el sistema inmune del organismo no puede defenderse él mismo contra infecciones.

Ser VIH-positivo, o ser infectado con el VIH no es lo mismo que haber adquirido el síndrome de inmunodeficiencia adquirida (SIDA). Alguien con VIH puede vivir por muchos años con pocos efectos de la enfermedad. Por supuesto, esto es siempre que su cuerpo reemplace las células T destrozadas por el virus. Sin embargo, una vez que el número de células T disminuye por debajo de un cierto umbral, los individuos infectados comenzarán a mostrar síntomas de SIDA. Tales individuos tendrán una respuesta inmune baja y una susceptibilidad alta a un alto rango de infecciones que son inofensivas para personas sanas. En el 2003, enfermedades asociadas con el VIH/SIDA causaron la muerte de aproximadamente 2.9 millones de personas en el mundo, incluido un estimado de 490000 niños menores de 15 años y desde que los primeros casos del SIDA fueron identificados en 1981, más de 20 millones de personas con VIH/SIDA han muerto [Jam04].

1.1.2 Función de la Proteasa y sus inhibidores

La quimioterapia anti-retroviral está dirigida, fundamentalmente, contra dos de las tres enzimas virales, Proteasa y Reverso Tranquistasa.

La Reverso Tanquistasa es usada por el VIH para hacer una copia del DNA de su genoma RNA. Hace una copia del DNA para producir la doble hélice que es capaz de insertarse en los cromosomas de la célula hospedera para formar un provirus. Por otro lado la Proteasa es esencial para la maduración viral, por lo que la inactivación de esta enzima conlleva a la producción de partículas virales no infecciosas. Específicamente nos detendremos en la Proteasa que es la enzima de estudio en este trabajo.

La Proteasa es un homodímero simétrico de 99 residuos por cadena. El VIH usa esta enzima para cortar las poliproteínas Gag-pol(p120), Gag (p55) y Env (p160) y formar proteínas ensenciales para la estructura del VIH y su enpaquetamiento RNA. El sitio activo de la Proteasa se une a las poliproteínas para producir las proteínas funcionales (Ver Anexo 1A). Una vez que el sitio activo de la Proteasa se une a las poliproteínas, ésta las corta y queda dividida en proteínas funcionales (Ver Anexo 1B).

Los inhibidores de la Proteasa son drogas que se unen al centro activo de la Proteasa codificada del VIH y previenen el corte de las poliproteíans gag y gag-pol evitando la formación de las proteínas que son esenciales para la estructura del VIH y para el envoltorio de RNA dentro de su nucleocápside. Como resultado, la maduración no ocurre y se producen partículas virales no infecciosas (Ver Anexo 1C).

Compuesto	**Abreviatura**
Saquinavir	SQV
Ritonavir	RTV
Indinavir	IDV
Nelfinavir	NFV
Amprenavir	APV
Lopinavir	LPV
Atazanavir	ATV

Tabla 1.1. Inhibidores de la Proteasa

Existe un gran número de inhibidores de la Proteasa hoy día, en la Tabla 1.1 se muestran 7 de los más conocidos y que serán los usados en este trabajo.

1.1.3 Pruebas de resistencia

Hoy en día existen un gran número de drogas anti-retrovirales aprobadas para tratar la infección del VIH. El tratamiento con combinaciones de estas drogas puede ofrecer la retención del virus y una oportunidad para la reconstrucción inmunológica. Sin embargo, a menos que la terapia retenga la replicación del virus completamente, el tratamiento anti-retroviral aumenta la aparición de variantes resistentes a la droga.

La aparición de estas variantes depende del desarrollo de variaciones genéticas en el virus que permiten que se libere de los efectos del inhibidor de una droga. Se dice que una variante de resistencia a droga es identificable desde un virus de referencia por la manifestación de mutaciones que contribuyen a reducir la susceptibilidad. Esto ocurre a través de un mecanismo natural, en particular, una de las causas de la variabilidad genética del VIH resulta de inhabilitar la Reverso Tranquistasa para reescribir la secuencia de nucleótidos durante la replicación. Esto está compuesto por el alto rango de replicación del VIH (alrededor de 10 millones de partículas/ día), a tasa de mutaciones espontáneas (aproximadamente una mutación/copia) y recombinación genética cuando los virus de diferentes secuencias infectan la misma célula.

Una vez que la variante se hace resistente a la droga, necesita niveles altos de la misma droga anti-retroviral para lograr la retención de la replicación del virus. Sin embargo, con altos niveles de la droga nosotros incrementamos el riesgo de efectos del lado adverso y el daño a un individuo. Por lo tanto, cuando la resistencia ocurre, los pacientes a menudo necesitan cambiar a un nuevo régimen de droga.

Se pueden usar las pruebas de resistencia para mostrar si una cepa particular del VIH es probable para que sea retenida por una droga o no. No obstante, hasta recientemente, las pruebas de resistencia fueron usadas exclusivamente como herramienta de búsqueda, para investigar el mecanismo de fracaso de la droga. En Julio de 1998, la idea de extender la metodología de pruebas de resistencia para manejo clínico de rutina, aunque lógica, no pudo ser recomendada debido a la falta de validación, estandarización y una definición concreta del rol de la prueba. Sin embargo, desde entonces, un número de estudios apareció indicando el valor de la prueba de resistencia para el manejo clínico y la falta de estandarización fue dirigida con el desarrollo de las pruebas de resistencia comerciales. Estos dos factores contribuyeron a una segunda declaración, publicada en los inicios del 2000, la cual explícitamente reconoce el valor de prueba de resistencia de las drogas del VIH, y finalmente en Mayo del 2000, la Sociedad Internacional del SIDA recomendó la incorporación de pruebas de resistencia de drogas en el manejo clínico de pacientes con

infección del VIH. Además, datos considerables soportan el uso de pruebas de resistencia a drogas que han sido ahora publicadas [Jam04].

La resistencia puede ser probada midiendo la actividad viral *in vitro* en presencia y ausencia de una droga mediante la prueba denominada "prueba de resistencia de fenotipo" o mediante la "prueba de resistencia de genotipo" que consiste en la secuenciación del código de los genes virales de la droga objetivo.

1.1.3.1 Pruebas de genotipo y fenotipo

Las pruebas de genotipo están dirigidas a analizar las regiones genómicas relevantes para la resistencia al fármaco. Después del aislamiento del RNA viral de un paciente, los segmentos relevantes del genoma son transcritos reverso *in vitro* y amplificados por una reacción PCR.

Originalmente las pruebas de genotipo envuelven el aislamiento de células mononuclerares de la sangre periféricas (PBMCs, del inglés: peripheral blood mononuclear cells) desde un paciente, el crecimiento del VIH en un cultivo de células y la medida de la abundancia de al proteína de cápside(CA, p24), que denotan la presencia y ausencia de compuestos antivirales.

Existen varias medidas para expresar la resistencia de la droga. La más comúnmente usada es IC50 que es la concentración de la droga necesaria para inhibir la replicación del virus por un 50%. A menudo se usa esta medida estandarizada, a partir de una secuencia de referencia, expresando cuántas veces es resistente la cepa analizada con respecto a esta secuencia de referencia. A esta medida la vamos a llamar factor de resistencia (FR).

$$FR = \frac{IC50(\text{sec}uencia\_mutada)}{IC50(\text{sec}uencia\_de\_referencia)} \tag{1.1}$$

Ambas pruebas tienen importancia para medir la resistencia, aunque producen resultados diferentes. Por un lado, el fenotipo cuantifica un valor simple de susceptibilidad a una droga, y por otro, el genotipo revela los patrones mutacionales. Las pruebas de genotipo son más rápidas que las de fenotipo (7-14 días vs. 2-4 días) y pueden detectar las mutaciones que preceden el desarrollo de un fenotipo resistente. Las pruebas de fenotipo son más caras que las de genotipo (US$ 750-vs 1000US$250-500), pero las primeras se prefieren para fármacos nuevos pues todavía no se conoce las mutaciones asociadas a la resistencia [Bee02].

Justamente por lo costoso de esta prueba de fenotipo se hace difícil su aplicación a cada una de las mutaciones que van surgiendo cada día y para cada uno de los fármacos que se van desarrollando. Para poder ayudar a esto se hace necesario el desarrollo de métodos computaciones para pronosticar la resistencia a partir un genotipo dado.

Actualmente existen muchas técnicas para clasificación que pueden ayudarnos a resolver este problema y que serán explicadas a continuación.

1.2 Métodos de clasificación.

Desde que se inventaron las computadoras nosotros nos preguntamos si ellas pueden estar hechas para aprender. Si nosotros pudiéramos entender como programarlas para aprender el impacto sería dramático. Todavía no conocemos como hacer que las computadoras aprendan de manera similar a como lo hacen las personas, porque primero debemos entender mejor como nosotros mismos vamos desarrollando el proceso de aprendizaje en el sentido biológico, y luego intentar interpretar esto mediante algoritmos que simulen este proceso. Sin embargo, sí se han desarrollado varios algoritmos para ciertos tipos de tareas de aprendizaje, y ha surgido un entendimiento teórico de aprendizaje.

Según [Mit97], como definición de aprendizaje tenemos que "un programa de computadora se dice que aprende desde una experiencia E con respecto a alguna clase de tareas T y una medida de representación P, si su representación en las tareas en T, como medida para P, mejora con la experiencia E".

Para tener un problema de aprendizaje bien definido, nosotros podemos identificar estos tres rasgos: la clase de tareas, la medida de representación y la fuente de experiencia.

En general, el campo de aprendizaje automático se relaciona con la respuesta de cómo construir programas de computadora que automáticamente mejoren con la experiencia. Típicamente, nosotros tenemos una clasificación cuantitativa o categórica, que deseamos predecir basada en un conjunto de características de entrada. Partimos de un conjunto de entrenamiento consistente de características a las que se les hace corresponder clases o categorías. A partir de este conjunto de datos, construimos un modelo de clasificación que posibilitará predecir una salida apropiada para casos que no han sido vistos con anterioridad.

El primer paso para construir un programa para aprender es considerar el tipo de experiencia de entrenamiento desde el cual el sistema aprenderá. Esto es importante porque el tipo y la disponibilidad del material de entrenamiento tienen un impacto

significativo en el éxito o fallo del aprendizaje. Una consideración importante es cómo el material de entrenamiento representa la población entera.

El próximo paso es determinar el tipo de conocimiento para aprender y cómo éste será usado como medida de representación. En particular se define una función objetivo:

$$F: C \rightarrow O \tag{1.2}$$

donde F acepta como entrada un conjunto de características C y produce como salida alguna clasificación O que sea verdadera desde C. El problema de la representación P para las tareas T es entonces reducible a encontrar una función F2 que represente, mejor que una función F1, a las tareas T, como medida para P [Mit97].

El paso final en la construcción de un sistema de aprendizaje, o sea, seleccionar un mecanismo de aprendizaje para derivar la función objetivo. Esto incluye determinar la representación de la función F. Una vez que un mecanismo es decidido entonces se necesita seleccionar un algoritmo para generar tales representaciones desde la experiencia de entrenamiento E [Jam04].

Resumiendo, podemos decir que para el diseño de un sistema de aprendizaje se necesita, según [Mit97]:

- Tarea T
- Medida de representación
- Experiencia de entrenamiento
- Tipo exacto del conocimiento que debe ser aprendido (Función objetivo)
- Mecanismo de aprendizaje

¿En qué se basa un método de clasificación? Para responder a esta pregunta vamos a basarnos en el concepto de aprendizaje. Es un mecanismo de aprendizaje, donde la tarea es tomar cada instancia y asignarla a una clase particular. Las clases entre las que puede elegir el procedimiento de clasificación se pueden describir de gran cantidad de formas. Su definición dependerá del problema en particular. Este tipo de método constituye una parte importante de muchas de las tareas de resolución de problemas. En su forma más simple, se presenta directamente como una tarea de reconocimiento.

Entre los métodos de clasificación más usados en esta área se encuentran KNN, árboles de decisión y redes neuronales. Estas técnicas son descritas a continuación.

1.2.1 Método del vecino más cercano (KNN)

El k-NN es un algoritmo puramente perezoso pues éste almacena el conjunto de entrenamiento y deja todo el procesamiento para la fase de clasificación [DAS91].

La entrada del clasificador es una instancia q, cuya clase se desconoce, y la salida es la predicción de su clase. Cada instancia $x = \{x_1, x_2, ..., x_{|F|}\}$ es un punto en el espacio multidimencional definido por el conjunto de rasgos y la clase F a la que pertenece x, dentro del conjunto de clases J. Los rasgos pueden ser de varios tipos: reales, enteros, simbólicos, símbolos ordenados, conjunto de símbolos, booleanos o difusos.

El error que puede cometerse al clasificar cada instancia del conjunto de entrenamiento es conocido como Error de Clasificación Dejando uno Fuera (LOOCE por sus siglas en inglés). El objetivo de este clasificador es minimizar el coeficiente *LOOCE*. La forma de calcularlo está en dependencia de si los valores de la clase son continuos o discretos. Para valores discretos se calcula como:

$$LOOCE = \sum_{q \in BC} \sum_{j \in J} \left(\delta_{q,j} - p_{q,j} \right) \tag{1.3}$$

Esto es, para cada instancia q que pertenece a la base de casos BC y para cada clase j que pertenece al conjunto de clases J, se calcula la sumatoria de la diferencia entre la función de membresía de clase (*class membership*) $\delta_{q,j}$ y la función de probabilidad de clase (*class probability*) $p_{q,j}$, definidas como:

$$\delta_{q,j} = \begin{cases} 1 & q_c = j \\ 0 & q_c \neq j \end{cases} \tag{1.4}$$

$$p_{q,j} = \frac{\sum_{r \in K} \delta_{r,j} \cdot sim(q,r)^2}{\sum_{r \in K} sim(q,r)^2} \tag{1.5}$$

Donde K es el conjunto de las instancias vecinas más similares.

Para clases con valores continuos el LOOCE se define como:

$$LOOCE = \sum_{q \in BC} (q_c - p_q)^2 \tag{1.6}$$

Donde q_c es la clase de la instancia q y p_q se calcula:

$$p_q = \frac{\sum_{r \in K} r_c \cdot sim(q,r)^2}{\sum_{r \in K} sim(q,r)^2} \tag{1.7}$$

Siendo r_c el valor de la clase de la instancia r.

Cuando se va a clasificar una instancia, dado que k-NN conoce el valor de la clase de q, este devuelve la clase más probable utilizando las siguientes fórmulas:

Si los valores de la clase son discretos, k-NN devuelve:

$q_c = \max_{j \in J} p_{q,j}$, donde $p_{q,j}$ está definida en la ecuación 1.5.

Si los valores de la clase son continuos, k-NN devuelve:

$q_c = p_q$, donde p_q está definida en la ecuación 1.7.

La similitud entre dos instancias se calcula como:

$$sim(q,c) = \sum_{a=1}^{n} w_a \cdot sim_a(q_a, c_a) \tag{1.8}$$

Donde sim_a es una función de similitud utilizada para comparar el valor del rasgo a de cada instancia, n la cantidad de características y w_a el peso del rasgo a. k-NN le da al peso de cada característica un valor constante:

$$\forall_{a \in F}\ w_a = s \tag{1.9}$$

Siendo s una constante escalar. La función expresión 1.9 da igual importancia a todos los rasgos por lo que en presencia de rasgos redundantes, irrelevantes, interactivos o ruidosos la precisión del el k-NN disminuye. Debido a esto se han creado muchas variantes que asignan a cada rasgo un peso que no tiene que ser necesariamente el mismo para todos [Zal03].

1.2.2 Árboles de decisión

El aprendizaje usando árboles de decisión es uno de los más usados. Es un método de aproximación de funciones objetivo de valores discretos, en el cual la función aprendida es representada por un árbol de decisión. Los árboles aprendidos pueden ser representados como conjuntos de reglas if-then. Este método de aprendizaje es uno de los más populares entre los algoritmos de inferencia inductiva.

Un árbol de decisión es un grafo acíclico con vértices interiores simbolizando pruebas para llevar a cabo en una característica o atributo y dejar clasificaciones indicadas. Estos árboles clasifican casos no conocidos mediante el ordenamiento del árbol desde la raíz hasta alguna hoja, la cual posee un valor de clasificación. O sea, como en el árbol cada nodo simboliza una prueba de algún atributo y cada arco que sale de un nodo corresponde a alguno de los posibles valores del atributo que representa ese nodo.

En términos generales, un árbol de decisión representa una disyunción de conjunciones de restricciones en los valores de los atributos. Cada camino de la raíz del grafo a una hoja traduce a una conjunción de prueba de atributos, y el grafo por si mismo traduce a una disyunción de esas conjunciones.

La mayoría de los algoritmos de aprendizaje que construyen árboles de decisión son variaciones del algoritmo base nombrado ID3 o su sucesor C4.5, que son algoritmos que emplean una búsqueda greedy sobre el espacio de posibles árboles de decisión.

La idea del algoritmo ID3 [Mit97] es seleccionar cuál atributo probar a cada nodo en el árbol. Como una medida de cuantificar el valor del atributo se define una propiedad estadística, llamada ganancia de información, que mide cuan bien un atributo dado separa los ejemplos del entrenamiento acorde a su clasificación objetivo. ID3 usa esta medida de información para seleccionar entre los atributos candidatos en cada paso mientras crece el árbol.

Entre estas medidas de ganancia de información la más usada es la entropía, que caracteriza la pureza de una colección de ejemplos. Dada una colección S, con ejemplos positivos y negativos, la entropía de S relativa a su clasificación booleana es:

$$\text{Entropy}(S) = -p_{+} \log 2 p_{+} - p_{-} \log 2\, p_{-} \qquad (1.10)$$

Donde: p_{+} es la proporción de ejemplos positivos en S y p_{-} es la proporción de ejemplos negativos en S.

1.2.3 Support Vector Machine

Support Vector Machine (SVM) es una técnica de aprendizaje supervisado que se desarrolló rápidamente y se usado ampliamente en varias ramas. Sus orígenes parten de la teoría de aprendizaje estadístico y se basa en el principio de minimización de riesgo estructural. Es usado tanto para clasificación, como para regresión.[1]

[1] http://www.svms.org/

En el caso de la clasificación, construye un hiperplano de separación en el espacio y en el caso de regresión, representa regresiones lineales en este espacio, pero sin pequeños errores de penalización [Vap95]. La clasificación puede ser lineal o no lineal.

Clasificación lineal

Cuando se usa SVM para clasificación éste crea un hiperplano de separación de los datos a dos clases con un margen máximo. Dado los ejemplos de entrenamiento, el hiperplano divide los ejemplos de entrenamiento según las clases, tal que la distancia desde los ejemplos más cercanos al hiperplano sea máxima.

Los parámetros del hiperplano de margen máximo son derivados de resolver un problema de optimización de programación cuadrática. Dentro de los métodos más comunes para resolver estos problemas está el algoritmo SMO de Platt[2].

Clasificación no lineal con *kernel trick*

El algoritmo del hiperplano óptimo fue propuesto por Vapnik en 1963 que fue un clasificador lineal. Sin embargo en [Ber92] se sugirió una forma de clasificador no lineal aplicando *kernel trick* para hiperplanos de margen máximo. El algoritmo es formalmente similar, excepto que cada producto de puntos es remplazado por una función de kernel no lineal. Esto permite que el algoritmo fije el hiperplano de margen máximo en el espacio de rasgos transformado.

Si el kernel usado es una función *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. Algunos de los kernels más comúnmente usados son[3]:

- Polinomial: $k(x, x') = \langle x \cdot x' \rangle^d$ (1.11)

- Radial Basis: $k(x, x') = \exp\left(\frac{\|x - x'\|}{2\sigma^2} \right)$ (1.12)

- Sigmoidal: $k(x, x') = \tanh\left(kx \cdot x' + c\right)$ (1.13)

[2] http://en.wikipedia.org/wiki/Support_vector_machine
[3] http://en.wikipedia.org/wiki/Support_vector_machine

1.2.4 Redes Neuronales

Muchos de los problemas del mundo real, debido a su carácter ambiguo e impreciso, no son susceptibles de ser resueltos con el empleo de enfoques simbólicos formales, o por medio de procedimientos algorítmicos bien establecidos. Por esta razón, se han desarrollado métodos sub-simbólicos, entre los cuales se encuentran las redes neuronales artificiales. Las redes neuronales se agrupan dentro de las técnicas conexionistas de la Inteligencia Artificial y constituyen una de las especialidades más ampliamente difundidas. El objeto principal de este enfoque es la red neuronal. 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 [Ham03] de funciones, siendo capaces de aprender las características relevantes de un conjunto de datos, para luego reproducirlas en entornos ruidosos o incompletos.

Una red neuronal puede definirse como un conjunto de unidades computacionales (neuronas) interconectadas por medio de arcos pesados a manera de grafo dirigido. El objetivo de tal red (que puede verse como una caja negra) es brindar (calcular) una salida Y a partir de una información X recibida con anterioridad. Usualmente las redes destinan un conjunto de neuronas para la entrada (neuronas de entrada) de la información proveniente del exterior y un conjunto distinto para ofrecer las salidas (neuronas de salida); el resto se consideran neuronas ocultas. Se concibe el cálculo general de la red a partir de la información que es procesada por cada una de sus neuronas en forma independiente. Cada una de ellas puede recibir información de las restantes y calcular su propia salida a partir de dicha entrada y de su estado actual, transitando eventualmente hacia un nuevo estado. Por lo general, el flujo de cálculo de la red avanza progresivamente desde las neuronas de entrada hacia las neuronas de salida, en un proceso en el que cada una de las neuronas ocultas va activándose progresivamente según el esquema de conexión particular de cada red [Hil95].

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

1.2.4.1 El Modelo de la Neurona.

Siguiendo el modelo natural (células neuronales en los animales), la neurona artificial posee un esquema simple de cálculo en una sola dirección. Recoge los niveles de señal en

sus entradas y las procesa según una función de activación (que puede ser lineal o no lineal y es dependiente del estado actual en que se encuentre dicha neurona), brindando un nivel concreto de señal en la salida. El modelo más frecuente es el que se describe en la figura 1.1.

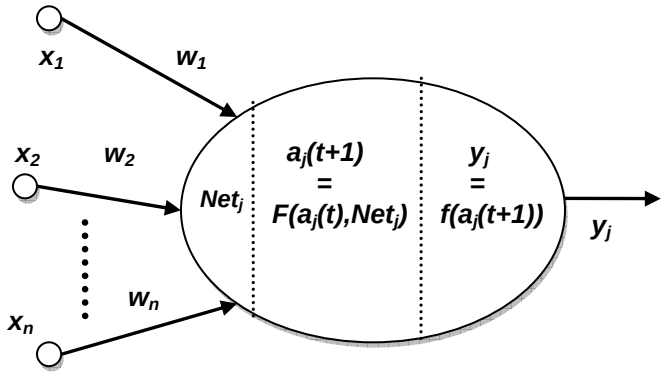


Figura 1.1. Modelo matemático de la neurona. (Tomado de [Hil95])

El valor de entrada neta Netj de la neurona es una combinación lineal de las señales provenientes de las neuronas de entrada más un valor constante w0 llamado umbral.

$$Net_j = \sum_i x_i w_i + w_0 \tag{1.14}$$

Los coeficientes de esta combinación lineal constituyen valores asociados a cada una de las conexiones de entrada a la neurona, denominados pesos de las conexiones. Estos constituyen los parámetros libres del modelo, que permiten el reajuste adecuado a la solución de los problemas concretos.

La función de activación F determina el nuevo estado de activación aj(t+1) teniendo en cuenta el estado actual aj(t) y la entrada neta Netj. La función de salida f brinda la salida real de la neurona a partir de su estado de activación.

Existe una amplia variedad de modelos de neuronas, cada uno se corresponde con un tipo determinado de función de activación y de salida.

1.2.4.2 El Esquema de Conexión de la Red Neuronal.

La forma específica de conexión (arquitectura) y la cantidad de neuronas conectadas (el número de parámetros libres) que intervienen en una red determina directamente el poder computacional de la misma. En los últimos años se han producido una amplia variedad de arquitecturas de redes neuronales, sin embargo, la mayoría de ellas se encuentran

ubicadas en dos grandes grupos: las redes muilticapa de alimentación hacia adelante (Feed-Forward Neuronal Networks, FFN) y las redes recurrentes (Recurrent Neuronal Networks, RNN) [Son02], [Hil95], [Trh64], [Ham03].

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.

Las RNN poseen una diferencia notable con la clase anterior: las redes recurrentes admiten conexiones hacia atrás, o sea, pueden formar ciclos en el grafo que describe sus conexiones. Estas conexiones hacia atrás, llamadas de retroalimentación (feedback loops) son las que permiten que la red sea capaz de guardar una memoria de los estados anteriores para su uso en el cálculo del estado y las salidas corrientes, 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. De manera general son aplicables a problemas reales que reflejan relaciones dinámicas y estructurales de orden complejo [Son02], [Per90], [Bal02].

En la figura 1.3 se muestra un ejemplo simple de una red recurrente de dos capas con conexiones hacia atrás. La misma puede emplearse para el análisis por etapas de una secuencia de valores de entrada donde cada salida está influenciada no sólo por la entrada actual, sino por el estado de la red en la etapa anterior. O sea, el estado computado con la entrada provista a la red en el tiempo t, es usado junto a la entrada provista en el tiempo t+1 para calcular el nuevo estado y la salida en t+1. Los operadores de desplazamiento (shift operators) q^{-1} representados en las conexiones hacia atrás señalan la sincronización de las dependencias entre etapas. Un exponente positivo advierte que la red requiere el estado computado en etapas anteriores y un exponente negativo, que se requiere el estado de etapas posteriores[4].

Existen otras arquitecturas, por ejemplo: las máquinas estocásticas (que usan las ideas de la mecánica estadística) y las redes Hopfield [Hil95], en las que todas las neuronas están conectadas entre sí.

[4] El uso de exponentes negativos puede no tener sentido en redes donde se simula la reconstrucción de señales en el tiempo; este tipo de exponentes se emplea con mayor frecuencia en problemas de análisis de datos estructurales. A las redes que resuelven este tipo de problema se le suele llamar *recursivas,* dejando el término *recurrentes* para aquellas donde se manejen relaciones temporales. En todos los casos el uso de los *shift operators* es un recurso para la notación en las redes discretas.

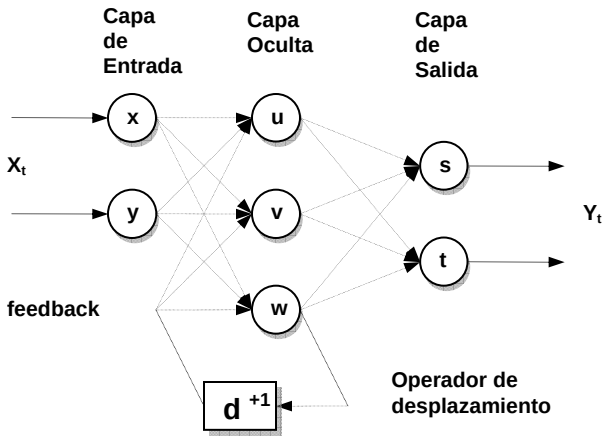


Figura 1.3. Red recurrente discreta con una conexión hacia una etapa anterior.

Una vez caracterizado un problema, la topología más adecuada para su solución debe ser seleccionada de manera empírica. Por un lado la potencia de cálculo para resolver los problemas complejos depende directamente de la cantidad de neuronas (cantidad de parámetros libres) de que disponga la red, y por otro, un número muy elevado de éstas puede afectar el desempeño de los algoritmos de entrenamiento. Tal situación conlleva a adoptar soluciones de compromiso entre el poder de representación de la red y su simplicidad. Por lo general la aplicación de heurísticas y técnicas especiales junto a la experiencia del especialista constituyen la forma más efectiva de abordar este problema.

1.2.4.3 Los Algoritmos de Aprendizaje.

Las redes neuronales poseen dos fases diferentes de trabajo: la fase de entrenamiento y la fase operativa. Ambas pueden estar mezcladas en ciertos modelos. La fase operativa es donde la red está disponible para el trabajo en modo óptimo de acuerdo al problema que trata de resolver. La fase de entrenamiento consiste en la aplicación de alguna técnica para lograr que la red ajuste sus parámetros, de manera que aprenda los aspectos esenciales del problema y logre la generalización de rasgos no previstos en el propio proceso de entrenamiento.

Los algoritmos de entrenamiento constituyen métodos que se aplican sobre los modelos de red para adaptar sus parámetros libres 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 (forma específica en que los parámetros libres son afectados). Existe una amplia variedad de algoritmos de entrenamiento

disponibles, la mayoría de ellos desarrollados para arquitecturas muy específicas. Cada uno se clasifica usualmente en supervisado o no supervisado [Hil95]. Los algoritmos supervisados son los que requieren, en su funcionamiento, información sobre las respuestas esperadas de la red, y los no supervisados los que no la requieren. Los algoritmos supervisados adaptan la red mediante la optimización de una función de error (que calcula las diferencias entre las respuestas deseadas y las obtenidas por el modelo, tomando como referencia un conjunto de datos fijos denominado conjunto de entrenamiento) sobre el espacio de los parámetros libres del modelo. Las técnicas supervisadas son caracterizadas por sus propiedades de convergencia global o local, rapidez con que logran dicha convergencia y los recursos en tiempo y espacio que requiere para su aplicación.

Entre las técnicas de entrenamiento supervisado más difundidas se encuentran aquellas que basan su funcionamiento en el método del gradiente descendente [Trh64]. Entre éstas, el algoritmo Backpropagation (BP, aplicado a redes feed-forward de forma directa) es el de más amplio uso. Backpropagation aplica la técnica gradiente descendente para la minimización del error de funcionamiento de la red. El error de la red para un patrón p determinado puede calcularse como la semisuma de los errores cuadráticos de cada unidad de salida al presentarse dicho patrón a la red:

$$E_p(W) = \frac{1}{2}\sum_{i \in O} (d_i - y_i)^2 \tag{1.15}$$

En la fórmula (1.15), W es el conjunto de pesos de las conexiones de la red (parámetros libres), yi y di corresponden respectivamente a la salida actual y la salida esperada de la i-ésima neurona de la red, dado un patrón de entrada determinado; O es el conjunto de neuronas de salida. El error total de funcionamiento de la red puede tomarse como la suma de los errores Ep(W) para cada uno de los patrones presentes en el conjunto de entrenamiento. El error calculado por la fórmula (1.15) es empleado frecuentemente en redes con funciones sigmoidales o lineales en la capa de salida. En tareas de clasificación múltiple donde se emplea softmax como función de activación, el error de un patrón puede calcularse mediante la fórmula conocida como cross-entropy para clasificación binomial o múltiple (1.16).[Sch99]

$$E_p(W) = -\sum_{i \in O} d_i \ln(y_i) \tag{1.16}$$

La función Ep(W) se representa sobre un espacio multidimensional de dimensión igual al número de pesos de la red. La búsqueda de la solución (conjunto de pesos) que minimice el error se realiza de acuerdo a la fórmula (1.17).

$$\Delta w_{ij} = -\alpha \frac{\partial E(W)}{\partial w_{ij}} \tag{1.17}$$

BP es capaz de calcular la magnitud de la influencia de cada peso sobre el error de respuesta de la red y realizar un reajuste de los mismos de manera proporcional (por medio de la constante de velocidad de entrenamiento α).

Las redes con aprendizaje no supervisado no requieren influencia externa para ajustar sus parámetros libres, o sea, no reciben información alguna sobre datos de entrenamiento. Su funcionamiento se basa en la construcción de clases o categorías a partir de las características comunes extraídas a los datos de entrada presentados. Dada una entrada, son capaces de construir la clase a la que pertenece, aún si la información brindada es parcial o ruidosa. Entre las técnicas más difundidas se encuentra el aprendizaje hebbiano y el aprendizaje competitivo y cooperativo [Hil95]. En el aprendizaje hebbiano, la idea básica consiste en establecer la actualización de los pesos entre neuronas a partir de una magnitud proporcional a la activación conjunta de éstas, de manera que una conexión es reforzada si ocurre la activación simultánea de sus neuronas y es debilitada siempre que una de ellas (o ambas) se encuentre desactivada. Entre los modelos más conocidos se encuentran las Memorias Asociativas Bidireccionales (Bidirectional Associative Memory BAM) [Kos88]. En el aprendizaje competitivo y cooperativo las neuronas compiten entre ellas a través de conexiones inhibitorias o participan cooperativamente a través de conexiones excitadoras en la realización de una tarea dada. En este tipo de redes solo una neurona o un grupo de ellas es activada cada vez que es presentado un patrón en la entrada. La red evoluciona de acuerdo a las contribuciones de cada neurona hacia sus vecinas, logrando sobresalir (activarse) o minimizar su influencia (desactivarse) total en la respuesta final. Entre las técnicas más representativas se encuentra la denominada Learning Vector Quantization (LVQ) y los llamados mapas auto-organizativos (self-organization maps) [Koh96], [Som99], [Tog92].

Referido al problema mencionado en el epígrafe anterior sobre la selección adecuada de la topología, han sido propuestas varias técnicas (para redes FFN y RNN) que permiten su modificación dinámicamente como parte del proceso de entrenamiento. Entre las técnicas más usadas se encuentran los algoritmos constructivos y la poda. Los métodos constructivos, llamados también de crecimiento, permiten comenzar con una

configuración mínima e ir añadiendo neuronas y conexiones paulatinamente hasta obtener una topología adaptada al problema en cuestión (Cascade-correlation [Fah90], Tiling, Upstart [Cam97]). La poda, al contrario, consiste en ir eliminando progresivamente las neuronas y conexiones menos útiles a partir de una red neuronal con una complejidad inicial arbitraria [Tin97], [Tin99].

1.2.4.4 Redes feedforward

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.

Las redes Multi Layer Perceptron (MLP) constituyen un ejemplo genérico de las redes FFN. En general se encuentran formadas por un conjunto de capas de neuronas ordenadas secuencialmente de la forma siguiente: una capa de entrada, un conjunto de capas intermedias denominadas capas ocultas y una capa de salida. En la figura 1.2 se muestra un ejemplo típico de este tipo de red.

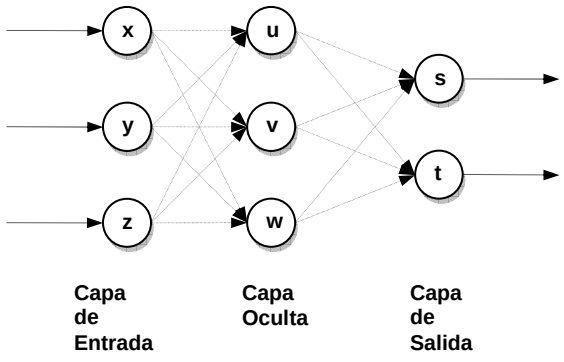


Figura 1.2. Una red MLP de 3 capas.

Cuando la red no tiene capas ocultas (single-layer network) ésta corresponde al Perceptron de Rosemblatt [Hil95], que constituyó el modelo precursor de las redes MLP.

La principal diferencia entre el perceptron y el MLP reside en la potencia de cómputo: el perceptron solo es capaz de separar linealmente (por un hiperplano) las entradas; mientras que el MLP, usando neuronas ocultas con funciones no lineales, es capaz de aproximar cualquier tipo de función continua y brindar excelentes resultados en las tareas de clasificación [Zur92], [Ham03]. El poder de representación de las redes MLP está relacionado con la existencia de al menos una capa de neuronas ocultas no lineales, las cuales transforman la entrada en una representación interna. Esta representación interna

puede servir luego a las capas subsecuentes para resolver los problemas tratados. En este sentido Lippman [Lip87] señala el poder de representación arbitrariamente complejo que puede llegar a presentar un MLP de 4 capa.

1.2.4.5 Redes Recurrentes

Las redes recurrentes se consideran sistemas dinámicos cuyo estado evoluciona siguiendo un marcado comportamiento no lineal. Como se describió en el acápite anterior, constituyen redes con conexiones de retroalimentación (feedback) que exhiben dos tipos básicos de funcionamiento [Son02], [Per90]:

Sistema autónomo convergente. Consiste en la evolución hacia un estado estable (punto fijo) siguiendo un proceso autónomo y partiendo de un conjunto de condiciones externas fijas,

Sistema no autónomo no convergente. Consiste en la variación del comportamiento en el tiempo de manera no autónoma, es decir exhibiendo cambios en el estado de las neuronas a partir de la influencia de estímulos variables recibidos en la entrada de la red a lo largo del tiempo.

El primer tipo de comportamiento permite realizar tareas de asociación (como las resueltas por los modelos de memorias asociativas [Hil95]), 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. En la figura 1.4 se muestra una red recurrente sencilla que es capaz de resolver el problema del XOR con una evolución hacia un punto fijo [Per90].

La red de la figura 1.4 posee dos neuronas, una para cada una de las señales de entrada I1 e I2; una neurona de salida o y una neurona oculta h. Una conexión de retroalimentación (conexión recurrente) desde la neurona o hacia la neurona h permite que la salida de la red dependa en cada momento del estado anterior. Suponiendo que la red trabaje actualizando continuamente los estados de activación de sus neuronas, la figura 1.5 muestra el funcionamiento de la misma en modo convergente (estabilización hacia un punto fijo dependiente de las condiciones iniciales). En la figura se muestra el comportamiento asintótico de la red $y \xrightarrow{\infty} 0$ una vez que se fija en sus entradas la información correspondiente (I1=1, I2=0).

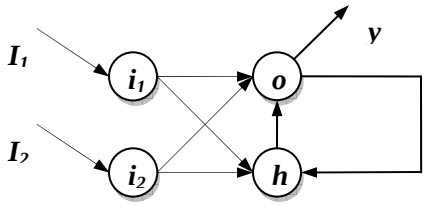


Figura 1.4. Red recurrente continua para el problema XOR.

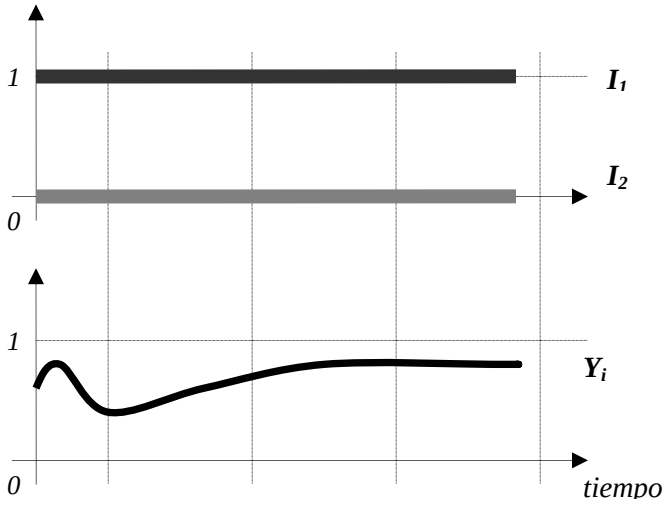


Figura 1.5. Funcionamiento convergente de la red recurrente para el problema XOR.

La figura 1.6 muestra el comportamiento de la red en modo no convergente. La diferencia esencial es que la red no evoluciona esta vez hacia un estado estable, sino que varía su salida de manera dinámica, exhibiendo un comportamiento dependiente en cada momento de los valores de las señales de entrada.

De manera general, una red recurrente puede expresarse como un conjunto de ecuaciones de la forma (1.18) [Zip95].

$$y_i(t) = \Phi_i\left(x_i(t), I_i(t)\right) \tag{1.18}$$

Donde yi denota el estado de activación de la neurona i en el tiempo t,

$$x_i = \sum_j y_j w_{ji} \tag{1.19}$$

Ii(t) denota la entrada exterior de la neurona i y Φi la función que gobierna el comportamiento del estado de activación de la neurona i en el tiempo t.

Φi constituye una función influida por la estructura[5] (topología) de la red (expresado en los valores de entrada netos xi) y por la función de entrada Ii(t) que aporta las señales provenientes del exterior.

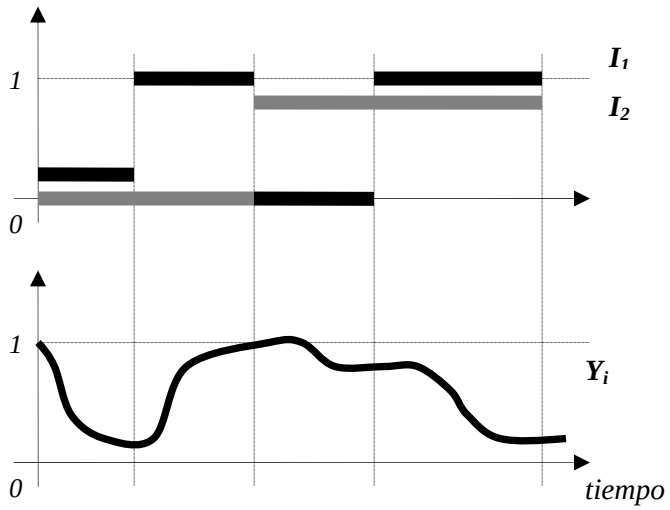


Figura 1.6. Funcionamiento no convergente de la red recurrente para el problema XOR.

Lo anteriormente dicho puede expresarse a través de las relaciones siguientes

$$N = (W, G(V,E), \{\Phi_i\}) \quad |\{\Phi_i\}| = |V|, \ \Phi_i : f \longrightarrow g, \ f, g \in F_{[\Re]} \tag{1.20}$$

$$L(W, G(V,E)) \longrightarrow \{\Phi_i\} \ W \in \Re^{|E|}, G \in \Im$$

N representa una red neuronal con una topología que puede ser caracterizada por el grafo dirigido G, con un conjunto de nodos (neuronas) V y un conjunto de arcos (conexiones) E. Los pesos de las conexiones están expresados en el vector W. Estos elementos describen estructuralmente a la red. El conjunto de funciones de activación $\{\Phi_i\}$ brinda una caracterización dinámica del comportamiento de la red neuronal en el tiempo. La segunda expresión representa la aplicación L con dominio en $\Re^{|E|} \times \Im$ ($\Re$ es el conjunto

[5] Nótese que una red recurrente es, como otras redes, un flujo de funciones interactuando entre sí (la salida de una es la entrada de otras incluyéndose a ella misma), por lo que una función de activación depende directamente de las demás, y la forma específica de esta dependencia está determinada por la estructura de las conexiones.

de los número reales y $\Im$ el conjunto de los grafos dirigidos; cada elemento del conjunto producto representa una topología y un conjunto de pesos determinado para la red) e imagen en el conjunto potencia del conjunto de las funciones que transforman una función real en otra (en esta formalización solo se han tenido en cuenta las redes recurrentes con dominio de trabajo en los reales). La aplicación L expresa la relación existente entre estructura y comportamiento dinámico de las redes recurrentes [Zip95], [Ati00], [Tso98], [Ham02a]. El objetivo principal de los investigadores en redes recurrentes estriba en el conocimiento de dicha relación y la producción de técnicas (algoritmos de entrenamiento, heurísticas) 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.

Resolver un problema determinado significa frecuentemente (como en el caso del problema XOR) ajustar las salidas de la red de manera que ésta exhiba el comportamiento requerido (en el caso del problema XOR se requiere que la salida Y se aproxime al valor 1, cuando las señales de entrada sean diferentes, y al valor 0 cuando sean iguales). Los valores de salida de la red a lo largo del tiempo son caracterizadas por las funciones de activación (que traducen una función de entrada I(t) en una función de salida y(t)); y éstas a su vez dependen de la interacción de los parámetros libres (valores de los pesos en las conexiones wij) en un entorno (topología) determinado; por lo que, ajustar las salidas de la red, significa encontrar la topología G y el vector de pesos W que mejor aproximan cada función de activación de la red a la función real deseada. Esta situación es válida tanto para redes continuas como discretas. La diferencia sustancial entre estos dos tipos de redes reside en los mecanismos que se emplean para deducir y aplicar las fórmulas relativas a las técnicas de entrenamiento.

Las redes con funciones reales que operan sobre una escala de tiempo continua se denominan redes recurrentes continuas. Igualmente se denomina redes recurrentes discretas a aquellas que expresan el comportamiento del estado de activación de las neuronas solo en puntos determinados t0 , t1, …, tn en el eje del tiempo.

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 [Ati00], en 5 clases generales:

- Backpropagation ThroughTime (BPTT),
- Forward Propagation o Real Time Recurrent Learning (RTRL),
- Fast Forward Propagation,

- Funciones de Green
- Actualización en Bloque (Block Update).

Cada una de las formulaciones puede ser derivada para redes discretas y redes continuas de forma muy similar. En el caso continuo las técnicas se reducen, por lo general, a la aplicación de herramientas matemáticas propias a la solución de sistemas de ecuaciones diferenciales [Per90]; y en el caso discreto, la actualización de los estados de activación y de los pesos de las conexiones se realiza sincrónicamente por etapas siguiendo un orden determinado.

Entre las técnicas de gradiente descendente más empleadas para el entrenamiento de redes recurrentes se encuentran las variantes BPTT y RTRL [Zip95], [Bod01], [Ati00], [Tso97], [Per90], [Trh64], [Ham03], [Wer90]. El algoritmo BPTT es una variante extendida del backpropagation original que posee requerimientos de tiempo y memoria crecientes, proporcional a la longitud de las secuencias que procesa. Solo es capaz de reajustar sus pesos una vez que toda la secuencia de entrada ha sido procesada, por lo que su aplicación a entornos de tiempo real es restringida. RTRL es un algoritmo derivado del BPTT por William y Zipser [Zip95] para obtener una versión on-line del mismo. La principal ventaja de este algoritmo radica en limitar considerablemente la cantidad de memoria requerida por BPTT para su operación, a la vez que disfruta de la generalidad de aplicación de este último.

Existen variaciones del algoritmo BPTT (Backpropagation ThroughStructure, BPTS) para procesar estructuras de datos jerárquicas sin necesidad de transformarlos a un formato de secuencia. Starita y Sperduti [Sta97] han producido un enfoque sistemático para adaptar redes recurrentes y algoritmos de entrenamiento al análisis de estructuras de datos complejas. Hammer [Ham02] presenta un estudio sobre las redes recurrentes aplicadas al procesamiento de estructuras de dato complejas; Siu-Yeung Cho [Cho03] señala una heurística que mejora el desempeño de BPTS.

Bengio [Ben94] realiza un análisis de la efectividad del BPTT aplicado a problemas donde se exige el aprendizaje de dependencias en secuencias muy grandes. Señala la imposibilidad de lograr, bajo ciertas circunstancias, el aprendizaje adecuado de cadenas de dependencias muy largas y a la vez lograr la capacidad de generalización en entornos ruidosos (problema del desvanecimiento del gradiente). A partir de un análisis teórico y experimental aporta sugerencias para mejorar los algoritmos y brindar conocimiento a priori en el proceso de entrenamiento.

Hammer brinda un informe abarcador en [Ham02a] sobre las perspectiva actuales de desarrollo de los algoritmos de aprendizaje en redes neuronales recurrentes. En el siguiente epígrafe centramos nuestra atención en el algoritmo Backpropagation ThroughTime para modelos de redes discretas.

1.2.5 Redes Recurrentes Discretas. Backpropagation ThroughTime (BPTT).

Una red discreta conserva el principio de funcionamiento general para las redes recurrentes, la diferencia radica en que la actualización del estado de activación de cada neurona se realiza en momentos puntuales de tiempo t1, t2,…tn, etc. La red para el problema XOR de la figura 1.4 trabajando en modo convergente podría mostrar un comportamiento discretizado en el tiempo como se aprecia en la figura 1.7.

En cada momento de tiempo ti la función de activación propia de cada neurona es calculada siguiendo el orden correspondiente. En la variante discreta de la red XOR resulta contradictorio la actualización simultánea de la neurona de salida ***o*** y la neurona oculta ***h***, pues existe un ciclo de dependencia. En este caso, para el cálculo de la entrada neta de las neuronas ***o*** y ***h*** se emplean los valores de activación calculados en el momento de tiempo anterior (deben prefijarse los valores de los estados de activación para estas neuronas en el momento t_0, usualmente se escoge el valor 0). Si aplicamos esta idea regresivamente desde el momento t_i hacia t_0 podemos establecer una relación de dependencia como la que se muestra en la figura 1.8.

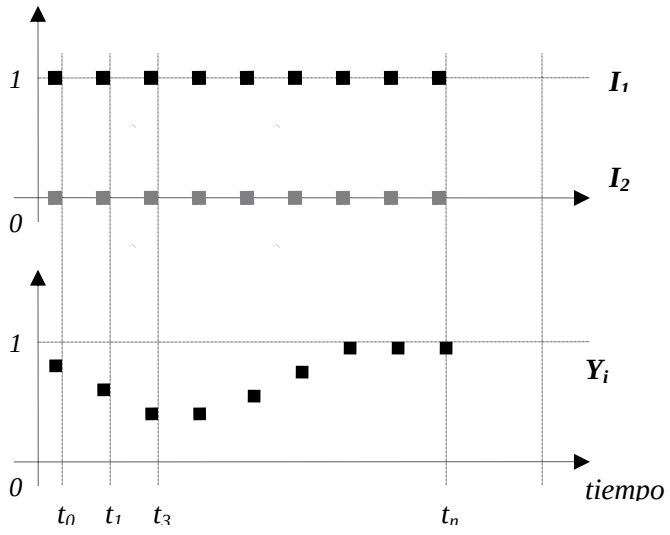


Figura 1.7. Funcionamiento convergente de una red recurrente discreta para el problema XOR.

La red neuronal así obtenida no es más que el resultado de la replicación de la red recurrente original (desprovista de las conexiones recurrentes) una vez por momento (etapa) de tiempo considerado en la secuencia. Las conexiones recurrentes conectan la neurona origen de la etapa anterior con la neurona destino en la etapa próxima. Cada conexión replicada es exactamente el mismo parámetro y comparte su valor wij en todas las etapas. El proceso mediante el cual se obtiene la red aumentada a partir de la red recurrente original se denomina despliegue (unfolding). Nótese que la red desplegada no es más que una red feed-forward, y es susceptible de la aplicación del algoritmo Backpropagation para su entrenamiento. Ésta es precisamente la idea subyacente en el algoritmo BPTT.

La red discreta puede representarse más concisamente con el empleo de los operadores de desplazamiento (shift operators). La figura 1.9 muestra las conexiones hacia la etapa posterior con el operador d+1.

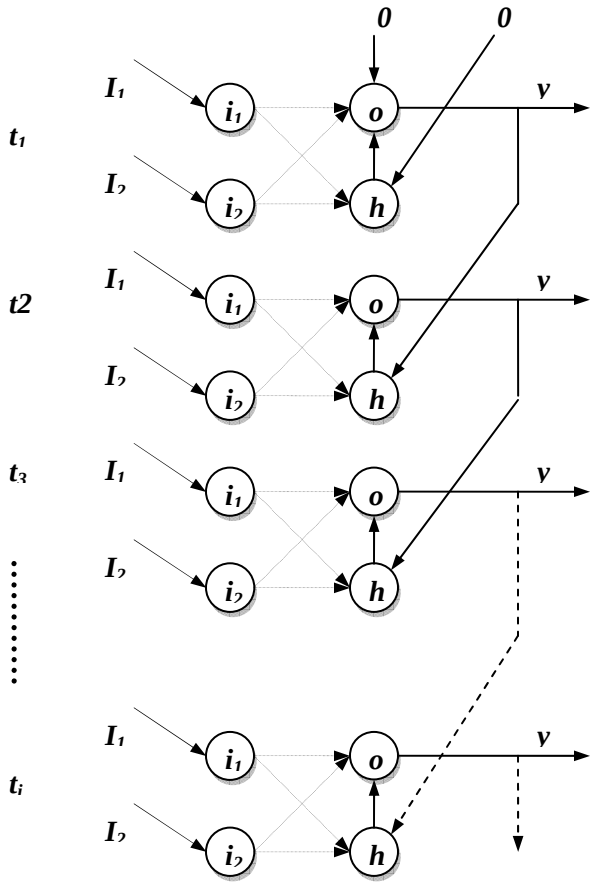


Figura 1.8. *Unfolding* de la red recurrente discreta para el problema XOR.

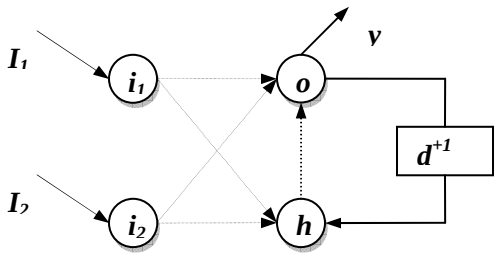


Figura 1.9. Red Recurrente Discreta para el problema XOR con operadores de desplazamiento.

En el ejemplo descrito hemos supuesto la conexión recurrente desde la etapa inmediata anterior. En sentido general, la conexión puede realizarse con un desplazamiento (*delay*) mayor que la unidad. Esta propiedad se emplea cuando se requiere que la red calcule los estados de las próximas etapas a partir de una información histórica más larga (la memoria de almacenamiento de los estados anteriores puede hacerse tan larga como se necesite, siempre que los algoritmos de entrenamiento puedan hacer un uso efectivo de ella [Ben94]).

Para las redes discretas la formulación (1.20) puede ser expresada más específicamente como:

$$y_i(t) = f(x_i(t)) \tag{1.21}$$

$$x_i(t) = \sum_{j \in U} y_j(t) w_{ij} + \sum_{j \in I} x_j^{in} w_{ij} + \sum_{j \in S} y_j(t - \tau_{ij}) w_{ij} \tag{1.22}$$

f se refiere a la función de activación del modelo de la neurona (frecuentemente una variante sigmoidal o tangencial); U denota los índices de las neuronas de la red (sin considerar las neuronas de entrada), I los índices de las neuronas de entrada, x_j^{in} es la j-ésima neurona de entrada. S denota los índices de las neuronas que almacenan la información de estados anteriores de la red (memoria de estado). $\tau_{ij} \geq 0$ es una magnitud entera que refiere el desplazamiento de las conexiones feedback en el tiempo (*shift operator* $q^{-\tau}$, $q^{+\tau}$).

Más concisamente:

$$x_i(t) = \sum_{j \in U, j \in I} z_j(t - \tau_{ij}) w_{ij} \tag{1.23}$$

$$z_j = \begin{cases} y_{j,} \, j \in U \\ x_j^{in}, \, j \in I \end{cases}$$

La notación *shift operator* puede usarse indistintamente siendo $q^{-\tau}$ zj = zj(t -τ).

BPTT consiste en aplicar el proceso de *unfolding* a la red recurrente original, y ejecutar Backpropagation a la red feed-forward obtenida. Backpropagation posee dos fases: la primera, el cálculo de las salidas de las neuronas de la red (proceso *forward*), la segunda, la propagación del error hacia atrás (proceso *backward*). En el proceso de retropropagación del error cada neurona j es caracterizada por una magnitud de error δj. Para neuronas de la capa de salida (función de activación sigmoidal) esta magnitud se calcula según:

$$\delta_j(t) = (d_j - y_j) y_j (1 - y_j) \tag{1.24}$$

Para neuronas ocultas:

$$\delta_j(t) = y_j(1 - y_j) \sum_{i \in Suc(j)} wij \delta_i \tag{1.25}$$

La actualización de los pesos de las conexiones se realiza según:

$$\Delta w_{ji}^{t+1} = \alpha \delta_j y_j \tag{1.26}$$

Donde α corresponde a la tasa de convergencia.

En la fórmula, t no se refiere a las etapas de la red recurrente sino a un ordenamiento sucesivo de las actualizaciones de los pesos en el proceso de entrenamiento.

Suponiendo una red recurrente N y un conjunto de secuencias de valores de entrada y salida (conjunto de entrenamiento); un orden general de los pasos del algoritmo podría ser:

$P \longleftarrow\!\!\!- LearningParameters$, $\xi \longleftarrow\!\!\!- MeanError$

$S \longleftarrow\!\!\!- \{(\bar{I}, \bar{d})\}$, Learning Pattern Set

$E \longleftarrow\!\!\!- 0$, $\forall s \in S$

$N' \longleftarrow\!\!\!- Unfolding(N, |s|)$

$O \longleftarrow\!\!\!- Fordware(N', \bar{I}_s)$

$e_s \longleftarrow\!\!\!- \delta - errorBackware(N', \bar{d}_s, O, P)$

$N \longleftarrow\!\!\!- ActualizeSharedWeights(N')$

$E \longleftarrow\!\!\!- E + e_s$

$E \longleftarrow\!\!\!- \frac{E}{|S|}$.

If $E > \xi$ goto 1

End.

La entrada del algoritmo consiste en un conjunto de patrones de entrenamiento (secuencia de entrada $\bar{I}$ con su correspondiente salida $\bar{d}$), los parámetros de entrenamiento P, y el error medio de entrenamiento ξ esperado sobre el conjunto S. El primer paso consiste en la aplicación del procedimiento backpropagation en la red feed-forward N' asociada a la red recurrente original, que es obtenida por el unfolding sobre cada secuencia de entrada particular (1.a – 1.c). Las conexiones replicadas en la red N' que corresponden a la misma conexión original en la red N se consideran un solo parámetro libre compartido (usualmente las actualizaciones diferentes de este mismo parámetro obtenidas por el proceso de retropropagación en las distintas etapas, son combinadas para obtener un único resultado de actualización sobre la red original N). El paso 1.d realiza la actualización de cada conexión compartida wij de la red original N a partir de los valores Δwijt correspondientes en cada etapa. Una posible forma de combinar estos valores podría ser la simple suma de las actualizaciones obtenidas en cada etapa [Bod01], [Son02], [Bal02], [Bal01].

$$\Delta w_{ij} = \sum_{t} \Delta w_{ijt}$$

La siguiente etapa verifica si se ha superado el parámetro de error aceptable para el aprendizaje y concluye la ejecución.

1.3 Trabajos relacionados en la clasificación de resistencia a fármacos del VIH.

Varios métodos estadísticos han sido usados para investigar la relación entre los resultados de ensayos genotípicos y fenotípicos del VIH-1. Análisis de cluster, análisis discriminante linear, métricas de almacenamiento heurístico, el vecino más cercano, redes neuronales y árboles de decisión han sido usados para correlacionar la susceptibilidad a la droga con el genotipo. Sin embargo, el problema de relacionar el resultado de ensayos genotípicos y fenotípicos proveen varios retos y el éxito de esos métodos han sido mezclados. Primeramente, el fenotipo debe ser considerado como una consecuencia de un número largo de posibles mutaciones. Como previamente se ha mencionado, éste está compuesto por el hecho de que el efecto de mutaciones a cualquier posición dada es influenciado por la presencia de mutaciones a otras posiciones y es, sin embargo, necesario para detectar interacciones globales.

El uso de análisis de cluster y análisis discriminante linear es descrito en [Sev00]. Los resultados de la investigación de mutaciones de resistencia a drogas de dos inhibidores de la Proteasa (SQV y IDV) fueron comparados. En particular, ambos análisis fueron útiles para identificar la asociación de mutaciones a las posiciones de los aminoácidos 10, 63, 71 y 90 con resistencia in vitro a SQV y IDV.

Típicamente un conjunto de indicadores de variables es creado para cada posición de aminoácido y un vector de variables de indicador es entonces usado para representar una secuencia. La distancia entre dos secuencias puede entonces ser definida como la distancia Euclidiana estándar entre los dos vectores correspondientes. Tal medida puede ser usada entonces para crear grupos de secuencias de aminoácidos con genotipos similares. Sin embargo, la distancia entre todos los pares de miembros de los dos grupos puede ser definida como el promedio de las distancias individuales entre todos los pares de miembros de los dos grupos. Crear jerarquías de tales grupos o clusters entonces facilita la investigación del grado para el cual similares genotipos tienen fenotipos similares [Jam04].

Por otro lado, el análisis discriminante linear puede ser usado para determinar cuáles mutaciones predicen mejor la sensibilidad a las drogas, como definido por el valor

fenotípico IC50. Un conjunto de datos consistente de pares de genotipo y fenotipo es dividido en dos grupos, resistentes o susceptibles, dependiendo del cut-off de IC50. Una función discriminante linear es entonces usada para predecir a cuál grupo pertenece un genotipo desconocido. Donde la función discriminante es simplemente una combinación linear de predictores o variables de indicadores.

En contraste, una simple métrica es usada en [Scm00] para predecir la resistencia a inhibidores de la Proteasa. Aquí fue analizada una base de datos de pares de genotipo y fenotipo. Fue encontrado que estos ejemplos con uno o dos mutaciones en la Proteasa fueron susceptibles fenotípicamente y ejemplos con 5 o más mutaciones fueron resistentes contra todos los inhibidores. Una lista de todas las mutaciones presente en la base de datos fue compilada y dividida hacia dos grupos. Esto fue mejorado por la incorporación de una base secundaria que simplemente tiene en cuenta el número total de mutaciones asociadas a resistencia en la Proteasa. Este logró una sensibilidad alta (92.5-98.5%) pero especificidad baja (57.9-77.3%) en la base de prueba.

En [Wan03] las redes neuronales fueron usadas para predecir resistencia a lopinavir desde genotipo. En [Wan03] dos modelos de redes neuronales fueron desarrollados. El primero fue basado en cambios de 11 posiciones de aminoácidos en la Proteasa, como descrito en la literatura, y la segunda fue basada en 28 posiciones de aminoácidos resultando de análisis de preponderancia de categoría. Un conjunto de 1322 ejemplos de clínicas fueron utilizados para entrenar, validar y probar los modelos. Los resultados fueron expresados en términos del coeficiente de correlación. En términos simples de coeficiente de correlación indica a cuáles extender la predicción y valores de verdad en línea recta. Fue encontrada que el modelo de 28 mutaciones probado para ser más exacto que el modelo de 11 mutaciones para predecir la resistencia del lopinavir.

Alternativamente los clasificadores con árboles de decisión fueron generados por medio de divisiones recursivas en [Bee02]. Estos modelos fueron entonces usados para identificar características de los patrones de genotipo de resistencia para 14 antirretrovirales. Se usaron divisiones recursivas para la construcción de un árbol de decisión. Los algoritmos de divisiones recursivas comienzan por dividir una población de datos a sub-poblaciones pro la determinación de un atributo que es mejor divisor de la población inicial. Continúa por repetir la identificación repetida de atributos que dividen mejor cada sub-población resultante. Una vez que una sub-población contiene individuos del mismo tipo no se hacen más divisiones y se asigna la clasificación a esta población. Un caso desconocido es entonces dado a una clasificación por el ordenamiento bajo el árbol desde la raíz hasta una hoja usando el atributo como prueba específica. Aquí la

población inicial fue una base de casos de pares de genotipo fenotipo, consistiendo de 471 ejemplos clínicos. Para cada droga en el estudio, se construye un árbol de decisión para predecir la resistencia desde el genotipo usando una división recursiva. Estos modelos fueron calculados usando experimentos leave-one-out y errores de predicción estuvieron en el rango 9.6-32%.

En [Jam04] se pueden ver los últimos resultados reportados. Aquí se utilizó una variante de árbol de decisión y el algoritmo KNN para predecir la resistencia donde se obtuvieron buenos resultados comparándolos con los resultados reportados en [Bee02] descritos anteriormente. Para los árboles de decisión los errores de predicción estaban entre 6.0 y 24.7%, obteniéndose buenos resultados para unos fármacos y malos para otros, pero fueron superiores a los reportados anteriormente que estaban entre 8.1 y 51%. Con el KNN se obtuvieron errores de predicción entre 18 y 46.2 %.

1.4 Posibilidades de uso de nuevas energías.

Uno de los pasos para aplicar un método de clasificación, como se ha dicho anteriormente, es encontrar unos descriptores para representar la información de entrada. En algunos trabajos se usa la simple representación de vectores de de 20 elementos para representar cada aminoácido, donde tiene valor uno el elemento que coincide con la posición del aminoácido que representa y cero todos los demás. En [Jam04] y [Bee02] se utilizan los *profile* de información mutua para representar cada secuencia de la enzima de la Proteasa. Esta representación ha dado buenos resultados para representar la secuencia.

En busca de una representación lógica se analizó que la energía de contacto de los aminoácidos es la que propicia y determina el despliegue de la enzima en el espacio (Ver Anexo 6). El cambio de un aminoácido influye en el cambio de la energía del aminoácido para atraer al resto de los aminoácidos, y este cambio influye en la estructura 3D de la enzima. Por esta razón en este trabajo se usa la energía para representar los aminoácidos de la secuencia de la Proteasa.

En [Miy94] se mostró que las energías de contacto cambian la estructura de la proteína. El reemplazo de un simple aminoácido es suficiente para los valores observados de las energías libres cambien. [Miy96].

1.5 Conclusiones parciales. Hipótesis de Investigación.

En este capítulo se abordó el estado del arte para los problemas de clasificación de la resistencia a fármacos del VIH, donde se introduce la problemática desde el punto de

vista biológico, los métodos de clasificación más usados y los resultados alcanzados para predecir esta resistencia hasta el momento.

En [Jam04] y [Bee02] podemos ver los trabajos más recientes para la clasificación de resistencia de enzimas como la Proteasa del VIH. En estos dos trabajos usan métodos de clasificación como KNN, árboles de decisión y SVM. En este trabajo pretendemos usar SVM también y comparar esos resultados con los que se alcancen usando redes neuronales MLP y redes recurrentes bidireccionales. El uso de estos métodos será descrito en el próximo capítulo.

Como se ha dicho antes el inhibidor de la Proteasa se une al sitio activo de la misma. En la práctica es necesario ver la estructura 3D de esta enzima que es la que le permite a los científicos buscar la forma del fármaco para que acople en el sitio activo. Precisamente, como se ha explicado aquí el cambio de un aminoácido cambia energía de contacto de los mismo y así su despliegue en el espacio, cambiando su estructura tridimensional. Por todo esto se ha escogido la energía de contacto como un descriptor de los aminoácidos de la secuencia de la Proteasa para este problema.

Para este problemas se han planteado las siguientes hipótesis:

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 Es posible usar redes recurrentes bidireccionales para predecir la resistencia de mutaciones de la Proteasa ante determinados fármacos.

Capítulo II: Modelación del problema de clasificación de la resistencia a fármacos del VIH.

Como se explicó en el epígrafe 1.2 el diseño de un sistema de aprendizaje, tiene como pasos: la definición de la tarea, la forma de medir el aprendizaje del sistema, la experiencia para el entrenamiento, la definición de la función objetivo y por último la selección del mecanismo de aprendizaje.

Los dos primeros pasos se puede decir que ayudan a la definición de la problemática. El resto ayudan a la búsqueda del método de solución para ese problema.

Como el problema que estamos analizando es de clasificación, ya habíamos dicho que la tarea consiste en tomar cada instancia y asignarla a una clase particular. En este trabajo la tarea específica podemos definirla como: la clasificación de secuencias de la Proteasa en resistentes o susceptible ante el fármaco analizado. En nuestro caso como son 7 fármacos estamos en presencia de 7 problemas de clasificación, uno para cada fármaco.

La medida de representación estará dada por el porciento de secuencias correctamente clasificadas, que es precisamente lo que me dará una idea del aprendizaje del método que se analice.

La experiencia para el entrenamiento y el tipo exacto del conocimiento aprendido es precisamente lo que explicaremos en el siguiente epígrafe.

2.1 Construcción de las Bases de Conocimiento para la clasificación de resistencia a 7 fármacos del VIH.

Después de definir la tarea y la medida de representación del problema, queda definida nuestra problemática. El tercer paso es buscar la información que servirá de entrenamiento al sistema.

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. Las bases de datos más conocidas y con más casos disponibles son: “Los Álamos” y “Stanford HIV Resistance Database”. Estas son las bases de datos más usadas para experimentos relacionados con este problema. La primera contiene la información de la secuencia de codones de la Proteasa y la segunda la información de la secuencia de aminoácidos.

La representación de la estructura primaria de una secuencia en codones tiene más información que la representación mediante la secuencia de aminoácidos. Para un mismo

aminoácido pueden codificar varios codones, de hecho existen 20 aminoácidos y 64 codones diferentes (Ver Código Genético en el Anexo2). Haciendo este análisis se puede ver que el uso de los casos de esta base de datos podía ser importante, pues se contaba con una representación más completa de la enzima de la Proteasa. Pero por otro lado se tenía el problema de la representación de la resistencia asociada al fármaco, pues en esta base no se reporta la misma en todos los casos.

A partir de la bibliografía reportada se encontró la resistencia asociada a todos los casos de la base pero su representación es muy variada, ya que existen varias formas para medir la resistencia y los autores las usan indistintamente. No se encontró vía de unificar estas medidas y por esta razón se decidió desechar estos casos.

En la base de datos de Stanford 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 contiene información a cerca de 7 fármacos, y puede ser bajada en: http://hivdb.stanford.edu/cgibin/PIResiNote.cgi.

En la figura 2.1 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 y el aminoácido que fue cambiado. El fenotipo está representado por el factor de resistencia basado en la concentración de cada uno de los 7 inhibidores de la Proteasa como se explicó en el epígrafe 1.1.4.2 (ver ecuación 1.1).

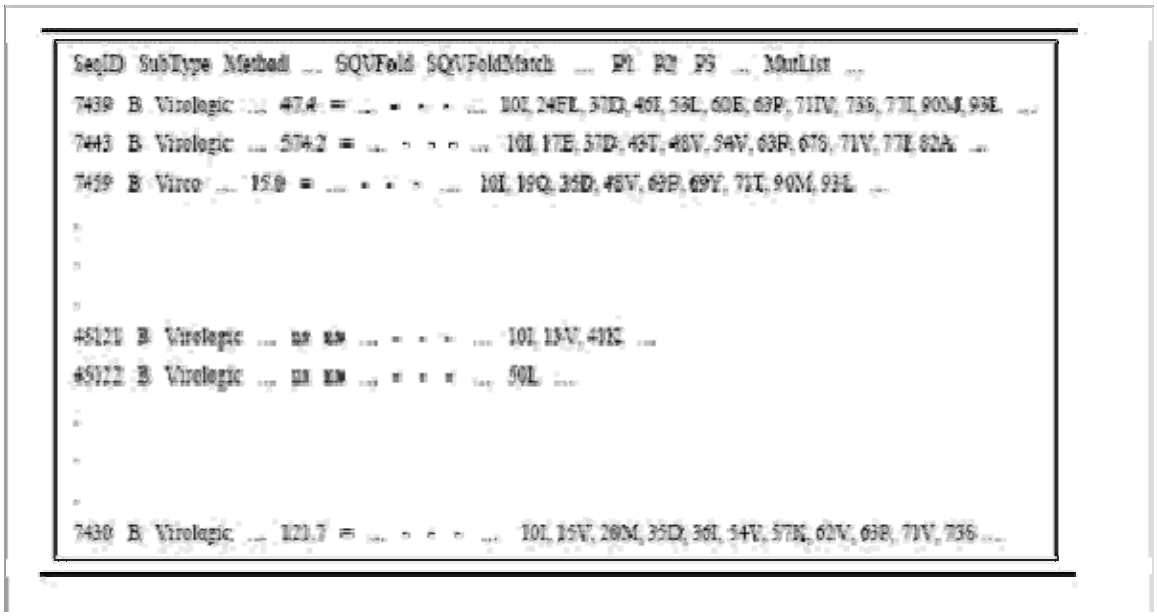


Figura 2.1 Base de datos de Stanford

Las mutaciones que describen a cada secuencia están basadas en una secuencia referencia, HXB2, que es conocida como secuencia del virus salvaje. 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 fig2.2.

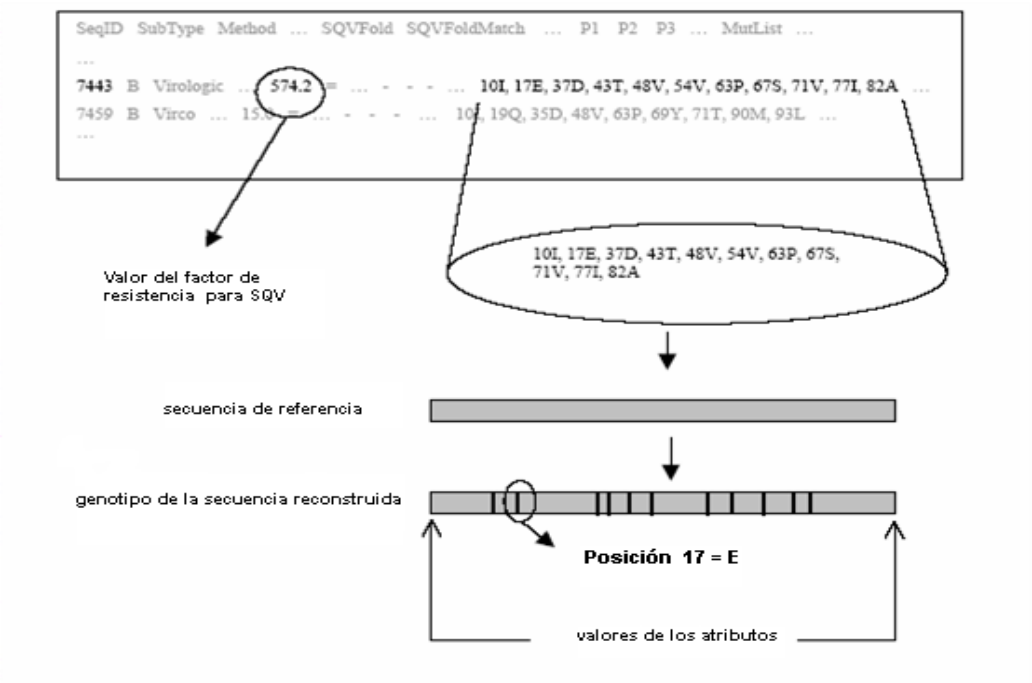


Figura 2.2 Conversión de los casos de la base de Stanford para la base de conocimientos.

De esta manera ya contamos con la información de experiencia que necesitamos 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. Por lo que podemos decir que tenemos 7 bases de casos con esta información, una para cada fármaco. Hasta ahora ya tenemos nuestro problema de aprendizaje bien definido, el próximo paso en la modelación del sistema de clasificación es definir la función objetivo, o sea una representación para este conocimiento.

Por todo lo analizado en el epígrafe 1.4, se seleccionó la energía de contacto de los aminoácidos para representar la secuencia de la Proteasa. En este problema vamos a analizar dos representaciones, una con la energía asociada a cada aminoácido y otra con la variación de la energía con respecto a la secuencia de referencia, o sea que en las posiciones en que la secuencia analizada haya mutado habrá una variación de energía y en el resto no habrá variación. A esta variación le denominaremos en lo adelante Δ*Energía*.

En la secuencia reconstruida, para cada caso, en la primera variante, se sustituyó cada aminoácido por la energía de contacto correspondiente. Y de esta manera cada caso tiene 99 parámetros de entrada, ya que la secuencia de la Proteasa tiene una longitud de 99 aminoácidos. Y en la segunda variante de representación se hizo de igual manera, pero sustituyendo por la Δ*Energía*.

La clase de cada caso está basada en el factor de resistencia para el que, según la bibliografía [Jam04], se definió un corte (cut-off) para cada fármaco, que casualmente en el caso de estos 7 fármacos es el mismo: 3.5. Si la resistencia es superior al corte se clasifica en resistente, pues significa que hay que suministrar más de 3.5 veces la concentración del fármaco para inhibir la Proteasa que se está analizando que la secuencia de referencia, y en caso de ser menor que 3.5 se clasifica como susceptible. De esta manera el problema de convierte en un problema de clasificación de dos clases.

La función objetivo se define como:

F: C → O,

donde, $C \in R^{99}$, para este caso la base consta con secuencias de igual tamaño, 99 aminoácidos. Cada elemento de C es un aminoácido representado por su *Energía* o Δ*Energía* que es un valor real, o sea C va a representar la secuencia de la Proteasa.

O = {resistente, susceptible}.

De los casos de la base de datos se eliminaron aquellos que tenían cambios por aminoácidos desconocidos, para eliminar la ausencia de información. En total se construyeron 7 bases de conocimiento, con la cantidad de casos que se muestra en la tabla 2.1.

Fármaco	Número de casos
SQV	386
LPV	175
RTV	306
APV	373
INV	377
ATV	121
NFV	403

Table 2.1. Número de casos en las bases de conocimiento de cada fármaco.

Finalmente ya tenemos nuestro diseño del problema, sólo nos queda buscar el mecanismo de aprendizaje apropiado. Para esto seleccionamos 3 variantes: SVM, redes neuronales MLP y redes recurrentes bidireccionales, que serán descritos en los próximos epígrafes.

2.2 Uso de SVM para predecir la resistencia

Como se describió en el epígrafe 1.2.3, SVM es un método nuevo que puede usarse en problemas de clasificación y que ha demostrado sus potencialidades en algunas aplicaciones. Además en [Bee02] son usados, precisamente, para resolver el problema que nos acomete, siendo justamente el método con mejores resultados reportado hasta el momento.

Actualmente existen varios software para crear SVM. En este trabajo se usó la biblioteca LIBSVM desarrollada por Chang en [Cha01] para el entrenamiento y prueba de SVM para estas bases de conocimiento, con vistas a comparar los resultados con los obtenidos en [Bee02], donde fue utilizado también. De esta forma se intentaba comprobar que el uso de las energías como descriptores de la secuencia era satisfactorio y efectivamente podían ser usadas como parámetros de entrada en este problema.

Este software permite hacer clasificación lineal y no lineal. Para la clasificación no lineal tiene los 3 kernel descritos en el epígrafe 1.2.3 (Ver ecuaciones 1.11, 1.12 y 1.13).

2.3 Uso de MLP para predecir la resistencia

El Multilayer Perceptron es un modelo de red neuronal artificial que simula una de las incontables funciones de nuestro sistema nervioso: la clasificación. Para esto, se vale de la simulación estructural y funcional de parte del mismo. Justamente ésta fue una de las causas por las que se escogió el MLP para aplicarlo a este problema. Como los casos de las bases con que contamos tienen secuencias con una longitud fija, se hizo posible el uso de estas redes.

Dada la complejidad de la labor de diseño e implementación de los MLP y la lentitud con que se ejecutaban estos procesos se han elaborado softwares que ayudan al diseñador de la red en estas tareas y que implementan algoritmos de entrenamiento que logran iguales resultados que los tradicionales, en intervalos menores de tiempo. SmartMLP, es una aplicación desarrollada justamente para facilitar el manejo de este tipo de redes. Esta aplicación permite crear, entrenar y explotar un MLP. Es difícil definir a priori que cantidad de capas y neuronas se necesitan para un problema dado. El uso de este programa ayuda a construir distintas RNA y al final el usuario puede decidir cual usar.

2.3.1 Facilidades del SmartMLP

SmartMLP hizo algunas mejoras del algoritmo de entrenamiento Backpropagation, para lograr una mayor eficiencia de éste. Se le realizaron varios cambios y adiciones. Las mejoras introducidas fueron las siguientes:

- Uso de una función heurística para el cálculo del coeficiente de aprendizaje. Esto contribuye a mejorar la convergencia del algoritmo, acelerando la velocidad del mismo:

$$\eta = \sqrt{N_1^2 + N_2^2 + \cdots + N_n^2} \tag{2.1}$$

  donde N_i es la cantidad de ejemplos del conjunto de aprendizaje pertenecientes a la clase i y n es total de clases

- Normalización de las entradas.
- Introducción de técnicas pedagógicas de selección de ejemplos durante el entrenamiento. Específicamente Repetir Hasta Aprender (RHA), la cual contribuye a que cada actualización de pesos se haga basándose en los ejemplos que generan mayor error, por tanto el cambio en los pesos es mayor y por consiguiente la cantidad de pasos a dar para alcanzar los pesos adecuados es menor.
- Adición de un término adicional a la función de actualización de pesos –Método del Momento -, el cual provoca que la red tienda a mantener la dirección en la actualización que llevaba. De manera que si la red está convergiendo hacia un mínimo y se lo salta, adquiriendo con esto un sentido contrario al que llevaba, la inercia de la actualización hará que los pasos siguientes sean menores, intentando atraer nuevamente a la red hacia el mínimo.
- Inclusión de dos criterios de paradas extras que pueden contribuir a eliminar iteraciones innecesarias del algoritmo de entrenamiento.

Analizando todas estas mejoras que introduce esta aplicación, se decidió su uso para la construcción de un MLP para cada uno de los 7 fármacos. La red construida para cada fármaco tiene una estructura como la de la figura 2.4. Como se puede ver, la red tiene tres capas, una capa de entrada que tiene 99 neuronas de entrada una asociada a cada descriptor, una capa oculta con n neuronas y una capa de salida con 2 neuronas, una para cada clase (susceptible y resistente).

Como sabemos, uno de los problemas cuando usamos estas redes neuronales es definir el número de capas ocultas y la cantidad de neuronas que tendrán. Se hicieron pruebas con varias variantes, quedando como las de mejores resultados, en el caso de las redes con entrada como energías de contacto una capa oculta con 4 neuronas, y en el caso del cambio de energía(*ΔEnergía*) una capa oculta con 20 neuronas.

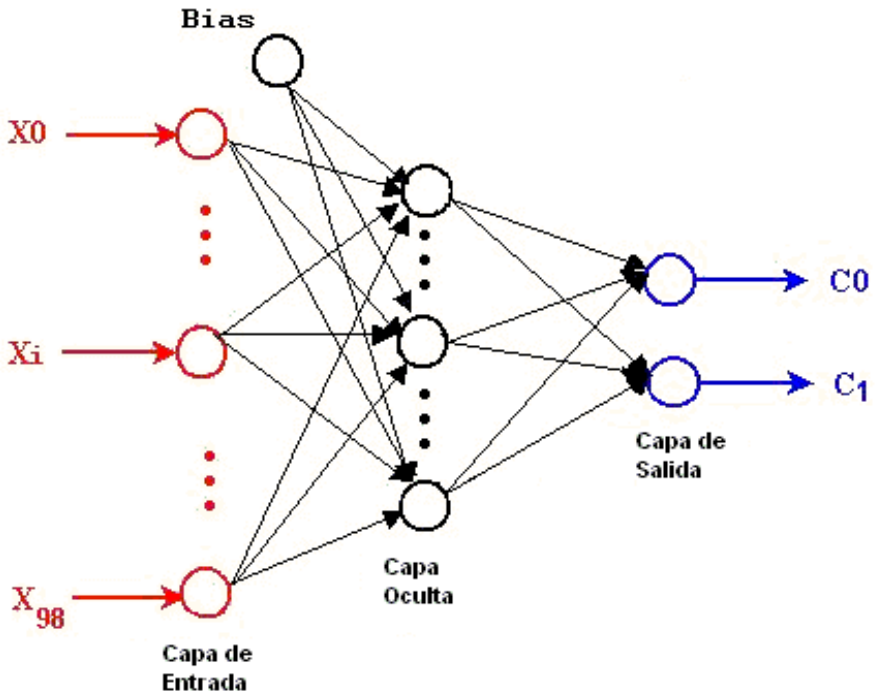


Figura 2.3 Topología de las redes MLP usadas para clasificar la resistencia a cada fármaco

Se usó como parámetro α, el calculado automáticamente por el software, descrito en la ecuación 2.1, que tiene en cuenta el tamaño de la muestra. Los valores se muestran en la tabla 2.2. Véase que el parámetro tiene igual valor independientemente de la representación, pues éste, como se muestra en la ecuación 2.1, depende sólo del la cantidad de ejemplos de la muestra, por clases.

Y se usó como parámetro **β,** 0.9 para todas las redes, tanto las construidas para la representación con *Energía*, como para las construidas con la representación con *ΔEnergía*.

Fármaco	Valor del α para redes con Energía y Δ*Energía*
SQV	0.0037878
LPV	0.0085714
RTV	0.0047468
APV	0.0040214
INV	0.0039787
ATV	0.0123966
NFV	0.0037220

Tabla 2.2 Valores del alfa calculado por SmartMLP para cada red neuronal.

2.4 Redes Recurrentes Bidireccionales Dinámicas como método de clasificación de resistencia.

La creación de métodos de aprendizaje computacionales ha estado basada en lo que el hombre conoce de su propio aprendizaje. Sería bueno poder diseñar un problema y que la propia naturaleza del problema nos llevara al diseño de un método como las redes neuronales.

Si pensamos un poco en el problema que estamos analizando aquí, como se describió en el capítulo anterior estamos en presencia de un problema de bioinformática, donde pretendemos a partir de la información de la secuencia primaria de la Proteasa del VIH predecir su resistencia ante un inhibidor. En la práctica el hombre la vía más segura sería ver la estructura tridimensional de dicha enzima y la estructura del fármaco y probar su acople.

Partiendo de esto, podemos tratar primero de representar la secuencia de forma que brinde información asociada con su estructura 3D. La vía que escogimos para esto es precisamente la representación basada en la *Energía* y Δ*Energía*.

Por otro lado, tenemos que la interacción de los aminoácidos por la atracción de la energía determina la posición final de los aminoácidos en el espacio, que es muy diferente a la estructura lineal de la secuencia primaria, teniendo en cuanta esto puede ser importante que la secuencia se analice como un todo o que, si se particiona, cada parte tenga en cuenta el resto de las partes de la secuencia.

Los problemas de aprendizaje de secuencias, todavía hoy en día se exploran, a pesar de las numerosas aplicaciones que se han desarrollado. En particular, la traslación secuencial es generalmente una tarea dura y los modelos actuales parecen tener varias limitaciones. Una de estas limitaciones es la suposición de que el sistema es causal [Bal00].

Un sistema dinámico se denomina causal si la salida en el tiempo t no depende de entradas futuras. Y se denomina no-causal si en la salida del tiempo t se tiene en cuenta la información de los tiempos pasados y futuros. Para ciertas categorías de secuencias finitas, la información del tiempo pasado y futuro es muy importante, por ejemplo para el ADN y las secuencias de proteínas, donde la secuencia puede depender fuertemente de eventos localizados arriba y debajo de la región que se analiza, algunas veces a distancias considerables, como por ejemplo en la predicción de estructuras, donde la secuencia primaria no indica como están dispuestas [Wan03].

Precisamente en nuestro problema se necesita tratar la secuencia de forma que ésta reciba información de todos los aminoácidos que tiene a la derecha y todos los que tiene a la izquierda. Por este motivo se decidió el uso de una red no-causal.

Después de este análisis se decidió que una estructura de red neuronal donde la secuencia se analizara en tres tiempos, de forma que el tiempo del medio tuviera en cuenta el primero y el tercero pudiera representar un poco mejor la naturaleza del problema.

Otro de los problemas que se tuvo en cuenta para la elección del método de aprendizaje es que se está trabajando con mutaciones de secuencias. Una secuencia puede mutar de tres formas: por el simple cambio de un aminoácido, por la inserción o por la eliminación de un aminoácido. En el caso de la primera, la longitud de la secuencia se mantiene, pero el caso de las dos últimas la longitud varía. Teniendo en cuenta esto para que nuestro sistema pueda predecir una secuencia mutada producida por inserciones y/o eliminaciones, necesitamos un método que me permita una entrada dinámica.

Los modelos conexionistas para el aprendizaje en los dominios secuenciales son, usualmente, sistemas dinámicos que necesitan estados ocultos para almacenar información contextual. Estos modelos pueden adaptarse a tiempos variables, en dependencia de las entradas del mismo, lo cual puede permitir el trabajo con secuencias de tamaños diferentes.

Existen varias topologías para redes recurrentes que han sido usadas en la literatura para resolver diferentes problemas. Dentro de los problemas de bioinformática en [Bal02] se describe una topología bidireccional dinámica que se usa para predicción de estructuras secundarias.

Debido a esto, se decidió el uso de redes bidireccionales dinámicas, como otro método de aprendizaje para resolver el problema.

Esta topología desarrollada por Baldi, consiste de dos bloques de contexto, uno que 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.

En la figura 2.4 se muestra un ejemplo de esta topología para nuestro problema. Téngase en cuenta, que en la figura, cada una de las flechas que van de capa a capa, significan que hay 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. Esto puede verse más claro cuando la red se despliega en el tiempo, como se muestra en la figura 2.5, que el tiempo t, en este caso el 2, depende del 1 y del 3 a la vez. En el caso del tiempo 1 que no tiene tiempo anterior por tanto las neuronas de la parte derecha se inicializan en cero y ocurre igual en el caso del tiempo 3, pero para las neuronas de la capa de la izquierda, pues no tiene tiempo futuro.

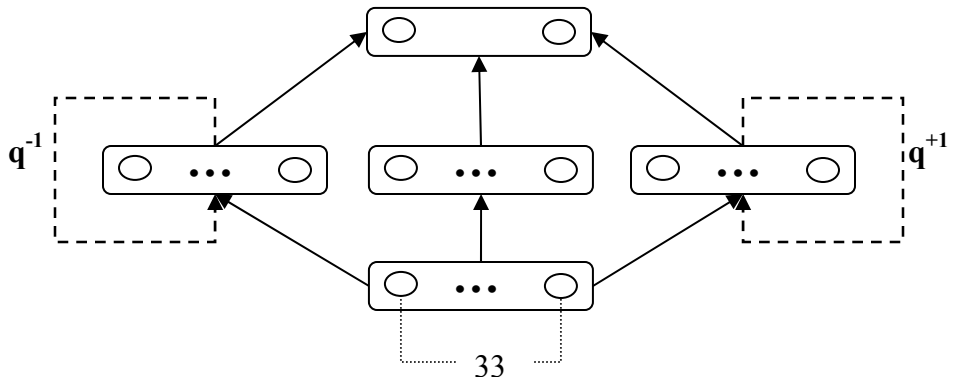


Figura 2.4 Topología de una Red Neuronal Bidireccional.

En la figura se muestra que la red tiene 33 neuronas de entrada y 2 neuronas de salida y las de las capas ocultas descritas en la tabla 2.2, según el fármaco que se esté tratando. En este caso las 33 entradas son valores reales y las salidas son (0,1) ó (1,0), significando (0,1) que es resistente y (1,0) que es susceptible.

Este tipo de problema no es el típico para resolver en este tipo de red pues no tenemos una salida asociada a cada tiempo. Pero sí podemos decir que cada tiempo de la secuencia está asociado a la misma salida. Específicamente la secuencia se dividió en tres tiempos, o sea 33 entradas en cada tiempo. A cada tiempo se le asignó la resistencia asociada a la secuencia.

Ahora el problema está en cuál es la salida final de la red cuando termina un caso. Pues, como se ve en la figura, la red tiene tres valores de salida, o sea la salida es un vector con 3 componentes. En nuestra base de casos estas salidas siempre van a ser la misma para un mismo caso, pero en la predicción esto no tiene que ser así. La salida puede ser cualquier combinación de resistente y susceptible en las tres salidas. Pueden existir varias variantes para llegar a una única salida con estas tres. En este trabajo vamos a tener en cuenta dos. Una primera variante de salida es la moda de las tres y una segunda variante es la salida correspondiente al tiempo del medio. En el caso de la primera variante estamos ofertando el valor que fue más frecuente en los diferentes tiempos, lo cual le da igual peso a todas las partes de la secuencia. En el segundo caso se tiene en cuenta sólo la salida del medio,

pero es válido recordar que esta salida va a estar influenciada por los otros dos tiempos, por tanto de los tres es el que más información tiene de la secuencia completa. En la figura 2.5 se puede ver la topología de la red 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.

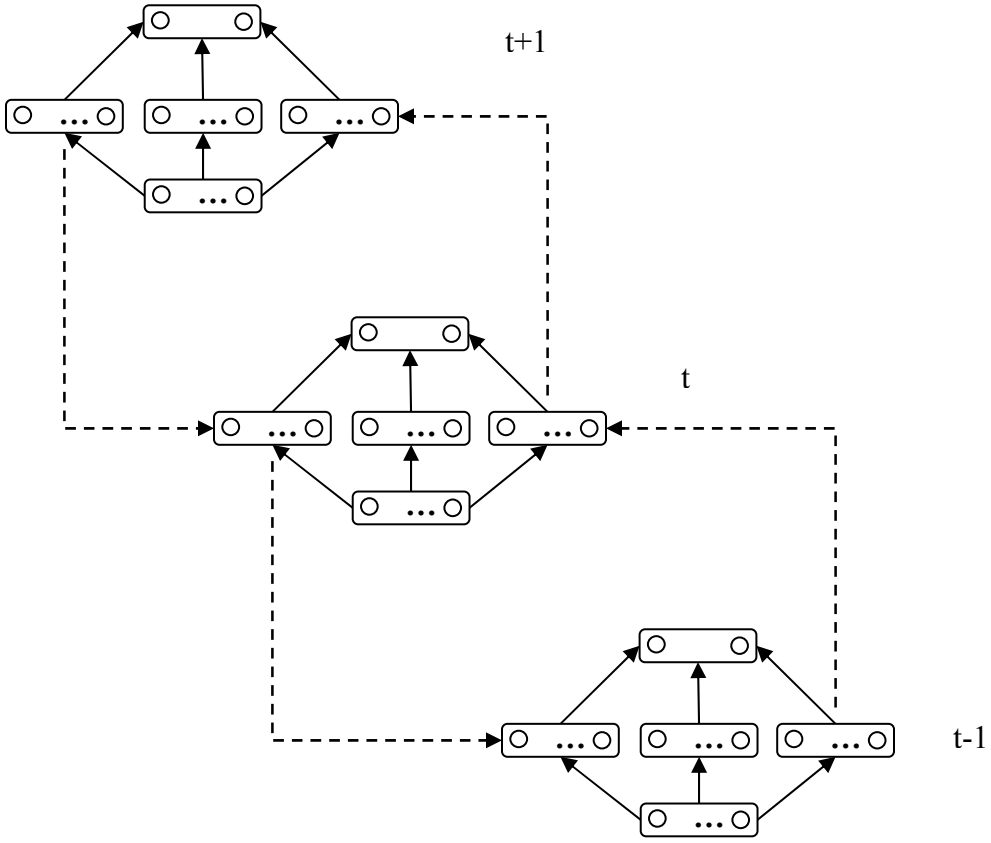


Figura 2.5 Despliegue de la red de la fig 2.4 en sus tres tiempos.

Finalmente se usó la topología descrita de red recurrente bidireccional como muestran las figuras 2.4 y 2.5. Como algoritmo de entrenamiento se usó el Backpropagation Trought Time descrito en el epígrafe 1.2.5.

La topología de este tipo de red es tan o más difícil de ajustar que la de las redes MLP, pues como se ve tenemos 3 capas de neuronas ocultas, para las que hay que definir la

cantidad de neuronas asociadas a cada una. En este problema se decidió que la influencia del tiempo anterior y posterior fuera similar, para esto se estableció que las capas de contextos asociadas tuvieran la misma cantidad de neuronas. De esta manera se hicieron múltiples pruebas hasta encontrar la red que mejor se ajustaba a cada base. Las topologías usadas finalmente, son descritas en la tabla 2.2.

Fármaco	**HB=HF**	**HO**
SQV	11	11
LPV	11	11
RTV	20	20
APV	20	20
IDV	27	20
ATV	32	32
NFV	20	20

Tabla 2.2. Número de neuronas asociadas a las capas de contexto backward(HB), forward(HF) y a la capa oculta del centro de la red(HO) para cada red neuronal.

Como función objetivo se usó Cross-Entropy y como función de activación de salida se usó Softmax.

Los parámetros α y β se probaron con varios valores, obteniendo los mejores resultados con:

- β =0.0001
- α = 0.001

Los valores de los pesos de la red se inicializan aleatoreamente con valores entre -0.5 y 0.5.

Como se puede ver hasta el momento, se necesita una herramienta que me permita crear este tipo de redes tan compleja, ya que no existe ningún software disponible con estas facilidades. Por este motivo surgió la necesidad de implementación de la herramienta NEngine que será descrita a continuación.

2.5 Diseño e implementación del NEngine.

Existen varias topologías definidas para redes recurrentes en la literatura. La gran variedad de redes de este tipo que existen, hace que las topologías sean diversas así como sus algoritmos de aprendizaje. Para poder hacer una plataforma que permita construir redes recurrentes se necesita que ésta tenga una gran flexibilidad.

Existen varios software en la literatura para el tratamiento con redes neuronales. En realidad, hasta ahora han sido creados varios simuladores (SNNS [Zel95], NeuroSolution [Koc98], etc.) y lenguajes de especificación (Aspirin [Lei92], CONNECT[6], etc.) con este propósito; pero todos presentan limitaciones de flexibilidad y alcance que dificultan el desarrollo sistemático de soluciones a problemas reales. Todos estos softwares tienen redes fijas, los más comunes usan redes recurrentes de Elman y otras, pero no permiten usar topologías de nueva creación, como la que pretendemos construir para nuestro trabajo.

En efecto, se hace evidente la necesidad de disponer de un ambiente de desarrollo que brinde la posibilidad de especificación y ejecución de los más diversos modelos de redes neuronales existentes, así como la posibilidad de extensión, combinación y creación de nuevos modelos (sin tener que modificar la herramienta misma) si es requerido, todo ello de manera programática.

El N-Engine v1.0 [Sal05] es un programa para Sistemas Operativos Windows 9x/2000/2003 construido bajo un diseño de máquina virtual, concebido en su primera versión para el diseño, entrenamiento y validación de modelos de redes feed-fordward multicapas (redes Multi Layer Perceptron) y Recurrente basado en Backpropagation Trought Time [Pea90]. Opera en un entorno aislado como un nodo individual, aunque presenta una arquitectura de software para su fácil extensión a entornos distribuidos y a otros modelos de redes diferentes. La plataforma N-Engine implementa el algoritmo de aprendizaje Backpropagation y Backpropagation Trough Time basado en gradiente descendiente [Ham02]. Básicamente está constituido de tres módulos fundamentales: el

[6] Neural Dimension Inc. http://www.nd.com/

núcleo N-Kernel v1.0, la Interfaz gráfica N-GraphConsole v1.0 y el intérprete de lote de comandos N-CBC v1.0 (Neuronal Command Batch Compiler).

N-Kernel v1.0 posee una arquitectura basada en subprocesos dedicados que corren concurrentes dentro de la aplicación, cada uno destinado a tareas específicas; con una comunicación interna basada en intercambio de mensajes. Un planificador incorporado basado en un modelo de operadores de exclusión controla la actividad entre los subprocesos, de manera que cada uno no interfiera la actividad de los demás. Exporta un conjunto de rutinas, denominadas genéricamente API, que abstrae la funcionalidad del núcleo hacia el exterior.

La interfaz gráfica N-GraphConsole v1.0 permite a los usuarios la operación del software de una forma cómoda y fácil de aprender. Presenta visores que asisten gráficamente el proceso de diseño, el control del entrenamiento y un editor de texto para el trabajo con los comandos y las bases de casos.

N-CBC v1.0 constituye un intérprete de comando incorporado al sistema, capaz de analizar una secuencia de sentencias en el lenguaje NCBL (Neuronal Command Batch Language) y traducirlas a las rutinas correspondiente de la API. Es a través de este intérprete que el usuario se comunica con el sistema y controla su funcionalidad.

El NCBL es un lenguaje sencillo creado para la comunicación del usuario con el núcleo del sistema. Inicialmente presenta forma de lotes de instrucciones atómicas que pueden ir incrementando su repertorio a medida que el sistema aumente su complejidad. Está constituido por tres bloques fundamentales de instrucciones: sentencias de entrada/salida de información, sentencias para el ajuste de parámetros, sentencias para el diseño de topologías y sentencias para la ejecución de los algoritmos de entrenamiento[7].

Típicamente el trabajo con N-Engine involucra la creación de una topología de red específica para cada problema.

// crear topologia para red MLP para el problema XOR

[7] El NCBL comentado en el presente trabajo constituye una versión muy reducida para la operación con el N-Engine v1.0 y es un subconjunto de la versión original de la cual se prevé presente una estructura más sofisticada e incorpore instrucciones para el control de ejecución, manejo de expresiones y la descripción, mediante refinamiento, de nuevos modelo de redes: capacidades previstas en el diseño de la Máquina Virtual original.

```
Create Layer LayIn as Length 2, double, input
Create Layer LayMd as Length 2, double, middle
Create Layer LayOut as Length 1, double, output
Create Connection Con1 as delay 0, source LayIn, target LayMd
Create Connection Con2 as delay 0, source LayMd, target LayOut
// crear una neurona
Create Neuron Ner1 as binary
```

Las cinco primeras sentencias crean la topología para una red que soluciona el problema del XOR[8] (una capa de entrada LayIn de dos neuronas que manejan información de tipo real, una capa intermedia LayMd con las mismas características que la anterior y una capa de salida con una neurona tipo real). Las tres primeras crean objetos tipo *layer* y la dos siguientes de tipo *connection*. Los objetos tipo *layer* brindan una abstracción de diseño para el manejo de capas enteras de neuronas, cada una recibe un nombre distinto por el cual puede ser referenciada en próximas sentencias. Los objetos *layer* poseen una longitud o cantidad de neuronas determinada y un tipo de valor de información que procesa (real en nuestro ejemplo), además de un modificador que caracteriza la misma en capa de entrada, intermedia o de salida. Los objetos de tipo *connection* poseen un nombre de objeto origen y destino que puede ser un objeto tipo *layer* o *neuron*, adicionalmente son modificados por un operador *delay*, característico de los modelos de red recurrentes. La próxima sentencia muestra la manera de crear una neurona de nombre Ner1 usando el lenguaje NCBL. La creación de un objeto tipo *connection* que involucre dos *layers* significará la creación implícita de n*m conexiones, donde n y m significan las cantidades de neuronas de cada una de las capas. Un objeto tipo *neuron* es tratado como un *layer* de una sola neurona.

El software también permite eliminar objetos creados, salvar y cargar un proyecto. Se puede asignar el valor de parámetros de entrenamiento.

Finalmente se creó una herramienta que brinda las siguientes facilidades:

- Incorpora Backpropagation Through Time sobre un algoritmo backpropagation (BP) estándar.

[8] El problema XOR es un problema típico en la literatura que describe las redes MLP, consiste en la simulación de la función lógica xor, mediante el uso de una red de una sola capa intermedia.

- Posee heurísticas para la aceleración de la convergencia:
    - Normalización de las entradas
    - Presentación de los casos siguiendo una distribución uniforme.
    - Selección del parámetro alfa según la base de casos.
- Dispone de varios modelos de neuronas para la capa de salida y de funciones objetivo que pueden ser combinadas según la naturaleza del problema.

  Se le implementaron como funciones objetivo:
    - Cross entropy
    - Suma cuadrática de errores

  Y como funciones de salida:
    - Softmax
    - Sigmoidal
    - Lineal

  Permite las combinaciones:
    - Para problemas de clasificación: cross-entropy con softmax (binomial, multinomial); cross-entropy con sigmoidal.
    - Para regresión: suma cuadrática de errores con lineal.
- Interacción del usuario a través de un lenguaje de instrucciones.
- Posibilidades de diseño de cualquier topología con el uso de instrucciones para crear capas de neuronas y conexiones.
- Permite la edición de la base de casos.
- Existe un grupo de instrucciones para configurar el entrenamiento.
- Visualización del error de la red y los porcientos de clasificación durante el proceso de entrenamiento.

- Finalizado el entrenamiento se reporta la matriz de confusión, error y porciento de clasificación correcta por clase.
- Permite realizar validación cruzada (*cross-validation Random Sampling*) para estimar los parámetros de eficiencia de la red.
- En el caso de *cross-validation*, visualiza el porciento de clasificación tanto de la base de entrenamiento como de la de prueba, lo que permite que el usuario pueda darse cuenta de un posible sobreajuste de la red.
- La red entrenada se almacena en disco, y puede ser manejada desde aplicaciones externas empleando rutinas contenidas en una dll.

La arquitectura, como ya se ha dicho, se basa en una estructura de clases para la gestión de procesos concurrentes, con facilidades para su extensión a una plataforma unificadora de redes neuronales.

En la fig 2.5 se puede ver una muestra de la ventana del software.

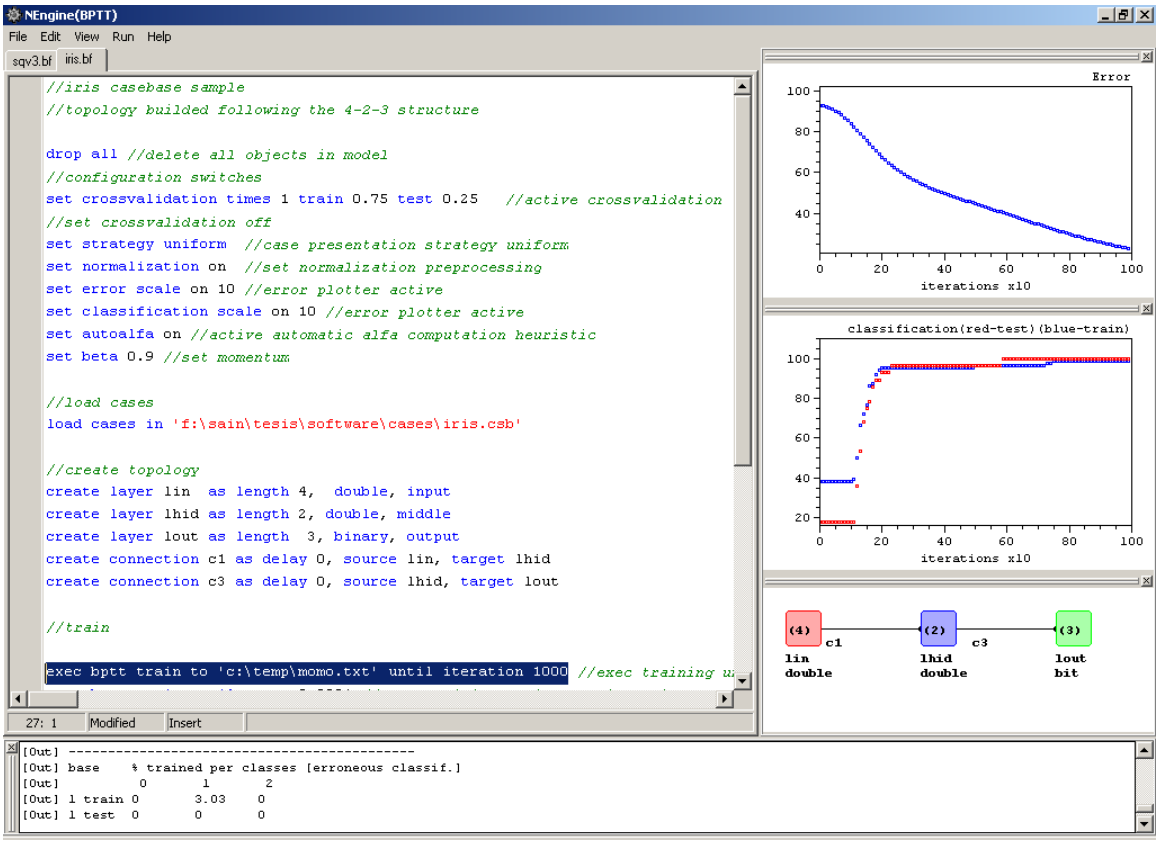


Figura 2.5 Ventanas de NEngine.

Con este software se crearon entrenaron y probaron las 7 redes neuronales bidireccionales de este trabajo.

2.6 Conclusiones Parciales

En este capítulo nos hemos centrado en el diseño de un sistema de aprendizaje, que quedó definido de la siguiente manera:

- Tarea: Clasificación de secuencias de la Proteasa en resistentes o susceptible ante el fármaco analizado.
- Medida de representación: porciento de secuencias correctamente clasificadas.
- Experiencia para el entrenamiento: Casos de la base de datos de Stanford para los 7 fármacos contra la Proteasa del VIH
- Función objetivo: F: C → O,

  donde, $C \in \mathbb{R}^{99}$, en este caso particular en nuestra base todas las secuencias tienen igual tamaño, 99 aminoácidos. Cada elemento de C es un aminoácido representado por su *Energía* o Δ*Energía* que es un valor real, o sea, C va a representar la secuencia de la Proteasa. O = {resistente, susceptible}.
- Mecanismo de aprendizaje: SVM, redes neuronales artificiales de tipo MLP y redes neuronales artificiales recurrentes bidireccionales dinámicas.

En este capítulo de hace una descripción de cómo son usados cada uno de estos mecanismo de aprendizaje en el problema enunciado. Se hace énfasis en el último como método que tiene asociación natural con el problema, por su propiedad de analizar un tiempo teniendo en cuenta tiempos anteriores y tiempos posteriores, y con el que se obtienen mejores resultados.

En este capítulo también se describió el software NEngine desarrollado con el objetivo de construir, entrenar y explotar redes neuronales recurrentes. Este software se usó en nuestro trabajo para poder construir las redes bidireccionales asociada a cada una de las 7 bases de casos.

El software tiene las siguientes facilidades:

- lenguaje de especificación
- construcción de distintas topologías
- selección de distintas funciones de error y de activación en la capa de salida

- uso de *crossvalidation* para estimar la eficiencia de la red
- entrenamiento con el uso de los métodos: BP y BPTT.

En el próximo capítulo se hará la validación de los resultados usando SVM, MLP y redes bidireccionales. Se hace la comparación de estos resultados con los reportados hasta ahora en la literatura.

Capítulo III: Validación.

3.1 Croos-validation

Como se explicó en el capítulo anterior, se utilizaron 3 métodos de aprendizaje para predecir la resistencia de secuencias del VIH ante 7 fármacos. Estos métodos fueron: SVM, MLP y red recurrente bidireccional.

Estas técnicas usadas tienen parámetros libres y esto conlleva a dos problemas: selección del modelo, que ha sido analizado y explicado en el capítulo anterior y la validación, que el tema, precisamente de este capítulo. ¿Una vez que seleccionamos el modelo, cómo estimamos el error? ¿Cómo probamos el nivel de generalización?

Cuando contamos con un número ilimitado de ejemplos en nuestra base de casos, estas preguntas pueden ser respondidas rápidamente, pues se selecciona el modelo de más bajo error en la base completa. Pero en las aplicaciones reales, generalmente sólo poseemos un número finito de ejemplos, que generalmente es más pequeño de lo esperado o necesario, porque formar conjuntos de datos es un proceso caro.

Aunque muchas veces estamos tentados a usar el conjunto de datos de entrenamiento para estimar el clasificador óptimo, esto nos puede traer dos problemas fundamentales:

- No se está teniendo en cuenta el *overfitting*, y este problema está presente, comúnmente, en modelos con un largo número de parámetros como lo son las redes neuronales.
- Se puede llegar a conclusiones apresuradas pues no se está teniendo en cuenta como reacciona el modelo ante casos que no ha visto antes, que es lo que le daría la medida de generalización.

Existen varios métodos para probar la generalización de un modelo, el más viejo y quizás el primero es el método "*holdout*". Este método consiste en dividir la base de casos en dos conjuntos, uno de entrenamiento y otro de prueba. Se usa el conjunto de entrenamiento para que el modelo aprenda y el de prueba para saber en que medida es capaz de generalizar. Pero este método puede no ser útil si la cantidad de casos es extremadamente pequeña y, por otro lado, puede dar una solución "buena" a partir de haber hecho una división de la base muy oportuna.

Hoy en día los métodos más usados para validar es la validación cruzada (*cross-validation*) [Gou97] [Sha93] [Sto77]. Existen varios métodos de *cross-validation*, entre

los más usados se encuentran: submuestreo aleatorio *(Random Sampling)*, *cross-validation k-fold* y *leave-v-out*.

Random sampling se basa en hacer *k* divisiones de la de la base de datos, seleccionando un número fijo de ejemplos. Para cada división de la base, se entrena el clasificador con los ejemplos de entrenamiento y se mide el error con los ejemplos de prueba, o sea, pudiera decirse que se hace *k* veces un *holdout*. La estimación del error estaría dada por el promedio de los errores de cada *holdout*, como se muestra en la ecuación 3.1:

$$E = \frac{1}{k}\sum_{i=1}^{K} E_i \tag{3.1}$$

donde E_i es el error de cada división.

Cross-validation k-fold se basa en dividir la base en *k* partes iguales. Se realizan *k* entrenamientos del modelo, al igual que en el método anterior, tomando siempre una parte para prueba y las restantes para entrenamiento. La diferencia con el método anterior es que la base se divide según estén organizados los casos. Esto puede ser un problema, si por ejemplo, en la base tenemos los casos organizados por clases la selección puede ser hecha de tal manera que en la base de prueba sólo tengamos una clase y la de entrenamiento pocos ejemplos de esa clase. Por otra parte la ventaja de este método es que todos los ejemplos de la base de casos son, eventualmente usados para ambos procesos, entrenamiento y prueba.

Leave-v-out se dice que es una versión más elaborada y a la vez más costosa que el *cross-validation k-fold*, donde se construyen todos los posibles subconjuntos de la base de tamaño *v*.

Todos estos métodos de *cross-validation* pueden servir tanto para la estimación del error de un modelo, como para la selección del mejor modelo, dentro de varios probados, para la solución de un problema determinado.

En nuestro trabajo, en todos los casos se usó *Random sampling,* con un k=10, o sea 10 divisiones de la base de casos, con sus respectivos entrenamientos y pruebas, dando como resultado el porciento de predicción (medida contraria al error), y dado por la ecuación 3.2.

$$C = \frac{1}{k}\sum_{i=1}^{k} C_i \tag{3.2}$$

En los epígrafes siguientes se describen los resultados obtenidos en cada uno de los métodos de aprendizaje utilizados en las dos variantes de representación de nuestro problema.

3.1.1 Cross-validación como validación de los resultados obtenidos con los métodos usados para predecir resistencia del virus VIH ante fármacos de la proteasa

En las últimas 2 columnas de la siguiente tabla podemos ver los resultados obtenidos con el software SmartMLP, en el uso de un MLP para predecir la resistencia de las mutaciones del VIH ante alguno de los fármacos. Las secuencias son representadas aquí por las energías de contacto (penúltima columna) y las Δ-Energías (última columna). Los resultados que se muestran están basados en *Random sampling*, con un k=10, como se dijo anteriormente, y son la medida de la predicción en las bases de prueba.

Fármaco	KNN	New Dtree	Dtree (James)	DTree (Beerenwinkel)	SVM	SmartMLP Energías	SmartMLP ΔEnergías
SQV	81.7	85.7	80	87.5	**88.1**	85.47	87.88
LPV	81.1	89.5				**92.33**	87.88
RTV	82	89.5	89	89.8	**92.7**	90.92	90.71
APV	80.9	75.8	75.8	87.4	**89.8**	82.17	80.65
IDV	80.6	85.5	85	89.1	**92**	86.96	92.55
ATV						**80.00**	74.16
NFV	73.6	**93.7**	91.8	88.5	90	86.63	87.13

Tabla 3.1 Porcientos de predicción de métodos usados y de las redes MLP creadas usando energías como representación de las secuencias.

Las tres primeras columnas corresponden a los resultados reportados por Roberto James en [Jam04] usando KNN, una variante de un árbol de decisión usada por él y el árbol de decisión clásico usando ID3. Las columnas sucesivas corresponden a los resultados reportados por Niko Beerenwinkel en [Bee02] usando un árbol de decisión clásico y el SVM creado usando el software de [Cha01], LIBSVM. Como se puede ver a simple vista los resultados del *SVM de Beerenwinkel* son superiores a todos los demás resultados pero también es cierto que Beerenwinkel (y también James) uilizaron más casos que nosotros (ver tabla 3.2).

Por otro lado tenemos que los resultados del SmartMLP con *ΔEnergías*, son bastante similares a los de *Energías*.

Fármaco	N_James	N_Beerenwinkel	N
SQV	830	662	386
LPV	381	315	175
RTV	773	662	306
APV	701	313	373
IDV	822	662	377
ATV		313	121
NFV	855	661	403

Tabla 3.2 Cantidad de casos usados por James(N_James), Beerenwinkel(N_Beerenwinkel) y en este trabajo(N).

La otra técnica que se usó fue el mismo SVM que usó Beerenwinkel en su trabajo, usando tanto las *Energías* como las *ΔEnergías*. En la tabla 3.3 se muestran los resultados obtenidos con la variante de *Energías* para SVM lineal, usando kernel polinomial de grado 1(otra variante de lineal), grado 2, grado 3, y radial basis.

Fármaco	SVM lineal	SVM Grado 1	SVM Grado2	SVM Grado3	SVM Radial Basis
SQV	88.05	82.33	87.27	84.67	84.67
LPV	88.57	85.14	88.00	87.42	85.14
RTV	92.15	84.96	91.83	91.17	86.92
APV	82.30	77.47	83.91	82.57	78.82
IDV	91.53	86.73	91.51	89.65	88.59
ATV	74.38	71.07	75.20	68.59	70.24
NFV	84.86	75.68	84.86	86.60	79.90

Tabla 3.3 Resultados obtenidos usando diferentes variantes de SVM con representación de la secuencia usando *Energías*.

A simple vista también podemos ver que no hay muchas diferencias en los resultados obtenidos con los diferentes *kernels*, y además para los diferentes fármacos existe un método distinto que se convierte en superior.

Para la representación usando *ΔEnergías* también se usaron estas variantes de SVM. En la tabla 3.4 se pueden ver que los resultados obtenidos son similares, lo cual demuestra que tanto la representación usando *Energías* como la representación usando *ΔEnergías* son buenas para la secuencia de entrada en estos casos.

Fármaco	SVM lineal	SVM Grado 1	SVM Grado2	SVM Grado3	SVM Radial Basis
SQV	87.82	80.82	69.68	85.23	82.38
LPV	88.57	85.14	85.14	88.57	85.14
RTV	91.83	84.96	79.08	92.15	86.92
APV	82.30	77.47	77.47	83.64	78.82
IDV	91.51	86.73	83.81	92.57	88.59
ATV	74.38	71.07	68.59	72.72	70.24
NFV	84.86	75.68	70.96	84.86	80.14

Tabla 3.4 Resultados obtenidos usando diferentes variantes de SVM con representación de la secuencia usando *ΔEnergías*.

Como se explicó en el capítulo anterior, se usaron redes bidireccionales, también, para resolver este problema. Los resultados se muestran en la tabla 3.5. Para este método se tomó como representación de la secuencia las *ΔEnergías.* Cómo puede verse a simple vista, los resultados son superiores al SVM usado por Beerenwinkel. Por ello pudiéramos decir que la red bidireccional efectivamente es un buen método para resolver este problema de predicción de resistencia. Tanto en la variante de selección de la moda de las tres salidas, como en la de selección de la salida del tiempo del medio, se obtienen mejores resultados en cuanto a la predicción usando *cross-validation random sampling* con k=10.

Fármaco	SVM	BRR (moda)	BRNN (valor del medio)
SQV	88.14	**91.16**	**91.16**
LPV		**94.42**	94.39
RTV	92.70	93.42	**94.73**
APV	**89.80**	89.25	88.71
IDV	92.04	**92.55**	**92.55**
ATV		**82.67**	81.33
NFV	90.00	**94.06**	93.07

Tabla 3.5 Resultados obtenidos usando redes bidireccionales

Como podemos ver, en un solo fármaco (APV) los resultados del uso de redes bidireccionales no son superiores a los reportados por Beerenwinkel con el SVM, pero al menos son similares a este. En el resto de los casos es superior a este método y superior a todos los descritos anteriormente.

A pesar de que *cross-validation* se considera como un método de validación para seleccionar el mejor método de aprendizaje, se hicieron pruebas estadísticas que pudieran reforzar las conclusiones de los resultados obtenidos. En los siguientes epígrafes se detallarán los resultados de las pruebas.

Debemos tener en cuenta que contamos con pocas muestras, 7 bases de casos en el mejor de los casos, y esto hace más difícil obtener significación en la comparación de poblaciones.

3.1.2 Especificidad y Sensitividad

La especificidad y la sensitividad son otras medidas para validar los métodos utilizados. Este tipo de medidas es muy usado en problemas de la bioinformática, y en problemas relacionados con la medicina.

Supongamos que la tabla de resultados para dos clases como las que tenemos en nuestro problema (resistente y susceptible) se resume esquemáticamente como en la Tabla 3.6. Se calculan los porcientos de predicción para cada una de las clases, en función de los valores posibles de la misma

	Casos Clasificados como susceptible	**Casos Clasificados como resistente**
susceptible	a	b
resistente	c	d

Tabla 3.6 Tabla de resultados de la base de prueba

A partir de estos resultados la sensitividad será el porciento de ejemplos de virus resistente que fueron clasificados correctamente como resistentes. La especificidad, es entonces, el porciento de ejemplos que fueron clasificados correctamente como susceptibles, o sea, se pueden calcular según las ecuaciones siguientes:

$$sensitividad = \frac{d}{c+d} \tag{3.3}$$

$$especificidad = \frac{a}{a+b} \tag{3.4}$$

El cálculo de la susceptibilidad y sensitividad en todos los métodos empleados se pueden observar en la tabla 3.7 y 3.8.

Fármacos	SmartMLP Energías		SmartMLP ΔEnergías	
	sensitivity	specificity	sensitivity	specificity
SQV	84.82	87.58	84.70	90.57
LPV	95.73	76.29	84.70	90.57
RTV	85.00	94.28	88.41	94.52
APV	74.34	87.37	80.95	80.39
IDV	87.39	89.98	95.72	89.36
ATV	85.42	75.39	78.53	73.33
NFV	86.39	87.25	89.37	79.31

Tabla 3.7 Sensitividad y Especificidad del SmartMLP usando *Energías* y *ΔEnergías*

Fármaco	BRR(moda)		BRNN(medio)	
	sensitivity	specificity	sensitivity	specificity
SQV	90.48	92.59	90.48	92.59
LPV	74.12	100	96.25	77.78
RTV	100	88.1	89.74	100
APV	84.71	93.07	84.81	87.85
IDV	89.13	95.83	95.35	90.2
ATV	94.29	77.5	87.5	80
NFV	95.83	89.66	94.44	89.66

Tabla 3.8 Sensitividad y Especificidad de las dos variantes de redes bidireccionales.

Podemos ver que los resultados son buenos también, en el epígrafe 3.3 se hará el análisis estadístico de los mismos.

3.2 Análisis estadístico de los resultados de SmartMLP con energías de contacto

3.2.1 Análisis estadístico de SmartMLP vs KNN, New DTree y DTree

Se usó una Prueba de Friedman para comparar los resultados entre los métodos usados por James y los del SmartMLP. En este caso contamos con 5 casos en que se reportan todos los datos como se puede ver en la tabla 3.1. Los resultados de esta prueba pueden verse en la Figura 3.1.

El análisis de varianza de Friedmann demuestra que hay diferencias en general entre los tres métodos seguidos por James y los resultados del *SmartMLP*. Por los rangos promedios podemos ver que los de mayor rango son el *SmartMLP* (*Smart MLP Energías*) y la variante de árbol de decisión creada por James (*New DTree James*). Por esto se procedió a hacer una prueba de Wilcoxon entre estos dos métodos. Los resultados se muestran en la Figura 3.2.

Rangos

	Rango promedio
% KNN James	1.60
% New DTree James	3.10
% Clasical DTree James	1.90
% Smart MLP Energías	3.40

Estadísticos de contraste [a]

N			5
Chi-cuadrado			7.163
gl			3
Sig. Monte Carlo	Sig.		**.056**
	Intervalo de confianza de 99%	Límite inferior	.050
		Límite superior	.061

a. Prueba de Friedman

Figura 3.1 Prueba de Friedman comparando los resultados de James con los del *SmartMLP*.

El análisis de Wilcoxon demuestra que no tenemos razones suficientes para pensar que *SmartMLP* sea significativamente mejor que los resultados de *New DTree*, pero sí podemos decir que los resultados obtenidos usando el *SmartMLP* con Energías como representación de la secuencia son comparables con los resultados obtenidos por James usando su árbol de decisión usando información mutua como representación de la secuencia.

Rangos

		N	Rango promedio	Suma de rangos
% Smart MLP Energías - % New DTree James	Rangos negativos	2[a]	3.50	7.00
	Rangos positivos	4[b]	3.50	14.00
	Empates	0[c]		
	Total	6		

a. % Smart MLP Energías < % New DTree James

b. % Smart MLP Energías > % New DTree James

c. % New DTree James = % Smart MLP Energías

Estadísticos de contraste[b,c]

			% Smart MLP Energías - % New DTree James
Z			-.734[a]
Sig. Monte Carlo (bilateral)	Sig.		**.563**
	Intervalo de confianza de 99%	Límite inferior	.540
		Límite superior	.586

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 299883525.

Figura 3.2 Prueba de los rangos con signo de Wilcoxon comparando resultados del *SmartMLP* con los resultados del *NewDTree* de James.

3.2.2 Análisis estadístico de SmartMLP vs. DTree y SVM

También se hizo el análisis de los resultados del SmartMLP y los resultados de Beerenwinkel, de la misma manera que en el anterior se realizó una prueba de Friedman para comparar los tres métodos que demostró que existían diferencias significativas (Significación =0.008). De los tres algoritmos el de mayor rango promedio es el SVM usado pro Beerenwinkel, por esto se realizó una prueba de Wilcoxon entre este método y el SmartMLP que nos demostró que hay diferencias medianamente significativas, por lo que podemos concluir que el primero es mejor.

Rangos

	Rango promedio
% Clasical DTree Kiko	1.80
% SVM Kiko	3.00
% Smart MLP Energías	1.20

Estadísticos de contraste [a]

N			5
Chi-cuadrado			8.400
gl			2
Sig. Monte Carlo	Sig.		**.008**
	Intervalo de confianza de 99%	Límite inferior	.006
		Límite superior	.010

a. Prueba de Friedman

Figura 3.3 Prueba de Friedman comparando resultados del *SmartMLP* con los resultados del *SVM* de Beerenwinkel.

Rangos

		N	Rango promedio	Suma de rangos
% Smart MLP Energías - % SVM Niko	Rangos negativos	5[a]	3.00	15.00
	Rangos positivos	0[b]	.00	.00
	Empates	0[c]		
	Total	5		

a. % Smart MLP Energías < % SVM Kiko

b. % Smart MLP Energías > % SVM Kiko

c. % SVM Kiko = % Smart MLP Energías

Estadísticos de contraste [b,c]

			% Smart MLP Energías - % SVM Niko
Z			-2.023[a]
Sig. Monte Carlo (bilateral)	Sig.		**.064**
	Intervalo de confianza de 99%	Límite inferior	.055
		Límite superior	.073

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 624387341.

Figura 3.4 Prueba de los rangos con signo de Wilcoxon comparando resultados del *SmartMLP* con los resultados del *SVM* de Beerenwinkel.

3.3 *Análisis estadístico de los resultados de SVM con energías de contacto*

El otro método usado con energías de contacto como representación, es el SVM en sus diferentes variantes, cuyos resultados se mostraron en la tabla 3.3.

Estos SVMs se compararon entre sí, mediante una prueba de Friedman arrojando diferencias significativas. Por los rangos promedios se puede ver que los mejores resultados se obtienen con el SVM lineal y el polinomial de grado 2, entre los que se comprobó con una prueba de Wilcoxon que no había razones suficientes para decir que eran diferentes, o sea que son comparables. El SVM lineal que es el de mayor rango promedio se comparó con el SVM usado por Beerenwinkel y la prueba de Wilcoxon nos demostró que había diferencias significativas, siendo superior este último.

Rangos

		N	Rango promedio	Suma de rangos
% SVM Lineal Energías - % SVM Niko	Rangos negativos	5[a]	3.00	15.00
	Rangos positivos	0[b]	.00	.00
	Empates	0[c]		
	Total	5		

a. % SVM Lineal Energías < % SVM Kiko

b. % SVM Lineal Energías > % SVM Kiko

c. % SVM Kiko = % SVM Lineal Energías

Estadísticos de contraste [b,c]

			% SVM Lineal Energías - % SVM Kiko
Z			-2.023[a]
Sig. Monte Carlo (bilateral)	Sig.		**.063**
	Intervalo de confianza de 99%	Límite inferior	.054
		Límite superior	.071

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 112562564.

Figura 3.5 Prueba de Wilcoxon comparando resultados del *SVM lineal* usando *Energías* con los resultados del *SVM* de Beerenwinkel.

3.3.1 SVM vs SmartMLP con energías de contacto

Se comparó el SVMLineal, que es el de mejores resultados entre las diferentes variantes usando energías contra los resultados del SmartMLP, como se puede ver en la Figura 3.6, demostrando que no hay razones para pensar que los resultados de los métodos sean diferentes, debemos decir que los métodos son comparables en sus resultados.

Rangos

		N	Rango promedio	Suma de rangos
% SVM Lineal Energías - % Smart MLP Energías	Rangos negativos	3[a]	5.00	15.00
	Rangos positivos	4[b]	3.25	13.00
	Empates	0[c]		
	Total	7		

a. % SVM Lineal Energías < % Smart MLP Energías

b. % SVM Lineal Energías > % Smart MLP Energías

c. % Smart MLP Energías = % SVM Lineal Energías

Estadísticos de contraste[b,c]

			% SVM Lineal Energías - % Smart MLP Energías
Z			-.169[a]
Sig. Monte Carlo (bilateral)	Sig.		**.940**
	Intervalo de confianza de 99%	Límite inferior	.914
		Límite superior	.965

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 221623949.

Figura 3.6 Prueba de Wilcoxon comparando resultados del *SVM lineal* usando *Energías* con los resultados del *SmartMLP* usando *Energías*.

3.4 Análisis estadístico de los resultados de SmartMLP y SVM con delta energías de contacto vs. los resultados obtenidos con energías de contacto

3.4.1 SmartMLP con delta energías vs. SmartMLP con energías

Para esta representación se realizaron las mismas comparaciones anteriores con los otros métodos, obteniendo resultados similares al SmartMLP con Energías. Se hizo una prueba

de Wilcoxon para comparar el SmartMLP en sus dos variantes y nos demostró que efectivamente no existen diferencias significativas.

Posteriormente se compararon los distintos SVM y se obtuvieron resultados similares, obteniendo como mejores resultados los del SVM lineal y de grado 3 esta vez.

Se compararon los resultados del SVM lineal y de grado 3 con los del SmartMLP, obteniéndose como resultado que los métodos esta vez también son comparables.

Por último se hizo una prueba de Wilcoxon para comparar los resultados del SmartMLP, en sus variantes de representación de las entradas, obteniéndose como resultado que son comparables.

Rangos

		N	Rango promedio	Suma de rangos
% Smart MLP d-Energías - % Smart MLP Energías	Rangos negativos	4[a]	4.00	16.00
	Rangos positivos	3[b]	4.00	12.00
	Empates	0[c]		
	Total	7		

a. % Smart MLP d-Energías < % Smart MLP Energías

b. % Smart MLP d-Energías > % Smart MLP Energías

c. % Smart MLP Energías = % Smart MLP d-Energías

Estadísticos de contraste[b,c]

			% Smart MLP d-Energías - % Smart MLP Energías
Z			-.338[a]
Sig. Monte Carlo (bilateral)	Sig.		**.807**
	Intervalo de confianza de 99%	Límite inferior	.782
		Límite superior	.832

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 1335104164.

Figura 3.7 Prueba de Wilcoxon comparando resultados de los resultados del *SmartMLP* usando *Energías* y *ΔEnergías*.

Por todo lo anterior, podemos concluir que tanto las Energías como ΔEnergías, son buenas representaciones para la secuencia en este problema, obteniéndose resultados similares a los vistos en la bibliografía hasta el momento, siendo superados sólo por los

resultados de Beerenwinkel con un SVM. Con respecto a esto último, tenemos la problemática de que el autor contaba con mayor cantidad de casos en las bases de datos, lo cual pudo influir en que obtuviera mejores resultados.

3.5 Análisis estadístico de los resultados de las Redes Bidireccionales vs. los resultados obtenidos con SmartMLP y SVM

Como ya se ha dicho anteriormente se usaron dos variantes para el análisis de la salida de las redes recurrentes bidireccionales construidas, la moda de las tres salidas y la elección del resultado de la salida del medio. Para el análisis de estas dos variantes se hizo una prueba de Wilcoxon donde se demostró que los dos métodos son comparables en cuanto a sus resultados.

Posteriormente se comparó con los mejores resultados de los SVMs y el SmartMLP usando ΔEnergías, ya que esta fue la representación seleccionada para este método. Todas estas pruebas arrojaron diferencias significativas entre los métodos, siendo mejor las variantes de red bidireccional en todos los casos.

Rangos

	Rango promedio
% BRR con la moda d-Energias	2.40
% BRR con valor del medio d-Energias	2.20
% SVM Niko	1.40

Estadísticos de contraste[a]

N			5
Chi-cuadrado			3.111
gl			2
Sig. Monte Carlo	Sig.		**.255**
	Intervalo de confianza de 99%	Límite inferior	.244
		Límite superior	.267

a. Prueba de Friedman

Figura 3.8 Prueba de Friedman comparando resultados de los resultados de las dos variantes de red bidireccional con el SVM de Beerenwinkel.

Por último se hizo una prueba de Wilcoxon con el SVM de Beerenwinkel, para el que no hay razones suficientes para decir que hay diferencias entre los mismo, o sea que los resultados con los dos métodos son comparables.

En los rangos promedios se puede ver que las redes bidireccionales son superiores al SVM. Además habíamos analizado antes que contamos con menos caso que este autor, por tanto un resultado como este, que los métodos son comparables, es muy bueno y demuestra que efectivamente las redes bidireccionales son un método de aprendizaje apropiado para ser usado en este problema de clasificación.

3.6 Análisis estadístico de la Especificidad y la Sensitividad

Como se vio en le epígrafe 3.1.2 esta es otra medida para validar la efectividad de un método de aprendizaje. En este epígrafe se hicieron comparaciones con las pruebas de Friedman y de Wilcoxon para comparar los resultados de estas medidas, tal y como se comparó en los epígrafes anteriores, la medida de predicción.

Los resultados de estas pruebas se pueden ver en el Anexo 4. Analizaremos, entonces, los métodos de redes bidireccionales que son los que mejores resultados nos han ofertado.

En cuanto a la sensitividad, se comparó con todos los métodos de James y Beerenwinkel, analizados anteriormente.

Como se pude ver en el Anexo 5, se puede decir que los resultados de la sensitividad son comparables con todos menos con el KNN de James, que los resultados de la red bidireccional son superiores.

En cuanto a la especificidad sólo es inferior al KNN que tiene mejores resultados.

De esta manera se ha demostrado que los resultados en la especificad y sensitividad también son buenos, para el método de redes bidireccionales.

3.7 Conclusiones Parciales

En este capítulo se ha hecho la validación de los resultados obtenidos con los tres tipos de métodos de clasificación usados, para predecir la resistencia del virus del VIH ante 7 inhibidores de la proteasa del mismo.

Se demostró que tanto, la energía de contacto de los aminoácidos, como los cambios de energía respecto a la secuencia de referencia, son buenos descriptores del genotipo del VIH, comparable con los *profiles* de información mutua, que son la representación de mejores resultados hasta el momento, reportados en [Jam04] y [Bee02].

También se comprobó que las redes bidireccionales son un buen método de clasificación para este problema, obteniéndose resultados de predicción entre el 81.4% y el 94.7%. Para la variante de la selección de la moda se obtuvieron valores de especificidad y sensitividad entre 74.1-100% y 77.5-95.8% respectivamente. En la variante de selección del valor del medio los resultados de la sensitividad y especificidad estuvieron en el orden de 84.8-96.2% y 77.7-100% respectivamente.

Conclusiones

En este trabajo se realizó un análisis de los resultados obtenidos hasta el momento, para predecir la resistencia del virus del VIH ante inhibidores de la Proteasa, a partir de la información del genotipo y fenotipo.

Se demostró que la energía 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, comparable con la descripción usando *profiles* de información mutua, que han conducido a los mejores resultados de clasificación que se habían reportado hasta el momento. También, la diferencia de energía respecto a la secuencia de referencia, es una representación comparable con las anteriores.

Se comprobó que las redes bidireccionales pueden ser usadas como un método de clasificación para este problema, obteniéndose los resultados de predicción más altos o comparables con resultados obtenidos partiendo incluso de bases de casos superiores a las aquí analizadas. Los resultados de predicción se encuentran entre el 81.4% y el 94.7%. Las dos variantes de tratamiento de la salida de estas redes son válidas y tienen resultados similares, concluyendo que efectivamente ambas pueden ser usadas en este problema. Para la variante de la selección de la moda se obtuvieron valores de especificidad y sensitividad entre 74.1-100% y 77.5-95.8% respectivamente. En la variante de selección del valor del medio los resultados sensitividad y especificidad estuvieron en el orden de 84.8-96.2% y 77.7-100% respectivamente.

Recomendaciones

1. Probar el uso de redes bidireccionales con los *profiles* de información mutua como descriptores de la secuencia.
2. Incrementar la cantidad de bases de casos, así como la cantidad de casos en ellas. Por ejemplo, se podría probar con los inhibidores de la Reverso Tranquistasa reportados en la base de Stanford.
3. Unir todas las bases en una base única donde tengamos descriptores apropiados para todos los fármacos que ellas abarcan. Ello permitiría no solo predecir si una mutación dada del VIH es o no resistente a un fármaco fijado a priori, sino también predecir los descriptores del fármaco para el cual será resistente.

Referencias bibliográficas

[Ati00] Atiya, Amir; Parlos Alexander, New Results on Recurrent Networks Training: Unifying the Algorithms and Accelerating Convergence, IEEE Transactions on Neural Network, 11(3):697-709, 2000.

[Bal98] Baldi, P., Brunak, S. (1998). Bioinformatics: The Machine Learning Approach. MIT Press, Cambridge, MA.

[Bal00] Baldi, P., Brunak, S. Frasconi P. Pollastri G. Soda G.(2000) Bidirectional Dynamics for Protein Secondary Structure Prediction In R. Sun and C.L. Giles (Eds.): Sequence Learning, LNAI 1828, pp. 80–104.

[Bal01] Baldi P., Soren B. Bioinformatics: The machine learning Approach. 2da.Edición. MIT Press. 2001.

[Bal02] Baldi P. New Machine Learning Methods for the Prediction of Protein Topologies, Universidad de Firenze, Italia, 2002.

[Bee02] Beerenwinkel Niko, Scmidt Barbara, Hauke Walter et al. Diversity and complexity of HIV-1 drug resistance: A bioinformatics approach to predicting phenotype from genotype PNAS (2002) Vol. 99 8271-8276

[Ben94] Bengio, Y. Learning Long-Term dependencies with Gradient descent is difficult, IEEE Transactions on Neural Network, 5(2):157-166, 1994.

[Ber92] Bernhard E. Boser, Isabelle M. Guyon, Vladimir N. Vapnik, A Training Algorithm for Optimal Margin Classifiers, In: D. Haussler, Proceedings of the 5th Annual Workshop on Computational Learning Theory, ACM Press, Pittsburgh, PA, 1992, pp. 144 – 152.

[Bis95] Bishop, C.M. Neural Networks for pattern recognition. Oxford University Press, 1995.

[Bod01] Boden, Mikael. A Guide to Recurrent Neural Networks and Backpropagation. Halmstad University, 2001.

[Cam97] Campbell, C. Constructives Learning Techniques for Designing Neural Netwoks Systems. Bristol University, 1997.

[Cha01] Chang C., Lin C.J. (2001), Libsvm, http://www.csie.ntu.edu.tw/cjlin/libsvm.

[Cho03] Cho Siu-Yeung. Efficient Learning in Adaptive Processing of Data Struictures. Neural Processing Letters, 17, pp. 175-190. 2003.

[DAS91] Dasarathy, B. V. (ED.). Nearest neighbor (NN) norms: NN pattern classification techniques. Los Alamitos, CA: IEEE Computer Society Press. 1991.

[Fah90] Fahlman S.E., Lebiere C. The cascade-correlation learning architecture. Advances in Neural Processing Systems. Vol. 2 pp.524-532, 1990.

[Gou97] Goutte, C. (1997), "Note on free lunches and cross-validation," Neural Computation, 9, 1211-1215,

ftp://eivind.imm.dtu.dk/dist/1997/ goutte.nflcv.ps.gz.

[Ham02] Hammer, B. Recurrent networks for structured data – a unifying approach and its properties. Cognitive Systems Research 3(2), pp. 145-165, 2002.

[Ham02a] Hammer, B., Steil, J. Tutorial: Perspectivas on Learning with RNNs. European Symposium on Artificial Neural Networks'2002, d-side publi, pp. 357-368. 2002.

[Ham03] Hammer B., Villmann T. Mathematical Aspects of Neural Networks. European Symposium on Artificial Neural Networks'2003, d-side public, pp.59-72. 2003.

[Hil95] Hilera J. Martínez V. Redes Neuronales Artificiales: Fundamentos, modelos y aplicaciones. Addison-Wesley, 1995.

[Jam04] James Murray, Robert. Predicting Human Immunodeficiency Virus Type 1 Drug Resistance From Genotype Using Machine Learning. Master of Science. University of Edinburgh. 2004

[Koh96] Kohonen, T. Self-Organizing Maps of Symbol Strings. Technical Report Lab. CISC, Nro A42, 1996.

[Kos88] Kosko, B. Bidirectional Associatives Memories. IEEE Transactions on Systems, Nro 18. pp.42-60, 1988.

[Lip87] Lipmann, R. An Introduction to Computing with Neural Nets. IEEE ASSP Magazine, Vol 4, Nro. 2, 1987, pp. 4-22.

[Miy94] 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, 1209–1220.

[Miy96] Miyazawa S., Jernigan R. L. Residue–Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading. J. Mol. Biol. 1996; 256, 623–644.

[Mit97] Mitchell Tom, Machine Learning, McGraw-Hill, 1997.

[Per90] Pearlmutter B. Dynamic Recurrent Neural Networks. DARPA Research, 1990.

[Sal05] Salazar, Sain. NEngine Versión 1.0: Herramienta para redes neuronales recurrentes.

[Sev00] Sevin Anne D., DeGruttola Victor, Nijhuis Monique et al. 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: 59-67

[Sch99] Schuster, M. On Supervised Learning from Sequential Data with application for Speech Recognition. NIST, 1999.

[Scm00] Scmidt Barbara, Walter Hauke, Moschik Brigitte et al. Simple algorithm derived from ageno-/phenotypic database to predict HIV-1 protease inhibitor resistance AIDS 2000,14:1731-1738

[Sha93] Shao, J. (1993), "Linear model selection by cross-validation," J. of the American Statistical Association, 88, 486-494.

[Som99] Somervuo, P., Kohonen, T. Self-Organizing Maps and Learning Vector Quantization for feature sequences. Neural Processing Letters, 10(2), pp.151-159, 1999.

[Son02] Sona, Diego. PhD Tesis, A proposal for an Abstract Neural Machine, Pisa University, 2002.

[Sta97] Starita, A., Sperduti, A. Supervised neural networks for the classification of structures. IEEE Transactions on Neural Networks, 8(3):714-735. 1997.

[Sto77] Stone, M. (1977), "Asymptotics for and against cross-validation", Biometrika, 64, 29-35.

[Tin99] Tin-Yau Kwok, Dit-Yan Yeung. Objective Functions for Training New Hidden Units in Constructive Neural Networks. Transactions on Neural Networks, Vol. 20, 1999.

[Tin97] Tin-Yau Kwok, Dit-Yan Yeung. Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. Transactions on Neural Networks. 1997.

[Tog92] Togneri, R. etal. A Comparison of the LBG, LVQ, SOM and GMM Algorithms for Vector Quantization Clustering Analysis. Proc. Fourth Australian International Conference on Speech Science and Technology, pp. 173-177, 1992.

[Trh64] Trhun S. A General Feed-Forward Algorithm for Gradient Descent in Connectionist Networks, Germany National Research Center, 1990.

[Tso98] Tsoi, A. Recurent neural network architectures: An overview. LNCS, Vol 1387:1-26, 1998.

[Tso97] Tsoi,A., Back,A. Discrete time recurrent neural network architectures: A unifying review, Neurocomputing, 15,(3,4):183-223, 1997.

[Vap95] Vapnik V. N., The Nature of Statistical Learning Theory. Berlin: Springer-Verlag, 1995

[Wan03] Wang Dechao, Larder Brendan. Enhanced Prediction of Lopinavir Resistance from Genotype by Use of Artificial Neural Networks The Journal Of Infectious Diseases 2003;188;653-60

[Wer90] Werbos, P. J. Backpropagation ThroughTime: What it does and How to do it. Proceedings of IEEE, Vol. 78. Nro. 10. 1990.

[Zal03] Zaldívar García, José Manuel. KNN Workshop 1.0: Suite para el desarrollo de clasificadores basados en instancias. Trabajo de diploma. Universidad Central de Las Villas 2003.

[Zip95] Zipser, David; William, Ronald, "Gradient-Based Learning Algorithm for RNN and their Computation Complexity": BackPropagation: Theory, Architectures and Applications, Y.Chauvin & Rumelhart Editions, Hillsdale, 1995.

[Zur92] Zurada, Jacek M. Introduction to artificial neural systems. West Publishing Company. 1992.

[Zel95] Zell, A. Mamier G., et al, SNSS. Stuttgart Neural Network Simulator. User Manual, Version 4.1, 1995.

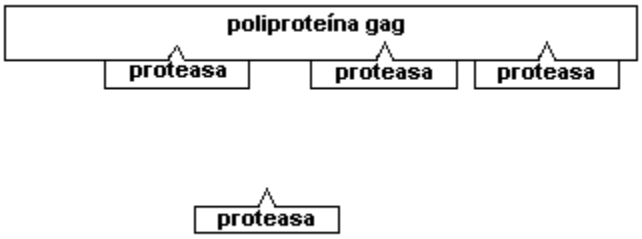
[Koc98] Kock, G. et al. Neurosolution: Integrated hardware and software for the development of neural applications. System Analysis, Modeling, Simulation (SAMS), 1998.

[Lei92] Leighton,R.R, The Aspirin / MIGRAINES Neural Network Software:User's Manual. MITRE Corporation, 1992. http://yoda.cis.temple.edu:8080/aspirin /doc/manual.ps

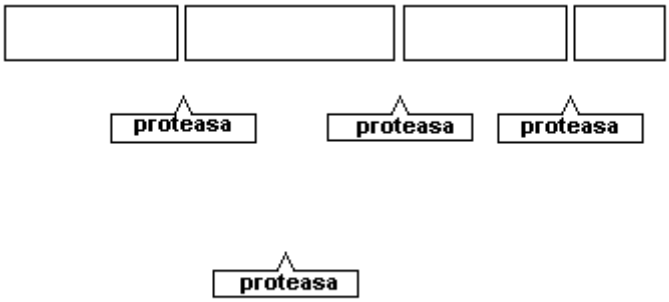
Anexos

Anexo1. Función de la Proteasa al cortar la poliproteína gag

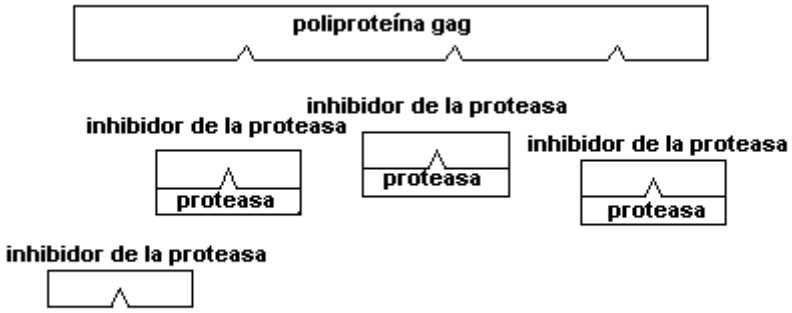
A)



B)



C)



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. Resultados del análisis estadístico para los resultados de predicción.

A) Prueba de Friedman para comparar los resultados de los SVMs usando *Energías*

Rangos

	Rango promedio
% SVM Lineal Energías	5.14
% SVM Grado 1 Energías	1.50
% SVM Grado 2 Energías	5.07
% SVM Grado 3 Var a Energías	3.64
% SVM Grado 3 Var b Energías	3.43
% SVM Radial Energias	2.21

Estadísticos de contraste[a]

N			7
Chi-cuadrado			22.426
gl			5
Sig. Monte Carlo	Sig.		**.000**
	Intervalo de confianza de 99%	Límite inferior	.000
		Límite superior	.000

a. Prueba de Friedman

B) Pruebas de los rangos con signo de Wilcoxon para comparar los SVMs usando *Energías*.

Rangos

		N	Rango promedio	Suma de rangos
% SVM Lineal d-Energías - % SVM Lineal Energías	Rangos negativos	3[a]	2.00	6.00
	Rangos positivos	0[b]	.00	.00
	Empates	4[c]		
	Total	7		
% SVM Grado 1 d-Energías - % SVM Grado 1 Energías	Rangos negativos	1[d]	1.00	1.00
	Rangos positivos	0[e]	.00	.00
	Empates	6[f]		
	Total	7		
% SVM Grado 2 d-Energías - % SVM Grado 2 Energías	Rangos negativos	7[g]	4.00	28.00
	Rangos positivos	0[h]	.00	.00
	Empates	0[i]		
	Total	7		
% SVM Grado 3 Var a d-Energías - % SVM Grado 3 Var a Energías	Rangos negativos	7[j]	4.00	28.00
	Rangos positivos	0[k]	.00	.00
	Empates	0[l]		
	Total	7		
% SVM Grado 3 Var b d-Energías - % SVM Grado 3 Var b Energías	Rangos negativos	1[m]	5.00	5.00
	Rangos positivos	6[n]	3.83	23.00
	Empates	0[o]		
	Total	7		
% SVM Radial d-Energias - % SVM Radial Energias	Rangos negativos	1[p]	2.00	2.00
	Rangos positivos	1[q]	1.00	1.00
	Empates	5[r]		
	Total	7		

a. % SVM Lineal d-Energías < % SVM Lineal Energías

b. % SVM Lineal d-Energías > % SVM Lineal Energías

c. % SVM Lineal Energías = % SVM Lineal d-Energías

d. % SVM Grado 1 d-Energías < % SVM Grado 1 Energías

e. % SVM Grado 1 d-Energías > % SVM Grado 1 Energías

f. % SVM Grado 1 Energías = % SVM Grado 1 d-Energías

g. % SVM Grado 2 d-Energías < % SVM Grado 2 Energías

h. % SVM Grado 2 d-Energías > % SVM Grado 2 Energías

i. % SVM Grado 2 Energías = % SVM Grado 2 d-Energías

j. % SVM Grado 3 Var a d-Energías < % SVM Grado 3 Var a Energías

k. % SVM Grado 3 Var a d-Energías > % SVM Grado 3 Var a Energías

l. % SVM Grado 3 Var a Energías = % SVM Grado 3 Var a d-Energías

m. % SVM Grado 3 Var b d-Energías < % SVM Grado 3 Var b Energías

n. % SVM Grado 3 Var b d-Energías > % SVM Grado 3 Var b Energías

o. % SVM Grado 3 Var b Energías = % SVM Grado 3 Var b d-Energías

p. % SVM Radial d-Energias < % SVM Radial Energias

q. % SVM Radial d-Energias > % SVM Radial Energias

r. % SVM Radial Energias = % SVM Radial d-Energias

Estadísticos de contraste[c,d]

			% SVM Lineal d-Energías - % SVM Lineal Energías	% SVM Grado 1 d-Energías - % SVM Grado 1 Energías	**% SVM Grado 2 d-Energías - % SVM Grado 2 Energías**	**% SVM Grado 3 Var a d-Energías - % SVM Grado 3 Var a Energías**	% SVM Grado 3 Var b d-Energías - % SVM Grado 3 Var b Energías	% SVM Radial d-Energias - % SVM Radial Energias
Z			-1.604[a]	-1.000[a]	-2.366[a]	-2.366[a]	-1.521[b]	-.447[a]
Sig. Monte Carlo (bilateral)	Sig.		**.255**	**1.000**	**.016**	**.016**	**.156**	**1.000**
	Intervalo de confianza de 99%	Límite inferior	.238	.974	.011	.011	.143	.974
		Límite superior	.272	1.000	.020	.020	.170	1.000

a. Basado en los rangos positivos.

b. Basado en los rangos negativos.

c. Prueba de los rangos con signo de Wilcoxon

d. Basado en 10000 tablas muestrales con semilla de inicio 329836257.

D) Prueba de Friedman para comparar los resultados de los SVMs usando *ΔEnergías*.

Rangos

	Rango promedio
% SVM Lineal d-Energías	5.43
% SVM Grado 1 d-Energías	3.00
% SVM Grado 2 d-Energías	2.07
% SVM Grado 3 Var a d-Energías	1.29
% SVM Grado 3 Var b d-Energías	5.57
% SVM Radial d-Energias	3.64

Estadísticos de contraste[a]

N			7
Chi-cuadrado			32.294
gl			5
Sig. Monte Carlo	Sig.		**.000**
	Intervalo de confianza de 99%	Límite inferior	.000
		Límite superior	.000

a. Prueba de Friedman

E) Pruebas de los rangos con signo de Wilcoxon para comparar las redes bidireccionales usando *ΔEnergías*.

Rangos

		N	Rango promedio	Suma de rangos
% BRR con valor del medio d-Energias - % BRR con la moda d-Energias	Rangos negativos	4[a]	2.75	11.00
	Rangos positivos	1[b]	4.00	4.00
	Empates	2[c]		
	Total	7		

a. % BRR con valor del medio d-Energias < % BRR con la moda d-Energias

b. % BRR con valor del medio d-Energias > % BRR con la moda d-Energias

c. % BRR con la moda d-Energias = % BRR con valor del medio d-Energias

Estadísticos de contraste[b,c]

			% BRR con valor del medio d-Energias - % BRR con la moda d-Energias
Z			-.944[a]
Sig. Monte Carlo (bilateral)	Sig.		**.439**
	Intervalo de confianza de 99%	Límite inferior	.418
		Límite superior	.461

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 1573343031.

F) Prueba de Friedman para comparar las redes bidireccionales con el SmartMLP usando *ΔEnergías*.

Rangos

	Rango promedio
% Smart MLP d-Energías	1.00
% BRR con la moda d-Energias	2.71
% BRR con valor del medio d-Energias	2.29

Estadísticos de contraste[a]

N			7
Chi-cuadrado			12.000
gl			2
Sig. Monte Carlo	Sig.		**.001**
	Intervalo de confianza de 99%	Límite inferior	.000
		Límite superior	.001

a. Prueba de Friedman

G) Prueba de los rangos con signo de Wilcoxon para comparar las redes bidireccionales contra los SVM usando *ΔEnergías*.

Rangos

		N	Rango promedio	Suma de rangos
% BRR con la moda d-Energias - % SVM Lineal d-Energías	Rangos negativos	0[a]	.00	.00
	Rangos positivos	7[b]	4.00	28.00
	Empates	0[c]		
	Total	7		
% BRR con la moda d-Energias - % SVM Grado 3 Var b d-Energías	Rangos negativos	1[d]	1.00	1.00
	Rangos positivos	6[e]	4.50	27.00
	Empates	0[f]		
	Total	7		

a. % BRR con la moda d-Energias < % SVM Lineal d-Energías

b. % BRR con la moda d-Energias > % SVM Lineal d-Energías

c. % SVM Lineal d-Energías = % BRR con la moda d-Energias

d. % BRR con la moda d-Energias < % SVM Grado 3 Var b d-Energías

e. % BRR con la moda d-Energias > % SVM Grado 3 Var b d-Energías

f. % SVM Grado 3 Var b d-Energías = % BRR con la moda d-Energias

Estadísticos de contraste[b,c]

			% BRR con la moda d-Energias - % SVM Lineal d-Energías	% BRR con la moda d-Energias - % SVM Grado 3 Var b d-Energías
Z			-2.366[a]	-2.197[a]
Sig. asintót. (bilateral)			.018	.028
Sig. Monte Carlo (bilateral)	Sig.		**.015**	**.032**
	Intervalo de confianza de 99%	Límite inferior	.010	.025
		Límite superior	.019	.038

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 113410539.

Anexo 4. Análisis estadísticos de la especificidad.

A) Prueba de los rangos con signo de Wilcoxon para comparar la especificidad de las dos variantes de redes bidireccionales.

Rangos

		N	Rango promedio	Suma de rangos
Espe BRR con el valor del medio d-Energias - Espe BRR con la moda d-Energias	Rangos negativos	3[a]	3.33	10.00
	Rangos positivos	2[b]	2.50	5.00
	Empates	2[c]		
	Total	7		

a. Espe BRR con el valor del medio d-Energias < Espe BRR con la moda d-Energias

b. Espe BRR con el valor del medio d-Energias > Espe BRR con la moda d-Energias

c. Espe BRR con la moda d-Energias = Espe BRR con el valor del medio d-Energias

Estadísticos de contraste[b,c]

			Espe BRR con el valor del medio d-Energias - Espe BRR con la moda d-Energias
Z			-.674[a]
Sig. Monte Carlo (bilateral)	Sig.		**.621**
	Intervalo de confianza de 99%	Límite inferior	.597
		Límite superior	.644

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 1660843777.

B) Prueba de los rangos con signo de Wilcoxon para comparar las redes bidireccionales con los otros métodos.

Rangos

		N	Rango promedio	Suma de rangos
Espe BRR con el valor del medio d-Energias - Espe KNN James	Rangos negativos	4[a]	4.25	17.00
	Rangos positivos	2[b]	2.00	4.00
	Empates	0[c]		
	Total	6		
Espe BRR con el valor del medio d-Energias - Espe New Dtree James	Rangos negativos	2[d]	3.50	7.00
	Rangos positivos	4[e]	3.50	14.00
	Empates	0[f]		
	Total	6		
Espe BRR con el valor del medio d-Energias - Espe Clas DTree James	Rangos negativos	0[g]	.00	.00
	Rangos positivos	5[h]	3.00	15.00
	Empates	0[i]		
	Total	5		
Espe BRR con el valor del medio d-Energias - Espe Clas DTree Niko	Rangos negativos	0[j]	.00	.00
	Rangos positivos	5[k]	3.00	15.00
	Empates	0[l]		
	Total	5		

a. Espe BRR con el valor del medio d-Energias < Espe KNN James

b. Espe BRR con el valor del medio d-Energias > Espe KNN James

c. Espe KNN James = Espe BRR con el valor del medio d-Energias

d. Espe BRR con el valor del medio d-Energias < Espe New Dtree James

e. Espe BRR con el valor del medio d-Energias > Espe New Dtree James

f. Espe New Dtree James = Espe BRR con el valor del medio d-Energias

g. Espe BRR con el valor del medio d-Energias < Espe Clas DTree James

h. Espe BRR con el valor del medio d-Energias > Espe Clas DTree James

i. Espe Clas DTree James = Espe BRR con el valor del medio d-Energias

j. Espe BRR con el valor del medio d-Energias < Espe Clas DTree Niko

k. Espe BRR con el valor del medio d-Energias > Espe Clas DTree Niko

l. Espe Clas DTree Niko = Espe BRR con el valor del medio d-Energias

Estadísticos de contraste[c,d]

			Espe BRR con el valor del medio d-Energias - Espe KNN James	Espe BRR con el valor del medio d-Energias - Espe New DTree James	Espe BRR con el valor del medio d-Energias - Espe Clas DTree James	Espe BRR con el valor del medio d-Energias - Espe Clas DTree Niko
Z			-1.363[a]	-.734[b]	-2.023[b]	-2.023[b]
Sig. Monte Carlo (bilateral)	Sig.		**.217**	**.555**	**.064**	**.064**
	Intervalo de confi de 99%	Límite inferior	.201	.532	.055	.055
		Límite superior	.233	.578	.073	.073

a. Basado en los rangos positivos.

b. Basado en los rangos negativos.

c. Prueba de los rangos con signo de Wilcoxon

d. Basado en 10000 tablas muestrales con semilla de inicio 677935123.

Rangos

		N	Rango promedio	Suma de rangos
Espe BRR con la moda d-Energias - Espe KNN James	Rangos negativos	4[a]	3.50	14.00
	Rangos positivos	2[b]	3.50	7.00
	Empates	0[c]		
	Total	6		
Espe BRR con la moda d-Energias - Espe New Dtree James	Rangos negativos	0[d]	.00	.00
	Rangos positivos	6[e]	3.50	21.00
	Empates	0[f]		
	Total	6		
Espe BRR con la moda d-Energias - Espe Clas DTree James	Rangos negativos	0[g]	.00	.00
	Rangos positivos	5[h]	3.00	15.00
	Empates	0[i]		
	Total	5		
Espe BRR con la moda d-Energias - Espe Clas DTree Niko	Rangos negativos	1[j]	1.00	1.00
	Rangos positivos	4[k]	3.50	14.00
	Empates	0[l]		
	Total	5		

a. Espe BRR con la moda d-Energias < Espe KNN James

b. Espe BRR con la moda d-Energias > Espe KNN James

c. Espe KNN James = Espe BRR con la moda d-Energias

d. Espe BRR con la moda d-Energias < Espe New Dtree James

e. Espe BRR con la moda d-Energias > Espe New Dtree James

f. Espe New Dtree James = Espe BRR con la moda d-Energias

g. Espe BRR con la moda d-Energias < Espe Clas DTree James

h. Espe BRR con la moda d-Energias > Espe Clas DTree James

i. Espe Clas DTree James = Espe BRR con la moda d-Energias

j. Espe BRR con la moda d-Energias < Espe Clas DTree Niko

k. Espe BRR con la moda d-Energias > Espe Clas DTree Niko

l. Espe Clas DTree Niko = Espe BRR con la moda d-Energias

Estadísticos de contraste[c,d]

			Espe BRR con la moda d-Energias - Espe KNN James	Espe BRR con la moda d-Energias - Espe New DTree James	Espe BRR con la moda d-Energias - Espe Clas DTree James	Espe BRR con la moda d-Energias - Espe Clas DTree Niko
Z			-.734[a]	-2.201[b]	-2.023[b]	-1.753[b]
Sig. Monte Carlo (bilateral)	Sig.		**.565**	**.029**	**.062**	**.125**
	Intervalo de confi de 99%	Límite inferior	.541	.022	.053	.112
		Límite superior	.588	.035	.071	.137

a. Basado en los rangos positivos.

b. Basado en los rangos negativos.

c. Prueba de los rangos con signo de Wilcoxon

d. Basado en 10000 tablas muestrales con semilla de inicio 1201206483.

Anexo 5. Análisis estadístico de la sensitividad

A) Prueba de los rangos con signo de Wilcoxon para comparar la sensitividad de las dos variantes de redes bidireccionales.

Rangos

		N	Rango promedio	Suma de rangos
Sens BRR con el valor del medio d-Energias - Sens BRR con la moda d-Energias	Rangos negativos	3[a]	3.67	11.00
	Rangos positivos	3[b]	3.33	10.00
	Empates	1[c]		
	Total	7		

a. Sens BRR con el valor del medio d-Energias < Sens BRR con la moda d-Energias

b. Sens BRR con el valor del medio d-Energias > Sens BRR con la moda d-Energias

c. Sens BRR con la moda d-Energias = Sens BRR con el valor del medio d-Energias

Estadísticos de contraste[b,c]

			Sens BRR con el valor del medio d-Energias - Sens BRR con la moda d-Energias
Z			-.105[a]
Sig. asintót. (bilateral)			**.917**
Sig. Monte Carlo (bilateral)	Intervalo de confianza de 99%	Límite inferior	.974
		Límite superior	1.000

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 1310155034.

B) Prueba de Wilcoxon comparando resultados de los resultados de sensitividad de las dos variantes de red bidireccional con todos los métodos antes reportados

Rangos

		N	Rango promedio	Suma de rangos
Sens BRR con la moda d-Energias - Sens New DTree James	Rangos negativos	3[a]	3.00	9.00
	Rangos positivos	3[b]	4.00	12.00
	Empates	0[c]		
	Total	6		
Sens BRR con la moda d-Energias - Sens KNN James	Rangos negativos	0[d]	.00	.00
	Rangos positivos	6[e]	3.50	21.00
	Empates	0[f]		
	Total	6		
Sens BRR con la moda d-Energias - Sens Clas DTree James	Rangos negativos	2[g]	2.50	5.00
	Rangos positivos	3[h]	3.33	10.00
	Empates	0[i]		
	Total	5		
Sens BRR con la moda d-Energias - Sens Clas DTree Niko	Rangos negativos	2[j]	2.00	4.00
	Rangos positivos	3[k]	3.67	11.00
	Empates	0[l]		
	Total	5		

a. Sens BRR con la moda d-Energias < Sens New DTree James

b. Sens BRR con la moda d-Energias > Sens New DTree James

c. Sens New DTree James = Sens BRR con la moda d-Energias

d. Sens BRR con la moda d-Energias < Sens KNN James

e. Sens BRR con la moda d-Energias > Sens KNN James

f. Sens KNN James = Sens BRR con la moda d-Energias

g. Sens BRR con la moda d-Energias < Sens Clas DTree James

h. Sens BRR con la moda d-Energias > Sens Clas DTree James

i. Sens Clas DTree James = Sens BRR con la moda d-Energias

j. Sens BRR con la moda d-Energias < Sens Clas DTree Niko

k. Sens BRR con la moda d-Energias > Sens Clas DTree Niko

l. Sens Clas DTree Niko = Sens BRR con la moda d-Energias

Estadísticos de contraste[b,c]

			Sens BRR con la moda d-Energias - Sens New DTree James	Sens BRR con la moda d-Energias - Sens KNN James	Sens BRR con la moda d-Energias - Sens Clas DTree James	Sens BRR con la moda d-Energias - Sens Clas DTree Niko
Z			-.314[a]	-2.201[a]	-.674[a]	-.944[a]
Sig. Monte Carlo (bilateral)	Sig.		**.852**	**.031**	**.631**	**.440**
	Intervalo de confia de 99%	Límite inferior	.826	.024	.607	.419
		Límite superior	.877	.037	.655	.461

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

c. Basado en 10000 tablas muestrales con semilla de inicio 876491272.

Anexo 6 Energías de contacto de los aminoácidos

Aminoácido	eir 1996	qi, 1996	**Energías** 1996 -0,6*qi*eir/2 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